

GNSS-SDR

0.0.13

Generated by Doxygen 1.8.14

Contents

1	Main Page	1
1.1	Contents	1
1.2	Overview	2
1.3	Building GNSS-SDR	3
1.3.1	Debug and Release builds	5
1.3.2	Updating GNSS-SDR	5
1.4	Using GNSS-SDR	5
1.5	Control plane	6
1.5.1	Configuration	6
1.5.2	GNSS block factory	7
1.6	Signal Processing plane	7
1.6.1	Signal Source	8
1.6.2	Signal Conditioner	8
1.6.3	Channel	9
1.6.3.1	Acquisition	9
1.6.3.2	Tracking	10
1.6.3.3	Decoding of the navigation message	11
1.6.4	Observables	11
1.6.5	Computation of Position, Velocity and Time	11
1.7	About the software license	12
1.8	Publications and Credits	13
1.9	Ok, now what?	14

2	Reference Documents	15
2.1	Interface Control Documents	15
2.1.1	GPS	15
2.1.2	GLONASS	15
2.1.3	Galileo	16
2.1.4	BeiDou	16
2.1.5	Satellite Based Augmentation Systems (SBAS)	17
2.2	Other Standards	17
2.2.1	RINEX	17
2.2.2	NMEA	18
2.2.3	KML	18
2.2.4	C++ Standards	18
2.2.5	Positioning protocols in wireless communication networks	18
3	Signal model	21
3.1	GNSS signal model	21
3.1.1	Global Positioning System (GPS) signal in space	22
3.1.2	GLONASS signal in space	23
3.1.3	Galileo signal in space	24
3.1.4	Reference	27
4	Todo List	29
5	Module Index	31
5.1	Modules	31
6	Hierarchical Index	33
6.1	Class Hierarchy	33
7	Class Index	41
7.1	Class List	41
8	File Index	53
8.1	File List	53

9	Module Documentation	67
9.1	Common definitions	67
9.1.1	Detailed Description	67
9.2	GPS_L2	68
9.2.1	Detailed Description	68
9.3	Gps_cnav_decoder	69
9.3.1	Detailed Description	69
9.3.2	Macro Definition Documentation	69
9.3.2.1	GPS_L2_V27_HISTORY_LENGTH_BITS	69
9.3.2.2	GPS_L2C_V27_DECODE_BITS	69
9.3.2.3	GPS_L2C_V27_DELAY_BITS	70
9.3.2.4	GPS_L2C_V27_INIT_BITS	70
10	Class Documentation	71
10.1	Acq_Conf Class Reference	71
10.1.1	Detailed Description	72
10.2	Acquisition_Dump_Reader Class Reference	72
10.2.1	Detailed Description	72
10.2.2	Constructor & Destructor Documentation	72
10.2.2.1	Acquisition_Dump_Reader() [1/2]	73
10.2.2.2	Acquisition_Dump_Reader() [2/2]	73
10.2.3	Member Function Documentation	73
10.2.3.1	operator=() [1/2]	73
10.2.3.2	operator=() [2/2]	73
10.3	Acquisition_msg_rx Class Reference	73
10.3.1	Detailed Description	74
10.3.2	Constructor & Destructor Documentation	74
10.3.2.1	~Acquisition_msg_rx()	74
10.4	AcquisitionInterface Class Reference	75
10.4.1	Detailed Description	76
10.5	Ad9361FpgaSignalSource Class Reference	76

10.5.1 Detailed Description	76
10.5.2 Member Function Documentation	76
10.5.2.1 implementation()	77
10.6 Agnss_Ref_Location Class Reference	77
10.6.1 Detailed Description	77
10.6.2 Constructor & Destructor Documentation	77
10.6.2.1 Agnss_Ref_Location()	78
10.6.3 Member Function Documentation	78
10.6.3.1 serialize()	78
10.7 Agnss_Ref_Time Class Reference	78
10.7.1 Detailed Description	79
10.7.2 Constructor & Destructor Documentation	79
10.7.2.1 Agnss_Ref_Time()	79
10.7.3 Member Function Documentation	79
10.7.3.1 serialize()	79
10.8 alm_t Struct Reference	79
10.8.1 Detailed Description	80
10.9 ambc_t Struct Reference	80
10.9.1 Detailed Description	80
10.10 ArraySignalConditioner Class Reference	80
10.10.1 Detailed Description	81
10.10.2 Constructor & Destructor Documentation	81
10.10.2.1 ArraySignalConditioner()	81
10.10.2.2 ~ArraySignalConditioner()	81
10.10.3 Member Function Documentation	82
10.10.3.1 implementation()	82
10.11 Bayesian_estimator Class Reference	82
10.11.1 Detailed Description	82
10.12 beamformer Class Reference	83
10.12.1 Detailed Description	83

10.13BeamformerFilter Class Reference	83
10.13.1 Detailed Description	84
10.13.2 Member Function Documentation	84
10.13.2.1 implementation()	84
10.14beidou_b1i_telemetry_decoder_gs Class Reference	84
10.14.1 Detailed Description	85
10.14.2 Constructor & Destructor Documentation	85
10.14.2.1 ~beidou_b1i_telemetry_decoder_gs()	85
10.14.3 Member Function Documentation	85
10.14.3.1 general_work()	86
10.14.3.2 set_channel()	86
10.14.3.3 set_satellite()	86
10.15beidou_b3i_telemetry_decoder_gs Class Reference	86
10.15.1 Detailed Description	87
10.15.2 Constructor & Destructor Documentation	87
10.15.2.1 ~beidou_b3i_telemetry_decoder_gs()	87
10.15.3 Member Function Documentation	87
10.15.3.1 general_work()	87
10.15.3.2 set_channel()	88
10.15.3.3 set_satellite()	88
10.16Beidou_Dnav_Almanac Class Reference	88
10.16.1 Detailed Description	89
10.16.2 Constructor & Destructor Documentation	89
10.16.2.1 Beidou_Dnav_Almanac()	89
10.16.3 Member Data Documentation	89
10.16.3.1 d_A_f0	89
10.16.3.2 d_A_f1	89
10.16.3.3 d_e_eccentricity	89
10.16.3.4 d_M_0	90
10.16.3.5 d_OMEGA	90

10.16.3.6 d_OMEGA0	90
10.16.3.7 d_OMEGA_DOT	90
10.16.3.8 d_sqrt_A	90
10.16.3.9 d_Toa	91
10.16.3.10 d_satellite_PRN	91
10.16.3.11 d_SV_health	91
10.17 Beidou_Dnav_Ephemeris Class Reference	91
10.17.1 Detailed Description	94
10.17.2 Constructor & Destructor Documentation	94
10.17.2.1 Beidou_Dnav_Ephemeris()	94
10.17.3 Member Function Documentation	94
10.17.3.1 satellitePosition()	94
10.17.3.2 serialize()	95
10.17.3.3 sv_clock_drift()	96
10.17.3.4 sv_clock_relativistic_term()	96
10.17.4 Member Data Documentation	96
10.17.4.1 b_alert_flag	96
10.17.4.2 b_antispoofing_flag	97
10.17.4.3 b_fit_interval_flag	97
10.17.4.4 b_integrity_status_flag	97
10.17.4.5 d_A_f0	97
10.17.4.6 d_A_f1	98
10.17.4.7 d_A_f2	98
10.17.4.8 d_AODC	98
10.17.4.9 d_AODE	98
10.17.4.10 d_Cic	99
10.17.4.11 d_Cis	99
10.17.4.12 d_Crc	99
10.17.4.13 d_Crs	99
10.17.4.14 d_Cuc	100

10.17.4.15d_Cus	100
10.17.4.16d_Delta_n	100
10.17.4.17d_dtr	100
10.17.4.18d_eccentricity	101
10.17.4.19d_i_0	101
10.17.4.20d_IDOT	101
10.17.4.21d_M_0	101
10.17.4.22d_OMEGA	102
10.17.4.23d_OMEGA0	102
10.17.4.24d_OMEGA_DOT	102
10.17.4.25d_satClkDrift	102
10.17.4.26d_satpos_X	103
10.17.4.27d_satpos_Y	103
10.17.4.28d_satpos_Z	103
10.17.4.29d_satvel_X	103
10.17.4.30d_satvel_Y	103
10.17.4.31d_satvel_Z	104
10.17.4.32d_sqrt_A	104
10.17.4.33d_TGD1	104
10.17.4.34d_TGD2	104
10.17.4.35d_Toc	104
10.17.4.36d_Toe	105
10.17.4.37d_TOW	105
10.17.4.38_AODO	105
10.17.4.39_BEIDOU_week	105
10.17.4.40_nav_type	106
10.17.4.41_satellite_PRN	106
10.17.4.42_sig_type	106
10.17.4.43_SV_accuracy	106
10.17.4.44_satelliteBlock	106

10.18	Beidou_Dnav_Iono Class Reference	107
10.18.1	Detailed Description	107
10.18.2	Constructor & Destructor Documentation	107
10.18.2.1	Beidou_Dnav_Iono()	108
10.18.3	Member Function Documentation	108
10.18.3.1	serialize()	108
10.18.4	Member Data Documentation	108
10.18.4.1	d_alpha0	108
10.18.4.2	d_alpha1	108
10.18.4.3	d_alpha2	109
10.18.4.4	d_alpha3	109
10.18.4.5	d_beta0	109
10.18.4.6	d_beta1	109
10.18.4.7	d_beta2	110
10.18.4.8	d_beta3	110
10.18.4.9	valid	110
10.19	Beidou_Dnav_Navigation_Message Class Reference	110
10.19.1	Detailed Description	111
10.19.2	Constructor & Destructor Documentation	111
10.19.2.1	Beidou_Dnav_Navigation_Message()	111
10.19.3	Member Function Documentation	112
10.19.3.1	d1_subframe_decoder()	112
10.19.3.2	d2_subframe_decoder()	112
10.19.3.3	get_ephemeris()	112
10.19.3.4	get_iono()	112
10.19.3.5	get_utc_model()	112
10.19.3.6	have_new_almanac()	113
10.19.3.7	have_new_ephemeris()	113
10.19.3.8	have_new_iono()	113
10.19.3.9	have_new_utc_model()	113

10.19.3.10	<code>satellitePosition()</code>	113
10.19.3.11	<code>set_satellite_PRN()</code>	113
10.19.3.12	<code>sv_clock_correction()</code>	114
10.19.3.13	<code>utc_time()</code>	114
10.20	Beidou_Dnav_Utc_Model Class Reference	114
10.20.1	Detailed Description	115
10.20.2	Member Data Documentation	115
10.20.2.1	<code>d_A0_GAL</code>	115
10.20.2.2	<code>d_A0_GLO</code>	115
10.20.2.3	<code>d_A0_GPS</code>	115
10.20.2.4	<code>d_A0_UTC</code>	115
10.20.2.5	<code>d_A1_GAL</code>	116
10.20.2.6	<code>d_A1_GLO</code>	116
10.20.2.7	<code>d_A1_GPS</code>	116
10.20.2.8	<code>d_A1_UTC</code>	116
10.20.2.9	<code>d_DeltaT_LSF</code>	116
10.20.2.10	<code>d_DeltaT_LS</code>	117
10.20.2.11	<code>i_DN</code>	117
10.20.2.12	<code>WN_LSF</code>	117
10.21	BeidouB1iDIIPIITracking Class Reference	117
10.21.1	Detailed Description	118
10.21.2	Member Function Documentation	118
10.21.2.1	<code>set_channel()</code>	118
10.21.2.2	<code>set_gnss_synchro()</code>	118
10.21.2.3	<code>stop_tracking()</code>	119
10.22	BeidouB1iPcpsAcquisition Class Reference	119
10.22.1	Detailed Description	120
10.22.2	Member Function Documentation	120
10.22.2.1	<code>implementation()</code>	120
10.22.2.2	<code>init()</code>	120

10.22.2.3 mag()	121
10.22.2.4 reset()	121
10.22.2.5 set_channel()	121
10.22.2.6 set_channel_fsm()	121
10.22.2.7 set_doppler_max()	121
10.22.2.8 set_doppler_step()	122
10.22.2.9 set_gnss_synchro()	122
10.22.2.10 set_local_code()	122
10.22.2.11 set_resampler_latency()	122
10.22.2.12 set_state()	122
10.22.2.13 set_threshold()	123
10.22.2.14 stop_acquisition()	123
10.23 BeidouB1iTelemetryDecoder Class Reference	123
10.23.1 Detailed Description	124
10.23.2 Member Function Documentation	124
10.23.2.1 implementation()	124
10.24 BeidouB3iDIIPIITracking Class Reference	124
10.24.1 Detailed Description	125
10.24.2 Member Function Documentation	125
10.24.2.1 set_channel()	125
10.24.2.2 set_gnss_synchro()	125
10.24.2.3 stop_tracking()	126
10.25 BeidouB3iPcpsAcquisition Class Reference	126
10.25.1 Detailed Description	127
10.25.2 Member Function Documentation	127
10.25.2.1 implementation()	127
10.25.2.2 init()	127
10.25.2.3 mag()	128
10.25.2.4 reset()	128
10.25.2.5 set_channel()	128

10.25.2.6 <code>set_channel_fsm()</code>	128
10.25.2.7 <code>set_doppler_max()</code>	129
10.25.2.8 <code>set_doppler_step()</code>	129
10.25.2.9 <code>set_gnss_synchro()</code>	129
10.25.2.10 <code>set_local_code()</code>	129
10.25.2.11 <code>set_resampler_latency()</code>	129
10.25.2.12 <code>set_state()</code>	130
10.25.2.13 <code>set_threshold()</code>	130
10.25.2.14 <code>stop_acquisition()</code>	130
10.26 <code>BeidouB3iTelemetryDecoder</code> Class Reference	130
10.26.1 Detailed Description	131
10.26.2 Member Function Documentation	131
10.26.2.1 <code>implementation()</code>	131
10.27 <code>byte_x2_to_complex_byte</code> Class Reference	131
10.27.1 Detailed Description	132
10.28 <code>ByteToShort</code> Class Reference	132
10.28.1 Detailed Description	133
10.28.2 Member Function Documentation	133
10.28.2.1 <code>implementation()</code>	133
10.29 <code>Channel</code> Class Reference	133
10.29.1 Detailed Description	134
10.29.2 Constructor & Destructor Documentation	134
10.29.2.1 <code>Channel()</code>	135
10.29.2.2 <code>~Channel()</code>	135
10.29.3 Member Function Documentation	135
10.29.3.1 <code>connect()</code>	135
10.29.3.2 <code>get_left_block()</code>	135
10.29.3.3 <code>get_left_block_acq()</code>	136
10.29.3.4 <code>get_left_block_trk()</code>	136
10.29.3.5 <code>implementation()</code>	136

10.29.3.6 <code>set_signal()</code>	136
10.29.3.7 <code>start_acquisition()</code>	136
10.29.3.8 <code>stop_channel()</code>	137
10.30 <code>Channel_Event</code> Class Reference	137
10.30.1 Detailed Description	137
10.31 <code>channel_msg_receiver_cc</code> Class Reference	137
10.31.1 Detailed Description	138
10.31.2 Constructor & Destructor Documentation	138
10.31.2.1 <code>~channel_msg_receiver_cc()</code>	138
10.32 <code>channel_status_msg_receiver</code> Class Reference	138
10.32.1 Detailed Description	139
10.32.2 Constructor & Destructor Documentation	139
10.32.2.1 <code>~channel_status_msg_receiver()</code>	139
10.32.3 Member Function Documentation	139
10.32.3.1 <code>get_current_status_map()</code>	139
10.32.3.2 <code>get_current_status_pvt()</code>	140
10.33 <code>ChannelFsm</code> Class Reference	140
10.33.1 Detailed Description	140
10.34 <code>ChannelInterface</code> Class Reference	141
10.34.1 Detailed Description	141
10.35 <code>cl_fft_plan</code> Struct Reference	142
10.35.1 Detailed Description	142
10.36 <code>clFFT_Complex</code> Struct Reference	142
10.36.1 Detailed Description	142
10.37 <code>clFFT_Dim3</code> Struct Reference	143
10.37.1 Detailed Description	143
10.38 <code>clFFT_SplitComplex</code> Struct Reference	143
10.38.1 Detailed Description	143
10.39 <code>cnav_msg_decoder_t</code> Struct Reference	143
10.39.1 Detailed Description	144

10.39.2 Member Data Documentation	144
10.39.2.1 part1	144
10.39.2.2 part2	144
10.40cnav_msg_t Struct Reference	144
10.40.1 Detailed Description	145
10.40.2 Member Data Documentation	145
10.40.2.1 alert	145
10.40.2.2 msg_id	145
10.40.2.3 prn	145
10.40.2.4 raw_msg	146
10.40.2.5 tow	146
10.41cnav_v27_part_t Struct Reference	146
10.41.1 Detailed Description	146
10.41.2 Member Data Documentation	147
10.41.2.1 crc_ok	147
10.41.2.2 dec	147
10.41.2.3 decisions	147
10.41.2.4 decoded	147
10.41.2.5 init	147
10.41.2.6 invert	148
10.41.2.7 message_lock	148
10.41.2.8 n_crc_fail	148
10.41.2.9 n_decoded	148
10.41.2.10n_symbols	148
10.41.2.11preamble_seen	149
10.41.2.12symbols	149
10.42Command_Event Class Reference	149
10.42.1 Detailed Description	149
10.43complex_byte_to_float_x2 Class Reference	150
10.43.1 Detailed Description	150

10.44	complex_float_to_complex_byte Class Reference	150
10.44.1	Detailed Description	151
10.45	Concurrent_Map< Data > Class Template Reference	151
10.45.1	Detailed Description	151
10.46	Concurrent_Queue< Data > Class Template Reference	152
10.46.1	Detailed Description	152
10.47	ConfigurationInterface Class Reference	152
10.47.1	Detailed Description	153
10.48	conjugate_cc Class Reference	153
10.48.1	Detailed Description	154
10.49	conjugate_ic Class Reference	154
10.49.1	Detailed Description	154
10.50	conjugate_sc Class Reference	155
10.50.1	Detailed Description	155
10.51	ControlThread Class Reference	155
10.51.1	Detailed Description	156
10.51.2	Constructor & Destructor Documentation	156
10.51.2.1	ControlThread() [1/2]	156
10.51.2.2	ControlThread() [2/2]	156
10.51.2.3	~ControlThread()	157
10.51.3	Member Function Documentation	157
10.51.3.1	flowgraph()	157
10.51.3.2	run()	157
10.51.3.3	set_control_queue()	157
10.52	Cpu_Multicorrelator Class Reference	158
10.52.1	Detailed Description	158
10.53	Cpu_Multicorrelator_16sc Class Reference	158
10.53.1	Detailed Description	159
10.54	Cpu_Multicorrelator_Real_Codes Class Reference	159
10.54.1	Detailed Description	159

10.55cshort_to_float_x2 Class Reference	160
10.55.1 Detailed Description	160
10.56CubatureFilter Class Reference	160
10.56.1 Detailed Description	161
10.57cuda_multicorrelator Class Reference	161
10.57.1 Detailed Description	161
10.58CustomUDPSignalSource Class Reference	161
10.58.1 Detailed Description	162
10.58.2 Member Function Documentation	162
10.58.2.1 implementation()	162
10.59dgps_t Struct Reference	162
10.59.1 Detailed Description	163
10.60direct_resampler_conditioner_cb Class Reference	163
10.60.1 Detailed Description	163
10.61direct_resampler_conditioner_cc Class Reference	164
10.61.1 Detailed Description	164
10.62direct_resampler_conditioner_cs Class Reference	164
10.62.1 Detailed Description	165
10.63DirectResamplerConditioner Class Reference	165
10.63.1 Detailed Description	166
10.63.2 Member Function Documentation	166
10.63.2.1 implementation()	166
10.64Dll_Pll_Conf Class Reference	166
10.64.1 Detailed Description	167
10.65Dll_Pll_Conf_Fpga Class Reference	167
10.65.1 Detailed Description	169
10.66dll_pll_veml_tracking Class Reference	169
10.66.1 Detailed Description	169
10.67dll_pll_veml_tracking_fpga Class Reference	170
10.67.1 Detailed Description	170

10.67.2 Constructor & Destructor Documentation	170
10.67.2.1 ~dll_pll_veml_tracking_fpga()	171
10.67.3 Member Function Documentation	171
10.67.3.1 general_work()	171
10.67.3.2 reset()	171
10.67.3.3 set_channel()	171
10.67.3.4 set_gnss_synchro()	171
10.67.3.5 start_tracking()	172
10.67.3.6 stop_tracking()	172
10.68eph_t Struct Reference	172
10.68.1 Detailed Description	173
10.69erp_t Struct Reference	173
10.69.1 Detailed Description	173
10.70erpd_t Struct Reference	173
10.70.1 Detailed Description	173
10.71Exponential_Smoother Class Reference	174
10.71.1 Detailed Description	174
10.71.2 Constructor & Destructor Documentation	174
10.71.2.1 Exponential_Smoother() [1/2]	174
10.71.2.2 ~Exponential_Smoother()	175
10.71.2.3 Exponential_Smoother() [2/2]	175
10.71.3 Member Function Documentation	175
10.71.3.1 operator=()	175
10.71.3.2 set_alpha()	175
10.71.3.3 set_samples_for_initialization()	175
10.72exterr_t Struct Reference	176
10.72.1 Detailed Description	176
10.73fcbd_t Struct Reference	176
10.73.1 Detailed Description	176
10.74file_t Struct Reference	177

10.74.1 Detailed Description	177
10.75FileConfiguration Class Reference	177
10.75.1 Detailed Description	178
10.76FileSignalSource Class Reference	178
10.76.1 Detailed Description	179
10.76.2 Member Function Documentation	179
10.76.2.1 implementation()	179
10.77FirFilter Class Reference	179
10.77.1 Detailed Description	180
10.77.2 Constructor & Destructor Documentation	180
10.77.2.1 FirFilter()	180
10.77.2.2 ~FirFilter()	180
10.77.3 Member Function Documentation	181
10.77.3.1 implementation()	181
10.78FlexibandSignalSource Class Reference	181
10.78.1 Detailed Description	182
10.78.2 Member Function Documentation	182
10.78.2.1 implementation()	182
10.79Fmcomms2SignalSource Class Reference	182
10.79.1 Detailed Description	183
10.79.2 Member Function Documentation	183
10.79.2.1 implementation()	183
10.80Fpga_Acquisition Class Reference	183
10.80.1 Detailed Description	184
10.80.2 Constructor & Destructor Documentation	184
10.80.2.1 Fpga_Acquisition()	184
10.80.2.2 ~Fpga_Acquisition()	184
10.80.3 Member Function Documentation	185
10.80.3.1 close_device()	185
10.80.3.2 configure_acquisition()	185

10.80.3.3 open_device()	185
10.80.3.4 read_acquisition_results()	185
10.80.3.5 read_fpga_total_scale_factor()	185
10.80.3.6 reset_acquisition()	186
10.80.3.7 run_acquisition()	186
10.80.3.8 set_block_exp()	186
10.80.3.9 set_doppler_max()	186
10.80.3.10set_doppler_step()	186
10.80.3.11set_doppler_sweep()	187
10.80.3.12set_local_code()	187
10.80.3.13write_local_code()	187
10.81Fpga_dynamic_bit_selection Class Reference	187
10.81.1 Detailed Description	188
10.81.2 Constructor & Destructor Documentation	188
10.81.2.1 Fpga_dynamic_bit_selection()	188
10.81.2.2 ~Fpga_dynamic_bit_selection()	188
10.81.3 Member Function Documentation	188
10.81.3.1 bit_selection()	188
10.82Fpga_Multicorrelator_8sc Class Reference	189
10.82.1 Detailed Description	190
10.82.2 Constructor & Destructor Documentation	190
10.82.2.1 Fpga_Multicorrelator_8sc()	190
10.82.2.2 ~Fpga_Multicorrelator_8sc()	190
10.82.3 Member Function Documentation	190
10.82.3.1 Carrier_wipeoff_multicorrelator_resampler()	191
10.82.3.2 disable_secondary_codes()	191
10.82.3.3 enable_secondary_codes()	191
10.82.3.4 free()	191
10.82.3.5 initialize_secondary_code()	191
10.82.3.6 lock_channel()	192

10.82.3.7 read_sample_counter()	192
10.82.3.8 set_channel()	192
10.82.3.9 set_initial_sample()	192
10.82.3.10set_local_code_and_taps()	192
10.82.3.11set_output_vectors()	193
10.82.3.12set_secondary_code_lengths()	193
10.82.3.13unlock_channel()	193
10.82.3.14update_local_code()	193
10.82.3.15update_prn_code_length()	193
10.83Fpga_Switch Class Reference	194
10.83.1 Detailed Description	194
10.83.2 Constructor & Destructor Documentation	194
10.83.2.1 Fpga_Switch()	194
10.83.2.2 ~Fpga_Switch()	194
10.83.3 Member Function Documentation	194
10.83.3.1 set_switch_position()	195
10.84FreqXlatingFirFilter Class Reference	195
10.84.1 Detailed Description	195
10.84.2 Member Function Documentation	196
10.84.2.1 implementation()	196
10.85FrontEndCal Class Reference	196
10.85.1 Detailed Description	196
10.85.2 Member Function Documentation	196
10.85.2.1 estimate_doppler_from_eph()	197
10.85.2.2 get_ephemeris()	197
10.85.2.3 GPS_L1_front_end_model_E4000()	197
10.85.2.4 set_configuration()	197
10.86ftp_t Struct Reference	198
10.86.1 Detailed Description	198
10.87Galileo_Almanac Class Reference	198

10.87.1 Detailed Description	199
10.87.2 Constructor & Destructor Documentation	199
10.87.2.1 Galileo_Almanac()	199
10.87.3 Member Data Documentation	199
10.87.3.1 d_A_f0	200
10.87.3.2 d_A_f1	200
10.87.3.3 d_Delta_i	200
10.87.3.4 d_Delta_sqrt_A	200
10.87.3.5 d_e_eccentricity	200
10.87.3.6 d_M_0	201
10.87.3.7 d_OMEGA	201
10.87.3.8 d_OMEGA0	201
10.87.3.9 d_OMEGA_DOT	201
10.87.3.10 satellite_PRN	201
10.88 Galileo_Almanac_Helper Class Reference	202
10.88.1 Detailed Description	203
10.88.2 Constructor & Destructor Documentation	203
10.88.2.1 Galileo_Almanac_Helper()	203
10.89 Galileo_E1_Tcp_Connector_Tracking_cc Class Reference	203
10.89.1 Detailed Description	204
10.90 galileo_e5a_noncoherentIQ_acquisition_caf_cc Class Reference	204
10.90.1 Detailed Description	205
10.90.2 Constructor & Destructor Documentation	205
10.90.2.1 ~galileo_e5a_noncoherentIQ_acquisition_caf_cc()	206
10.90.3 Member Function Documentation	206
10.90.3.1 general_work()	206
10.90.3.2 init()	206
10.90.3.3 mag()	206
10.90.3.4 set_active()	206
10.90.3.5 set_channel()	207

10.90.3.6 <code>set_channel_fsm()</code>	207
10.90.3.7 <code>set_doppler_max()</code>	207
10.90.3.8 <code>set_doppler_step()</code>	208
10.90.3.9 <code>set_gnss_synchro()</code>	208
10.90.3.10 <code>set_local_code()</code>	208
10.90.3.11 <code>set_state()</code>	208
10.90.3.12 <code>set_threshold()</code>	209
10.91 Galileo_Ephemeris Class Reference	209
10.91.1 Detailed Description	211
10.91.2 Member Function Documentation	211
10.91.2.1 <code>Galileo_System_Time()</code>	212
10.91.2.2 <code>satellitePosition()</code>	212
10.91.2.3 <code>serialize()</code>	212
10.91.2.4 <code>sv_clock_drift()</code>	212
10.91.2.5 <code>sv_clock_relativistic_term()</code>	212
10.91.3 Member Data Documentation	213
10.91.3.1 <code>A_1</code>	213
10.91.3.2 <code>af0_4</code>	213
10.91.3.3 <code>af1_4</code>	213
10.91.3.4 <code>af2_4</code>	213
10.91.3.5 <code>BGD_E1E5a_5</code>	214
10.91.3.6 <code>BGD_E1E5b_5</code>	214
10.91.3.7 <code>C_ic_4</code>	214
10.91.3.8 <code>C_is_4</code>	214
10.91.3.9 <code>C_rc_3</code>	215
10.91.3.10 <code>C_rs_3</code>	215
10.91.3.11 <code>C_uc_3</code>	215
10.91.3.12 <code>C_us_3</code>	215
10.91.3.13 <code>d_satpos_X</code>	216
10.91.3.14 <code>d_satpos_Y</code>	216

10.91.3.15d_satpos_Z	216
10.91.3.16d_satvel_X	216
10.91.3.17d_satvel_Y	216
10.91.3.18d_satvel_Z	217
10.91.3.19delta_n_3	217
10.91.3.20E1B_DVS_5	217
10.91.3.21E1B_HS_5	217
10.91.3.22E5a_DVS	218
10.91.3.23E5a_HS	218
10.91.3.24E5b_DVS_5	218
10.91.3.25E5b_HS_5	218
10.91.3.26e_1	219
10.91.3.27Galileo_dtr	219
10.91.3.28_0_2	219
10.91.3.29_satellite_PRN	219
10.91.3.30Dot_2	220
10.91.3.31M0_1	220
10.91.3.32OMEGA_0_2	220
10.91.3.33omega_2	220
10.91.3.34OMEGA_dot_3	221
10.91.3.350c_4	221
10.91.3.360e_1	221
10.91.3.37TOW_5	221
10.91.3.38WN_5	222
10.92Galileo_Fnav_Message Class Reference	222
10.92.1 Detailed Description	223
10.93Galileo_Inav_Message Class Reference	223
10.93.1 Detailed Description	224
10.94Galileo_Iono Class Reference	224
10.94.1 Detailed Description	225

10.94.2 Constructor & Destructor Documentation	225
10.94.2.1 Galileo_Iono()	225
10.94.3 Member Function Documentation	225
10.94.3.1 serialize()	225
10.94.4 Member Data Documentation	225
10.94.4.1 ai0_5	226
10.94.4.2 ai1_5	226
10.94.4.3 ai2_5	226
10.94.4.4 Region1_flag_5	226
10.94.4.5 Region2_flag_5	227
10.94.4.6 Region3_flag_5	227
10.94.4.7 Region4_flag_5	227
10.94.4.8 Region5_flag_5	227
10.94.4.9 TOW_5	228
10.94.4.10WN_5	228
10.95galileo_pcps_8ms_acquisition_cc Class Reference	228
10.95.1 Detailed Description	229
10.95.2 Constructor & Destructor Documentation	229
10.95.2.1 ~galileo_pcps_8ms_acquisition_cc()	230
10.95.3 Member Function Documentation	230
10.95.3.1 general_work()	230
10.95.3.2 init()	230
10.95.3.3 mag()	230
10.95.3.4 set_active()	230
10.95.3.5 set_channel()	231
10.95.3.6 set_channel_fsm()	231
10.95.3.7 set_doppler_max()	231
10.95.3.8 set_doppler_step()	232
10.95.3.9 set_gnss_synchro()	232
10.95.3.10set_local_code()	232

10.95.3.11	<code>set_state()</code>	232
10.95.3.12	<code>set_threshold()</code>	234
10.96	<code>galileo_telemetry_decoder_gs</code> Class Reference	234
10.96.1	Detailed Description	235
10.96.2	Member Function Documentation	235
10.96.2.1	<code>general_work()</code>	235
10.96.2.2	<code>set_channel()</code>	235
10.96.2.3	<code>set_satellite()</code>	235
10.97	<code>Galileo_Utc_Model</code> Class Reference	236
10.97.1	Detailed Description	236
10.97.2	Constructor & Destructor Documentation	236
10.97.2.1	<code>Galileo_Utc_Model()</code>	237
10.97.3	Member Function Documentation	237
10.97.3.1	<code>GST_to_UTC_time()</code>	237
10.97.3.2	<code>serialize()</code>	237
10.97.4	Member Data Documentation	237
10.97.4.1	<code>t0t_6</code>	237
10.97.4.2	<code>WNot_6</code>	238
10.98	<code>GalileoE1BTelemetryDecoder</code> Class Reference	238
10.98.1	Detailed Description	238
10.98.2	Member Function Documentation	239
10.98.2.1	<code>implementation()</code>	239
10.99	<code>GalileoE1DIIPIVemlTracking</code> Class Reference	239
10.99.1	Detailed Description	240
10.99.2	Member Function Documentation	240
10.99.2.1	<code>implementation()</code>	240
10.99.2.2	<code>set_channel()</code>	240
10.99.2.3	<code>set_gnss_synchro()</code>	240
10.99.2.4	<code>stop_tracking()</code>	241
10.100	<code>GalileoE1DIIPIVemlTrackingFpga</code> Class Reference	241

10.100.1Detailed Description	242
10.100.2Constructor & Destructor Documentation	242
10.100.2.1GalileoE1DIIPIIVemlTrackingFpga()	242
10.100.2.2~GalileoE1DIIPIIVemlTrackingFpga()	242
10.100.3Member Function Documentation	242
10.100.3.1connect()	243
10.100.3.2disconnect()	243
10.100.3.3get_left_block()	243
10.100.3.4get_right_block()	243
10.100.3.5implementation()	243
10.100.3.6item_size()	244
10.100.3.7role()	244
10.100.3.8set_channel()	244
10.100.3.9set_gnss_synchro()	244
10.100.3.10start_tracking()	245
10.100.3.11stop_tracking()	245
10.10GalileoE1Pcps8msAmbiguousAcquisition Class Reference	245
10.101.1Detailed Description	246
10.101.2Member Function Documentation	246
10.101.2.1implementation()	247
10.101.2.2nit()	247
10.101.2.3mag()	247
10.101.2.4reset()	247
10.101.2.5set_channel()	247
10.101.2.6set_channel_fsm()	248
10.101.2.7set_doppler_max()	248
10.101.2.8set_doppler_step()	248
10.101.2.9set_gnss_synchro()	248
10.101.2.10set_local_code()	249
10.101.2.11set_threshold()	249

10.101.2.1	stop_acquisition()	249
10.102	GalileoE1PcpsAmbiguousAcquisition Class Reference	249
10.102.1	Detailed Description	250
10.102.2	Member Function Documentation	250
10.102.2.1	implementation()	251
10.102.2.2	init()	251
10.102.2.3	mag()	251
10.102.2.4	reset()	251
10.102.2.5	set_channel()	251
10.102.2.6	set_channel_fsm()	252
10.102.2.7	set_doppler_center()	252
10.102.2.8	set_doppler_max()	252
10.102.2.9	set_doppler_step()	252
10.102.2.10	set_gnss_synchro()	252
10.102.2.11	set_local_code()	253
10.102.2.12	set_resampler_latency()	253
10.102.2.13	set_state()	253
10.102.2.14	set_threshold()	253
10.102.2.15	stop_acquisition()	253
10.103	GalileoE1PcpsAmbiguousAcquisitionFpga Class Reference	254
10.103.1	Detailed Description	255
10.103.2	Constructor & Destructor Documentation	255
10.103.2.1	GalileoE1PcpsAmbiguousAcquisitionFpga()	255
10.103.2.2	~GalileoE1PcpsAmbiguousAcquisitionFpga()	255
10.103.3	Member Function Documentation	255
10.103.3.1	connect()	256
10.103.3.2	disconnect()	256
10.103.3.3	get_left_block()	256
10.103.3.4	get_right_block()	256
10.103.3.5	implementation()	256

10.103.3.6	init()	257
10.103.3.7	item_size()	257
10.103.3.8	mag()	257
10.103.3.9	reset()	257
10.103.3.10	role()	257
10.103.3.11	set_channel()	258
10.103.3.12	set_channel_fsm()	258
10.103.3.13	set_doppler_center()	258
10.103.3.14	set_doppler_max()	258
10.103.3.15	set_doppler_step()	258
10.103.3.16	set_gnss_synchro()	259
10.103.3.17	set_local_code()	259
10.103.3.18	set_resampler_latency()	259
10.103.3.19	set_state()	259
10.103.3.20	set_threshold()	259
10.103.3.21	stop_acquisition()	260
10.104	GalileoE1PcpsCccwsrAmbiguousAcquisition Class Reference	260
10.104.1	Detailed Description	261
10.104.2	Member Function Documentation	261
10.104.2.1	implementation()	261
10.104.2.2	init()	261
10.104.2.3	mag()	262
10.104.2.4	reset()	262
10.104.2.5	set_channel()	262
10.104.2.6	set_channel_fsm()	262
10.104.2.7	set_doppler_max()	263
10.104.2.8	set_doppler_step()	263
10.104.2.9	set_gnss_synchro()	263
10.104.2.10	set_state()	263
10.104.2.11	set_threshold()	263

10.104.2.1	stop_acquisition()	264
10.105	GalileoE1PcpsQuickSyncAmbiguousAcquisition Class Reference	264
10.105.1	Detailed Description	265
10.105.2	Member Function Documentation	265
10.105.2.1	implementation()	265
10.105.2.2	init()	265
10.105.2.3	mag()	266
10.105.2.4	reset()	266
10.105.2.5	set_channel()	266
10.105.2.6	set_channel_fsm()	266
10.105.2.7	set_doppler_max()	267
10.105.2.8	set_doppler_step()	267
10.105.2.9	set_gnss_synchro()	267
10.105.2.10	set_local_code()	267
10.105.2.11	set_state()	267
10.105.2.12	set_threshold()	268
10.105.2.13	stop_acquisition()	268
10.106	GalileoE1PcpsTongAmbiguousAcquisition Class Reference	268
10.106.1	Detailed Description	269
10.106.2	Member Function Documentation	269
10.106.2.1	implementation()	270
10.106.2.2	init()	270
10.106.2.3	mag()	270
10.106.2.4	reset()	270
10.106.2.5	set_channel()	270
10.106.2.6	set_channel_fsm()	271
10.106.2.7	set_doppler_max()	271
10.106.2.8	set_doppler_step()	271
10.106.2.9	set_gnss_synchro()	271
10.106.2.10	set_local_code()	272

10.106.2.1	set_state()	272
10.106.2.2	set_threshold()	272
10.106.2.3	stop_acquisition()	272
10.107	GalileoE1TcpConnectorTracking Class Reference	273
10.107.1	Detailed Description	273
10.107.2	Member Function Documentation	273
10.107.2.1	implementation()	274
10.107.2.2	set_channel()	274
10.107.2.3	set_gnss_synchro()	274
10.107.2.4	stop_tracking()	274
10.108	GalileoE5aDIIPITracking Class Reference	275
10.108.1	Detailed Description	275
10.108.2	Member Function Documentation	275
10.108.2.1	implementation()	276
10.108.2.2	set_channel()	276
10.108.2.3	set_gnss_synchro()	276
10.108.2.4	stop_tracking()	276
10.109	GalileoE5aDIIPITrackingFpga Class Reference	277
10.109.1	Detailed Description	277
10.109.2	Constructor & Destructor Documentation	278
10.109.2.1	GalileoE5aDIIPITrackingFpga()	278
10.109.2.2	~GalileoE5aDIIPITrackingFpga()	278
10.109.3	Member Function Documentation	278
10.109.3.1	connect()	278
10.109.3.2	disconnect()	278
10.109.3.3	get_left_block()	279
10.109.3.4	get_right_block()	279
10.109.3.5	implementation()	279
10.109.3.6	item_size()	279
10.109.3.7	role()	279

10.109.3.8	set_channel()	280
10.109.3.9	set_gnss_synchro()	280
10.109.3.10	start_tracking()	280
10.109.3.11	stop_tracking()	280
10.110	GalileoE5aNoncoherentIQAcquisitionCaf Class Reference	281
10.110.1	Detailed Description	282
10.110.2	Member Function Documentation	282
10.110.2.1	implementation()	282
10.110.2.2	init()	282
10.110.2.3	mag()	282
10.110.2.4	reset()	282
10.110.2.5	set_channel()	283
10.110.2.6	set_channel_fsm()	283
10.110.2.7	set_doppler_max()	283
10.110.2.8	set_doppler_step()	283
10.110.2.9	set_gnss_synchro()	284
10.110.2.10	set_local_code()	284
10.110.2.11	set_state()	284
10.110.2.12	set_threshold()	284
10.110.2.13	stop_acquisition()	285
10.111	GalileoE5aPcpsAcquisition Class Reference	285
10.111.1	Detailed Description	286
10.111.2	Member Function Documentation	286
10.111.2.1	init()	286
10.111.2.2	mag()	286
10.111.2.3	reset()	287
10.111.2.4	set_channel()	287
10.111.2.5	set_channel_fsm()	287
10.111.2.6	set_doppler_center()	287
10.111.2.7	set_doppler_max()	287

10.111.2.8	set_doppler_step()	288
10.111.2.9	set_gnss_synchro()	288
10.111.2.10	set_local_code()	288
10.111.2.11	set_resampler_latency()	288
10.111.2.12	set_state()	288
10.111.2.13	set_threshold()	289
10.111.2.14	stop_acquisition()	289
10.112	GalileoE5aPcpsAcquisitionFpga Class Reference	289
10.112.1	Detailed Description	291
10.112.2	Constructor & Destructor Documentation	291
10.112.2.1	GalileoE5aPcpsAcquisitionFpga()	291
10.112.2.2	~GalileoE5aPcpsAcquisitionFpga()	291
10.112.3	Member Function Documentation	291
10.112.3.1	connect()	291
10.112.3.2	disconnect()	292
10.112.3.3	get_left_block()	292
10.112.3.4	get_right_block()	292
10.112.3.5	implementation()	292
10.112.3.6	init()	292
10.112.3.7	item_size()	293
10.112.3.8	mag()	293
10.112.3.9	reset()	293
10.112.3.10	role()	293
10.112.3.11	set_channel()	293
10.112.3.12	set_channel_fsm()	294
10.112.3.13	set_doppler_center()	294
10.112.3.14	set_doppler_max()	294
10.112.3.15	set_doppler_step()	294
10.112.3.16	set_gnss_synchro()	294
10.112.3.17	set_local_code()	295

10.112.3.1	set_resampler_latency()	295
10.112.3.1	set_single_doppler_flag()	295
10.112.3.2	set_state()	295
10.112.3.2	set_threshold()	295
10.112.3.2	stop_acquisition()	296
10.113	GalileoE5aTelemetryDecoder Class Reference	296
10.113.1	Detailed Description	297
10.113.2	Member Function Documentation	297
10.113.2.1	implementation()	297
10.114	GalileoE5bDIPIITracking Class Reference	297
10.114.1	Detailed Description	298
10.114.2	Member Function Documentation	298
10.114.2.1	connect()	298
10.114.2.2	disconnect()	299
10.114.2.3	get_left_block()	299
10.114.2.4	get_right_block()	299
10.114.2.5	implementation()	299
10.114.2.6	set_channel()	299
10.114.2.7	set_gnss_synchro()	300
10.114.2.8	stop_tracking()	300
10.115	GalileoE5bPcpsAcquisition Class Reference	300
10.115.1	Detailed Description	301
10.115.2	Constructor & Destructor Documentation	301
10.115.2.1	GalileoE5bPcpsAcquisition()	302
10.115.2.2	~GalileoE5bPcpsAcquisition()	302
10.115.3	Member Function Documentation	302
10.115.3.1	connect()	302
10.115.3.2	disconnect()	302
10.115.3.3	get_left_block()	302
10.115.3.4	get_right_block()	303

10.115.3.5	implementation()	303
10.115.3.6	init()	303
10.115.3.7	item_size()	303
10.115.3.8	mag()	303
10.115.3.9	reset()	304
10.115.3.10	role()	304
10.115.3.11	set_channel()	304
10.115.3.12	set_channel_fsm()	304
10.115.3.13	set_doppler_center()	305
10.115.3.14	set_doppler_max()	305
10.115.3.15	set_doppler_step()	305
10.115.3.16	set_gnss_synchro()	305
10.115.3.17	set_local_code()	305
10.115.3.18	set_resampler_latency()	306
10.115.3.19	set_state()	306
10.115.3.20	set_threshold()	306
10.115.3.21	stop_acquisition()	306
10.116	GalileoE5bPcpsAcquisitionFpga Class Reference	307
10.116.1	Detailed Description	308
10.116.2	Constructor & Destructor Documentation	308
10.116.2.1	GalileoE5bPcpsAcquisitionFpga()	308
10.116.2.2	~GalileoE5bPcpsAcquisitionFpga()	308
10.116.3	Member Function Documentation	309
10.116.3.1	connect()	309
10.116.3.2	disconnect()	309
10.116.3.3	get_left_block()	309
10.116.3.4	get_right_block()	309
10.116.3.5	implementation()	310
10.116.3.6	init()	310
10.116.3.7	item_size()	310

10.116.3.8	mag()	310
10.116.3.9	reset()	310
10.116.3.10	role()	311
10.116.3.11	set_channel()	311
10.116.3.12	set_channel_fsm()	311
10.116.3.13	set_doppler_center()	311
10.116.3.14	set_doppler_max()	312
10.116.3.15	set_doppler_step()	312
10.116.3.16	set_gnss_synchro()	312
10.116.3.17	set_local_code()	312
10.116.3.18	set_resampler_latency()	312
10.116.3.19	set_single_doppler_flag()	313
10.116.3.20	set_state()	313
10.116.3.21	set_threshold()	313
10.116.3.22	stop_acquisition()	313
10.117	GalileoE5bTelemetryDecoder Class Reference	314
10.117.1	Detailed Description	314
10.117.2	Member Function Documentation	314
10.117.2.1	connect()	315
10.117.2.2	disconnect()	315
10.117.2.3	get_left_block()	315
10.117.2.4	get_right_block()	315
10.117.2.5	implementation()	315
10.118	GenSignalSource Class Reference	316
10.118.1	Detailed Description	316
10.118.2	Constructor & Destructor Documentation	316
10.118.2.1	GenSignalSource()	316
10.118.2.2	~GenSignalSource()	317
10.118.3	Member Function Documentation	317
10.118.3.1	implementation()	317

10.118	GeoJSON_Printer Class Reference	317
10.119	Detailed Description	317
10.120	Geph_t Struct Reference	318
10.120	Detailed Description	318
10.121	Glonass_Gnav_Almanac Class Reference	318
10.121	Detailed Description	319
10.121	Constructor & Destructor Documentation	320
10.121.2	Glonass_Gnav_Almanac()	320
10.121.3	Member Function Documentation	320
10.121.3	serialize()	320
10.121.4	Member Data Documentation	320
10.121.4	id_C_n	320
10.121.4	id_Delta_i_n_A	321
10.121.4	id_Delta_T_n_A	321
10.121.4	id_Delta_T_n_A_dot	321
10.121.4	id_epsilon_n_A	321
10.121.4	id_H_n_A	322
10.121.4	id_KP	322
10.121.4	id_I_n	322
10.121.4	id_lambda_n_A	322
10.121.4	id_M_n_A	323
10.121.4	id_n_A	323
10.121.4	id_omega_n_A	323
10.121.4	id_t_lambda_n_A	323
10.121.4	id_tau_n_A	324
10.121.4	id_satellite_freq_channel	324
10.121.4	id_satellite_PRN	324
10.121.4	id_satellite_slot_number	324
10.122	Glonass_Gnav_Ephemeris Class Reference	325
10.122	Detailed Description	327

10.122.2	Constructor & Destructor Documentation	327
10.122.2.1	Glonass_Gnav_Ephemeris()	327
10.122.3	Member Function Documentation	327
10.122.3.1	compute_GLONASS_time()	328
10.122.3.2	glot_to_gpst()	328
10.122.3.3	glot_to_utc()	328
10.122.3.4	serialize()	329
10.122.3.5	sv_clock_drift()	330
10.122.4	Member Data Documentation	330
10.122.4.1	d_AXn	330
10.122.4.2	d_AYn	331
10.122.4.3	d_AZn	331
10.122.4.4	d_B_n	331
10.122.4.5	d_Delta_tau_n	331
10.122.4.6	d_dtr	332
10.122.4.7	d_E_n	332
10.122.4.8	d_F_T	332
10.122.4.9	d_gamma_n	332
10.122.4.10	d_iode	333
10.122.4.11	d_l3rd_n	333
10.122.4.12	d_l5th_n	333
10.122.4.13	d_m	333
10.122.4.14	d_M	334
10.122.4.15	d_n	334
10.122.4.16	d_N_T	334
10.122.4.17	d_P	334
10.122.4.18	d_P_1	335
10.122.4.19	d_P_2	335
10.122.4.20	d_P_3	335
10.122.4.21	d_P_4	335

10.122.4.22_satClkDrift	336
10.122.4.23_t_b	336
10.122.4.24_t_k	336
10.122.4.25_tau_c	336
10.122.4.26_tau_n	336
10.122.4.27_tod	337
10.122.4.28_TOW	337
10.122.4.29_VXn	337
10.122.4.30_VYn	337
10.122.4.31_VZn	337
10.122.4.32_WN	338
10.122.4.33_Xn	338
10.122.4.34_Yn	338
10.122.4.35_yr	338
10.122.4.36_Zn	338
10.122.4.37_satellite_freq_channel	339
10.122.4.38_satellite_PRN	339
10.122.4.39_satellite_slot_number	339
10.123.1_Glonass_Gnav_Navigation_Message Class Reference	339
10.123.1.1 Detailed Description	340
10.123.2 Constructor & Destructor Documentation	340
10.123.2.1 Glonass_Gnav_Navigation_Message()	341
10.123.3 Member Function Documentation	341
10.123.3.1 CRC_test()	341
10.123.3.2 get_almanac()	341
10.123.3.3 get_ephemeris()	341
10.123.3.4 get_frame_number()	342
10.123.3.5 get_utc_model()	342
10.123.3.6 have_new_almanac()	342
10.123.3.7 have_new_ephemeris()	342

10.123.3.8	<code>have_new_utc_model()</code>	342
10.123.3.9	<code>string_decoder()</code>	342
10.124	<code>Glonass_Gnav_Utc_Model</code> Class Reference	343
10.124.1	Detailed Description	344
10.124.2	Constructor & Destructor Documentation	344
10.124.2.1	<code>Glonass_Gnav_Utc_Model()</code>	344
10.124.3	Member Function Documentation	344
10.124.3.1	<code>serialize()</code>	344
10.124.3.2	<code>utc_time()</code>	345
10.124.4	Member Data Documentation	345
10.124.4.1	<code>d_B1</code>	345
10.124.4.2	<code>d_B2</code>	345
10.124.4.3	<code>d_N_4</code>	345
10.124.4.4	<code>d_N_A</code>	346
10.124.4.5	<code>d_tau_c</code>	346
10.124.4.6	<code>d_tau_gps</code>	346
10.125	<code>Glonass_I1_ca_dll_pll_c_aid_tracking_cc</code> Class Reference	346
10.125.1	Detailed Description	347
10.126	<code>Glonass_I1_ca_dll_pll_c_aid_tracking_sc</code> Class Reference	347
10.126.1	Detailed Description	348
10.127	<code>Glonass_L1_Ca_Dll_Pll_Tracking_cc</code> Class Reference	348
10.127.1	Detailed Description	349
10.128	<code>Glonass_I1_ca_telemetry_decoder_gs</code> Class Reference	349
10.128.1	Detailed Description	350
10.128.2	Constructor & Destructor Documentation	350
10.128.2.1	<code>~glonass_I1_ca_telemetry_decoder_gs()</code>	350
10.128.3	Member Function Documentation	350
10.128.3.1	<code>general_work()</code>	350
10.128.3.2	<code>set_channel()</code>	350
10.128.3.3	<code>set_satellite()</code>	351

10.129	Glonass_L2_ca_dll_pll_c_aid_tracking_cc Class Reference	351
10.129.1	Detailed Description	351
10.130	Glonass_L2_ca_dll_pll_c_aid_tracking_sc Class Reference	352
10.130.1	Detailed Description	352
10.131	Glonass_L2_Ca_DLL_Pll_Tracking_cc Class Reference	352
10.131.1	Detailed Description	353
10.132	Glonass_L2_ca_telemetry_decoder_gs Class Reference	353
10.132.1	Detailed Description	354
10.132.2	Constructor & Destructor Documentation	354
10.132.2.1	~Glonass_L2_ca_telemetry_decoder_gs()	354
10.132.3	Member Function Documentation	354
10.132.3.1	general_work()	354
10.132.3.2	set_channel()	354
10.132.3.3	set_satellite()	355
10.133	GlonassL1CaDIIPICAidTracking Class Reference	355
10.133.1	Detailed Description	356
10.133.2	Member Function Documentation	356
10.133.2.1	implementation()	356
10.133.2.2	set_channel()	356
10.133.2.3	set_gnss_synchro()	356
10.133.2.4	stop_tracking()	357
10.134	GlonassL1CaDIIPIITracking Class Reference	357
10.134.1	Detailed Description	358
10.134.2	Member Function Documentation	358
10.134.2.1	implementation()	358
10.134.2.2	set_channel()	358
10.134.2.3	set_gnss_synchro()	358
10.134.2.4	stop_tracking()	359
10.135	GlonassL1CaPcpsAcquisition Class Reference	359
10.135.1	Detailed Description	360

10.135.2	Member Function Documentation	360
10.135.2.1	implementation()	360
10.135.2.2	init()	360
10.135.2.3	mag()	361
10.135.2.4	reset()	361
10.135.2.5	set_channel()	361
10.135.2.6	set_channel_fsm()	361
10.135.2.7	set_doppler_max()	362
10.135.2.8	set_doppler_step()	362
10.135.2.9	set_gnss_synchro()	362
10.135.2.10	set_local_code()	362
10.135.2.11	set_state()	362
10.135.2.12	set_threshold()	363
10.135.2.13	stop_acquisition()	363
10.136	GlonassL1CaTelemetryDecoder Class Reference	363
10.136.1	Detailed Description	364
10.136.2	Member Function Documentation	364
10.136.2.1	implementation()	364
10.137	GlonassL2CaDIPIICAidTracking Class Reference	364
10.137.1	Detailed Description	365
10.137.2	Member Function Documentation	365
10.137.2.1	implementation()	365
10.137.2.2	set_channel()	365
10.137.2.3	set_gnss_synchro()	366
10.137.2.4	stop_tracking()	366
10.138	GlonassL2CaDIPIITracking Class Reference	366
10.138.1	Detailed Description	367
10.138.2	Member Function Documentation	367
10.138.2.1	implementation()	367
10.138.2.2	set_channel()	367

10.138.2.3	set_gnss_synchro()	367
10.138.2.4	stop_tracking()	368
10.139	GlonassL2CaPcpsAcquisition Class Reference	368
10.139.1	Detailed Description	369
10.139.2	Member Function Documentation	369
10.139.2.1	implementation()	369
10.139.2.2	init()	369
10.139.2.3	mag()	370
10.139.2.4	reset()	370
10.139.2.5	set_channel()	370
10.139.2.6	set_channel_fsm()	370
10.139.2.7	set_doppler_max()	371
10.139.2.8	set_doppler_step()	371
10.139.2.9	set_gnss_synchro()	371
10.139.2.10	set_local_code()	371
10.139.2.11	set_state()	371
10.139.2.12	set_threshold()	372
10.139.2.13	stop_acquisition()	372
10.140	GlonassL2CaTelemetryDecoder Class Reference	372
10.140.1	Detailed Description	373
10.140.2	Member Function Documentation	373
10.140.2.1	implementation()	373
10.141	Gn3sSignalSource Class Reference	373
10.141.1	Detailed Description	374
10.141.2	Member Function Documentation	374
10.141.2.1	implementation()	374
10.142	GnMaxSignalSource Class Reference	374
10.142.1	Detailed Description	375
10.142.2	Member Function Documentation	375
10.142.2.1	implementation()	375

10.143.1	Gnss_circular_deque< T > Class Template Reference	375
10.143.1	Detailed Description	376
10.143.2	Constructor & Destructor Documentation	376
10.143.2.1	Gnss_circular_deque() [1/2]	376
10.143.2.2	Gnss_circular_deque() [2/2]	376
10.143.3	Member Function Documentation	376
10.143.3.1	at()	376
10.143.3.2	back()	377
10.143.3.3	clear()	377
10.143.3.4	front()	377
10.143.3.5	get()	377
10.143.3.6	pop_front()	378
10.143.3.7	push_back()	378
10.143.3.8	reset() [1/2]	378
10.143.3.9	reset() [2/2]	378
10.143.3.10	size()	379
10.144.1	Gnss_Satellite Class Reference	379
10.144.1	Detailed Description	380
10.144.2	Constructor & Destructor Documentation	380
10.144.2.1	Gnss_Satellite() [1/4]	380
10.144.2.2	Gnss_Satellite() [2/4]	380
10.144.2.3	~Gnss_Satellite()	380
10.144.2.4	Gnss_Satellite() [3/4]	381
10.144.2.5	Gnss_Satellite() [4/4]	381
10.144.3	Member Function Documentation	381
10.144.3.1	get_block()	381
10.144.3.2	get_PRN()	381
10.144.3.3	get_rf_link()	381
10.144.3.4	get_system()	382
10.144.3.5	get_system_short()	382

10.144.3.6operator=() [1/2]	382
10.144.3.7operator=() [2/2]	382
10.144.3.8update_PRN()	382
10.144.3.9what_block()	382
10.144.4Friends And Related Function Documentation	383
10.144.4.1operator<<	383
10.144.4.2operator==	383
10.145gnss_sdr_fpga_sample_counter Class Reference	383
10.145.1Detailed Description	384
10.146gnss_sdr_sample_counter Class Reference	384
10.146.1Detailed Description	384
10.147Gnss_Sdr_Supl_Client Class Reference	384
10.147.1Detailed Description	386
10.147.2Member Function Documentation	386
10.147.2.1load_cnav_ephemeris_xml()	387
10.147.2.2load_cnav_utc_xml()	387
10.147.2.3load_ephemeris_xml()	387
10.147.2.4load_gal_almanac_xml()	387
10.147.2.5load_gal_ephemeris_xml()	387
10.147.2.6load_gal_iono_xml()	387
10.147.2.7load_gal_utc_xml()	388
10.147.2.8load_glo_utc_xml()	388
10.147.2.9load_gnav_ephemeris_xml()	388
10.147.2.10load_gps_almanac_xml()	388
10.147.2.11load_iono_xml()	388
10.147.2.12load_ref_location_xml()	388
10.147.2.13load_ref_time_xml()	389
10.147.2.14load_utc_xml()	389
10.147.2.15save_cnav_ephemeris_map_xml()	389
10.147.2.16save_cnav_utc_xml()	389

10.147.2.15	save_ephemeris_map_xml()	389
10.147.2.16	save_gal_almanac_xml()	390
10.147.2.17	save_gal_ephemeris_map_xml()	390
10.147.2.18	save_gal_iono_xml()	390
10.147.2.19	save_gal_utc_xml()	390
10.147.2.20	save_glo_utc_xml()	390
10.147.2.21	save_gnav_ephemeris_map_xml()	391
10.147.2.22	save_gps_almanac_xml()	391
10.147.2.23	save_iono_xml()	391
10.147.2.24	save_ref_location_xml()	391
10.147.2.25	save_ref_time_xml()	391
10.147.2.26	save_utc_xml()	392
10.148	Gnss_sdr_time_counter Class Reference	392
10.148.1	Detailed Description	392
10.149	Gnss_Sdr_Valve Class Reference	393
10.149.1	Detailed Description	393
10.150	Gnss_Signal Class Reference	393
10.150.1	Detailed Description	394
10.150.2	Member Function Documentation	394
10.150.2.1	get_satellite()	394
10.150.2.2	get_signal_str()	394
10.150.3	Friends And Related Function Documentation	394
10.150.3.1	operator<<	395
10.150.3.2	operator==	395
10.151	Gnss_Synchro Class Reference	395
10.151.1	Detailed Description	397
10.151.2	Constructor & Destructor Documentation	397
10.151.2.1	Gnss_Synchro() [1/3]	397
10.151.2.2	~Gnss_Synchro()	397
10.151.2.3	Gnss_Synchro() [2/3]	397

10.151.2.4Gnss_Synchro() [3/3]	397
10.151.3Member Function Documentation	398
10.151.3.1operator=() [1/2]	398
10.151.3.2operator=() [2/2]	398
10.151.3.3Serialize()	398
10.151.4Member Data Documentation	399
10.151.4.1Acq_delay_samples	399
10.151.4.2Acq_doppler_hz	399
10.151.4.3Acq_doppler_step	399
10.151.4.4Acq_samplestamp_samples	399
10.151.4.5Carrier_Doppler_hz	400
10.151.4.6Carrier_phase_rads	400
10.151.4.7Channel_ID	400
10.151.4.8CN0_dB_hz	400
10.151.4.9Code_phase_samples	401
10.151.4.10correlation_length_ms	401
10.151.4.11flag_valid_acquisition	401
10.151.4.12flag_valid_pseudorange	401
10.151.4.13flag_valid_symbol_output	402
10.151.4.14flag_valid_word	402
10.151.4.15	402
10.151.4.16interp_TOW_ms	402
10.151.4.17PRN	403
10.151.4.18Prompt_I	403
10.151.4.19Prompt_Q	403
10.151.4.20pseudorange_m	403
10.151.4.21RX_time	404
10.151.4.22Signal	404
10.151.4.23System	404
10.151.4.24TOW_at_current_symbol_ms	404

10.151.4.2	Tracking_sample_counter	405
10.152	gnss_synchro_monitor Class Reference	405
10.152.1	Detailed Description	405
10.152.2	Constructor & Destructor Documentation	406
10.152.2.1	~gnss_synchro_monitor()	406
10.153	Gnss_Synchro_Udp_Sink Class Reference	406
10.153.1	Detailed Description	406
10.154	GNSSBlockFactory Class Reference	406
10.154.1	Detailed Description	407
10.154.2	Member Function Documentation	407
10.154.2.1	GetBlock()	407
10.155	GNSSBlockInterface Class Reference	408
10.155.1	Detailed Description	409
10.156	GNSSFlowgraph Class Reference	409
10.156.1	Detailed Description	410
10.156.2	Constructor & Destructor Documentation	410
10.156.2.1	GNSSFlowgraph()	410
10.156.2.2	~GNSSFlowgraph()	410
10.156.3	Member Function Documentation	410
10.156.3.1	acquisition_manager()	410
10.156.3.2	apply_action()	411
10.156.3.3	connect()	411
10.156.3.4	disconnect()	411
10.156.3.5	get_pvt()	411
10.156.3.6	priorize_satellites()	411
10.156.3.7	send_telemetry_msg()	412
10.156.3.8	set_configuration()	412
10.156.3.9	start()	412
10.156.3.10	stop()	412
10.156.3.11	wait()	412

10.157. G nuplot Class Reference	413
10.157.1. Detailed Description	414
10.157.2. Constructor & Destructor Documentation	414
10.157.2.1. Gnuplot()	414
10.157.3. Member Function Documentation	415
10.157.3.1. operator<<()	415
10.157.3.2. replot()	415
10.157.3.3. unset_title()	415
10.158. G nuplotException Class Reference	415
10.158.1. Detailed Description	416
10.159. G ps_Acq_Assist Class Reference	416
10.159.1. Detailed Description	417
10.159.2. Constructor & Destructor Documentation	417
10.159.2.1. Gps_Acq_Assist()	417
10.159.3. Member Data Documentation	417
10.159.3.1. Azimuth	417
10.159.3.2. Code_Phase	417
10.159.3.3. Code_Phase_int	417
10.159.3.4. Code_Phase_window	418
10.159.3.5. d_Doppler0	418
10.159.3.6. d_Doppler1	418
10.159.3.7. d_TOW	418
10.159.3.8. dopplerUncertainty	418
10.159.3.9. Elevation	419
10.159.3.10. GPS_Bit_Number	419
10.159.3.11. satellite_PRN	419
10.160. G ps_Almanac Class Reference	419
10.160.1. Detailed Description	420
10.160.2. Constructor & Destructor Documentation	420
10.160.2.1. Gps_Almanac()	420

10.160.3	Member Data Documentation	421
10.160.3.1d	A_f0	421
10.160.3.2d	A_f1	421
10.160.3.3d	Delta_i	421
10.160.3.4d	e_eccentricity	421
10.160.3.5d	M_0	421
10.160.3.6d	OMEGA	422
10.160.3.7d	OMEGA0	422
10.160.3.8d	OMEGA_DOT	422
10.160.3.9d	sqrt_A	422
10.160.3.1i0	AS_status	422
10.160.3.1i1	satellite_PRN	423
10.160.3.1i2	SV_health	423
10.160.3.1i3	Toa	423
10.160.3.1i4	WNa	423
10.16	Gps_CNAV_Ephemeris Class Reference	423
10.161.1	Detailed Description	426
10.161.2	Constructor & Destructor Documentation	426
10.161.2.1	Gps_CNAV_Ephemeris()	426
10.161.3	Member Function Documentation	426
10.161.3.1	satellitePosition()	426
10.161.3.2	Serialize()	427
10.161.3.3	sv_clock_drift()	428
10.161.3.4	sv_clock_relativistic_term()	428
10.161.4	Member Data Documentation	428
10.161.4.1b	alert_flag	428
10.161.4.2b	antispoofing_flag	429
10.161.4.3b	integrity_status_flag	429
10.161.4.4d	A_DOT	429
10.161.4.5d	A_f0	429

10.161.4.6d_A_f1	430
10.161.4.7d_A_f2	430
10.161.4.8d_Cic	430
10.161.4.9d_Cis	430
10.161.4.10_Crc	431
10.161.4.1d_Crs	431
10.161.4.12_Cuc	431
10.161.4.13_Cus	431
10.161.4.14_DELTA_A	432
10.161.4.15_DELTA_DOT_N	432
10.161.4.16_Delta_n	432
10.161.4.17_DELTA_OMEGA_DOT	432
10.161.4.18_dtr	432
10.161.4.19_e_eccentricity	433
10.161.4.20_i_0	433
10.161.4.21_IDOT	433
10.161.4.22_M_0	433
10.161.4.23_OMEGA	434
10.161.4.24_OMEGA0	434
10.161.4.25_satClkDrift	434
10.161.4.26_satpos_X	434
10.161.4.27_satpos_Y	434
10.161.4.28_satpos_Z	435
10.161.4.29_satvel_X	435
10.161.4.30_satvel_Y	435
10.161.4.31_satvel_Z	435
10.161.4.32_TGD	435
10.161.4.33_Toc	436
10.161.4.34_Toe1	436
10.161.4.35_Toe2	436

10.161.4.36_Top	436
10.161.4.37_TOW	437
10.161.4.38_URA0	437
10.161.4.39_URA1	437
10.161.4.40_URA2	437
10.161.4.41_GPS_week	437
10.161.4.42_signal_health	438
10.161.4.43_URA	438
10.162_Gps_CNAV_Iono Class Reference	438
10.162.1 Detailed Description	439
10.162.2 Constructor & Destructor Documentation	439
10.162.2.1 Gps_CNAV_Iono()	439
10.162.3 Member Function Documentation	439
10.162.3.1 serialize()	439
10.162.4 Member Data Documentation	439
10.162.4.1 d_alpha0	439
10.162.4.2 d_alpha1	440
10.162.4.3 d_alpha2	440
10.162.4.4 d_alpha3	440
10.162.4.5 d_beta0	440
10.162.4.6 d_beta1	441
10.162.4.7 d_beta2	441
10.162.4.8 d_beta3	441
10.162.4.9 valid	441
10.163_Gps_CNAV_Navigation_Message Class Reference	442
10.163.1 Detailed Description	442
10.163.2 Constructor & Destructor Documentation	442
10.163.2.1 Gps_CNAV_Navigation_Message()	442
10.163.3 Member Function Documentation	442
10.163.3.1 get_ephemeris()	443

10.163.3.2	get_iono()	443
10.163.3.3	get_utc_model()	443
10.163.3.4	have_new_ephemeris()	443
10.163.3.5	have_new_iono()	443
10.163.3.6	have_new_utc_model()	443
10.164	Gps_CNAV_Utc_Model Class Reference	444
10.164.1	Detailed Description	444
10.164.2	Constructor & Destructor Documentation	444
10.164.2.1	Gps_CNAV_Utc_Model()	445
10.164.3	Member Data Documentation	445
10.164.3.1	d_A0	445
10.164.3.2	d_A1	445
10.164.3.3	d_A2	445
10.164.3.4	d_DeltaT_LS	445
10.164.3.5	d_DeltaT_LSF	446
10.164.3.6	d_t_OT	446
10.164.3.7	d_DN	446
10.164.3.8	d_WN_LSF	446
10.164.3.9	d_WN_T	446
10.165	Gps_Ephemeris Class Reference	447
10.165.1	Detailed Description	449
10.165.2	Constructor & Destructor Documentation	449
10.165.2.1	Gps_Ephemeris()	449
10.165.3	Member Function Documentation	449
10.165.3.1	satellitePosition()	450
10.165.3.2	Serialize()	450
10.165.3.3	sv_clock_drift()	451
10.165.3.4	sv_clock_relativistic_term()	451
10.165.4	Member Data Documentation	452
10.165.4.1	b_alert_flag	452

10.165.4.2b_antispoofing_flag	452
10.165.4.3b_fit_interval_flag	452
10.165.4.4b_integrity_status_flag	452
10.165.4.5b_L2_P_data_flag	453
10.165.4.6d_A_f0	453
10.165.4.7d_A_f1	453
10.165.4.8d_A_f2	453
10.165.4.9d_Cic	454
10.165.4.10_Cis	454
10.165.4.1d_Crc	454
10.165.4.12_Crs	454
10.165.4.13_Cuc	455
10.165.4.14_Cus	455
10.165.4.15_Delta_n	455
10.165.4.16_dtr	455
10.165.4.17_e_eccentricity	456
10.165.4.18_i_0	456
10.165.4.19_IDOT	456
10.165.4.20_IODC	456
10.165.4.21_IODE_SF2	457
10.165.4.22_IODE_SF3	457
10.165.4.23_M_0	457
10.165.4.24_OMEGA	457
10.165.4.25_OMEGA0	458
10.165.4.26_OMEGA_DOT	458
10.165.4.27_satClkDrift	458
10.165.4.28_satpos_X	458
10.165.4.29_satpos_Y	458
10.165.4.30_satpos_Z	459
10.165.4.31_satvel_X	459

10.165.4.32_satvel_Y	459
10.165.4.33_satvel_Z	459
10.165.4.34_sqrt_A	459
10.165.4.35_TGD	460
10.165.4.36_Toc	460
10.165.4.37_Toe	460
10.165.4.38_TOW	460
10.165.4.39_AODO	461
10.165.4.40_code_on_L2	461
10.165.4.41_GPS_week	461
10.165.4.42_SV_accuracy	461
10.165.4.43_satelliteBlock	462
10.166.Gps_Iono Class Reference	462
10.166.1.Detailed Description	463
10.166.2.Constructor & Destructor Documentation	463
10.166.2.1.Gps_Iono()	463
10.166.3.Member Function Documentation	463
10.166.3.1.serialize()	463
10.166.4.Member Data Documentation	463
10.166.4.1d_alpha0	463
10.166.4.2d_alpha1	464
10.166.4.3d_alpha2	464
10.166.4.4d_alpha3	464
10.166.4.5d_beta0	464
10.166.4.6d_beta1	465
10.166.4.7d_beta2	465
10.166.4.8d_beta3	465
10.166.4.9.valid	465
10.167.Gps_L1_Ca_Dll_Pll_Tracking_GPU_cc Class Reference	466
10.167.1.Detailed Description	466

10.166	Gps_L1_Ca_Kf_Tracking_cc Class Reference	466
10.168	Detailed Description	467
10.166	Gps_L1_Ca_Tcp_Connector_Tracking_cc Class Reference	467
10.169	Detailed Description	468
10.170	Gps_L1_ca_telemetry_decoder_gs Class Reference	468
10.170	Detailed Description	469
10.170	Member Function Documentation	469
10.170.2	general_work()	469
10.170.2	set_channel()	469
10.170.2	set_satellite()	469
10.171	Gps_L2c_telemetry_decoder_gs Class Reference	470
10.171	Detailed Description	470
10.171	Member Function Documentation	470
10.171.2	general_work()	470
10.171.2	set_channel()	471
10.171.2	set_satellite()	471
10.172	Gps_L5_telemetry_decoder_gs Class Reference	471
10.172	Detailed Description	472
10.172	Member Function Documentation	472
10.172.2	set_channel()	472
10.172.2	set_satellite()	472
10.173	Gps_Navigation_Message Class Reference	472
10.173	Detailed Description	473
10.173	Constructor & Destructor Documentation	473
10.173.2	Gps_Navigation_Message()	473
10.173	Member Function Documentation	474
10.173.3	get_ephemeris()	474
10.173.3	get_flag_iono_valid()	474
10.173.3	get_flag_utc_model_valid()	474
10.173.3	get_GPS_week()	474

10.173.3.5	get_iono()	474
10.173.3.6	get_satellite_PRN()	475
10.173.3.7	get_TOW()	475
10.173.3.8	get_utc_model()	475
10.173.3.9	set_channel()	475
10.173.3.10	set_satellite_PRN()	475
10.173.3.11	subframe_decoder()	476
10.173.3.12	utc_time()	476
10.174	Gps_Utc_Model Class Reference	476
10.174.1	Detailed Description	477
10.174.2	Constructor & Destructor Documentation	477
10.174.2.1	Gps_Utc_Model()	477
10.174.3	Member Data Documentation	477
10.174.3.1	id_A0	477
10.174.3.2	id_A1	477
10.174.3.3	id_A2	477
10.174.3.4	id_DeltaT_LS	478
10.174.3.5	id_DeltaT_LSF	478
10.174.3.6	id_t_OT	478
10.174.3.7	id_DN	478
10.174.3.8	id_WN_LSF	478
10.174.3.9	id_WN_T	479
10.175	GpsL1CaDIIPIITracking Class Reference	479
10.175.1	Detailed Description	480
10.175.2	Member Function Documentation	480
10.175.2.1	implementation()	480
10.175.2.2	set_channel()	480
10.175.2.3	set_gnss_synchro()	480
10.175.2.4	stop_tracking()	481
10.176	GpsL1CaDIIPIITrackingFpga Class Reference	481

10.176.1	Detailed Description	482
10.176.2	Constructor & Destructor Documentation	482
10.176.2.1	GpsL1CaDllPllTrackingFpga()	482
10.176.2.2	~GpsL1CaDllPllTrackingFpga()	482
10.176.3	Member Function Documentation	482
10.176.3.1	connect()	483
10.176.3.2	disconnect()	483
10.176.3.3	get_left_block()	483
10.176.3.4	get_right_block()	483
10.176.3.5	implementation()	483
10.176.3.6	item_size()	484
10.176.3.7	role()	484
10.176.3.8	set_channel()	484
10.176.3.9	set_gnss_synchro()	484
10.176.3.10	start_tracking()	485
10.176.3.11	stop_tracking()	485
10.177	GpsL1CaDllPllTrackingGPU Class Reference	485
10.177.1	Detailed Description	486
10.177.2	Member Function Documentation	486
10.177.2.1	implementation()	486
10.177.2.2	set_channel()	486
10.177.2.3	set_gnss_synchro()	486
10.177.2.4	stop_tracking()	487
10.178	GpsL1CaKfTracking Class Reference	487
10.178.1	Detailed Description	488
10.178.2	Member Function Documentation	488
10.178.2.1	implementation()	488
10.178.2.2	set_channel()	488
10.178.2.3	set_gnss_synchro()	488
10.178.2.4	stop_tracking()	489

10.179	GpsL1CaPcpsAcquisition Class Reference	489
10.179.1	Detailed Description	490
10.179.2	Member Function Documentation	490
10.179.2.1	implementation()	490
10.179.2.2	nit()	490
10.179.2.3	mag()	491
10.179.2.4	reset()	491
10.179.2.5	set_channel()	491
10.179.2.6	set_channel_fsm()	491
10.179.2.7	set_doppler_center()	491
10.179.2.8	set_doppler_max()	492
10.179.2.9	set_doppler_step()	492
10.179.2.10	set_gnss_synchro()	492
10.179.2.11	set_local_code()	492
10.179.2.12	set_resampler_latency()	492
10.179.2.13	set_state()	493
10.179.2.14	set_threshold()	493
10.179.2.15	stop_acquisition()	493
10.180	GpsL1CaPcpsAcquisitionFineDoppler Class Reference	493
10.180.1	Detailed Description	494
10.180.2	Member Function Documentation	494
10.180.2.1	implementation()	495
10.180.2.2	nit()	495
10.180.2.3	mag()	495
10.180.2.4	reset()	495
10.180.2.5	set_channel()	495
10.180.2.6	set_channel_fsm()	496
10.180.2.7	set_doppler_max()	496
10.180.2.8	set_doppler_step()	496
10.180.2.9	set_gnss_synchro()	496

10.180.2.1	get_state()	497
10.180.2.1	get_threshold()	497
10.180.2.1	stop_acquisition()	497
10.180	GpsL1CaPcpsAcquisitionFpga Class Reference	497
10.181.1	Detailed Description	499
10.181.2	Constructor & Destructor Documentation	499
10.181.2.1	GpsL1CaPcpsAcquisitionFpga()	499
10.181.2.2	~GpsL1CaPcpsAcquisitionFpga()	499
10.181.3	Member Function Documentation	499
10.181.3.1	connect()	499
10.181.3.2	disconnect()	500
10.181.3.3	get_left_block()	500
10.181.3.4	get_right_block()	500
10.181.3.5	implementation()	500
10.181.3.6	init()	500
10.181.3.7	item_size()	501
10.181.3.8	mag()	501
10.181.3.9	reset()	501
10.181.3.10	role()	501
10.181.3.11	set_channel()	501
10.181.3.12	set_channel_fsm()	502
10.181.3.13	set_doppler_center()	502
10.181.3.14	set_doppler_max()	502
10.181.3.15	set_doppler_step()	502
10.181.3.16	set_gnss_synchro()	502
10.181.3.17	set_local_code()	503
10.181.3.18	set_resampler_latency()	503
10.181.3.19	set_state()	503
10.181.3.20	set_threshold()	503
10.181.3.21	stop_acquisition()	503

10.182	GpsL1CaPcpsAssistedAcquisition Class Reference	504
10.182.1	Detailed Description	505
10.182.2	Member Function Documentation	505
10.182.2.1	implementation()	505
10.182.2.2	init()	505
10.182.2.3	mag()	505
10.182.2.4	reset()	505
10.182.2.5	set_channel()	506
10.182.2.6	set_channel_fsm()	506
10.182.2.7	set_doppler_max()	506
10.182.2.8	set_doppler_step()	506
10.182.2.9	set_gnss_synchro()	507
10.182.2.10	set_threshold()	507
10.182.2.11	stop_acquisition()	507
10.183	GpsL1CaPcpsOpenCIAcquisition Class Reference	507
10.183.1	Detailed Description	508
10.183.2	Member Function Documentation	508
10.183.2.1	implementation()	509
10.183.2.2	init()	509
10.183.2.3	mag()	509
10.183.2.4	reset()	509
10.183.2.5	set_channel()	509
10.183.2.6	set_channel_fsm()	510
10.183.2.7	set_doppler_max()	510
10.183.2.8	set_doppler_step()	510
10.183.2.9	set_gnss_synchro()	510
10.183.2.10	set_local_code()	511
10.183.2.11	set_threshold()	511
10.183.2.12	stop_acquisition()	511
10.184	GpsL1CaPcpsQuickSyncAcquisition Class Reference	511

10.184.1	Detailed Description	512
10.184.2	Member Function Documentation	512
10.184.2.1	implementation()	513
10.184.2.2	nit()	513
10.184.2.3	mag()	513
10.184.2.4	reset()	513
10.184.2.5	set_channel()	513
10.184.2.6	set_channel_fsm()	514
10.184.2.7	set_doppler_max()	514
10.184.2.8	set_doppler_step()	514
10.184.2.9	set_gnss_synchro()	514
10.184.2.10	set_local_code()	515
10.184.2.11	set_state()	515
10.184.2.12	set_threshold()	515
10.184.2.13	stop_acquisition()	515
10.185	5psL1CaPcpsTongAcquisition Class Reference	516
10.185.1	Detailed Description	517
10.185.2	Member Function Documentation	517
10.185.2.1	implementation()	517
10.185.2.2	nit()	517
10.185.2.3	mag()	517
10.185.2.4	reset()	517
10.185.2.5	set_channel()	518
10.185.2.6	set_channel_fsm()	518
10.185.2.7	set_doppler_max()	518
10.185.2.8	set_doppler_step()	518
10.185.2.9	set_gnss_synchro()	519
10.185.2.10	set_local_code()	519
10.185.2.11	set_state()	519
10.185.2.12	set_threshold()	519

10.185.2.1	stop_acquisition()	519
10.186	GpsL1CaTcpConnectorTracking Class Reference	520
10.186.1	Detailed Description	520
10.186.2	Member Function Documentation	520
10.186.2.1	implementation()	521
10.186.2.2	set_channel()	521
10.186.2.3	set_gnss_synchro()	521
10.186.2.4	stop_tracking()	521
10.187	GpsL1CaTelemetryDecoder Class Reference	522
10.187.1	Detailed Description	522
10.187.2	Member Function Documentation	522
10.187.2.1	implementation()	522
10.188	GpsL2CTelemetryDecoder Class Reference	523
10.188.1	Detailed Description	523
10.188.2	Member Function Documentation	523
10.188.2.1	implementation()	523
10.189	GpsL2MDIIPITracking Class Reference	524
10.189.1	Detailed Description	524
10.189.2	Member Function Documentation	524
10.189.2.1	implementation()	525
10.189.2.2	set_channel()	525
10.189.2.3	set_gnss_synchro()	525
10.189.2.4	stop_tracking()	525
10.190	GpsL2MDIIPITrackingFpga Class Reference	526
10.190.1	Detailed Description	526
10.190.2	Member Function Documentation	526
10.190.2.1	implementation()	527
10.190.2.2	set_channel()	527
10.190.2.3	set_gnss_synchro()	527
10.190.2.4	stop_tracking()	527

10.190	GpsL2MPcpsAcquisition Class Reference	528
10.191	Detailed Description	529
10.191.2	Member Function Documentation	529
10.191.2.1	implementation()	529
10.191.2.2	nit()	529
10.191.2.3	mag()	529
10.191.2.4	reset()	530
10.191.2.5	set_channel()	530
10.191.2.6	set_channel_fsm()	530
10.191.2.7	set_doppler_center()	530
10.191.2.8	set_doppler_max()	530
10.191.2.9	set_doppler_step()	531
10.191.2.10	set_gnss_synchro()	531
10.191.2.11	set_local_code()	531
10.191.2.12	set_resampler_latency()	531
10.191.2.13	set_state()	531
10.191.2.14	set_threshold()	532
10.191.2.15	stop_acquisition()	532
10.192	GpsL2MPcpsAcquisitionFpga Class Reference	532
10.192	Detailed Description	533
10.192.2	Member Function Documentation	533
10.192.2.1	implementation()	534
10.192.2.2	nit()	534
10.192.2.3	mag()	534
10.192.2.4	reset()	534
10.192.2.5	set_channel()	534
10.192.2.6	set_channel_fsm()	535
10.192.2.7	set_doppler_max()	535
10.192.2.8	set_doppler_step()	535
10.192.2.9	set_gnss_synchro()	535

10.192.2.1	set_local_code()	536
10.192.2.1	set_state()	536
10.192.2.1	set_threshold()	536
10.192.2.1	stop_acquisition()	536
10.193	GpsL5DIIPIITracking Class Reference	537
10.193.1	Detailed Description	537
10.193.2	Member Function Documentation	537
10.193.2.1	implementation()	538
10.193.2.2	set_channel()	538
10.193.2.3	set_gnss_synchro()	538
10.193.2.4	stop_tracking()	538
10.194	GpsL5DIIPIITrackingFpga Class Reference	539
10.194.1	Detailed Description	539
10.194.2	Constructor & Destructor Documentation	540
10.194.2.1	GpsL5DIIPIITrackingFpga()	540
10.194.2.2	~GpsL5DIIPIITrackingFpga()	540
10.194.3	Member Function Documentation	540
10.194.3.1	connect()	540
10.194.3.2	disconnect()	540
10.194.3.3	get_left_block()	541
10.194.3.4	get_right_block()	541
10.194.3.5	implementation()	541
10.194.3.6	item_size()	541
10.194.3.7	role()	541
10.194.3.8	set_channel()	542
10.194.3.9	set_gnss_synchro()	542
10.194.3.10	start_tracking()	542
10.194.3.11	stop_tracking()	542
10.195	GpsL5IPcpsAcquisition Class Reference	543
10.195.1	Detailed Description	544

10.195.2	Member Function Documentation	544
10.195.2.1	implementation()	544
10.195.2.2	init()	544
10.195.2.3	mag()	544
10.195.2.4	reset()	545
10.195.2.5	set_channel()	545
10.195.2.6	set_channel_fsm()	545
10.195.2.7	set_doppler_center()	545
10.195.2.8	set_doppler_max()	545
10.195.2.9	set_doppler_step()	546
10.195.2.10	set_gnss_synchro()	546
10.195.2.11	set_local_code()	546
10.195.2.12	set_resampler_latency()	546
10.195.2.13	set_state()	546
10.195.2.14	set_threshold()	547
10.195.2.15	stop_acquisition()	547
10.196	GpsL5iPcpsAcquisitionFpga Class Reference	547
10.196.1	Detailed Description	549
10.196.2	Constructor & Destructor Documentation	549
10.196.2.1	GpsL5iPcpsAcquisitionFpga()	549
10.196.2.2	~GpsL5iPcpsAcquisitionFpga()	549
10.196.3	Member Function Documentation	549
10.196.3.1	connect()	549
10.196.3.2	disconnect()	550
10.196.3.3	get_left_block()	550
10.196.3.4	get_right_block()	550
10.196.3.5	implementation()	550
10.196.3.6	init()	550
10.196.3.7	item_size()	551
10.196.3.8	mag()	551

10.196.3.9	reset()	551
10.196.3.10	role()	551
10.196.3.11	set_channel()	551
10.196.3.12	set_channel_fsm()	552
10.196.3.13	set_doppler_center()	552
10.196.3.14	set_doppler_max()	552
10.196.3.15	set_doppler_step()	552
10.196.3.16	set_gnss_synchro()	552
10.196.3.17	set_local_code()	553
10.196.3.18	set_resampler_latency()	553
10.196.3.19	set_state()	553
10.196.3.20	set_threshold()	553
10.196.3.21	stop_acquisition()	553
10.197	GpsL5TelemetryDecoder Class Reference	554
10.197.1	Detailed Description	554
10.197.2	Member Function Documentation	554
10.197.2.1	implementation()	554
10.198	GPU_Complex Struct Reference	555
10.198.1	Detailed Description	555
10.199	GPU_Complex_Short Struct Reference	555
10.199.1	Detailed Description	555
10.200	Gpx_Printer Class Reference	556
10.200.1	Detailed Description	556
10.201	Gr_Complex_Ip_Packet_Source Class Reference	556
10.201.1	Detailed Description	557
10.202	time_t Struct Reference	557
10.202.1	Detailed Description	557
10.203	half_cyc_tag Struct Reference	557
10.203.1	Detailed Description	557
10.204	Hybrid_observables_gs Class Reference	558

10.204.1	Detailed Description	558
10.205	HybridObservables Class Reference	558
10.205.1	Detailed Description	559
10.205.2	Member Function Documentation	559
10.205.2.1	implementation()	559
10.205.2.2	item_size()	559
10.206	byteToCbyte Class Reference	560
10.206.1	Detailed Description	560
10.206.2	Member Function Documentation	560
10.206.2.1	implementation()	560
10.207	byteToComplex Class Reference	561
10.207.1	Detailed Description	561
10.207.2	Member Function Documentation	561
10.207.2.1	implementation()	561
10.208	byteToCshort Class Reference	562
10.208.1	Detailed Description	562
10.208.2	Member Function Documentation	562
10.208.2.1	implementation()	562
10.209	INIReader Class Reference	563
10.209.1	Detailed Description	563
10.209.2	Constructor & Destructor Documentation	563
10.209.2.1	INIReader()	563
10.209.3	Member Function Documentation	563
10.209.3.1	Get()	563
10.209.3.2	GetInteger()	564
10.209.3.3	ParseError()	564
10.210	MemoryConfiguration Class Reference	564
10.210.1	Detailed Description	565
10.211	Interleaved_byte_to_complex_byte Class Reference	565
10.211.1	Detailed Description	565

10.212	Interleaved_byte_to_complex_short Class Reference	566
10.212.1	Detailed Description	566
10.213	Interleaved_short_to_complex_short Class Reference	566
10.213.1	Detailed Description	567
10.214	ShortToComplex Class Reference	567
10.214.1	Detailed Description	568
10.214.2	Member Function Documentation	568
10.214.2.1	implementation()	568
10.215	ShortToCshort Class Reference	568
10.215.1	Detailed Description	569
10.215.2	Member Function Documentation	569
10.215.2.1	implementation()	569
10.216	Kernel_info_t Struct Reference	569
10.216.1	Detailed Description	569
10.217	Kml_Printer Class Reference	570
10.217.1	Detailed Description	570
10.218	Absat23_source Class Reference	570
10.218.1	Detailed Description	571
10.219	AbsatSignalSource Class Reference	571
10.219.1	Detailed Description	571
10.219.2	Member Function Documentation	571
10.219.2.1	implementation()	572
10.220	Tex_t Struct Reference	572
10.220.1	Detailed Description	572
10.221	Texeph_t Struct Reference	572
10.221.1	Detailed Description	573
10.222	Texion_t Struct Reference	573
10.222.1	Detailed Description	573
10.223	Txmsg_t Struct Reference	573
10.223.1	Detailed Description	573

10.224	MmseResamplerConditioner Class Reference	574
10.224.1	Detailed Description	574
10.225	ModelFunction Class Reference	574
10.225.1	Detailed Description	574
10.226	Monitor_Pvt Class Reference	575
10.226.1	Detailed Description	575
10.226.2	Member Function Documentation	576
10.226.2.1	Serialize()	576
10.227	Monitor_Pvt_Udp_Sink Class Reference	576
10.227.1	Detailed Description	576
10.228	hsm_h_t Struct Reference	576
10.228.1	Detailed Description	577
10.229	MultichannelFileSignalSource Class Reference	577
10.229.1	Detailed Description	577
10.229.2	Member Function Documentation	577
10.229.2.1	Implementation()	578
10.230	av_t Struct Reference	578
10.230.1	Detailed Description	579
10.231	Nmea_Printer Class Reference	579
10.231.1	Detailed Description	579
10.231.2	Constructor & Destructor Documentation	580
10.231.2.1	Nmea_Printer()	580
10.231.2.2	~Nmea_Printer()	580
10.231.3	Member Function Documentation	580
10.231.3.1	Print_Nmea_Line()	580
10.232	Notch Class Reference	580
10.232.1	Detailed Description	581
10.233	NotchFilter Class Reference	581
10.233.1	Detailed Description	581
10.233.2	Member Function Documentation	582

10.233.2.1implementation()	582
10.234.1NotchFilterLite Class Reference	582
10.234.1.1Detailed Description	582
10.234.2Member Function Documentation	583
10.234.2.1implementation()	583
10.235.1NotchLite Class Reference	583
10.235.1.1Detailed Description	584
10.236.1NsrFileSignalSource Class Reference	584
10.236.1.1Detailed Description	584
10.236.2Member Function Documentation	585
10.236.2.1implementation()	585
10.237.1trip_t Struct Reference	585
10.237.1.1Detailed Description	585
10.238.1Obs_Conf Class Reference	585
10.238.1.1Detailed Description	586
10.239.1Obs_t Struct Reference	586
10.239.1.1Detailed Description	586
10.240.1bsd_t Struct Reference	586
10.240.1.1Detailed Description	586
10.241.1Observables_Dump_Reader Class Reference	587
10.241.1.1Detailed Description	587
10.242.1ObservablesInterface Class Reference	587
10.242.1.1Detailed Description	588
10.243.1opt_t Struct Reference	588
10.243.1.1Detailed Description	588
10.244.1OsmosdrSignalSource Class Reference	588
10.244.1.1Detailed Description	589
10.244.2Member Function Documentation	589
10.244.2.1implementation()	589
10.245.1Pass_Through Class Reference	589

10.245.1	Detailed Description	590
10.245.2	Member Function Documentation	590
10.245.2.1	implementation()	590
10.246	clk_t Struct Reference	590
10.246.1	Detailed Description	591
10.247	pcps_acquisition Class Reference	591
10.247.1	Detailed Description	592
10.247.2	Member Function Documentation	592
10.247.2.1	general_work()	592
10.247.2.2	nit()	592
10.247.2.3	mag()	592
10.247.2.4	set_active()	592
10.247.2.5	set_channel()	593
10.247.2.6	set_channel_fsm()	593
10.247.2.7	set_doppler_center()	593
10.247.2.8	set_doppler_max()	594
10.247.2.9	set_doppler_step()	594
10.247.2.10	set_gnss_synchro()	594
10.247.2.11	set_local_code()	594
10.247.2.12	set_state()	595
10.247.2.13	set_threshold()	595
10.248	pcps_acquisition_fine_doppler_cc Class Reference	595
10.248.1	Detailed Description	596
10.248.2	Constructor & Destructor Documentation	596
10.248.2.1	~pcps_acquisition_fine_doppler_cc()	597
10.248.3	Member Function Documentation	597
10.248.3.1	general_work()	597
10.248.3.2	nit()	597
10.248.3.3	mag()	597
10.248.3.4	nextPowerOf2()	597

10.248.3.5	set_active()	598
10.248.3.6	set_channel()	598
10.248.3.7	set_channel_fsm()	598
10.248.3.8	set_doppler_max()	599
10.248.3.9	set_doppler_step()	600
10.248.3.10	set_gnss_synchro()	600
10.248.3.11	set_local_code()	600
10.248.3.12	set_state()	601
10.248.3.13	set_threshold()	601
10.249	pcps_acquisition_fpga Class Reference	601
10.249.1	Detailed Description	602
10.249.2	Constructor & Destructor Documentation	602
10.249.2.1	~pcps_acquisition_fpga()	602
10.249.3	Member Function Documentation	603
10.249.3.1	init()	603
10.249.3.2	mag()	603
10.249.3.3	reset_acquisition()	603
10.249.3.4	set_active()	603
10.249.3.5	set_channel()	603
10.249.3.6	set_channel_fsm()	605
10.249.3.7	set_doppler_center()	605
10.249.3.8	set_doppler_max()	605
10.249.3.9	set_doppler_step()	606
10.249.3.10	set_gnss_synchro()	606
10.249.3.11	set_local_code()	606
10.249.3.12	set_state()	606
10.249.3.13	set_threshold()	607
10.250	pcps_assisted_acquisition_cc Class Reference	607
10.250.1	Detailed Description	608
10.250.2	Constructor & Destructor Documentation	608

10.250.2.1~pcps_assisted_acquisition_cc()	608
10.250.3Member Function Documentation	608
10.250.3.1general_work()	609
10.250.3.2nit()	609
10.250.3.3mag()	609
10.250.3.4set_active()	609
10.250.3.5set_channel()	609
10.250.3.6set_channel_fsm()	610
10.250.3.7set_doppler_max()	610
10.250.3.8set_doppler_step()	610
10.250.3.9set_gnss_synchro()	611
10.250.3.10set_local_code()	611
10.250.3.11set_threshold()	611
10.251pcps_cccwsr_acquisition_cc Class Reference	612
10.251.1Detailed Description	613
10.251.2Constructor & Destructor Documentation	613
10.251.2.1~pcps_cccwsr_acquisition_cc()	613
10.251.3Member Function Documentation	613
10.251.3.1general_work()	613
10.251.3.2nit()	613
10.251.3.3mag()	613
10.251.3.4set_active()	613
10.251.3.5set_channel()	614
10.251.3.6set_channel_fsm()	614
10.251.3.7set_doppler_max()	614
10.251.3.8set_doppler_step()	615
10.251.3.9set_gnss_synchro()	615
10.251.3.10set_local_code()	615
10.251.3.11set_state()	615
10.251.3.12set_threshold()	616

10.252. pcps_openc1_acquisition_cc Class Reference	616
10.252.1. Detailed Description	617
10.252.2. Constructor & Destructor Documentation	617
10.252.2.1. ~pcps_openc1_acquisition_cc()	618
10.252.3. Member Function Documentation	618
10.252.3.1. general_work()	618
10.252.3.2. nit()	618
10.252.3.3. mag()	618
10.252.3.4. set_active()	618
10.252.3.5. set_channel()	619
10.252.3.6. set_channel_fsm()	619
10.252.3.7. set_doppler_max()	619
10.252.3.8. set_doppler_step()	620
10.252.3.9. set_gnss_synchro()	620
10.252.3.10. set_local_code()	620
10.252.3.11. set_state()	620
10.252.3.12. set_threshold()	622
10.253. pcps_quicksync_acquisition_cc Class Reference	622
10.253.1. Detailed Description	623
10.253.2. Constructor & Destructor Documentation	623
10.253.2.1. ~pcps_quicksync_acquisition_cc()	623
10.253.3. Member Function Documentation	624
10.253.3.1. general_work()	624
10.253.3.2. nit()	624
10.253.3.3. mag()	624
10.253.3.4. set_active()	624
10.253.3.5. set_channel()	625
10.253.3.6. set_channel_fsm()	625
10.253.3.7. set_doppler_max()	625
10.253.3.8. set_doppler_step()	625

10.253.3.9	set_gnss_synchro()	626
10.253.3.10	set_local_code()	626
10.253.3.11	set_state()	626
10.253.3.12	set_threshold()	627
10.254	pcps_tong_acquisition_cc Class Reference	627
10.254.1	Detailed Description	628
10.254.2	Constructor & Destructor Documentation	628
10.254.2.1	~pcps_tong_acquisition_cc()	628
10.254.3	Member Function Documentation	628
10.254.3.1	general_work()	629
10.254.3.2	nit()	629
10.254.3.3	mag()	629
10.254.3.4	set_active()	629
10.254.3.5	set_channel()	629
10.254.3.6	set_channel_fsm()	630
10.254.3.7	set_doppler_max()	630
10.254.3.8	set_doppler_step()	630
10.254.3.9	set_gnss_synchro()	631
10.254.3.10	set_local_code()	631
10.254.3.11	set_state()	631
10.254.3.12	set_threshold()	631
10.255	pcpsconf_fpga_t Struct Reference	632
10.255.1	Detailed Description	632
10.256	pcv_t Struct Reference	632
10.256.1	Detailed Description	633
10.257	pcvs_t Struct Reference	633
10.257.1	Detailed Description	633
10.258	peph_t Struct Reference	633
10.258.1	Detailed Description	633
10.259	PlutosdrSignalSource Class Reference	634

10.259.1 Detailed Description	634
10.259.2 Member Function Documentation	634
10.259.2.1 implementation()	634
10.260 pppcorr_t Struct Reference	635
10.260.1 Detailed Description	635
10.261 prcpt_t Struct Reference	635
10.261.1 Detailed Description	636
10.262 pulse_blanking_cc Class Reference	636
10.262.1 Detailed Description	637
10.263 PulseBlankingFilter Class Reference	637
10.263.1 Detailed Description	637
10.263.2 Member Function Documentation	638
10.263.2.1 implementation()	638
10.264 Pvt_Conf Class Reference	638
10.264.1 Detailed Description	639
10.265 Pvt_Solution Class Reference	639
10.265.1 Detailed Description	641
10.265.2 Member Function Documentation	641
10.265.2.1 get_avg_height()	641
10.265.2.2 get_avg_latitude()	641
10.265.2.3 get_avg_longitude()	641
10.265.2.4 get_clock_drift_ppm()	641
10.265.2.5 get_course_over_ground()	641
10.265.2.6 get_height()	642
10.265.2.7 get_latitude()	642
10.265.2.8 get_longitude()	642
10.265.2.9 get_num_valid_observations()	642
10.265.2.10 get_speed_over_ground()	642
10.265.2.11 get_time_offset_s()	642
10.265.2.12 set_averaging_depth()	643

10.265.2.1	set_clock_drift_ppm()	643
10.265.2.1	set_course_over_ground()	643
10.265.2.1	set_num_valid_observations()	643
10.265.2.1	set_pre_2009_file()	643
10.265.2.1	set_rx_pos()	643
10.265.2.1	set_rx_vel()	644
10.265.2.1	set_speed_over_ground()	644
10.265.2.2	set_time_offset_s()	644
10.266	EvtInterface Class Reference	644
10.266.1	Detailed Description	645
10.267	Raw_t Struct Reference	645
10.267.1	Detailed Description	646
10.268	RawArraySignalSource Class Reference	646
10.268.1	Detailed Description	647
10.268.2	Member Function Documentation	647
10.268.2.1	implementation()	647
10.269	Rinex_Printer Class Reference	647
10.269.1	Detailed Description	652
10.269.2	Constructor & Destructor Documentation	652
10.269.2.1	Rinex_Printer()	652
10.269.2.2	~Rinex_Printer()	652
10.269.3	Member Function Documentation	652
10.269.3.1	compute_BDS_time()	652
10.269.3.2	compute_Galileo_time()	653
10.269.3.3	compute_GPS_time() [1/2]	653
10.269.3.4	compute_GPS_time() [2/2]	653
10.269.3.5	compute_UTC_time() [1/2]	653
10.269.3.6	compute_UTC_time() [2/2]	653
10.269.3.7	get_leap_second()	654
10.269.3.8	log_rinex_nav() [1/10]	654

10.269.3.9og_rinex_nav() [2/10]	654
10.269.3.10g_rinex_nav() [3/10]	654
10.269.3.11g_rinex_nav() [4/10]	655
10.269.3.12g_rinex_nav() [5/10]	655
10.269.3.13g_rinex_nav() [6/10]	655
10.269.3.14g_rinex_nav() [7/10]	655
10.269.3.15g_rinex_nav() [8/10]	655
10.269.3.16g_rinex_nav() [9/10]	656
10.269.3.17g_rinex_nav() [10/10]	656
10.269.3.18g_rinex_obs() [1/12]	656
10.269.3.19g_rinex_obs() [2/12]	656
10.269.3.20g_rinex_obs() [3/12]	657
10.269.3.21g_rinex_obs() [4/12]	657
10.269.3.22g_rinex_obs() [5/12]	657
10.269.3.23g_rinex_obs() [6/12]	657
10.269.3.24g_rinex_obs() [7/12]	658
10.269.3.25g_rinex_obs() [8/12]	658
10.269.3.26g_rinex_obs() [9/12]	658
10.269.3.27g_rinex_obs() [10/12]	658
10.269.3.28g_rinex_obs() [11/12]	659
10.269.3.29g_rinex_obs() [12/12]	659
10.269.3.30nex_nav_header() [1/14]	659
10.269.3.31nex_nav_header() [2/14]	659
10.269.3.32nex_nav_header() [3/14]	660
10.269.3.33nex_nav_header() [4/14]	660
10.269.3.34nex_nav_header() [5/14]	660
10.269.3.35nex_nav_header() [6/14]	660
10.269.3.36nex_nav_header() [7/14]	661
10.269.3.37nex_nav_header() [8/14]	661
10.269.3.38nex_nav_header() [9/14]	661

10.269.3.39	inex_nav_header() [10/14]	661
10.269.3.40	inex_nav_header() [11/14]	662
10.269.3.41	inex_nav_header() [12/14]	662
10.269.3.42	inex_nav_header() [13/14]	662
10.269.3.43	inex_nav_header() [14/14]	662
10.269.3.44	inex_obs_header() [1/12]	663
10.269.3.45	inex_obs_header() [2/12]	663
10.269.3.46	inex_obs_header() [3/12]	663
10.269.3.47	inex_obs_header() [4/12]	663
10.269.3.48	inex_obs_header() [5/12]	664
10.269.3.49	inex_obs_header() [6/12]	664
10.269.3.50	inex_obs_header() [7/12]	664
10.269.3.51	inex_obs_header() [8/12]	664
10.269.3.52	inex_obs_header() [9/12]	665
10.269.3.53	inex_obs_header() [10/12]	665
10.269.3.54	inex_obs_header() [11/12]	665
10.269.3.55	inex_obs_header() [12/12]	665
10.269.3.56	inex_sbs_header()	666
10.269.3.57	update_date_time()	666
10.269.3.58	update_nav_header()	666
10.269.4	Member Data Documentation	666
10.269.4.1	navBdsFile	666
10.269.4.2	navFile	667
10.269.4.3	navGalFile	667
10.269.4.4	navGloFile	667
10.269.4.5	navMixFile	667
10.269.4.6	obsFile	667
10.269.4.7	sbsFile	668
10.270	Rtcm Class Reference	668
10.270.1	Detailed Description	671

10.270.2	Constructor & Destructor Documentation	671
10.270.2.1	Rtcm()	671
10.270.3	Member Function Documentation	671
10.270.3.1	bin_to_binary_data()	671
10.270.3.2	bin_to_double()	672
10.270.3.3	bin_to_hex()	672
10.270.3.4	bin_to_uint()	672
10.270.3.5	binary_data_to_bin()	672
10.270.3.6	check_CRC()	672
10.270.3.7	hex_to_bin()	672
10.270.3.8	hex_to_int()	673
10.270.3.9	hex_to_uint()	673
10.270.3.10	is_server_running()	673
10.270.3.11	lock_time() [1/4]	673
10.270.3.12	lock_time() [2/4]	673
10.270.3.13	lock_time() [3/4]	674
10.270.3.14	lock_time() [4/4]	674
10.270.3.15	print_MSM_1()	674
10.270.3.16	print_MSM_2()	675
10.270.3.17	print_MSM_3()	675
10.270.3.18	print_MSM_4()	675
10.270.3.19	print_MSM_5()	676
10.270.3.20	print_MSM_6()	676
10.270.3.21	print_MSM_7()	676
10.270.3.22	print_MT1001()	677
10.270.3.23	print_MT1002()	677
10.270.3.24	print_MT1003()	677
10.270.3.25	print_MT1004()	677
10.270.3.26	print_MT1005()	678
10.270.3.27	print_MT1005_test()	678

10.270.3.28	print_MT1006()	678
10.270.3.29	print_MT1008()	678
10.270.3.30	print_MT1009()	679
10.270.3.31	print_MT1010()	679
10.270.3.32	print_MT1011()	680
10.270.3.33	print_MT1012()	680
10.270.3.34	print_MT1019()	681
10.270.3.35	print_MT1020()	681
10.270.3.36	print_MT1029()	681
10.270.3.37	print_MT1045()	682
10.270.3.38	read_MT1005()	682
10.270.3.39	read_MT1019()	682
10.270.3.40	read_MT1020()	682
10.270.3.41	read_MT1045()	683
10.270.3.42	in_server()	683
10.270.3.43	send_message()	683
10.270.3.44	stop_server()	683
10.271	Rtcm_Printer Class Reference	684
10.271.1	Detailed Description	685
10.271.2	Constructor & Destructor Documentation	685
10.271.2.1	Rtcm_Printer()	685
10.271.2.2	~Rtcm_Printer()	685
10.271.3	Member Function Documentation	685
10.271.3.1	lock_time()	685
10.271.3.2	print_MT1005_test()	686
10.271.3.3	Print_Rtcm_MT1009()	686
10.271.3.4	Print_Rtcm_MT1010()	686
10.271.3.5	Print_Rtcm_MT1011()	687
10.271.3.6	Print_Rtcm_MT1012()	687
10.271.3.7	Print_Rtcm_MT1019()	688

10.271.3.8Print_Rtcm_MT1020()	688
10.271.3.9Print_Rtcm_MT1045()	689
10.272.Rtcm_t Struct Reference	689
10.272.1Detailed Description	690
10.273.Rtk_t Struct Reference	690
10.273.1Detailed Description	690
10.274.Rtklib_Pvt Class Reference	690
10.274.1Detailed Description	691
10.274.2Member Function Documentation	691
10.274.2.1implementation()	691
10.274.2.2item_size()	692
10.275.Rtklib_pvt_gs Class Reference	692
10.275.1Detailed Description	693
10.275.2Constructor & Destructor Documentation	693
10.275.2.1~rtklib_pvt_gs()	693
10.275.3Member Function Documentation	693
10.275.3.1clear_ephemeris()	693
10.275.3.2get_beidou_dnav_almanac_map()	693
10.275.3.3get_beidou_dnav_ephemeris_map()	693
10.275.3.4get_galileo_almanac_map()	694
10.275.3.5get_galileo_ephemeris_map()	694
10.275.3.6get_gps_almanac_map()	694
10.275.3.7get_gps_ephemeris_map()	694
10.275.3.8get_latest_PVT()	694
10.275.3.9work()	695
10.276.Rtklib_Solver Class Reference	695
10.276.1Detailed Description	696
10.276.2Member Data Documentation	696
10.276.2.1beidou_dnav_ephemeris_map	696
10.276.2.2galileo_ephemeris_map	696

10.276.2.3	glonass_gnav_almanac	697
10.276.2.4	glonass_gnav_ephemeris_map	697
10.276.2.5	glonass_gnav_utc_model	697
10.276.2.6	gps_cnav_ephemeris_map	697
10.276.2.7	gps_ephemeris_map	697
10.277	Rtklib_Solver_Dump_Reader Class Reference	698
10.277.1	Detailed Description	698
10.278	Rtksvr_t Struct Reference	698
10.278.1	Detailed Description	699
10.279	Rtl_Tcp_Dongle_Info Class Reference	699
10.279.1	Detailed Description	700
10.280	Rtl_tcp_signal_source_c Class Reference	700
10.280.1	Detailed Description	701
10.281	RtlTcpSignalSource Class Reference	701
10.281.1	Detailed Description	701
10.281.2	Member Function Documentation	701
10.281.2.1	implementation()	702
10.282	Rtkbas_Ephemeris Class Reference	702
10.282.1	Detailed Description	703
10.282.2	Member Data Documentation	703
10.282.2.1	b_sv_do_not_use	703
10.282.2.2	d_acc	703
10.282.2.3	d_af0	703
10.282.2.4	d_af1	703
10.282.2.5	d_pos	704
10.282.2.6	d_tof	704
10.282.2.7	d_vel	704
10.282.2.8	prn	704
10.282.2.9	sv_ura	704
10.282.2.10	t0	705

10.283	bas_l1_telemetry_decoder_gs Class Reference	705
10.283.1	Detailed Description	705
10.283.2	Member Function Documentation	706
10.283.2.1	general_work()	706
10.283.2.2	set_channel()	706
10.283.2.3	set_satellite()	706
10.284	basL1TelemetryDecoder Class Reference	706
10.284.1	Detailed Description	707
10.284.2	Member Function Documentation	707
10.284.2.1	implementation()	707
10.285	bs_t Struct Reference	707
10.285.1	Detailed Description	708
10.286	bsfcrr_t Struct Reference	708
10.286.1	Detailed Description	708
10.287	bsigp_t Struct Reference	708
10.287.1	Detailed Description	708
10.288	bsigpband_t Struct Reference	709
10.288.1	Detailed Description	709
10.289	bsion_t Struct Reference	709
10.289.1	Detailed Description	709
10.290	bslcorr_t Struct Reference	709
10.290.1	Detailed Description	710
10.291	bsmsg_t Struct Reference	710
10.291.1	Detailed Description	710
10.292	bssat_t Struct Reference	710
10.292.1	Detailed Description	710
10.293	bssatp_t Struct Reference	711
10.293.1	Detailed Description	711
10.294	eph_t Struct Reference	711
10.294.1	Detailed Description	711

10.295	Serdes_Gnss_Synchro Class Reference	711
10.295.1	Detailed Description	712
10.295.2	Constructor & Destructor Documentation	712
10.295.2.1	Serdes_Gnss_Synchro() [1/2]	712
10.295.2.2	Serdes_Gnss_Synchro() [2/2]	712
10.295.3	Member Function Documentation	712
10.295.3.1	createProtobuffer()	712
10.295.3.2	operator=() [1/2]	713
10.295.3.3	operator=() [2/2]	713
10.295.3.4	readProtobuffer()	713
10.296	Serdes_Monitor_Pvt Class Reference	714
10.296.1	Detailed Description	714
10.296.2	Constructor & Destructor Documentation	714
10.296.2.1	Serdes_Monitor_Pvt() [1/2]	714
10.296.2.2	Serdes_Monitor_Pvt() [2/2]	714
10.296.3	Member Function Documentation	715
10.296.3.1	createProtobuffer()	715
10.296.3.2	operator=() [1/2]	716
10.296.3.3	operator=() [2/2]	716
10.296.3.4	readProtobuffer()	716
10.297	Serial_t Struct Reference	716
10.297.1	Detailed Description	717
10.298	short_x2_to_cshort Class Reference	717
10.298.1	Detailed Description	717
10.299	signal_generator_c Class Reference	718
10.299.1	Detailed Description	718
10.299.2	Friends And Related Function Documentation	718
10.299.2.1	signal_make_generator_c	719
10.300	SignalConditioner Class Reference	719
10.300.1	Detailed Description	720

10.300.2	Constructor & Destructor Documentation	720
10.300.2.1	SignalConditioner()	720
10.300.2.2	~SignalConditioner()	720
10.300.3	Member Function Documentation	720
10.300.3.1	implementation()	720
10.301	SignalGenerator Class Reference	721
10.301.1	Detailed Description	721
10.301.2	Member Function Documentation	721
10.301.2.1	implementation()	721
10.302	nrmask_t Struct Reference	722
10.302.1	Detailed Description	722
10.303	sol_t Struct Reference	722
10.303.1	Detailed Description	722
10.304	solbuf_t Struct Reference	722
10.304.1	Detailed Description	723
10.305	soloapt_t Struct Reference	723
10.305.1	Detailed Description	723
10.306	solstat_t Struct Reference	723
10.306.1	Detailed Description	724
10.307	solstatbuf_t Struct Reference	724
10.307.1	Detailed Description	724
10.308	Spirit_Motion_Csv_Dump_Reader Class Reference	724
10.308.1	Detailed Description	725
10.309	SpirFileSignalSource Class Reference	726
10.309.1	Detailed Description	726
10.309.2	Member Function Documentation	726
10.309.2.1	implementation()	726
10.310	SpirGSS6450FileSignalSource Class Reference	727
10.310.1	Detailed Description	727
10.311	ssat_t Struct Reference	728

10.311. Detailed Description	728
10.312sr_t Struct Reference	728
10.312. Detailed Description	729
10.313ta_t Struct Reference	729
10.313. Detailed Description	729
10.314tec_t Struct Reference	729
10.314. Detailed Description	730
10.315stream_cfg Struct Reference	730
10.315. Detailed Description	730
10.316stream_t Struct Reference	730
10.316. Detailed Description	731
10.317StringConverter Class Reference	731
10.317. Detailed Description	731
10.318cp_Communication Class Reference	731
10.318. Detailed Description	732
10.319cp_Packet_Data Class Reference	732
10.319. Detailed Description	732
10.320cp_t Struct Reference	733
10.320. Detailed Description	733
10.321cpcli_t Struct Reference	733
10.321. Detailed Description	733
10.322TcpCmdInterface Class Reference	733
10.322. Detailed Description	734
10.322. Member Function Documentation	734
10.322.2.1get_LLH()	734
10.322.2.2get_utc_time()	734
10.323cpsvr_t Struct Reference	734
10.323. Detailed Description	734
10.324ec_t Struct Reference	735
10.324. Detailed Description	735

10.325. TelemetryDecoderInterface Class Reference	735
10.325.1. Detailed Description	736
10.326. Te_t Struct Reference	736
10.326.1. Detailed Description	736
10.327. Te_d_t Struct Reference	736
10.327.1. Detailed Description	737
10.328. Tlm_Dump_Reader Class Reference	737
10.328.1. Detailed Description	737
10.329. Tracking_2nd_DLL_filter Class Reference	737
10.329.1. Detailed Description	738
10.329.2. Member Function Documentation	738
10.329.2.1. get_code_nco()	738
10.329.2.2. initialize()	738
10.329.2.3. set_DLL_BW()	738
10.329.2.4. set_pdi()	738
10.330. Tracking_2nd_PLL_filter Class Reference	739
10.330.1. Detailed Description	739
10.330.2. Member Function Documentation	739
10.330.2.1. set_pdi()	739
10.330.2.2. set_PLL_BW()	739
10.331. Tracking_Dump_Reader Class Reference	740
10.331.1. Detailed Description	740
10.332. Tracking_FLL_PLL_filter Class Reference	740
10.332.1. Detailed Description	741
10.333. Tracking_loop_filter Class Reference	741
10.333.1. Detailed Description	741
10.333.2. Constructor & Destructor Documentation	742
10.333.2.1. Tracking_loop_filter()	742
10.333.3. Member Function Documentation	742
10.333.3.1. operator=()	742

10.334	Tracking_True_Obs_Reader Class Reference	742
10.334.1	Detailed Description	743
10.335	TrackingInterface Class Reference	743
10.335.1	Detailed Description	744
10.336	top_t Struct Reference	744
10.336.1	Detailed Description	744
10.337	True_Observables_Reader Class Reference	744
10.337.1	Detailed Description	745
10.338	TwoBitCpxFileSignalSource Class Reference	745
10.338.1	Detailed Description	745
10.338.2	Member Function Documentation	745
10.338.2.1	implementation()	746
10.339	TwoBitPackedFileSignalSource Class Reference	746
10.339.1	Detailed Description	747
10.339.2	Member Function Documentation	747
10.339.2.1	implementation()	747
10.340	UhdSignalSource Class Reference	747
10.340.1	Detailed Description	748
10.340.2	Member Function Documentation	748
10.340.2.1	implementation()	748
10.341	Unpack_2bit_samples Class Reference	748
10.341.1	Detailed Description	749
10.342	Unpack_byte_2bit_cpx_samples Class Reference	749
10.342.1	Detailed Description	749
10.343	Unpack_byte_2bit_samples Class Reference	750
10.343.1	Detailed Description	750
10.344	Unpack_byte_4bit_samples Class Reference	750
10.344.1	Detailed Description	751
10.345	Unpack_intspir_1bit_samples Class Reference	751
10.345.1	Detailed Description	751

10.346	unpack_spir_gss6450_samples Class Reference	752
10.346.1	Detailed Description	752
10.347	UnscentedFilter Class Reference	752
10.347.1	Detailed Description	753
10.348	rl_t Struct Reference	753
10.348.1	Detailed Description	753
10.349	27_decision_t Struct Reference	753
10.349.1	Detailed Description	753
10.350	27_poly_t Struct Reference	753
10.350.1	Detailed Description	754
10.351	27_t Struct Reference	754
10.351.1	Detailed Description	754
10.352	iterbi_Decoder Class Reference	754
10.352.1	Detailed Description	755
10.352.2	Member Function Documentation	755
10.352.2.1	decode_block()	755
11	File Documentation	757
11.1	acq_conf.h File Reference	757
11.1.1	Detailed Description	757
11.2	acquisition_dump_reader.h File Reference	757
11.2.1	Detailed Description	758
11.3	acquisition_interface.h File Reference	758
11.3.1	Detailed Description	758
11.4	acquisition_msg_rx.h File Reference	759
11.4.1	Detailed Description	759
11.5	ad9361_fpga_signal_source.h File Reference	759
11.5.1	Detailed Description	760
11.6	ad9361_manager.h File Reference	760
11.6.1	Detailed Description	761
11.7	agnss_ref_location.h File Reference	762

11.7.1 Detailed Description	762
11.8 agnss_ref_time.h File Reference	762
11.8.1 Detailed Description	762
11.9 array_signal_conditioner.h File Reference	763
11.9.1 Detailed Description	763
11.10 bayesian_estimation.h File Reference	763
11.10.1 Detailed Description	764
11.11 beamformer.h File Reference	764
11.11.1 Detailed Description	765
11.12 beamformer_filter.h File Reference	765
11.12.1 Detailed Description	765
11.13 Beidou_B1I.h File Reference	766
11.13.1 Detailed Description	766
11.13.2 Variable Documentation	767
11.13.2.1 BEIDOU_B1I_CODE_LENGTH_CHIPS	767
11.13.2.2 BEIDOU_B1I_CODE_PERIOD_MS	767
11.13.2.3 BEIDOU_B1I_CODE_PERIOD_S	767
11.13.2.4 BEIDOU_B1I_CODE_RATE_CPS	767
11.13.2.5 BEIDOU_B1I_FREQ_HZ	768
11.14 beidou_b1i_dll_pll_tracking.h File Reference	768
11.14.1 Detailed Description	768
11.15 beidou_b1i_pcps_acquisition.h File Reference	768
11.15.1 Detailed Description	769
11.16 beidou_b1i_signal_processing.h File Reference	769
11.16.1 Detailed Description	770
11.16.2 Function Documentation	770
11.16.2.1 beidou_b1i_code_gen_complex()	770
11.16.2.2 beidou_b1i_code_gen_complex_sampled() [1/2]	770
11.16.2.3 beidou_b1i_code_gen_complex_sampled() [2/2]	771
11.16.2.4 beidou_b1i_code_gen_float()	771

11.16.2.5 beidou_b1i_code_gen_int()	771
11.17beidou_b1i_telemetry_decoder.h File Reference	771
11.17.1 Detailed Description	772
11.18beidou_b1i_telemetry_decoder_gs.h File Reference	772
11.18.1 Detailed Description	773
11.19Beidou_B3I.h File Reference	773
11.19.1 Detailed Description	774
11.19.2 Variable Documentation	774
11.19.2.1 BEIDOU_B3I_CODE_LENGTH_CHIPS	774
11.19.2.2 BEIDOU_B3I_CODE_PERIOD_MS	774
11.19.2.3 BEIDOU_B3I_CODE_PERIOD_S	774
11.19.2.4 BEIDOU_B3I_CODE_RATE_CPS	775
11.19.2.5 BEIDOU_B3I_FREQ_HZ	775
11.19.2.6 BEIDOU_B3I_TELEMETRY_RATE_BITS_SECOND	775
11.20beidou_b3i_dll_pll_tracking.h File Reference	775
11.20.1 Detailed Description	776
11.21beidou_b3i_pcps_acquisition.h File Reference	776
11.21.1 Detailed Description	776
11.22beidou_b3i_signal_processing.h File Reference	777
11.22.1 Detailed Description	777
11.22.2 Function Documentation	777
11.22.2.1 beidou_b3i_code_gen_complex()	778
11.22.2.2 beidou_b3i_code_gen_complex_sampled() [1/2]	778
11.22.2.3 beidou_b3i_code_gen_complex_sampled() [2/2]	778
11.22.2.4 beidou_b3i_code_gen_float()	778
11.22.2.5 beidou_b3i_code_gen_int()	779
11.23beidou_b3i_telemetry_decoder.h File Reference	779
11.23.1 Detailed Description	779
11.24beidou_b3i_telemetry_decoder_gs.h File Reference	780
11.24.1 Detailed Description	780

11.25Beidou_DNAV.h File Reference	781
11.25.1 Detailed Description	785
11.25.2 Variable Documentation	785
11.25.2.1 BEIDOU_DNAV_SUBFRAME_DATA_BITS	785
11.26beidou_dnav_almanac.h File Reference	785
11.26.1 Detailed Description	786
11.27beidou_dnav_ephemeris.h File Reference	786
11.27.1 Detailed Description	786
11.28beidou_dnav_iono.h File Reference	787
11.28.1 Detailed Description	787
11.29beidou_dnav_navigation_message.h File Reference	787
11.29.1 Detailed Description	788
11.30beidou_dnav_utc_model.h File Reference	788
11.30.1 Detailed Description	788
11.31bits.h File Reference	789
11.31.1 Detailed Description	789
11.32byte_to_short.h File Reference	789
11.32.1 Detailed Description	790
11.33byte_x2_to_complex_byte.h File Reference	790
11.33.1 Detailed Description	791
11.34channel.h File Reference	791
11.34.1 Detailed Description	791
11.35channel_event.h File Reference	792
11.35.1 Detailed Description	792
11.36channel_fsm.h File Reference	792
11.36.1 Detailed Description	793
11.37channel_interface.h File Reference	793
11.37.1 Detailed Description	793
11.38channel_msg_receiver_cc.h File Reference	794
11.38.1 Detailed Description	794

11.39channel_status_msg_receiver.h File Reference	794
11.39.1 Detailed Description	795
11.40clFFT.h File Reference	795
11.40.1 Detailed Description	796
11.41cnav_msg.h File Reference	796
11.41.1 Detailed Description	797
11.42command_event.h File Reference	797
11.42.1 Detailed Description	798
11.43complex_byte_to_float_x2.h File Reference	798
11.43.1 Detailed Description	799
11.44complex_float_to_complex_byte.h File Reference	799
11.44.1 Detailed Description	799
11.45concurrent_map.h File Reference	800
11.45.1 Detailed Description	800
11.46concurrent_queue.h File Reference	800
11.46.1 Detailed Description	801
11.47configuration_interface.h File Reference	801
11.47.1 Detailed Description	801
11.48conjugate_cc.h File Reference	802
11.48.1 Detailed Description	802
11.49conjugate_ic.h File Reference	802
11.49.1 Detailed Description	803
11.50conjugate_sc.h File Reference	803
11.50.1 Detailed Description	804
11.51control_thread.h File Reference	804
11.51.1 Detailed Description	805
11.52convolutional.h File Reference	805
11.52.1 Detailed Description	806
11.52.2 Function Documentation	806
11.52.2.1 Gamma()	806

11.52.2.2 nsc_enc_bit()	807
11.52.2.3 nsc_transit()	807
11.52.2.4 parity_counter()	808
11.52.2.5 Viterbi()	808
11.53cpu_multicorrelator.h File Reference	809
11.53.1 Detailed Description	809
11.54cpu_multicorrelator_16sc.h File Reference	809
11.54.1 Detailed Description	810
11.55cpu_multicorrelator_real_codes.h File Reference	810
11.55.1 Detailed Description	810
11.56cshort_to_float_x2.h File Reference	811
11.56.1 Detailed Description	811
11.57cuda_multicorrelator.h File Reference	811
11.57.1 Detailed Description	812
11.58custom_udp_signal_source.h File Reference	812
11.58.1 Detailed Description	813
11.59direct_resampler_conditioner.h File Reference	813
11.59.1 Detailed Description	813
11.60direct_resampler_conditioner_cb.h File Reference	814
11.60.1 Detailed Description	814
11.61direct_resampler_conditioner_cc.h File Reference	814
11.61.1 Detailed Description	815
11.62direct_resampler_conditioner_cs.h File Reference	815
11.62.1 Detailed Description	816
11.63display.h File Reference	816
11.63.1 Detailed Description	817
11.64dll_pll_conf.h File Reference	817
11.64.1 Detailed Description	817
11.65dll_pll_conf_fpga.h File Reference	818
11.65.1 Detailed Description	818

11.66dll_pll_veml_tracking.h File Reference	818
11.66.1 Detailed Description	819
11.67dll_pll_veml_tracking_fpga.h File Reference	819
11.67.1 Detailed Description	820
11.68edc.h File Reference	820
11.68.1 Detailed Description	821
11.69exponential_smoother.h File Reference	821
11.69.1 Detailed Description	821
11.70fec.h File Reference	822
11.70.1 Detailed Description	822
11.71fft_base_kernels.h File Reference	822
11.71.1 Detailed Description	823
11.72fft_internal.h File Reference	823
11.72.1 Detailed Description	823
11.73file_configuration.h File Reference	824
11.73.1 Detailed Description	824
11.74file_signal_source.h File Reference	824
11.74.1 Detailed Description	825
11.75fir_filter.h File Reference	825
11.75.1 Detailed Description	826
11.76flexiband_signal_source.h File Reference	826
11.76.1 Detailed Description	826
11.77fmcomms2_signal_source.h File Reference	827
11.77.1 Detailed Description	827
11.78fpga_acquisition.h File Reference	827
11.78.1 Detailed Description	828
11.79fpga_dynamic_bit_selection.h File Reference	828
11.79.1 Detailed Description	828
11.80fpga_multicorrelator.h File Reference	829
11.80.1 Detailed Description	829

11.81	fpga_switch.h File Reference	829
11.81.1	Detailed Description	830
11.82	freq_xlating_fir_filter.h File Reference	830
11.82.1	Detailed Description	830
11.83	front_end_cal.h File Reference	831
11.83.1	Detailed Description	831
11.84	galileo_almanac.h File Reference	831
11.84.1	Detailed Description	832
11.85	galileo_almanac_helper.h File Reference	832
11.85.1	Detailed Description	832
11.86	Galileo_E1.h File Reference	832
11.86.1	Detailed Description	833
11.86.2	Variable Documentation	834
11.86.2.1	GALILEO_E1_B_CODE_LENGTH_CHIPS	834
11.86.2.2	GALILEO_E1_B_SAMPLES_PER_SYMBOL	834
11.86.2.3	GALILEO_E1_B_SYMBOL_RATE_BPS	834
11.86.2.4	GALILEO_E1_C_SECONDARY_CODE_LENGTH	834
11.86.2.5	GALILEO_E1_CODE_CHIP_RATE_CPS	835
11.86.2.6	GALILEO_E1_CODE_PERIOD_MS	835
11.86.2.7	GALILEO_E1_CODE_PERIOD_S	835
11.86.2.8	GALILEO_E1_FREQ_HZ	835
11.86.2.9	GALILEO_E1_HISTORY_DEEP	835
11.86.2.10	GALILEO_E1_OPT_ACQ_FS_SPS	836
11.86.2.11	GALILEO_E1_SUB_CARRIER_A_RATE_HZ	836
11.86.2.12	GALILEO_E1_SUB_CARRIER_B_RATE_HZ	836
11.87	galileo_e1_dll_pll_veml_tracking.h File Reference	836
11.87.1	Detailed Description	837
11.88	galileo_e1_dll_pll_veml_tracking_fpga.h File Reference	837
11.88.1	Detailed Description	837
11.89	galileo_e1_pcps_8ms_ambiguous_acquisition.h File Reference	838

11.89.1 Detailed Description	838
11.90galileo_e1_pcps_ambiguous_acquisition.h File Reference	838
11.90.1 Detailed Description	839
11.91galileo_e1_pcps_ambiguous_acquisition_fpga.h File Reference	839
11.91.1 Detailed Description	839
11.92galileo_e1_pcps_cccwsr_ambiguous_acquisition.h File Reference	840
11.92.1 Detailed Description	840
11.93galileo_e1_pcps_quicksync_ambiguous_acquisition.h File Reference	840
11.93.1 Detailed Description	841
11.94galileo_e1_pcps_tong_ambiguous_acquisition.h File Reference	841
11.94.1 Detailed Description	841
11.95galileo_e1_signal_processing.h File Reference	842
11.95.1 Detailed Description	842
11.95.2 Function Documentation	842
11.95.2.1 galileo_e1_code_gen_complex_sampled() [1/2]	843
11.95.2.2 galileo_e1_code_gen_complex_sampled() [2/2]	843
11.95.2.3 galileo_e1_code_gen_float_sampled() [1/2]	843
11.95.2.4 galileo_e1_code_gen_float_sampled() [2/2]	843
11.95.2.5 galileo_e1_code_gen_sinboc11_float()	844
11.96galileo_e1_tcp_connector_tracking.h File Reference	844
11.96.1 Detailed Description	844
11.97galileo_e1_tcp_connector_tracking_cc.h File Reference	845
11.97.1 Detailed Description	845
11.98galileo_e1b_telemetry_decoder.h File Reference	846
11.98.1 Detailed Description	846
11.99galileo_e5_signal_processing.h File Reference	846
11.99.1 Detailed Description	847
11.99.2 Function Documentation	847
11.99.2.1 galileo_e5_a_code_gen_complex_primary()	847
11.99.2.2 galileo_e5_a_code_gen_complex_sampled()	848

11.99.2.3 galileo_e5_b_code_gen_complex_primary()	848
11.99.2.4 galileo_e5_b_code_gen_complex_sampled()	848
11.100 galileo_E5a.h File Reference	848
11.100.1 Detailed Description	850
11.100.2 Variable Documentation	850
11.100.2.1 GALILEO_E5A_CODE_CHIP_RATE_CPS	850
11.100.2.2 GALILEO_E5A_CODE_LENGTH_CHIPS	850
11.100.2.3 GALILEO_E5A_CODE_PERIOD_MS	850
11.100.2.4 GALILEO_E5A_CODE_PERIOD_S	851
11.100.2.5 GALILEO_E5A_FREQ_HZ	851
11.100.2.6 GALILEO_E5A_I_SECONDARY_CODE_LENGTH	851
11.100.2.7 GALILEO_E5A_I_TIERED_CODE_PERIOD_S	851
11.100.2.8 GALILEO_E5A_OPT_ACQ_FS_SPS	851
11.100.2.9 GALILEO_E5A_Q_SECONDARY_CODE_LENGTH	852
11.100.2.10 GALILEO_E5A_Q_TIERED_CODE_PERIOD_S	852
11.100.2.11 GALILEO_E5A_SYMBOL_RATE_BPS	852
11.101 galileo_e5a_dll_pll_tracking.h File Reference	852
11.101.1 Detailed Description	853
11.102 galileo_e5a_dll_pll_tracking_fpga.h File Reference	853
11.102.1 Detailed Description	853
11.103 galileo_e5a_noncoherent_iq_acquisition_caf.h File Reference	854
11.103.1 Detailed Description	854
11.104 galileo_e5a_noncoherent_iq_acquisition_caf_cc.h File Reference	854
11.104.1 Detailed Description	855
11.105 galileo_e5a_pcps_acquisition.h File Reference	855
11.105.1 Detailed Description	856
11.106 galileo_e5a_pcps_acquisition_fpga.h File Reference	856
11.106.1 Detailed Description	856
11.107 galileo_e5a_telemetry_decoder.h File Reference	857
11.107.1 Detailed Description	857

11.108	Galileo_E5b.h File Reference	857
11.108.1	Detailed Description	858
11.108.2	variable Documentation	859
11.108.2.1	GALILEO_E5B_CODE_CHIP_RATE_CPS	859
11.108.2.2	GALILEO_E5B_CODE_LENGTH_CHIPS	859
11.108.2.3	GALILEO_E5B_CODE_PERIOD_MS	859
11.108.2.4	GALILEO_E5B_CODE_PERIOD_S	859
11.108.2.5	GALILEO_E5B_FREQ_HZ	859
11.108.2.6	GALILEO_E5B_I_SECONDARY_CODE_LENGTH	860
11.108.2.7	GALILEO_E5B_I_TIERED_CODE_PERIOD_S	860
11.108.2.8	GALILEO_E5B_OPT_ACQ_FS_SPS	860
11.108.2.9	GALILEO_E5B_Q_SECONDARY_CODE_LENGTH	860
11.108.2.10	GALILEO_E5B_Q_TIERED_CODE_PERIOD_S	860
11.108.2.11	GALILEO_E5B_SYMBOL_RATE_BPS	861
11.109	galileo_e5b_dll_pll_tracking.h File Reference	861
11.109.1	Detailed Description	861
11.110	galileo_e5b_pcps_acquisition.h File Reference	861
11.110.1	Detailed Description	862
11.111	galileo_e5b_pcps_acquisition_fpga.h File Reference	862
11.111.1	Detailed Description	863
11.112	galileo_e5b_telemetry_decoder.h File Reference	863
11.112.1	Detailed Description	863
11.113	galileo_ephemeris.h File Reference	864
11.113.1	Detailed Description	864
11.114	Galileo_FNAV.h File Reference	864
11.114.1	Detailed Description	867
11.115	galileo_fnav_message.h File Reference	868
11.115.1	Detailed Description	868
11.116	Galileo_INAV.h File Reference	868
11.116.1	Detailed Description	872

11.116.2/variable Documentation	872
11.116.2.1GALILEO_INAV_PAGE_PART_SYMBOLS	873
11.116.2.2GALILEO_INAV_PAGE_PART_WITH_PREAMBLE_SECONDS	873
11.116.2.3GALILEO_INAV_PAGE_SYMBOLS	873
11.117galileo_inav_message.h File Reference	873
11.117.1Detailed Description	874
11.118galileo_iono.h File Reference	874
11.118.1Detailed Description	874
11.119galileo_pcps_8ms_acquisition_cc.h File Reference	875
11.119.1Detailed Description	876
11.120galileo_telemetry_decoder_gs.h File Reference	876
11.120.1Detailed Description	877
11.121galileo_utc_model.h File Reference	877
11.121.1Detailed Description	877
11.122gen_signal_source.h File Reference	878
11.122.1Detailed Description	878
11.123geofunctions.h File Reference	878
11.123.1Detailed Description	879
11.123.2Function Documentation	880
11.123.2.1cart2geo()	880
11.123.2.2cart2utm()	880
11.123.2.3clksin()	881
11.123.2.4clsin()	881
11.123.2.5findUtmZone()	881
11.123.2.6Geo_to_ECEF()	881
11.123.2.7Gravity_ECEF()	882
11.123.2.8great_circle_distance()	882
11.123.2.9pv_Geo_to_ECEF()	882
11.123.2.10skew_symmetric()	882
11.123.2.11togeod()	883

11.123.2. 112 pocent()	883
11.124. 112 geojson_printer.h File Reference	883
11.124.1 Detailed Description	884
11.125. 112 lonass_gnav_almanac.h File Reference	884
11.125.1 Detailed Description	884
11.126. 112 lonass_gnav_ephemeris.h File Reference	885
11.126.1 Detailed Description	885
11.127. 112 lonass_gnav_navigation_message.h File Reference	885
11.127.1 Detailed Description	886
11.128. 112 lonass_gnav_utc_model.h File Reference	886
11.128.1 Detailed Description	887
11.129. 112 lonass_l1_ca_dll_pll_c_aid_tracking.h File Reference	887
11.129.1 Detailed Description	887
11.130. 112 lonass_l1_ca_dll_pll_c_aid_tracking_cc.h File Reference	888
11.130.1 Detailed Description	888
11.131. 112 lonass_l1_ca_dll_pll_c_aid_tracking_sc.h File Reference	889
11.131.1 Detailed Description	889
11.132. 112 lonass_l1_ca_dll_pll_tracking.h File Reference	890
11.132.1 Detailed Description	890
11.133. 112 lonass_l1_ca_dll_pll_tracking_cc.h File Reference	890
11.133.1 Detailed Description	891
11.134. 112 lonass_l1_ca_pcps_acquisition.h File Reference	891
11.134.1 Detailed Description	892
11.135. 112 lonass_l1_ca_telemetry_decoder.h File Reference	892
11.135.1 Detailed Description	892
11.136. 112 lonass_l1_ca_telemetry_decoder_gs.h File Reference	893
11.136.1 Detailed Description	893
11.137. 112 LONASS_L1_L2_CA.h File Reference	894
11.137.1 Detailed Description	897
11.137.2 Macro Definition Documentation	898

11.137.2.1	GLONASS_GNAV_PREAMBLE	898
11.137.3	Variable Documentation	898
11.137.3.1	GLONASS_C20	898
11.137.3.2	GLONASS_EARTH_INCLINATION	898
11.137.3.3	GLONASS_EARTH_RADIUS	899
11.137.3.4	GLONASS_F_M_A	899
11.137.3.5	GLONASS_FLATTENING	899
11.137.3.6	GLONASS_GNAV_HAMMING_CODE_BITS	899
11.137.3.7	GLONASS_GNAV_STRING_BITS	899
11.137.3.8	GLONASS_GNAV_STRING_SYMBOLS	900
11.137.3.9	GLONASS_GNAV_TELEMETRY_RATE_BITS_SECOND	900
11.137.3.10	GLONASS_GNAV_TELEMETRY_RATE_SYMBOLS_SECOND	900
11.137.3.11	GLONASS_GRAVITY	900
11.137.3.12	GLONASS_GRAVITY_CORRECTION	900
11.137.3.13	GLONASS_J2	901
11.137.3.14	GLONASS_J4	901
11.137.3.15	GLONASS_J6	901
11.137.3.16	GLONASS_J8	901
11.137.3.17	GLONASS_L1_CA_CHIP_PERIOD_S	901
11.137.3.18	GLONASS_L1_CA_CODE_LENGTH_CHIPS	902
11.137.3.19	GLONASS_L1_CA_CODE_PERIOD_S	902
11.137.3.20	GLONASS_L1_CA_CODE_RATE_CPS	902
11.137.3.21	GLONASS_L1_CA_DFREQ_HZ	902
11.137.3.22	GLONASS_L1_CA_FREQ_HZ	902
11.137.3.23	GLONASS_L2_CA_CHIP_PERIOD_S	903
11.137.3.24	GLONASS_L2_CA_CODE_LENGTH_CHIPS	903
11.137.3.25	GLONASS_L2_CA_CODE_PERIOD_S	903
11.137.3.26	GLONASS_L2_CA_CODE_RATE_CPS	903
11.137.3.27	GLONASS_L2_CA_DFREQ_HZ	903
11.137.3.28	GLONASS_L2_CA_FREQ_HZ	904

11.137.3.29	LONASS_LEAP_SECONDS	904
11.137.3.30	LONASS_MOON_ECCENTRICITY	904
11.137.3.31	LONASS_MOON_GM	905
11.137.3.32	LONASS_MOON_INCLINATION	905
11.137.3.33	LONASS_MOON_OMEGA_0	905
11.137.3.34	LONASS_MOON_OMEGA_1	905
11.137.3.35	LONASS_MOON_Q0	905
11.137.3.36	LONASS_MOON_Q1	906
11.137.3.37	LONASS_MOON_SEMI_MAJOR_AXIS	906
11.137.3.38	LONASS_SEMI_MAJOR_AXIS	906
11.137.3.39	LONASS_SUN_ECCENTRICITY	906
11.137.3.40	LONASS_SUN_GM	906
11.137.3.41	LONASS_SUN_OMEGA	907
11.137.3.42	LONASS_SUN_Q0	907
11.137.3.43	LONASS_SUN_Q1	907
11.137.3.44	LONASS_SUN_SEMI_MAJOR_AXIS	907
11.137.3.45	LONASS_TAU_0	907
11.137.3.46	LONASS_TAU_1	908
11.137.3.47	LONASS_U0	908
11.138	lonass_l1_signal_processing.h File Reference	908
11.138.1	Detailed Description	908
11.138.2	Function Documentation	909
11.138.2.1	lonass_l1_ca_code_gen_complex()	909
11.138.2.2	lonass_l1_ca_code_gen_complex_sampled() [1/2]	909
11.138.2.3	lonass_l1_ca_code_gen_complex_sampled() [2/2]	909
11.139	lonass_l2_ca_dll_pll_c_aid_tracking.h File Reference	909
11.139.1	Detailed Description	910
11.140	lonass_l2_ca_dll_pll_c_aid_tracking_cc.h File Reference	910
11.140.1	Detailed Description	911
11.141	lonass_l2_ca_dll_pll_c_aid_tracking_sc.h File Reference	911

11.141.1	Detailed Description	912
11.142	glonass_l2_ca_dll_pll_tracking.h File Reference	912
11.142.1	Detailed Description	912
11.143	glonass_l2_ca_dll_pll_tracking_cc.h File Reference	913
11.143.1	Detailed Description	913
11.144	glonass_l2_ca_pcps_acquisition.h File Reference	914
11.144.1	Detailed Description	914
11.145	glonass_l2_ca_telemetry_decoder.h File Reference	914
11.145.1	Detailed Description	915
11.146	glonass_l2_ca_telemetry_decoder_gs.h File Reference	915
11.146.1	Detailed Description	916
11.147	glonass_l2_signal_processing.h File Reference	916
11.147.1	Detailed Description	916
11.147.2	Function Documentation	917
11.147.2.1	glonass_l2_ca_code_gen_complex()	917
11.147.2.2	glonass_l2_ca_code_gen_complex_sampled() [1/2]	917
11.147.2.3	glonass_l2_ca_code_gen_complex_sampled() [2/2]	917
11.148	gnss_signal_source.h File Reference	917
11.148.1	Detailed Description	918
11.149	gnss_max_signal_source.h File Reference	918
11.149.1	Detailed Description	918
11.150	gnss_block_factory.h File Reference	919
11.150.1	Detailed Description	919
11.151	gnss_block_interface.h File Reference	919
11.151.1	Detailed Description	920
11.152	gnss_circular_deque.h File Reference	920
11.152.1	Detailed Description	920
11.153	gnss_flowgraph.h File Reference	921
11.153.1	Detailed Description	921
11.154	gnss_frequencies.h File Reference	921

11.154.1Detailed Description	922
11.154.2variable Documentation	923
11.154.2.1DFRQ1_GLO	923
11.154.2.2DFRQ2_GLO	923
11.154.2.3FREQ1	923
11.154.2.4FREQ1_BDS	923
11.154.2.5FREQ1_GLO	923
11.154.2.6FREQ2	924
11.154.2.7FREQ2_BDS	924
11.154.2.8FREQ2_GLO	924
11.154.2.9FREQ3_BDS	924
11.154.2.10FREQ3_GLO	924
11.154.2.11FREQ5	925
11.154.2.12FREQ6	925
11.154.2.13FREQ7	925
11.154.2.14FREQ8	925
11.154.2.15FREQ9	925
11.155nss_obs_codes.h File Reference	926
11.155.1Detailed Description	928
11.155.2variable Documentation	928
11.155.2.1CODE_L1A	928
11.155.2.2CODE_L1B	929
11.155.2.3CODE_L1C	929
11.155.2.4CODE_L1E	929
11.155.2.5CODE_L1I	929
11.155.2.6CODE_L1L	929
11.155.2.7CODE_L1M	930
11.155.2.8CODE_L1N	930
11.155.2.9CODE_L1P	930
11.155.2.10CODE_L1Q	930

11.155.2.11CODE_L1S	930
11.155.2.12CODE_L1W	931
11.155.2.13CODE_L1X	931
11.155.2.14CODE_L1Y	931
11.155.2.15CODE_L1Z	931
11.155.2.16CODE_L2C	931
11.155.2.17CODE_L2D	932
11.155.2.18CODE_L2I	932
11.155.2.19CODE_L2L	932
11.155.2.20CODE_L2M	932
11.155.2.21CODE_L2N	932
11.155.2.22CODE_L2P	933
11.155.2.23CODE_L2Q	933
11.155.2.24CODE_L2S	933
11.155.2.25CODE_L2W	933
11.155.2.26CODE_L2X	933
11.155.2.27CODE_L2Y	934
11.155.2.28CODE_L3I	934
11.155.2.29CODE_L3Q	934
11.155.2.30CODE_L3X	934
11.155.2.31CODE_L5A	934
11.155.2.32CODE_L5B	935
11.155.2.33CODE_L5C	935
11.155.2.34CODE_L5I	935
11.155.2.35CODE_L5Q	935
11.155.2.36CODE_L5X	935
11.155.2.37CODE_L6A	936
11.155.2.38CODE_L6B	936
11.155.2.39CODE_L6C	936
11.155.2.40CODE_L6I	936

11.155.2.41	CODE_L6L	936
11.155.2.42	CODE_L6Q	937
11.155.2.43	CODE_L6S	937
11.155.2.44	CODE_L6X	937
11.155.2.45	CODE_L6Z	937
11.155.2.46	CODE_L7I	937
11.155.2.47	CODE_L7Q	938
11.155.2.48	CODE_L7X	938
11.155.2.49	CODE_L8I	938
11.155.2.50	CODE_L8Q	938
11.155.2.51	CODE_L8X	938
11.155.2.52	CODE_L9A	939
11.155.2.53	CODE_L9B	939
11.155.2.54	CODE_L9C	939
11.155.2.55	CODE_L9X	939
11.155.2.56	CODE_NONE	939
11.155.2.57	MAXCODE	940
11.156	nss_satellite.h File Reference	940
11.156.1	Detailed Description	940
11.157	nss_sdr_create_directory.h File Reference	940
11.157.1	Detailed Description	941
11.158	nss_sdr_flags.h File Reference	941
11.158.1	Detailed Description	942
11.158.2	Function Documentation	942
11.158.2.1	DECLARE_bool()	942
11.158.2.2	DECLARE_double() [1/3]	943
11.158.2.3	DECLARE_double() [2/3]	943
11.158.2.4	DECLARE_double() [3/3]	943
11.158.2.5	DECLARE_int32() [1/7]	943
11.158.2.6	DECLARE_int32() [2/7]	943

11.158.2.7	DECLARE_int32() [3/7]	943
11.158.2.8	DECLARE_int32() [4/7]	944
11.158.2.9	DECLARE_int32() [5/7]	944
11.158.2.10	DECLARE_int32() [6/7]	944
11.158.2.11	DECLARE_int32() [7/7]	944
11.158.2.12	DECLARE_string() [1/7]	944
11.158.2.13	DECLARE_string() [2/7]	944
11.158.2.14	DECLARE_string() [3/7]	945
11.158.2.15	DECLARE_string() [4/7]	945
11.158.2.16	DECLARE_string() [5/7]	945
11.158.2.17	DECLARE_string() [6/7]	945
11.158.2.18	DECLARE_string() [7/7]	945
11.159	nss_sdr_fpga_sample_counter.h File Reference	945
11.159.1	Detailed Description	946
11.160	nss_sdr_make_unique.h File Reference	946
11.160.1	Detailed Description	946
11.161	nss_sdr_sample_counter.h File Reference	947
11.161.1	Detailed Description	947
11.162	nss_sdr_supl_client.h File Reference	948
11.162.1	Detailed Description	948
11.163	nss_sdr_time_counter.h File Reference	949
11.163.1	Detailed Description	949
11.164	nss_sdr_valve.h File Reference	949
11.164.1	Detailed Description	950
11.165	nss_signal.h File Reference	950
11.165.1	Detailed Description	951
11.166	nss_signal_processing.h File Reference	951
11.166.1	Detailed Description	951
11.166.2	Function Documentation	952
11.166.2.1	complex_exp_gen()	952

11.166.2.2complex_exp_gen_conj()	952
11.166.2.3hex_to_binary_converter()	952
11.166.2.4resampler() [1/2]	952
11.166.2.5resampler() [2/2]	953
11.167gnss_synchro.h File Reference	953
11.167.1Detailed Description	953
11.168gnss_synchro_monitor.h File Reference	953
11.168.1Detailed Description	954
11.169gnss_synchro_udp_sink.h File Reference	954
11.169.1Detailed Description	955
11.170nuplot_i.h File Reference	955
11.170.1Detailed Description	956
11.171gps_acq_assist.h File Reference	956
11.171.1Detailed Description	956
11.172gps_almanac.h File Reference	957
11.172.1Detailed Description	957
11.173GPS_CNAV.h File Reference	957
11.173.1Detailed Description	960
11.174gps_cnav_ephemeris.h File Reference	960
11.174.1Detailed Description	960
11.175gps_cnav_iono.h File Reference	961
11.175.1Detailed Description	961
11.176gps_cnav_navigation_message.h File Reference	961
11.176.1Detailed Description	962
11.177gps_cnav_utc_model.h File Reference	962
11.177.1Detailed Description	962
11.178gps_ephemeris.h File Reference	963
11.178.1Detailed Description	963
11.179gps_iono.h File Reference	963
11.179.1Detailed Description	964

11.180GPS_L1_CA.h File Reference	964
11.180.1Detailed Description	967
11.180.2Function Documentation	968
11.180.2.1ALPHA_0()	968
11.180.2.2T_OA()	968
11.180.3Variable Documentation	968
11.180.3.1GPS_CA_TELEMETRY_RATE_BITS_SECOND	968
11.180.3.2GPS_CA_TELEMETRY_RATE_SYMBOLS_SECOND	968
11.180.3.3GPS_L1_CA_BIT_PERIOD_MS	969
11.180.3.4GPS_L1_CA_CHIP_PERIOD_S	969
11.180.3.5GPS_L1_CA_CODE_LENGTH_CHIPS	969
11.180.3.6GPS_L1_CA_CODE_PERIOD_MS	969
11.180.3.7GPS_L1_CA_CODE_PERIOD_S	969
11.180.3.8GPS_L1_CA_CODE_RATE_CPS	970
11.180.3.9GPS_L1_CA_OPT_ACQ_FS_SPS	970
11.180.3.10GPS_L1_FREQ_HZ	970
11.180.3.11GPS_SUBFRAME_BITS	970
11.180.3.12GPS_SUBFRAME_LENGTH	970
11.180.3.13GPS_SUBFRAME_MS	971
11.180.3.14GPS_SUBFRAME_SECONDS	971
11.180.3.15GPS_WORD_BITS	971
11.180.3.16GPS_WORD_LENGTH	971
11.180.3.17MAX_TOA_DELAY_MS	971
11.180gps_l1_ca_dll_pll_tracking.h File Reference	972
11.181.1Detailed Description	972
11.180gps_l1_ca_dll_pll_tracking_fpga.h File Reference	972
11.182.1Detailed Description	973
11.180gps_l1_ca_dll_pll_tracking_gpu.h File Reference	973
11.183.1Detailed Description	973
11.180gps_l1_ca_dll_pll_tracking_gpu_cc.h File Reference	974

11.184. Detailed Description	974
11.185. ps_l1_ca_kf_tracking.h File Reference	975
11.185. Detailed Description	975
11.186. ps_l1_ca_kf_tracking_cc.h File Reference	975
11.186. Detailed Description	976
11.187. ps_l1_ca_pcps_acquisition.h File Reference	976
11.187. Detailed Description	977
11.188. ps_l1_ca_pcps_acquisition_fine_doppler.h File Reference	977
11.188. Detailed Description	978
11.189. ps_l1_ca_pcps_acquisition_fpga.h File Reference	978
11.189. Detailed Description	978
11.190. ps_l1_ca_pcps_assisted_acquisition.h File Reference	979
11.190. Detailed Description	979
11.191. ps_l1_ca_pcps_openc1_acquisition.h File Reference	979
11.191. Detailed Description	980
11.192. ps_l1_ca_pcps_quicksync_acquisition.h File Reference	980
11.192. Detailed Description	980
11.193. ps_l1_ca_pcps_tong_acquisition.h File Reference	981
11.193. Detailed Description	981
11.194. ps_l1_ca_tcp_connector_tracking.h File Reference	981
11.194. Detailed Description	982
11.195. ps_l1_ca_tcp_connector_tracking_cc.h File Reference	982
11.195. Detailed Description	983
11.196. ps_l1_ca_telemetry_decoder.h File Reference	983
11.196. Detailed Description	983
11.197. ps_l1_ca_telemetry_decoder_gs.h File Reference	984
11.197. Detailed Description	984
11.198. ps_l2_m_dll_pll_tracking.h File Reference	985
11.198. Detailed Description	985
11.199. ps_l2_m_dll_pll_tracking_fpga.h File Reference	985

11.199.1 Detailed Description	986
11.200gps_l2_m_pcps_acquisition.h File Reference	986
11.200.1 Detailed Description	986
11.200gps_l2_m_pcps_acquisition_fpga.h File Reference	987
11.201.1 Detailed Description	987
11.202GPS_L2C.h File Reference	987
11.202.1 Detailed Description	988
11.202.2 Variable Documentation	988
11.202.2.1GPS_L2_CNAV_DATA_PAGE_BITS	989
11.202.2.2GPS_L2_FREQ_HZ	989
11.202.2.3GPS_L2_L_CODE_LENGTH_CHIPS	989
11.202.2.4GPS_L2_L_CODE_RATE_CPS	989
11.202.2.5GPS_L2_L_PERIOD_S	989
11.202.2.6GPS_L2_M_CODE_LENGTH_CHIPS	990
11.202.2.7GPS_L2_M_CODE_RATE_CPS	990
11.202.2.8GPS_L2_M_PERIOD_S	990
11.202.2.9GPS_L2C_M_INIT_REG	990
11.202.2.10GPS_L2C_OPT_ACQ_FS_SPS	991
11.203gps_l2c_signal.h File Reference	991
11.203.1 Detailed Description	991
11.203.2 Function Documentation	991
11.203.2.1gps_l2c_m_code_gen_complex()	992
11.203.2.2gps_l2c_m_code_gen_complex_sampled()	992
11.204gps_l2c_telemetry_decoder.h File Reference	992
11.204.1 Detailed Description	992
11.205gps_l2c_telemetry_decoder_gs.h File Reference	993
11.205.1 Detailed Description	993
11.206GPS_L5.h File Reference	993
11.206.1 Detailed Description	994
11.206.2 Variable Documentation	995

11.206.2.1GPS_L5_CNAV_DATA_PAGE_BITS	995
11.206.2.2GPS_L5_FREQ_HZ	995
11.206.2.3GPS_L5_OPT_ACQ_FS_SPS	995
11.206.2.4GPS_L5I_CODE_LENGTH_CHIPS	995
11.206.2.5GPS_L5I_CODE_RATE_CPS	995
11.206.2.6GPS_L5I_PERIOD_MS	996
11.206.2.7GPS_L5I_PERIOD_S	996
11.206.2.8GPS_L5I_SYMBOL_PERIOD_MS	996
11.206.2.9GPS_L5I_SYMBOL_PERIOD_S	996
11.206.2.10GPS_L5Q_CODE_LENGTH_CHIPS	996
11.206.2.11GPS_L5Q_CODE_RATE_CPS	997
11.206.2.12GPS_L5Q_PERIOD_S	997
11.207gps_l5_dll_pll_tracking.h File Reference	997
11.207.1Detailed Description	997
11.208gps_l5_dll_pll_tracking_fpga.h File Reference	998
11.208.1Detailed Description	998
11.209gps_l5_signal.h File Reference	998
11.209.1Detailed Description	999
11.209.2Function Documentation	999
11.209.2.1gps_l5i_code_gen_complex()	999
11.209.2.2gps_l5i_code_gen_complex_sampled()	1000
11.209.2.3gps_l5i_code_gen_float()	1000
11.209.2.4gps_l5q_code_gen_complex()	1000
11.209.2.5gps_l5q_code_gen_complex_sampled()	1000
11.209.2.6gps_l5q_code_gen_float()	1000
11.210gps_l5_telemetry_decoder.h File Reference	1001
11.210.1Detailed Description	1001
11.211gps_l5_telemetry_decoder_gs.h File Reference	1001
11.211.1Detailed Description	1002
11.212gps_l5i_pcps_acquisition.h File Reference	1002

11.212. Detailed Description	1003
11.213. gps_l5i_pcps_acquisition_fpga.h File Reference	1003
11.213. Detailed Description	1003
11.214. gps_navigation_message.h File Reference	1004
11.214. Detailed Description	1004
11.215. gps_sdr_signal_processing.h File Reference	1004
11.215. Detailed Description	1005
11.215. Function Documentation	1005
11.215.2. gps_l1_ca_code_gen_complex()	1005
11.215.2.2. gps_l1_ca_code_gen_complex_sampled() [1/2]	1006
11.215.2.3. gps_l1_ca_code_gen_complex_sampled() [2/2]	1006
11.215.2.4. gps_l1_ca_code_gen_float()	1006
11.215.2.5. gps_l1_ca_code_gen_int()	1006
11.216. gps_utc_model.h File Reference	1006
11.216. Detailed Description	1007
11.217. gpx_printer.h File Reference	1007
11.217. Detailed Description	1007
11.218. gr_complex_ip_packet_source.h File Reference	1008
11.218. Detailed Description	1008
11.219. hybrid_observables.h File Reference	1008
11.219. Detailed Description	1009
11.220. hybrid_observables_gs.h File Reference	1009
11.220. Detailed Description	1010
11.221. byte_to_cbyte.h File Reference	1010
11.221. Detailed Description	1010
11.222. byte_to_complex.h File Reference	1011
11.222. Detailed Description	1011
11.223. byte_to_cshort.h File Reference	1011
11.223. Detailed Description	1012
11.224. h_memory_configuration.h File Reference	1012

11.224. Detailed Description	1012
11.225. ni.h File Reference	1013
11.225. Detailed Description	1013
11.226. NIReader.h File Reference	1013
11.226. Detailed Description	1014
11.227. interleaved_byte_to_complex_byte.h File Reference	1014
11.227. Detailed Description	1015
11.228. interleaved_byte_to_complex_short.h File Reference	1015
11.228. Detailed Description	1016
11.229. interleaved_short_to_complex_short.h File Reference	1016
11.229. Detailed Description	1016
11.230. short_to_complex.h File Reference	1017
11.230. Detailed Description	1017
11.231. short_to_cshort.h File Reference	1017
11.231. Detailed Description	1018
11.232. item_type_helpers.h File Reference	1018
11.232. Detailed Description	1018
11.232. Function Documentation	1019
11.232.2.1 item_type_is_complex()	1019
11.232.2.2 item_type_size()	1019
11.232.2.3 item_type_valid()	1019
11.232.2.4 make_vector_converter()	1019
11.233. ml_printer.h File Reference	1020
11.233. Detailed Description	1020
11.234. absat23_source.h File Reference	1021
11.234. Detailed Description	1021
11.235. absat_signal_source.h File Reference	1021
11.235. Detailed Description	1022
11.236. cck_detectors.h File Reference	1022
11.236. Detailed Description	1022

11.236.2	Function Documentation	1023
11.236.2.1	carrier_lock_detector()	1023
11.236.2.2	cn0_m2m4_estimator()	1023
11.236.2.3	cn0_svn_estimator()	1024
11.237	MATH_CONSTANTS.h File Reference	1024
11.237.1	Detailed Description	1027
11.237.2	Variable Documentation	1028
11.237.2.1	AS2R	1028
11.237.2.2	AU	1028
11.237.2.3	BEIDOU_F	1028
11.237.2.4	BEIDOU_GM	1028
11.237.2.5	BEIDOU_OMEGA_EARTH_DOT	1028
11.237.2.6	D2R	1029
11.237.2.7	GALILEO_F	1029
11.237.2.8	GALILEO_GM	1029
11.237.2.9	GLONASS_GM	1029
11.237.2.10	GLONASS_OMEGA_EARTH_DOT	1029
11.237.2.11	GNSS_OMEGA_EARTH_DOT	1030
11.237.2.12	GNSS_PI	1030
11.237.2.13	GPS_F	1030
11.237.2.14	GPS_GM	1030
11.237.2.15	HALF_PI	1030
11.237.2.16	BI_TWO_N19	1031
11.237.2.17	FI_TWO_N23	1031
11.237.2.18	BI_TWO_N31	1031
11.237.2.19	FI_TWO_N38	1031
11.237.2.20	FI_TWO_N43	1031
11.237.2.21	2D	1032
11.237.2.22	C2RAD	1032
11.237.2.23	SPEED_OF_LIGHT_M_MS	1032

11.237.2.23 SPEED_OF_LIGHT_M_S	1032
11.237.2.25 WO_N10	1032
11.237.2.26 WO_N11	1033
11.237.2.27 WO_N14	1033
11.237.2.28 WO_N15	1033
11.237.2.29 WO_N16	1033
11.237.2.30 WO_N17	1033
11.237.2.31 WO_N18	1034
11.237.2.32 WO_N19	1034
11.237.2.33 WO_N2	1034
11.237.2.34 WO_N20	1034
11.237.2.35 WO_N21	1034
11.237.2.36 WO_N23	1035
11.237.2.37 WO_N24	1035
11.237.2.38 WO_N25	1035
11.237.2.39 WO_N27	1035
11.237.2.40 WO_N29	1035
11.237.2.41 WO_N30	1036
11.237.2.42 WO_N31	1036
11.237.2.43 WO_N32	1036
11.237.2.44 WO_N33	1036
11.237.2.45 WO_N34	1036
11.237.2.46 WO_N35	1037
11.237.2.47 WO_N38	1037
11.237.2.48 WO_N39	1037
11.237.2.49 WO_N40	1037
11.237.2.50 WO_N43	1037
11.237.2.51 WO_N44	1038
11.237.2.52 WO_N46	1038
11.237.2.53 WO_N48	1038

11.237.2.54WO_N5	1038
11.237.2.55WO_N50	1038
11.237.2.56WO_N51	1039
11.237.2.57WO_N55	1039
11.237.2.58WO_N57	1039
11.237.2.59WO_N59	1039
11.237.2.60WO_N6	1039
11.237.2.61WO_N60	1040
11.237.2.62WO_N66	1040
11.237.2.63WO_N68	1040
11.237.2.64WO_N8	1040
11.237.2.65WO_N9	1040
11.237.2.66WO_P11	1041
11.237.2.67WO_P12	1041
11.237.2.68WO_P14	1041
11.237.2.69WO_P16	1041
11.237.2.70WO_P19	1041
11.237.2.71WO_P3	1042
11.237.2.72WO_P31	1042
11.237.2.73WO_P32	1042
11.237.2.74WO_P4	1042
11.237.2.75WO_P56	1042
11.237.2.76WO_P57	1043
11.237.2.77WO_PI	1043
11.238.1mse_resampler_conditioner.h File Reference	1043
11.238.1Detailed Description	1043
11.239.1monitor_pvt.h File Reference	1044
11.239.1Detailed Description	1044
11.240.1monitor_pvt_udp_sink.h File Reference	1044
11.240.1Detailed Description	1045

11.241	multichannel_file_signal_source.h File Reference	1045
11.241.1	Detailed Description	1045
11.242	mea_printer.h File Reference	1046
11.242.1	Detailed Description	1046
11.243	nonlinear_tracking.h File Reference	1046
11.243.1	Detailed Description	1047
11.244	notch_cc.h File Reference	1047
11.244.1	Detailed Description	1048
11.245	notch_filter.h File Reference	1048
11.245.1	Detailed Description	1048
11.246	notch_filter_lite.h File Reference	1049
11.246.1	Detailed Description	1049
11.247	notch_lite_cc.h File Reference	1049
11.247.1	Detailed Description	1050
11.248	sr_file_signal_source.h File Reference	1050
11.248.1	Detailed Description	1051
11.249	bs_conf.h File Reference	1051
11.249.1	Detailed Description	1051
11.250	bsdiff_flags.h File Reference	1052
11.250.1	Detailed Description	1052
11.251	observable_tests_flags.h File Reference	1052
11.251.1	Detailed Description	1053
11.252	observables_dump_reader.h File Reference	1053
11.252.1	Detailed Description	1053
11.253	observables_interface.h File Reference	1054
11.253.1	Detailed Description	1054
11.254	smosdr_signal_source.h File Reference	1054
11.254.1	Detailed Description	1055
11.255	pass_through.h File Reference	1055
11.255.1	Detailed Description	1055

11.256	pcps_acquisition.h File Reference	1056
11.256.1	Detailed Description	1056
11.257	pcps_acquisition_fine_doppler_cc.h File Reference	1057
11.257.1	Detailed Description	1058
11.258	pcps_acquisition_fpga.h File Reference	1058
11.258.1	Detailed Description	1059
11.259	pcps_assisted_acquisition_cc.h File Reference	1059
11.259.1	Detailed Description	1060
11.260	pcps_cccwsr_acquisition_cc.h File Reference	1060
11.260.1	Detailed Description	1061
11.261	pcps_openc1_acquisition_cc.h File Reference	1061
11.261.1	Detailed Description	1062
11.262	pcps_quicksync_acquisition_cc.h File Reference	1063
11.262.1	Detailed Description	1063
11.263	pcps_tong_acquisition_cc.h File Reference	1064
11.263.1	Detailed Description	1065
11.264	plutosdr_signal_source.h File Reference	1066
11.264.1	Detailed Description	1066
11.265	position_test_flags.h File Reference	1066
11.265.1	Detailed Description	1067
11.266	pulse_blanking_cc.h File Reference	1067
11.266.1	Detailed Description	1068
11.267	pulse_blanking_filter.h File Reference	1068
11.267.1	Detailed Description	1069
11.268	pvt_conf.h File Reference	1069
11.268.1	Detailed Description	1069
11.269	pvt_interface.h File Reference	1070
11.269.1	Detailed Description	1070
11.270	pvt_solution.h File Reference	1070
11.270.1	Detailed Description	1071

11.271 <code>law_array_signal_source.h</code> File Reference	1071
11.271.1 Detailed Description	1071
11.272 <code>anex_printer.h</code> File Reference	1072
11.272.1 Detailed Description	1072
11.273 <code>acm.h</code> File Reference	1073
11.273.1 Detailed Description	1073
11.274 <code>acm_printer.h</code> File Reference	1074
11.274.1 Detailed Description	1074
11.275 <code>aklib.h</code> File Reference	1074
11.275.1 Detailed Description	1083
11.275.2 Typedef Documentation	1084
11.275.2.1 <code>fatalfunc_t</code>	1084
11.275.3 Variable Documentation	1084
11.275.3.1 <code>ARMODE_CONT</code>	1084
11.275.3.2 <code>ARMODE_FIXHOLD</code>	1084
11.275.3.3 <code>ARMODE_INST</code>	1084
11.275.3.4 <code>ARMODE_OFF</code>	1085
11.275.3.5 <code>ARMODE_PPPAR</code>	1085
11.275.3.6 <code>ARMODE_PPPAR_ILS</code>	1085
11.275.3.7 <code>CHISQR</code>	1085
11.275.3.8 <code>DTTOL</code>	1086
11.275.3.9 <code>EFACT_BDS</code>	1086
11.275.3.10 <code>EFACT_GAL</code>	1086
11.275.3.11 <code>EFACT_GLO</code>	1086
11.275.3.12 <code>EFACT_GPS</code>	1086
11.275.3.13 <code>EFACT_IRN</code>	1087
11.275.3.14 <code>EFACT_QZS</code>	1087
11.275.3.15 <code>EFACT_SBS</code>	1087
11.275.3.16 <code>PHOPT_BRDC</code>	1087
11.275.3.17 <code>PHOPT_LEX</code>	1087

11.275.3.18PHOPT_PREC	1088
11.275.3.19PHOPT_SBAS	1088
11.275.3.20PHOPT_SSRAPC	1088
11.275.3.21PHOPT_SSRCOM	1088
11.275.3.22RR_BRDCI	1088
11.275.3.23RR_CBIAS	1089
11.275.3.24RR_SAAS	1089
11.275.3.25E_WGS84	1089
11.275.3.26TP_TIMEOUT	1089
11.275.3.27AP_RESION	1089
11.275.3.28ION	1090
11.275.3.29IT_SWAP_STAT	1090
11.275.3.30IT_SWAP_TRAC	1090
11.275.3.31NOOPT_BRDC	1090
11.275.3.32NOOPT_EST	1090
11.275.3.33NOOPT_IFLC	1091
11.275.3.34NOOPT_LEX	1091
11.275.3.35NOOPT_OFF	1091
11.275.3.36NOOPT_QZS	1091
11.275.3.37NOOPT_SBAS	1091
11.275.3.38NOOPT_STEC	1092
11.275.3.39NOOPT_TEC	1092
11.275.3.40AM_CARR	1092
11.275.3.41MAXANT	1092
11.275.3.42MAXBAND	1093
11.275.3.43MAXCLI	1093
11.275.3.44MAXDToe	1093
11.275.3.45MAXDToe_BDS	1093
11.275.3.46MAXDToe_GAL	1093
11.275.3.47MAXDToe_GLO	1094

11.275.3.49	MAXDTE_QZS	1094
11.275.3.49	MAXDTE_S	1094
11.275.3.50	MAXDTE_SBS	1094
11.275.3.51	MAXERRMSG	1094
11.275.3.52	MAXEXFILE	1095
11.275.3.53	MAXFREQ	1095
11.275.3.54	MAXGDOP	1095
11.275.3.55	MAXLEAPS	1095
11.275.3.56	MAXNGEO	1095
11.275.3.57	MAXNIGP	1096
11.275.3.58	MAXOBS	1096
11.275.3.59	MAXOBSBUF	1096
11.275.3.60	MAXOBSTYPE	1096
11.275.3.61	MAXPRNBDS	1096
11.275.3.62	MAXPRNGAL	1097
11.275.3.63	MAXPRNGLO	1097
11.275.3.64	MAXPRNGPS	1097
11.275.3.65	MAXPRNSBS	1097
11.275.3.66	MAXRAWLEN	1097
11.275.3.67	MAXRCV	1098
11.275.3.68	MAXSBSAGEF	1098
11.275.3.69	MAXSBSAGEL	1098
11.275.3.70	MAXSBSMSG	1098
11.275.3.71	MAXSBSURA	1098
11.275.3.72	MAXSOLBUF	1099
11.275.3.73	MAXSOLMSG	1099
11.275.3.74	MAXSOLQ	1099
11.275.3.75	MAXSTATMSG	1099
11.275.3.76	MAXSTRMSG	1099
11.275.3.77	MAXSTRPATH	1100

11.275.3.78	INPRNBDS	1100
11.275.3.79	INPRNGAL	1100
11.275.3.80	INPRNGLO	1100
11.275.3.81	INPRNGPS	1100
11.275.3.82	INPRNSBS	1101
11.275.3.83	EXOBS	1101
11.275.3.84	FREQ	1101
11.275.3.85	FREQGLO	1101
11.275.3.86	SATBDS	1101
11.275.3.87	SATGAL	1102
11.275.3.88	SATGLO	1102
11.275.3.89	SATGPS	1102
11.275.3.90	SATSBS	1102
11.275.3.91	SYS	1102
11.275.3.92	MODE_DGPS	1103
11.275.3.93	MODE_FIXED	1103
11.275.3.94	MODE_KINEMA	1103
11.275.3.95	MODE_MOVEB	1103
11.275.3.96	MODE_PPP_FIXED	1103
11.275.3.97	MODE_PPP_KINEMA	1104
11.275.3.98	MODE_PPP_STATIC	1104
11.275.3.99	MODE_SINGLE	1104
11.275.3.100	MODE_STATIC	1104
11.275.3.101	LYCRC24Q	1104
11.275.3.102	LYCRC32	1105
11.275.3.103	SOPT_RINEX	1105
11.275.3.104	IN_HWBIAS	1105
11.275.3.105	WGS84	1105
11.275.3.106	EL_HUMI	1105
11.275.3.107	ERIBUFFSIZE	1106

11.275.3.130LF_ENU	1106
11.275.3.130LF_GSIF	1106
11.275.3.130LF_LLH	1106
11.275.3.130LF_NMEA	1106
11.275.3.130LF_STAT	1107
11.275.3.130LF_XYZ	1107
11.275.3.130LQ_DGPS	1107
11.275.3.130LQ_DR	1107
11.275.3.130LQ_FIX	1107
11.275.3.130LQ_FLOAT	1108
11.275.3.130LQ_NONE	1108
11.275.3.130LQ_PPP	1108
11.275.3.130LQ_SBAS	1108
11.275.3.130LQ_SINGLE	1108
11.275.3.130YS_ALL	1109
11.275.3.130YS_BDS	1109
11.275.3.130YS_GAL	1109
11.275.3.130YS_GLO	1109
11.275.3.130YS_GPS	1109
11.275.3.130YS_IRN	1110
11.275.3.130YS_LEO	1110
11.275.3.130YS_NONE	1110
11.275.3.130YS_QZS	1110
11.275.3.130YS_SBS	1110
11.275.3.130MES_GPST	1111
11.275.3.130MES_JST	1111
11.275.3.130MES_UTC	1111
11.275.3.130METAGH_LEN	1111
11.275.3.130INTACT	1111
11.275.3.130PROPT_COR	1112

11.275.3.11BOPOPT_CORG	1112
11.275.3.11BOPOPT_EST	1112
11.275.3.11BOPOPT_ESTG	1112
11.275.3.11BOPOPT_OFF	1112
11.275.3.11BOPOPT_SAAS	1113
11.275.3.11BOPOPT_SBAS	1113
11.276.1rtklib_conversions.h File Reference	1113
11.276.1Detailed Description	1113
11.276.2Function Documentation	1114
11.276.2.1eph_to_rtklib()	1114
11.277.1rtklib_ephemeris.h File Reference	1114
11.277.1Detailed Description	1115
11.278.1rtklib_ionex.h File Reference	1115
11.278.1Detailed Description	1116
11.279.1rtklib_lambda.h File Reference	1116
11.279.1Detailed Description	1117
11.279.2Macro Definition Documentation	1117
11.279.2.1SWAP_LAMBDA	1118
11.280.1rtklib_pntpos.h File Reference	1118
11.280.1Detailed Description	1119
11.280.2Function Documentation	1119
11.280.2.1pntpos()	1120
11.280.3variable Documentation	1120
11.280.3.1ERR_ION	1120
11.280.3.2ERR_TROP	1120
11.280.3.3MAXITR	1120
11.280.3.4NX	1121
11.281.1rtklib_ppp.h File Reference	1121
11.281.1Detailed Description	1122
11.281.2Macro Definition Documentation	1123

11.281.2.1SWAP_D	1123
11.281.2.2SWAP_I	1123
11.282klib_preceph.h File Reference	1123
11.282.1Detailed Description	1124
11.283klib_pvt.h File Reference	1125
11.283.1Detailed Description	1125
11.284klib_pvt_gs.h File Reference	1125
11.284.1Detailed Description	1126
11.285klib_rtcn.h File Reference	1126
11.285.1Detailed Description	1127
11.286klib_rtcn2.h File Reference	1127
11.286.1Detailed Description	1128
11.287klib_rtcn3.h File Reference	1128
11.287.1Detailed Description	1130
11.287.2Variable Documentation	1131
11.287.2.1CODES_BDS	1131
11.287.2.2CODES_GAL	1131
11.287.2.3CODES_GLO	1131
11.287.2.4CODES_GPS	1132
11.287.2.5CODES_QZS	1132
11.287.2.6CODES_SBS	1132
11.287.2.7SSRUDINT	1132
11.288klib_rtkcmn.h File Reference	1133
11.288.1Detailed Description	1136
11.288.2Macro Definition Documentation	1136
11.288.2.1Rx	1136
11.288.2.2Ry	1137
11.288.2.3Rz	1137
11.289klib_rtkpos.h File Reference	1137
11.289.1Detailed Description	1139

11.290. rtklib_rtksvr.h File Reference	1139
11.290.1. Detailed Description	1140
11.290.2. Variable Documentation	1141
11.290.2.1. PRCOPT_DEFAULT	1141
11.290.2.2. SOLOPT_DEFAULT	1141
11.291. rtklib_sbss.h File Reference	1141
11.291.1. Detailed Description	1142
11.291.2. Variable Documentation	1143
11.291.2.1. IGPBAND1	1143
11.291.2.2. IGPBAND2	1144
11.292. rtklib_solution.h File Reference	1144
11.292.1. Detailed Description	1145
11.293. rtklib_solver.h File Reference	1146
11.293.1. Detailed Description	1146
11.294. rtklib_solver_dump_reader.h File Reference	1147
11.294.1. Detailed Description	1147
11.295. rtklib_stream.h File Reference	1147
11.295.1. Detailed Description	1149
11.296. rtklib_tides.h File Reference	1150
11.296.1. Detailed Description	1150
11.297. rtl_tcp_commands.h File Reference	1151
11.297.1. Detailed Description	1151
11.297.2. Enumeration Type Documentation	1151
11.297.2.1. RTL_TCP_COMMAND	1152
11.297.3. Function Documentation	1152
11.297.3.1. rtl_tcp_command()	1152
11.298. rtl_tcp_dongle_info.h File Reference	1152
11.298.1. Detailed Description	1152
11.299. rtl_tcp_signal_source.h File Reference	1153
11.299.1. Detailed Description	1153

11.300	tl_tcp_signal_source_c.h File Reference	1154
11.300.1	Detailed Description	1154
11.301	bas_ephemeris.h File Reference	1155
11.301.1	Detailed Description	1155
11.302	bas_l1_telemetry_decoder.h File Reference	1155
11.302.1	Detailed Description	1156
11.303	bas_l1_telemetry_decoder_gs.h File Reference	1156
11.303.1	Detailed Description	1157
11.304	erdes_gnss_synchro.h File Reference	1157
11.304.1	Detailed Description	1157
11.305	erdes_monitor_pvt.h File Reference	1158
11.305.1	Detailed Description	1158
11.306	short_x2_to_cshort.h File Reference	1158
11.306.1	Detailed Description	1159
11.307	signal_conditioner.h File Reference	1159
11.307.1	Detailed Description	1159
11.308	signal_generator.h File Reference	1160
11.308.1	Detailed Description	1160
11.309	signal_generator_c.h File Reference	1160
11.309.1	Detailed Description	1161
11.309.2	Function Documentation	1161
11.309.2.1	signal_make_generator_c()	1161
11.310	signal_generator_flags.h File Reference	1162
11.310.1	Detailed Description	1162
11.311	spir_file_signal_source.h File Reference	1163
11.311.1	Detailed Description	1163
11.312	spir_gss6450_file_signal_source.h File Reference	1163
11.312.1	Detailed Description	1164
11.313	spirent_motion_csv_dump_reader.h File Reference	1164
11.313.1	Detailed Description	1165

11.314	tring_converter.h File Reference	1165
11.314.1	Detailed Description	1165
11.315	swift_common.h File Reference	1166
11.315.1	Detailed Description	1166
11.316	cp_cmd_interface.h File Reference	1166
11.316.1	Detailed Description	1167
11.317	cp_communication.h File Reference	1167
11.317.1	Detailed Description	1168
11.318	cp_packet_data.h File Reference	1168
11.318.1	Detailed Description	1168
11.319	elemetry_decoder_interface.h File Reference	1168
11.319.1	Detailed Description	1169
11.320	test_flags.h File Reference	1169
11.320.1	Detailed Description	1169
11.321	lm_dump_reader.h File Reference	1170
11.321.1	Detailed Description	1170
11.322	tracking_2nd_DLL_filter.h File Reference	1170
11.322.1	Detailed Description	1170
11.323	tracking_2nd_PLL_filter.h File Reference	1171
11.323.1	Detailed Description	1171
11.324	tracking_discriminators.h File Reference	1171
11.324.1	Detailed Description	1172
11.324.2	Function Documentation	1172
11.324.2.1	dll_nc_e_minus_l_normalized()	1173
11.324.2.2	dll_nc_vemlp_normalized()	1173
11.324.2.3	ll_four_quadrant_atan()	1173
11.324.2.4	phase_unwrap()	1174
11.324.2.5	pll_cloop_two_quadrant_atan()	1174
11.324.2.6	pll_four_quadrant_atan()	1174
11.325	tracking_dump_reader.h File Reference	1174

11.325. Detailed Description	1175
11.326. tracking_FLL_PLL_filter.h File Reference	1175
11.326. Detailed Description	1175
11.327. tracking_interface.h File Reference	1175
11.327. Detailed Description	1176
11.328. tracking_loop_filter.h File Reference	1176
11.328. Detailed Description	1176
11.329. tracking_tests_flags.h File Reference	1177
11.329. Detailed Description	1178
11.330. tracking_true_obs_reader.h File Reference	1178
11.330. Detailed Description	1178
11.331. true_observables_reader.h File Reference	1179
11.331. Detailed Description	1179
11.332. two_bit_cpx_file_signal_source.h File Reference	1179
11.332. Detailed Description	1180
11.333. two_bit_packed_file_signal_source.h File Reference	1180
11.333. Detailed Description	1181
11.334. uhd_signal_source.h File Reference	1181
11.334. Detailed Description	1181
11.335. unpack_2bit_samples.h File Reference	1182
11.335. Detailed Description	1182
11.336. unpack_byte_2bit_cpx_samples.h File Reference	1183
11.336. Detailed Description	1183
11.337. unpack_byte_2bit_samples.h File Reference	1184
11.337. Detailed Description	1184
11.338. unpack_byte_4bit_samples.h File Reference	1184
11.338. Detailed Description	1185
11.339. unpack_intspir_1bit_samples.h File Reference	1185
11.339. Detailed Description	1186
11.340. unpack_spir_gss6450_samples.h File Reference	1186
11.340. Detailed Description	1186
11.341. viterbi_decoder.h File Reference	1187
11.341. Detailed Description	1187

Chapter 1

Main Page



Figure 1.1 GNSS-SDR logo

Welcome to GNSS-SDR!

GNSS-SDR is an open-source [GNSS software receiver](#) freely available to the research community. This project provides a common framework for GNSS signal processing which can operate in a variety of computer platforms. This tool is intended to foster collaboration, increase awareness, and reduce development costs in the field of GNSS receiver design and customized use of GNSS signals.

For details about GNSS-SDR and using it, please see the [main project page](#) or browse [the source code at GitHub](#). You could be also interested in [subscribing to the mailing list](#).

1.1 Contents

- [Overview](#)
- [Building GNSS-SDR](#)
- [Using GNSS-SDR](#)
- [Control plane](#)
- [Signal Processing plane](#)
- [About the software license](#)
- [Publications and Credits](#)
- [Ok, now what?](#)

More details on GNSS-SDR signal processing blocks:

- [Signal Source](#)

- Signal Conditioner
- Channel
 - Acquisition
 - Tracking
 - Decoding of the navigation message
- Observables
- Computation of Position, Velocity and Time

1.2 Overview

GNSS-SDR provides an interface to different suitable RF front-ends and implements all the receiver chain up to the navigation solution. Its design allows any kind of customization, including interchangeability of signal sources, signal processing algorithms, interoperability with other systems, output formats, and offers interfaces to all the intermediate signals, parameters and variables. The goal is to write efficient and truly reusable code, easy to read and maintain, with fewer bugs, and producing highly optimized executables in a variety of hardware platforms and operating systems. In that sense, the challenge consists of defining a gentle balance within level of abstraction and performance. GNSS-SDR runs in a personal computer and provides interfaces through USB and Ethernet buses to a variety of either commercially available or custom-made RF front-ends, adapting the processing algorithms to different sampling frequencies, intermediate frequencies and sample resolutions. This makes possible rapid prototyping of specific receivers intended, for instance, to geodetic applications, observation of the ionospheric impact on navigation signals, GNSS reflectometry, signal quality monitoring, or carrier-phase based navigation techniques.

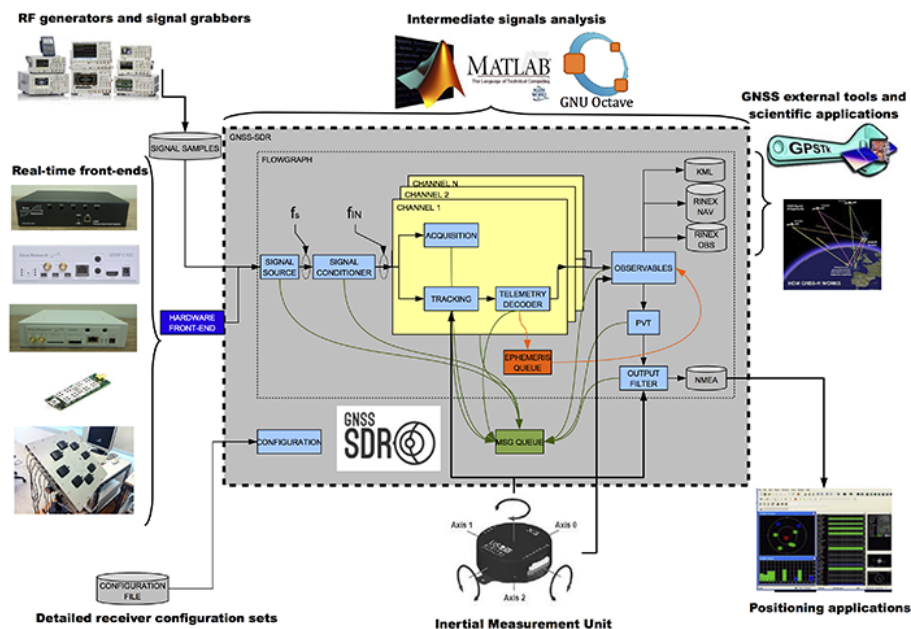


Figure 1.2 Overview

As signal inputs, it accepts:

- Raw data file captured with a data grabber (digitized at some intermediate frequency or directly at baseband).

- Any suitable RF configuration that can be driven by the Universal Software Radio Peripheral Hardware Driver ([UHD](#)). This includes all current and future [Ettus Research](#) products. The USRP1 + DBSRX 2.2 daughterboard is an example of working configuration for GPS L1 C/A and Galileo E1B and E1C signals.
- The [GN3S v2 USB dongle](#) (GN3S v3 might work with small modifications).
- Experimentally, with some [USB DVB-T dongles based on the Realtek RTL2832U chipset](#).
- For mathematical representations of the targeted signals, check out the [Signal model](#) page.

As outputs, it provides:

- Dump of intermediate signals (configurable by the user)
- The processing is logged at a system temporary folder (usually, `/tmp`)
- Observables in form of RINEX file (experimental)
- Navigation message data in form of RINEX file
- Position, Velocity and Time solution in KML format and NMEA

1.3 Building GNSS-SDR

In principle, GNSS-SDR can be built in any Unix-like system. In practice, it depends on being able to install all the required dependencies. See the [building guide](#) page for details about the project's dependencies and build process. Mainly, it consists on installing [GNU Radio](#) plus some few more libraries:

- [Armadillo](#), a C++ linear algebra library,
- [Boost](#), a set of free peer-reviewed portable C++ source libraries,
- [Gflags](#), a library that implements commandline flags processing,
- [Glog](#), a library that implements application-level logging,
- [Googletest](#), Google's framework for writing C++ tests,
- [Mako](#), a template library written in Python,
- [Matio](#), a MATLAB MAT File I/O Library,
- [Protocol Buffers](#), a language-neutral, platform-neutral extensible mechanism for serializing structured data,
- [PugiXML](#), a light-weight, simple and fast XML parser for C++ with XPath support,
- [Volk](#), a Vector-Optimized Library of Kernels which provides an abstraction of optimized math routines targeting several SIMD processors,

and, optionally,

- GNU Radio modules for hardware interface ([gr-uhd](#), [gr-osmosdr](#), [gr-iio](#)),
- [Benchmark](#), a library to benchmark code snippets,
- [Gperftools](#), which provides fast, multi-threaded malloc() and performance analysis tools.

After all dependencies are installed, clone the GNSS-SDR repository:

```
$ git clone https://github.com/gnss-sdr/gnss-sdr
```

This will create a folder named `gnss-sdr` with the following structure:

```
|-gnss-sdr
|---build      <- where gnss-sdr is built
|---cmake      <- CMake-related files
|---conf       <- Configuration files. Each file represents one receiver.
|---data       <- Populate this folder with your captured data.
|---docs       <- Contains documentation-related files
|---install    <- Executables
|---src        <- Source code folder
|-----algorithms
|-----PVT
|-----acquisition
|-----channel
|-----conditioner
|-----data_type_adapter
|-----input_filter
|-----libs
|-----observables
|-----resampler
|-----signal_source
|-----telemetry_decoder
|-----tracking
|-----core
|-----interfaces
|-----libs
|-----receiver
|-----system_parameters
|-----main
|-----tests
|-----utils    <- some utilities (e.g. Matlab scripts)
```

You are now ready to build GNSS-SDR by using **CMake** as building tool:

```
$ cd gnss-sdr/build
$ cmake ..
$ make
```

If everything goes well, three new executables will be created at `gnss-sdr/install`, namely `gnss-sdr`, `volk_gnsssdrr_profile` and `run_tests`. You can run them from that folder, but if you prefer to install `gnss-sdr` on your system and have it available anywhere else, do:

```
$ sudo make install
```

This will make a copy of the `conf/` folder into `/usr/local/share/gnss-sdr/conf` for your reference. We suggest to create a working directory at your preferred location and store your own configuration and data files there.

You can create the documentation by doing:

```
$ make doc
```

from the `gnss-sdr/build` folder. In both cases, **Doxygen** will generate HTML documentation that can be retrieved pointing your browser of preference to `gnss-sdr/docs/html/index.html`.

There are two more extra targets available. From the `gnss-sdr/build` folder:

```
$ make doc-clean
```

will remove the content of previously-generated documentation and, if a LaTeX installation is detected in your system,

```
$ make pdfmanual
```

will create a PDF manual at `gnss-sdr/docs/GNSS-SDR_manual.pdf`. Please note that the PDF generation requires some fonts to be installed on the host system. In Ubuntu, those fonts do not come by default. You can install them by doing:

```
$ sudo apt-get install texlive-fonts-recommended
```

and then run `cmake ../` and `make pdfmanual` again.

1.3.1 Debug and Release builds

By default, CMake will build the Release version, meaning that the compiler will generate a faster, optimized executable. This is the recommended build type when using a RF front-end and you need to attain real time. If working with a file (and thus without real-time constraints), you may want to obtain more information about the internals of the receiver, as well as more fine-grained logging. This can be done by building the Debug version, by doing:

```
$ cd gnss-sdr/build
$ cmake -DCMAKE_BUILD_TYPE=Debug ..
$ make
$ sudo make install
```

1.3.2 Updating GNSS-SDR

If you checked out GNSS-SDR some days ago, it is possible that some developer had updated files at the Git repository. You can update your local copy by doing:

```
$ git checkout next
$ git pull https://github.com/gnss-sdr/gnss-sdr next
```

Before rebuilding the source code, it is safe (and recommended) to remove the remainders of old builds:

```
$ cd gnss-sdr/build
$ sudo make uninstall
$ rm -rf *
```

You can also check [The Git Book](#) for more information about Git usage.

1.4 Using GNSS-SDR

With GNSS-SDR, you can define you own receiver, work with captured raw data or from a RF front-end, dump into files intermediate signals, or tune every single algorithm used in the [Signal Processing plane](#). All the configuration is done in a single file. Those configuration files reside at the `gnss-sdr/conf` folder. By default, the executable `gnss-sdr` will read the configuration available at `gnss-sdr/conf/gnss-sdr.conf`. You can edit that file to fit your needs, or even better, define a new `my_receiver.conf` file with your own configuration. This new receiver can be done by invoking `gnss-sdr` with the `-config_file` flag pointing to your configuration file:

```
$ gnss-sdr --config_file=../conf/my_receiver.conf
```

You can see a guide of available implementations at [the online documentation](#). That folder contains other working examples as well. If you have a working configuration and want to share it with others, please email it to the [GNSS-SDR developers mailing list](#) and we will be happy to upload it to the server.

You can use a single configuration file for processing different data files, specifying the file to be processed with the `-signal_source` flag:

```
$ gnss-sdr --config_file=../conf/my_receiver.conf --signal_source=../data/my_captured_data.dat
```

This will override the `SignalSource.filename` specified in the configuration file.

You can get a complete list of available commandline flags by doing:

```
$ gnss-sdr --help
```

1.5 Control plane

GNSS-SDR's main method initializes the logging library, processes the command line flags, if any, provided by the user and instantiates a [ControlThread](#) object. Its constructor reads the configuration file, creates a control queue and creates a flowgraph according to the configuration. Then, the program's main method calls the `run()` method of the instantiated object, an action that connects the flowgraph and starts running it. After that, and until a stop message is received, it reads control messages sent by the receiver's modules through a safe-thread queue and processes them. Finally, when a stop message is received, the main method executes the destructor of the [ControlThread](#) object, which deallocates memory, does other cleanup and exits the program.

The [GNSSFlowgraph](#) class is responsible for preparing the graph of blocks according to the configuration, running it, modifying it during run-time and stopping it. Blocks are identified by its role. This class knows which roles it has to instantiate and how to connect them. It relies on the configuration to get the correct instances of the roles it needs and then it applies the connections between GNU Radio blocks to make the graph ready to be started. The complexity related to managing the blocks and the data stream is handled by GNU Radio's `gr::top_block` class. [GNSSFlowgraph](#) wraps the `gr::top_block` instance so we can take advantage of the [GNSS block factory](#), the configuration system and the processing blocks. This class is also responsible for applying changes to the configuration of the flowgraph during run-time, dynamically reconfiguring channels: it selects the strategy for selecting satellites. This can range from a sequential search over all the satellites' ID to smarter approaches that determine what are the satellites most likely in-view based on rough estimations of the receiver position in order to avoid searching satellites in the other side of the Earth.

The Control Plane is in charge of creating a flowgraph according to the configuration and then managing the modules. Configuration allows users to define in an easy way their own custom receiver by specifying the flowgraph (type of signal source, number of channels, algorithms to be used for each channel and each module, strategies for satellite selection, type of output format, etc.). Since it is difficult to foresee what future module implementations will be needed in terms of configuration, we used a very simple approach that can be extended without a major impact in the code. This can be achieved by simply mapping the names of the variables in the modules with the names of the parameters in the configuration.

1.5.1 Configuration

Properties are passed around within the program using the [ConfigurationInterface](#) class. There are two implementations of this interface: [FileConfiguration](#) and [InMemoryConfiguration](#). [FileConfiguration](#) reads the properties (pairs of property name and value) from a file and stores them internally. [InMemoryConfiguration](#) does not read from a file; it remains empty after instantiation and property values and names are set using the `set` property method. [FileConfiguration](#) is intended to be used in the actual GNSS-SDR application whereas [InMemoryConfiguration](#) is intended to be used in tests to avoid file-dependency in the file system. Classes that need to read configuration parameters will receive instances of [ConfigurationInterface](#) from where they will fetch the values. For instance, parameters related to `SignalSource` should look like this:

```
SignalSource.parameter1=value1
SignalSource.parameter2=value2
```

The name of these parameters can be anything but one reserved word: `implementation`. This parameter indicates in its value the name of the class that has to be instantiated by the factory for that role. For instance, if our signal source is providing data already at baseband and thus we want to use the implementation [Pass_Through](#) for module [SignalConditioner](#), the corresponding line in the configuration file would be

```
SignalConditioner.implementation=Pass_Through
```

Since the configuration is just a set of property names and values without any meaning or syntax, the system is very versatile and easily extendable. Adding new properties to the system only implies modifications in the classes that will make use of these properties. In addition, the configuration files are not checked against any strict syntax so it is always in a correct status (as long as it contains pairs of property names and values in [INI format](#)).

1.5.2 GNSS block factory

Hence, the application defines a simple accessor class to fetch the configuration pairs of values and passes them to a factory class called [GNSSBlockFactory](#). This factory decides, according to the configuration, which class needs to be instantiated and which parameters should be passed to the constructor. Hence, the factory encapsulates the complexity of blocks' instantiation. With that approach, adding a new block that requires new parameters will be as simple as adding the block class and modifying the factory to be able to instantiate it. This loose coupling between the blocks' implementations and the syntax of the configuration enables extending the application capacities in a high degree. It also allows to produce fully customized receivers, for instance a testbed for acquisition algorithms, and to place observers at any point of the receiver chain.

1.6 Signal Processing plane

GNU Radio's class `gr::basic_block` is the abstract base class for all signal processing blocks, a bare abstraction of an entity that has a name and a set of inputs and outputs. It is never instantiated directly; rather, this is the abstract parent class of both `gr::hier_block2`, which is a recursive container that adds or removes processing or hierarchical blocks to the internal graph, and `gr::block`, which is the abstract base class for all the processing blocks.

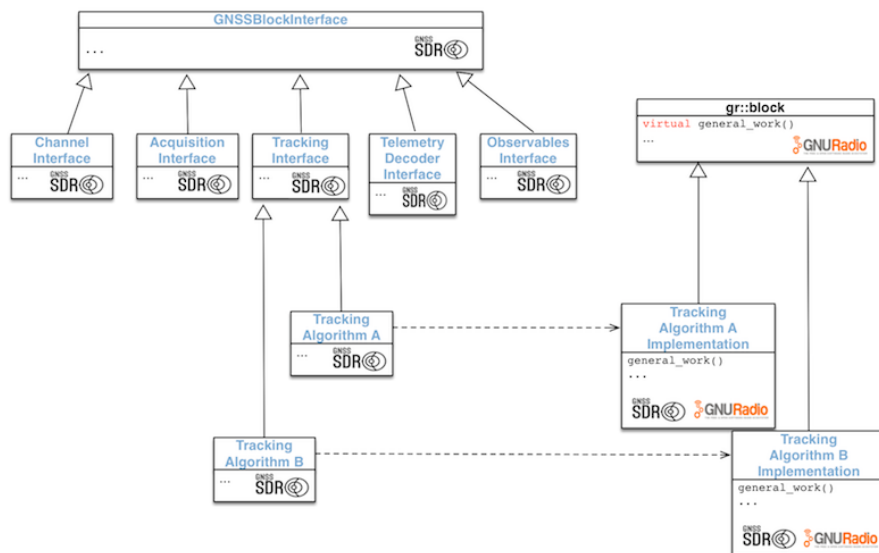


Figure 1.3 Class hierarchy of signal processing blocks

A signal processing flow is constructed by creating a tree of hierarchical blocks, which at any level may also contain terminal nodes that actually implement signal processing functions.

Class `gr::top_block` is the top-level hierarchical block representing a flowgraph. It defines GNU Radio run-time functions used during the execution of the program: `run()`, `start()`, `stop()`, `wait()`, etc. A subclass called [GNSSBlockInterface](#) is the common interface for all the GNSS-SDR modules. It defines pure virtual methods, that are required to be implemented by a derived class.

Subclassing [GNSSBlockInterface](#), we defined interfaces for the GNSS receiver blocks depicted in the figure above. This hierarchy provides the definition of different algorithms and different implementations, which will be instantiated according to the configuration. This strategy allows multiple implementations sharing a common interface, achieving the objective of decoupling interfaces from implementations: it defines a family of algorithms, encapsulates each one, and makes them interchangeable. Hence, we let the algorithm vary independently from the program that uses it.

1.6.1 Signal Source

The input of a software receiver are the raw bits that come out from the front-end's analog-to-digital converter (ADC). Those bits can be read from a file stored in the hard disk or directly in real-time from a hardware device through USB or Ethernet buses.

The Signal Source module is in charge of implementing the hardware driver, that is, the portion of the code that communicates with the RF front-end and receives the samples coming from the ADC. This communication is usually performed through USB or Ethernet buses. Since real-time processing requires a highly optimized implementation of the whole receiver, this module also allows to read samples from a file stored in a hard disk, and thus processing without time constraints. Relevant parameters of those samples are the intermediate frequency (or baseband I&Q components), the sampling rate and number of bits per sample, that must be specified by the user in the configuration file.

This module also performs bit-depth adaptation, since most of the existing RF front-ends provide samples quantized with 2 or 3 bits, while operations inside the processor are performed on 32- or 64-bit words, depending on its architecture. Although there are implementations of the most intensive computational processes (mainly correlation) that take advantage of specific data types and architectures for the sake of efficiency, the approach is processor-specific and hardly portable. We suggest to keep signal samples in standard data types and letting the compiler select the best library version (implemented using SIMD or any other processor-specific technology) of the required routines for a given processor.

Example: [FileSignalSource](#)

The user can configure the receiver for reading from a file, setting in the configuration file the data file location, sample format, and the sampling frequency and intermediate frequency at what the signal was originally captured.

```
##### SIGNAL_SOURCE CONFIG #####
SignalSource.implementation=File_Signal_Source
SignalSource.filename=/home/user/gnss-sdr/data/my_capture.dat
SignalSource.item_type=gr_complex
SignalSource.sampling_frequency=4000000 ; Sampling frequency in [Hz]
SignalSource.freq=1575420000 ; RF front-end center frequency in [Hz]
```

Example: [UhdSignalSource](#)

The user may prefer to use a UHD-compatible RF front-end and try real-time processing. For instance, for a USRP1 + DBSRX daughterboard, use:

```
##### SIGNAL_SOURCE CONFIG #####
SignalSource.implementation=UHD_Signal_Source
SignalSource.item_type=gr_complex
SignalSource.sampling_frequency=4000000 ; Sampling frequency in [Hz]
SignalSource.freq=1575420000 ; RF front-end center frequency in [Hz]
SignalSource.gain=60 ; Front-end gain in dB
SignalSource.subdevice=B:0 ; UHD subdevice specification (for USRP1 use A:0 or B:0)
```

Other examples are available at [gnss-sdr/conf](#).

1.6.2 Signal Conditioner

The signal conditioner is in charge of resampling the signal and delivering a reference sample rate to the downstream processing blocks, acting as a facade between the signal source and the synchronization channels, providing a simplified interface to the input signal. In case of multiband front-ends, this module would be in charge of providing a separated data stream for each band.

1.6.3 Channel

A channel encapsulates all signal processing devoted to a single satellite. Thus, it is a large composite object which encapsulates the [Acquisition](#), [Tracking](#) and [Decoding of the navigation message](#) modules. As a composite object, it can be treated as a single entity, meaning that it can be easily replicated. Since the number of channels is selectable by the user in the configuration file, this approach helps improving the scalability and maintainability of the receiver.

This module is also in charge of managing the interplay between acquisition and tracking. Acquisition can be initialized in several ways, depending on the prior information available (called cold start when the receiver has no information about its position nor the satellites almanac; warm start when a rough location and the approximate time of day are available, and the receiver has a recently recorded almanac broadcast; or hot start when the receiver was tracking a satellite and the signal line of sight broke for a short period of time, but the ephemeris and almanac data is still valid, or this information is provided by other means), and an acquisition process can finish deciding that the satellite is not present, that longer integration is needed in order to confirm the presence of the satellite, or declaring the satellite present. In the latter case, acquisition process should stop and trigger the tracking module with coarse estimations of the synchronization parameters.

The abstract class [ChannelInterface](#) represents an interface to a channel GNSS block. Check [Channel](#) for an actual implementation.

1.6.3.1 Acquisition

The first task of a GNSS receiver is to detect the presence or absence of in-view satellites. This is done by the acquisition system process, which also provides a coarse estimation of two signal parameters: the frequency shift with respect to the nominal IF frequency, and a delay term which allows the receiver to create a local code aligned with the incoming code. [AcquisitionInterface](#) is the common interface for all the acquisition algorithms and their corresponding implementations. Algorithms' interface, that may vary depending on the use of information external to the receiver, such as in Assisted GNSS, is defined in classes referred to as *adapters*. These adapters wrap the GNU Radio blocks interface into a compatible interface expected by [AcquisitionInterface](#). This allows the use of existing GNU Radio blocks derived from `gr::block`, and ensures that newly developed implementations will also be reusable in other GNU Radio-based applications. Moreover, it adds still another layer of abstraction, since each given acquisition algorithm can have different implementations (for instance using different numerical libraries). In such a way, implementations can be continuously improved without having any impact neither on the algorithm interface nor the general acquisition interface.

Check [GpsL1CaPcpsAcquisition](#) and [GalileoE1PcpsAmbiguousAcquisition](#) for examples of adapters from a Parallel Code Phase Search (PCPS) acquisition block, and `pcps_acquisition_cc` for an example of a block implementation. The source code of all the available acquisition algorithms is located at:

```
| -gnss-sdr
| ---src
| -----algorithms
| -----acquisition
| -----adapters          <- Adapters of the processing blocks to an AcquisitionInterface
| -----gnuradio_blocks   <- Signal processing blocks implementation
```

The user can select a given implementation for the algorithm to be used in each receiver channel, as well as their parameters, in the configuration file:

```
##### ACQUISITION GLOBAL CONFIG #####

#implementation: Acquisition algorithm selection for this channel:
Acquisition_1C.implementation=GPS_L1_CA_PCPS_Acquisition
#dump: Enable or disable the acquisition internal data file logging [true] or [false]
Acquisition_1C.dump=false
#filename: Log path and filename
Acquisition_1C.dump_filename=./acq_dump.dat
#item_type: Type and resolution for each of the signal samples. Use only gr_complex in this version.
```

```

Acquisition_1C.item_type=gr_complex
;#coherent_integration_time_ms: Signal block duration for the acquisition signal detection [ms]
Acquisition_1C.coherent_integration_time_ms=1
;#threshold: Acquisition threshold
Acquisition_1C.threshold=2.5
;#pfa: Acquisition false alarm probability. This option overrides the threshold option.
Acquisition_1C.pfa=0.0001
;#doppler_max: Maximum expected Doppler shift [Hz]
Acquisition_1C.doppler_max=5000
;#doppler_step: Doppler step in the grid search [Hz]
Acquisition_1C.doppler_step=250

```

1.6.3.2 Tracking

When a satellite is declared present, the parameters estimated by the acquisition module are then fed to the receiver tracking module, which represents the second stage of the signal processing unit, aiming to perform a local search for accurate estimates of code delay and carrier phase, and following their eventual variations.

Again, a class hierarchy consisting of a [TrackingInterface](#) class and subclasses implementing algorithms provides a way of testing different approaches, with full access to their parameters. Check [GpsL1CaDlIPllTracking](#) or [GalileoE1DlIPllVemlTracking](#) for examples of adapters, and [Gps_L1_Ca_Dll_Pll_Tracking_cc](#) for an example of a signal processing block implementation. There are also available some useful classes and functions for signal tracking; take a look at Correlator, [lock_detectors.h](#), [tracking_discriminators.h](#) or [tracking_2nd_DLL_filter.h](#).

The source code of all the available tracking algorithms is located at:

```

|-gnss-sdr
|---src
|----algorithms
|-----tracking
|-----adapters          <- Adapters of the processing blocks to a TrackingInterface
|-----gnuradio_blocks    <- Signal processing blocks implementation
|-----libs               <- libraries of tracking objects (e.g. correlators, discriminators, and so on)

```

The user can select a given implementation for the algorithm to be used in all the tracking blocks, as well as its parameters, in the configuration file:

```

;##### TRACKING GLOBAL CONFIG #####

;#implementation: Selected tracking algorithm
Tracking_1C.implementation=GPS_L1_CA_DLL_PLL_Tracking
;#item_type: Type and resolution for each of the signal samples.
Tracking_1C.item_type=gr_complex

;#dump: Enable or disable the Tracking internal binary data file logging [true] or [false]
Tracking_1C.dump=false

;#dump_filename: Log path and filename. Notice that the tracking channel will add "x.dat" where x is the channel
Tracking_1C.dump_filename=./tracking_ch_

;#pll_bw_hz: PLL loop filter bandwidth [Hz]
Tracking_1C.pll_bw_hz=50.0;

;#dll_bw_hz: DLL loop filter bandwidth [Hz]
Tracking_1C.dll_bw_hz=2.0;

;#fll_bw_hz: FLL loop filter bandwidth [Hz]
Tracking_1C.fll_bw_hz=10.0;

Tracking_1C.pll_filter_order=3 ; PLL loop filter order [2] or [3]
Tracking_1C.dll_filter_order=2 ; DLL loop filter order [1], [2] or [3]

;#early_late_space_chips: correlator early-late space [chips].
Tracking_1C.early_late_space_chips=0.5;

```


1.6.3.3 Decoding of the navigation message

Most of GNSS signal links are modulated by a navigation message containing the time the message was transmitted, orbital parameters of satellites (also known as ephemeris) and an almanac (information about the general system health, rough orbits of all satellites in the network as well as data related to error correction). Navigation data bits are structured in words, pages, subframes, frames and superframes. Sometimes, bits corresponding to a single parameter are spread over different words, and values extracted from different frames are required for proper decoding. Some words are for synchronization purposes, others for error control and others contain actual information. There are also error control mechanisms, from parity checks to forward error correction (FEC) encoding and interleaving, depending on the system.

The common interface is [TelemetryDecoderInterface](#). Check [GpsL1CaTelemetryDecoder](#) for an example of the GPS L1 NAV message decoding adapter, and `gps_l1_ca_telemetry_decoder_cc` for an actual implementation of a signal processing block. Configuration example:

```
;##### TELEMETRY DECODER CONFIG #####
TelemetryDecoder_1C.implementation=GPS_L1_CA_Telemetry_Decoder
TelemetryDecoder_1C.dump=false
```

See the [Reference Documents](#) for more information about the signal format.

1.6.4 Observables

GNSS systems provide different kinds of observations. The most commonly used are the code observations, also called pseudoranges. The *pseudo* comes from the fact that on the receiver side the clock error is unknown and thus the measurement is not a pure range observation. High accuracy applications also use the carrier phase observations, which are based on measuring the difference between the carrier phase transmitted by the GNSS satellites and the phase of the carrier generated in the receiver. Both observables are computed from the outputs of the tracking module and the decoding of the navigation message. This module collects all the data provided by every tracked channel, aligns all received data into a coherent set, and computes the observables.

The common interface is [ObservablesInterface](#).

Configuration example:

```
;##### OBSERVABLES CONFIG #####
Observables.implementation=Hybrid_Observables

;#dump: Enable or disable the Observables internal binary data file logging [true] or [false]
Observables.dump=false

;#dump_filename: Log path and filename.
Observables.dump_filename=./observables.dat
```

1.6.5 Computation of Position, Velocity and Time

Although data processing for obtaining high-accuracy PVT solutions is out of the scope of GNSS-SDR, we provide a module that can compute a simple least square solution and leaves room for more sophisticated positioning methods. The integration with libraries and software tools that are able to deal with multi-constellation data such as [GPSTk](#) or [gLAB](#) appears as a viable solution for high performance, completely customizable GNSS receivers.

The common interface is [PvtInterface](#). For instance, in order to use the implementation `RTKLIB_PVT`, add to the configuration file:

```

;##### PVT CONFIG #####
PVT.implementation=RTKLIB_PVT

;#nmea_dump_filename: NMEA log path and filename
PVT.nmea_dump_filename=./gnss_sdr_pvt.nmea;

;#flag_nmea_tty_port: Enable or disable the NMEA log to a serial TTY port (Can be used with real hardware or v
PVT.flag_nmea_tty_port=true;

;#nmea_dump_devname: serial device descriptor for NMEA logging
PVT.nmea_dump_devname=/dev/pts/4

;#dump: Enable or disable the PVT internal binary data file logging [true] or [false]
PVT.dump=false

```

This implementation allows tuning of the following parameters:

```

PVT.implementation=RTKLIB_PVT
PVT.positioning_mode=Single          ; options: Single, Static, Kinematic, PPP_Static, PPP_Kinematic
PVT.iono_model=Broadcast             ; options: OFF, Broadcast
PVT.trop_model=Saastamoinen          ; options: OFF, Saastamoinen
PVT.rinex_version=2                  ; options: 2 or 3
PVT.output_rate_ms=100               ; Period in [ms] between two PVT outputs
PVT.display_rate_ms=500              ; Position console print (std::out) interval [ms].
PVT.nmea_dump_filename=./gnss_sdr_pvt.nmea ; NMEA log path and filename
PVT.flag_nmea_tty_port=false         ; Enables the NMEA log to a serial TTY port
PVT.nmea_dump_devname=/dev/pts/4     ; serial device descriptor for NMEA logging
PVT.flag_rtcn_server=true            ; Enables or disables a TCP/IP server dispatching RTCM messages
PVT.flag_rtcn_tty_port=false         ; Enables the RTCM log to a serial TTY port
PVT.rtcn_dump_devname=/dev/pts/1     ; serial device descriptor for RTCM logging
PVT.rtcn_tcp_port=2101
PVT.rtcn_MT1019_rate_ms=5000
PVT.rtcn_MT1045_rate_ms=5000
PVT.rtcn_MT1097_rate_ms=1000
PVT.rtcn_MT1077_rate_ms=1000

```

1.7 About the software license

GNSS-SDR is released under the [General Public License \(GPL\) v3](#), thus securing practical usability, inspection, and continuous improvement by the research community, allowing the discussion based on tangible code and the analysis of results obtained with real signals. The GPL implies that:

- Copies may be distributed free of charge or for money, but the source code has to be shipped or provided free of charge (or at cost price) on demand. The receiver of the source code has the same rights meaning he can share copies free of charge or resell.
- The licensed material may be analyzed or modified.
- Modified material may be distributed under the same licensing terms but **do not** have to be distributed.

That means that modifications only have to be made available to the public if distribution happens. So it is perfectly fine to take the GNSS-SDR source code, modify it heavily and use it in a not distributed application / library. This is how companies like Google can run their own patched versions of Linux for example.

But what this also means is that non-GPL code cannot use GPL code. This means that you cannot modify / use GNSS-SDR, blend it with non-GPL code, and make money with the resulting software. You cannot distribute the resulting software under a non-disclosure agreement or contract. Distributors under the GPL also grant a license for any of their patents practiced by the software, to practice those patents in GPL software. You can sell a device that runs with GNSS-SDR, but if you distribute the code, it has to remain under GPL.

1.8 Publications and Credits

If you use GNSS-SDR to produce a research paper or Thesis, we would appreciate if you reference any of these articles to credit the GNSS-SDR project:

- C. Fernández-Prades, J. Arribas, L. Esteve, D. Pubill, P. Closas, *An Open Source Galileo E1 Software Receiver*, in Proc. of the 6th ESA Workshop on Satellite Navigation Technologies (NAVITEC 2012), ESTEC, Noordwijk, The Netherlands, Dec. 2012.
- J. Arribas, *GNSS Array-based Acquisition: Theory and Implementation*, PhD Thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, June 2012.
- C. Fernández-Prades, J. Arribas, P. Closas, C. Avilés, and L. Esteve, *GNSS-SDR: an open source tool for researchers and developers*, in Proc. of the ION GNSS 2011 Conference, Portland, Oregon, Sept. 19-23, 2011.
- C. Fernández-Prades, C. Avilés, L. Esteve, J. Arribas, and P. Closas, *Design patterns for GNSS software receivers*, in Proc. of the 5th ESA Workshop on Satellite Navigation Technologies (NAVITEC'2010), ESTEC, Noordwijk, The Netherlands, Dec. 2010. DOI:10.1109/NAVITEC.2010.5707981

For LaTeX users, these are the BibTeX cites for your convenience:

```
@INPROCEEDINGS{GNSS-SDR12,
  author = {C.~{Fern}\{a\}ndez--Prades} and J.~Arribas and L.~Esteve and D.~Pubill and P.~Closas},
  title = {An Open Source {G}alileo {E1} Software Receiver},
  booktitle = {Proc. of the 6th ESA Workshop on Satellite Navigation Technologies (NAVITEC'2012)},
  year = {2012},
  address = {ESTEC, Noordwijk, The Netherlands},
  month = {Dec.} }

@PHDTHESIS{Arribas12,
  author = {J.~Arribas},
  title = {{GNSS} Array-based Acquisition: Theory and Implementation},
  school = {Universitat Polit\`{e}cnica de Catalunya},
  year = {2012},
  address = {Barcelona, Spain},
  month = {June} }

@INPROCEEDINGS{GNSS-SDR11,
  AUTHOR = {C.~{Fern}\{a\}ndez--Prades} and J.~Arribas and P.~Closas and C.~Avil\{e\}s and L.~Esteve},
  TITLE = {{GNSS-SDR}: An Open Source Tool For Researchers and Developers},
  BOOKTITLE = {Proc. of the ION GNSS 2011 Conference},
  YEAR = {2011},
  address = {Portland, Oregon},
  month = {Sept.} }

@INPROCEEDINGS{GNSS-SDR10,
  AUTHOR = {C.~{Fern}\{a\}ndez--Prades} and C.~Avil\{e\}s and L.~Esteve and J.~Arribas and P.~Closas},
  TITLE = {Design patterns for {GNSS} software receivers},
  BOOKTITLE = {Proc. of the 5th ESA Workshop on Satellite Navigation Technologies (NAVITEC'2010)},
  YEAR = {2010},
  address = {ESTEC, Noordwijk, The Netherlands},
  month = {Dec.},
  note = {doi:10.1109/NAVITEC.2010.5707981} }
```

More papers related to GNSS-SDR are available at the [publications page](#).

1.9 Ok, now what?

In order to start using GNSS-SDR, you may want to populate `gnss-sdr/data` folder (or anywhere else on your system) with raw data files. By "raw data" we mean the output of a Radio Frequency front-end's Analog-to-Digital converter. GNSS-SDR needs signal samples already in baseband or in passband, at a suitable intermediate frequency (on the order of MHz). Prepare your configuration file, and then you are ready for going to the `gnss-sdr/install` folder, running `./gnss-sdr`, and see how the file is processed. Please ask the Developer Team for a signal sample if you need one, and they will do their best ;-)

Another interesting option is working in real-time with a RF front-end. We provide drivers for UHD-compatible hardware (see [Signal Source](#)), for the GN3S v2 USB dongle and for some DVB-T USB dongles. Start with a low number of channels and then increase it in order to test how many channels your processor can handle in real-time.

You can find more information at the [GNSS-SDR Documentation page](#) or directly asking to the [GNSS-SDR Developers mailing list](#).

You are also very welcome to contribute to the project, there are many ways to [participate in GNSS-SDR](#). If you need some special feature not yet implemented, the Developer Team would love to be hired for developing it. Please do not hesitate to [contact them](#).

Enjoy GNSS-SDR!

The Developer Team.

Chapter 2

Reference Documents

2.1 Interface Control Documents

2.1.1 GPS

All the current GPS Interface Control Documents can be downloaded from [GPS.gov](https://www.gps.gov), the official U.S. Government webpage for GPS.

- GPS L1 and L2C: Global Positioning System Directorate, **Interface Specification IS-GPS-200 Revision K**. March, 2019.
- GPS L1C (available with first Block III launch): Global Positioning System Directorate, **Interface Specification IS-GPS-800 Revision F**. March, 2019.
- GPS L5 (first Block IIF satellite launched on May, 2010): Global Positioning System Directorate, **Interface Specification IS-GPS-705 Revision F**. March, 2019.

2.1.2 GLONASS

Official GLONASS webpage: [Information-analytical centre official website](https://www.glonass-iac.ru/).

- Standard Accuracy (ST) signals at L1 and L2: Russian Institute of Space Device Engineering, Global Navigation Satellite System GLONASS. **Interface Control Document. Navigational radiosignal in bands L1, L2. Edition 5.1**, Moscow, Russia, 2008
- **GLONASS Interface Control Document. Open CDMA navigational radio signal in L1 band. Edition 1.0 (in Russian)**. Russian Space Systems OJSC. 2016.
- **GLONASS Interface Control Document. Open CDMA navigational radio signal in L2 band. Edition 1.0 (in Russian)**. Russian Space Systems OJSC. 2016.
- **GLONASS Interface Control Document. Open CDMA navigational radio signal in L3 band. Edition 1.0 (in Russian)**. Russian Space Systems OJSC. 2016.

2.1.3 Galileo

Check the [Galileo website of the European Global Navigation Satellite Systems Agency \(GSA\)](#) and the [Galileo website of the European Space Agency](#). There is a website with [Galileo constellation status information](#) from the GSA.

- Galileo E5, E6, and E1: European GNSS (Galileo) Open Service. **Signal In Space Interface Control Document. Ref: OS SIS ICD, Issue 1.3**, European Commission, Dec. 2016.

The European Commission is granting free access to the technical information on the future Galileo open service signal, i.e. the specifications manufacturers and developers need to process data received from satellites. This document informs receiver manufacturers, application developers and service providers on how to use the future Galileo system and what they can expect in terms of performance.

2.1.4 BeiDou

Official webpage at beidou.gov.cn

- **BeiDou Navigation Satellite System Signal In Space Interface Control Document Open Service Signal B1I (Version 3.0)**. China Satellite Navigation Office, Feb. 2019.
- **BeiDou Navigation Satellite System Signal In Space Interface Control Document Open Service Signal B1C (Version 1.0)**. China Satellite Navigation Office, Jun. 2018.
- **BeiDou Navigation Satellite System Signal In Space Interface Control Document Open Service Signal B3I (Version 1.0)**. China Satellite Navigation Office, Feb. 2018.
- **BeiDou Navigation Satellite System Signal In Space Interface Control Document Open Service Signal B2a (Version 1.0)**. China Satellite Navigation Office, Dec. 2017.
- **BeiDou Navigation Satellite System Signal In Space Interface Control Document Open Service Signal (Version 2.1)**. China Satellite Navigation Office, December 2016.

2.1.5 Satellite Based Augmentation Systems (SBAS)

- **Minimum Operational Performance Standards for Global Positioning System/Wide Area Augmentation System Airborne Equipment, DO-229D**, RTCA, Washington, DC, Dec. 13, 2006. The 'RTCA MOPS DO-229D - appendix A' is the reference standard for WAAS/EGNOS application development. RTCA is an advisory committee of the US federal government, and issues standards for civil airborne equipment, among other duties. One such standard is MOPS 229D (Minimum Operational Performance Standards for Global Positioning System/Wide Area Augmentation System Airborne Equipment version D), which describes the implementation of satellite-based augmentation services (SBAS) for receivers designed for civil aviation use. An annex to DO229D contains the specifications for the SBAS signal and message. The RTCA provides regular updates to these standards. MOPS 229D is available for a fee from the [RTCA website](#).
- **Global Positioning System Wide Area Augmentation System (WAAS) Performance Standard, 1st Edition**, Department of Transportation and Federal Aviation Administration, Oct. 31, 2008. This document defines the levels of performance the U.S. Government makes available to users of the GPS SPS augmented by the Wide Area Augmentation System.
- **EGNOS Data Access Service (EDAS) Service Definition Document. Revision 2.2**, European GNSS Agency (GSA), June, 2019. This is a complementary document to the RTCA DO229D, mentioned above. It describes the scope of services provided by the EGNOS EDAS Service to be used by end-users or Application Specific Service Providers. It details the general conditions relating to the use of the EGNOS service, a technical description of the Signal-in-Space (SIS), the reference receiver, environmental conditions, the service performance achieved and aspects relating to service provision.
- **EGNOS Safety of Life Service Definition Document. Revision 3.3**, European GNSS Agency (GSA), Mar, 2019. The EGNOS Safety of Life (SoL) Service is provided openly and is freely accessible without any direct charge and is tailored to safety-critical transport applications in various domains, in particular for aviation applications. The service is thus compliant with the aviation APV-I (Approach with Vertical Guidance) requirements, as defined by ICAO in Annex 10, but may support also applications in other SoL domains.
- **EGNOS Open Service Service Definition Document. Revision 2.3**, European GNSS Agency (GSA), Sep., 2017.

More information about EGNOS can be found through the [EGNOS Portal](#).

2.2 Other Standards

2.2.1 RINEX

The final output of a navigation receiver is usually its position, speed or other related physical quantities. However, the calculation of those quantities are based on a series of measurements from one or more satellite constellations. Although receivers calculate positions in real time, in many cases it is interesting to store intermediate measures for later post-processing. RINEX is the standard format that allows the management and disposal of the measures generated by a receiver, as well as their off-line processing by a multitude of applications.

- The most common version at present is **RINEX: The Receiver Independent Exchange Format Version 2.12**, which enables storage of measurements from pseudorange, carrier-phase and Doppler systems for GPS, GLONASS, Galileo along with data from EGNOS and WAAS satellite based augmentation systems (SBAS).
- The most recent version is **RINEX: The Receiver Independent Exchange Format Version 3.03** published in July, 2015. It includes Galileo and improves the handling of multi-constellation data files.
- There is also available the **RINEX Extensions to Handle Clock Information**, published in September, 2010.

2.2.2 NMEA

The [National Marine Electronics Association](#) released the NMEA 0183 Interface Standard, which defines electrical signal requirements, data transmission protocol and time, and specific sentence formats for a 4800-baud serial data bus. The standard is [available for purchase](#).

2.2.3 KML

KML is an XML language focused on geographic visualization, including annotation of maps and images. Geographic visualization includes not only the presentation of graphical data on the globe, but also the control of the user's navigation in the sense of where to go and where to look. Google submitted KML (formerly Keyhole Markup Language) to the Open Geospatial Consortium (OGC) to be evolved within the OGC consensus process with the following goal: KML Version 2.2 has been adopted as an OGC implementation standard.

- Open Geospatial Consortium, Inc., [OGC KML Version 2.2.0](#), April 2008.

2.2.4 C++ Standards

The C++ programming language is standardized by the International Organization for Standardization (ISO), with the latest standard version ratified and published by ISO in December 2017 as ISO/IEC 14882:2017 (informally known as C++17). The list of supported C++ standards (the highest available is automatically selected by the CMake script):

- **Draft C++20:** Check the [C++ standard draft sources at GitHub](#).
- **C++17:** The current ISO C++ standard is officially known as *ISO International Standard ISO/IEC 14882:2017 – Programming languages – C++*. You can get it from [ISO](#), [IEC](#) or [ANSI](#). The closest free working document available is [N4659](#).
- **C++14:** The former ISO C++ standard was officially known as *ISO International Standard ISO/IEC 14882:2014 – Programming languages – C++*. You can get it from [ISO](#) or [ANSI](#). The closest free working document available is [N4296](#).
- **C++11:** An older ISO C++ standard was ISO/IEC 14882:2011. You can get it from [ISO](#). The closest free working document available is [N3337](#).

2.2.5 Positioning protocols in wireless communication networks

Cellular industry location standards first appeared in the late 1990s, with the [3rd generation partnership project \(3GPP\)](#) radio resource location services protocol (RRLP) technical specification 44.031 positioning protocol for GSM networks. Today, RRLP is the de facto standardized protocol to carry GNSS assistance data to GNSS-enabled mobile devices, and the term "3GPP specification" now covers all GSM (including GPRS and EDGE), W-CDMA and LTE (including LTE-A) specifications. Precisely, the label "LTE-A" is applied to networks compliant with LTE Release 10 and beyond, which fulfill the requirements issued by the [International Telecommunication Union Radiocommunication Sector \(ITU-R\)](https://www.itu.int/en/ITU-R/Pages/default.aspx?target=_blank) in the global standard for international mobile telecommunications (IMT Advanced, also referred to as 4G) access technologies.

Control plane protocols:

- Radio Resource LCS Protocol (RRLP): [3GPP Technical Specification 44.031](#).

- LTE Positioning Protocol (LPP): [3GPP Technical Specification 36.355](#).

User plane protocols:

- Open Mobile Alliance (OMA), [Secure User Plane Location Architecture Version 1 \(SUPL 1.0\)](#), June 2007.
- Open Mobile Alliance (OMA), [Secure User Plane Location Architecture Version 2 \(SUPL 2.0\)](#), April 2012.

LTE Release 9 introduced extension hooks in LPP messages, so that the bodies external to 3GPP could extend the LPP feature set. OMA LPP extensions (LPPE), supported in SUPL 3.0, build on top of the 3GPP LPP reusing its procedures and data types. Check the [OMA Specifications webpage](#) for updated information about LPP Extensions (LPPE) Specification.

- The [OMA Mobile Location Protocol \(MLP\) V3.5](#) is an application-level protocol for getting the position of mobile stations (mobile phones, wireless personal digital assistants, etc.) independent of underlying network technology. The MLP serves as the interface between a Location Server and a Location Services (LCS) Client. This specification defines the core set of operations that a Location Server should be able to perform.

Chapter 3

Signal model

3.1 GNSS signal model

This page describes signals transmitted by GNSS space vehicles. Signal models are mathematical representations of the electromagnetic waves that are exciting the receiver's antenna. The current induced by those waves is then amplified, filtered and downconverted to a suitable frequency (can be at some intermediate frequency or directly to baseband), and then converted to 0s and 1s by the Analog-to-Digital Converter (ADC). That is the job of the Radio Frequency front-end, which at its output delivers a stream of digital samples. Those samples constitute the input of a software receiver, so for GNSS-SDR the signal models described below can be seen as *the rules of the game*.

GNSS' space vehicles are modern versions of lighthouses, but with better visibility. Each satellite is a reference point, and if we know our distance to several reference points, we can compute our location, just as mariners do when they see a couple of lighthouses. For each in-view satellite i of system s , we can write:

$$\rho_i = \sqrt{(x_i^{\text{Tx}} - x)^2 + (y_i^{\text{Tx}} - y)^2 + (z_i^{\text{Tx}} - z)^2} + c\Delta t^{(s)} + \sigma_e, \quad (3.1)$$

where $(x_i^{\text{Tx}}, y_i^{\text{Tx}}, z_i^{\text{Tx}})$ is the satellite's position (known from the navigation message), (x, y, z) the receiver's position, and σ_e gathers other sources of error. Since the receiver needs to estimate its own 3D position (three spatial unknowns) and its clock deviation with respect to the satellites' time basis, at least $3 + N_s$ satellites must be seen by the receiver at the same time, where N_s is the number of different navigation systems available (in-view) at a given time. Each received satellite signal, once synchronized and demodulated at the receiver, defines one equation such as the one defined above, forming a set of nonlinear equations that can be solved algebraically by means of the [Bancroft algorithm](#) or numerically, resorting to multidimensional Newton-Raphson and weighted least square methods. When *a priori* information is added we resort to Bayesian estimation, a problem that can be solved recursively by a Kalman filter or any of its variants. The problem can be further expanded by adding other unknowns (for instance, parameters of ionospheric and tropospheric models), sources of information from other systems, mapping information, and even motion models of the receiver. In the design of multi-constellation GNSS receivers, the vector of unknowns can also include the receiver clock offset with respect to each system in order to take advantage of a higher number of in-view satellites and using them jointly in the navigation solution, therefore increasing accuracy.

The [analytic representation](#) of a signal received from a GNSS satellite can be generically expressed as

$$r(t) = \alpha(t)s_T(t - \tau(t))e^{-j2\pi f_d(t)}e^{j2\pi f_c t} + n(t), \quad (3.2)$$

where $\alpha(t)$ is the amplitude, $s_T(t)$ is the complex baseband transmitted signal, $\tau(t)$ is the time-varying delay, $f_d(t) = f_c\tau(t)$ is the Doppler shift, f_c is the carrier frequency, and $n(t)$ is a noise term. These signals arrive to the Earth's surface at extremely low power (e.g. -158.5 dBW for GPS L1 C/A-code, -157 dBW for Galileo E1), well below the noise floor. In order to estimate its distances to satellites, the receiver must correlate time-aligned replicas of the corresponding pseudorandom code with the incoming signal, in a process called *despreading* that

provides processing gain only to the signal of interest. After a coarse and fine estimation stages of the synchronization parameters (usually known as acquisition and tracking, respectively), signal processing output is in form of *observables*:

i) the pseudorange (code) measurement, equivalent to the difference of the time of reception (expressed in the time frame of the receiver) and the time of transmission (expressed in the time frame of the satellite) of a distinct satellite signal; and optionally

ii) the carrier-phase measurement, actually being a measurement on the beat frequency between the received carrier of the satellite signal and a receiver-generated reference frequency. Carrier phase measurements are ambiguous, in the sense that the integer number of carrier wavelengths between satellite and the receiver's antenna is unknown. Techniques such as **Least-square AMBiguity Decorrelation Approach (LAMBDA)** or Multi Carrier Ambiguity Resolution (MCAR) can be applied to resolve such ambiguity and provide an accurate estimation of the distance between the satellite and the receiver.

Then, depending on the required accuracy, the navigation solution can range from pseudorange-only, computationally low demanding, and limited accuracy least squares methods to sophisticated combinations of code and phase observables at different frequencies for high demanding applications such as surveying, geodesy, and geophysics.

Next sections provide brief descriptions of the space segment of different GNSSs and their broadcast signal structures accessible by civilians.

3.1.1 Global Positioning System (GPS) signal in space

The Global Positioning System (GPS) is a space-based radio-navigation system owned by the United States Government (USG) and operated by the United States Air Force (USAF). GPS provides positioning and timing services to military and civilian users on a continuous, worldwide basis. Two GPS services are provided: the Precise Positioning Service (PPS), available primarily to the military of the United States and its allies, and the Standard Positioning Service (SPS) open to civilian users.

- **GPS L1**. Defined at **Interface Specification IS-GPS-200 Revision K**, this band is centered at $f_{\text{GPS L1}} = 1575.42$ MHz. The complex baseband transmitted signal can be written as

$$s_T^{(\text{GPS L1})}(t) = e_{L1I}(t) + je_{L1Q}(t), \quad (3.3)$$

with

$$e_{L1I}(t) = \sum_{l=-\infty}^{\infty} D_{\text{NAV}}[l]_{204600} \oplus C_{P(Y)}[l]_{L_{P(Y)}} p(t - lT_{c,P(Y)}), \quad (3.4)$$

$$e_{L1Q}(t) = \sum_{l=-\infty}^{\infty} D_{\text{NAV}}[l]_{20460} \oplus C_{C/A}[l]_{1023} p(t - lT_{c,C/A}), \quad (3.5)$$

where \oplus is the exclusive-or operation (modulo-2 addition), $|l|_L$ means l modulo L , $[l]_L$ means the integer part of $\frac{l}{L}$, D_{NAV} is the GPS navigation message bit sequence, transmitted at 50 bps, $T_{c,P(Y)} = \frac{1}{10.23} \mu\text{s}$, $T_{c,C/A} = \frac{1}{1.023} \mu\text{s}$, $L_{P(Y)} = 6.1871 \cdot 10^{12}$, and $p(t)$ is a rectangular pulse of a chip-period duration centered at $t = 0$ and filtered at the transmitter. According to the chip rate, the binary phase-shift keying modulations in the equations above are denoted as BPSK(10) and BPSK(1), respectively. The precision P codes (named Y codes whenever the anti-spoofing mode is activated, encrypting the code and thus denying non-U.S. military users) are sequences of 7 days in length. Regarding the modernization plans for GPS, it is worthwhile to mention that there is a new civilian-use signal planned, called L1C and defined at **Interface Specification IS-GPS-800 Revision F**, to be broadcast on the same L1 frequency that currently contains the C/A signal. The L1C will be available with first Block III launch, currently scheduled for 2013. The implementation will provide C/A code to ensure backward compatibility.

- **GPS L2C.** Defined at **Interface Specification IS-GPS-200 Revision K**, is only available on Block IIR-M and subsequent satellite blocks. Centered at $f_{\text{GPS L2}} = 1227.60$ MHz, the signal structure is the same than in (eq:GPSL1}), with the precision code in the In-phase component, just as in (eq:L1CAI}) but with an optional presence of the navigation message D_{NAV} . For the Quadrature-phase component, three options are defined:

$$e_{L2CQ}(t) = \sum_{l=-\infty}^{\infty} D_{\text{CNAV}}[l]_{10230} \oplus \left(C_{\text{CL}}[l]_{L_{\text{CL}}} p_{1/2}(t - lT_{c,L2C}) + \right. \quad (3.6)$$

$$\left. + C_{\text{CM}}[l]_{L_{\text{CM}}} p_{1/2}\left(t - \left(l + \frac{3}{4}\right) T_{c,L2C}\right) \right), \quad (3.7)$$

$$e_{L2CQ}(t) = \sum_{l=-\infty}^{\infty} D_{\text{NAV}}[l]_{20460} \oplus C_{\text{C/A}}[l]_{1023} p(t - lT_{c,C/A}), \text{ or} \quad (3.8)$$

$$e_{L2CQ}(t) = \sum_{l=-\infty}^{\infty} C_{\text{C/A}}[l]_{1023} p(t - lT_{c,C/A}), \quad (3.9)$$

where $T_{c,L2C} = \frac{1}{511.5}$ ms and $p_{1/2}(t)$ is a rectangular pulse of half chip-period duration, thus time-multiplexing both codes. The civilian long code C_{CL} is $L_{\text{CL}} = 767250$ chips long, repeating every 1.5 s, while the civilian moderate code C_{CM} is $L_{\text{CL}} = 10230$ chips long and its repeats every 20 ms. The CNAV data is an upgraded version of the original NAV navigation message, containing higher precision representation and nominally more accurate data than the NAV data. It is transmitted at 25 bps with forward error correction (FEC) encoding, resulting in 50 sps.

- **GPS L5.** The GPS L5 link, defined at **Interface Specification IS-GPS-705 Revision F**, is only available in Block IIF (first satellite launched on May, 2010) and subsequent satellite blocks. Centered at $f_{\text{GPS L5}} = 1176.45$ MHz, this signal in space can be written as:

$$s_T^{(\text{GPS L5})}(t) = e_{L5I}(t) + je_{L5Q}(t), \quad (3.10)$$

$$e_{L5I}(t) = \sum_{m=-\infty}^{+\infty} C_{nh10}[m]_{10} \oplus D_{\text{CNAV}}[m]_{10} \oplus \sum_{l=1}^{102300} C_{L5I}[l]_{10230} p(t - mT_{c,nh} - lT_{c,L5}), \quad (3.11)$$

$$e_{L5Q}(t) = \sum_{m=-\infty}^{+\infty} C_{nh20}[m]_{20} \oplus \sum_{l=1}^{102300} C_{L5Q}[l]_{10230} \cdot p(t - mT_{c,nh} - lT_{c,L5}), \quad (3.12)$$

where $T_{c,nh} = 1$ ms and $T_{c,L5} = \frac{1}{10.23}$ μ s, thus defining a BPSK(10) modulation. Both L5I and L5Q contain synchronization sequences. {itemize}

3.1.2 GLONASS signal in space

The nominal baseline constellation of the Russian Federation's Global Navigation Satellite System (GLONASS) comprises 24 GLONASS-M satellites that are uniformly deployed in three roughly circular orbital planes at an inclination of 64.8° to the equator. The altitude of the orbit is 19,100 km. The orbit period of each satellite is 11 hours, 15 minutes, and 45 seconds. The orbital planes are separated by 120° right ascension of the ascending node. Eight satellites are equally spaced in each plane with 45° argument of latitude. Moreover, the orbital planes have an argument of latitude displacement of 15° relative to each other.

GLONASS civil signal-in-space is defined at **Interface Control Document. Navigational radiosignal in bands L1, L2. Edition 5.1**. This system makes use of a frequency-division multiple access (FDMA) signal structure, transmitting in two bands: $f_{\text{GLOL1}}^{(k)} = 1602 + k \cdot 0.5625$ MHz and $f_{\text{GLOL2}}^{(k)} = 1246 + k \cdot 0.4375$ MHz, where $k \in \{-7, -6, \dots, 5, 6\}$ is the channel number. Satellites in opposite points of an orbit plane transmit signals on equal frequencies, as these satellites will never be in view simultaneously by a ground-based user.

- **GLONASS L1.** Two kind of signals are transmitted: a standard precision (SP) and an obfuscated high precision (HP) signal. The complex baseband transmitted signal can be written as

$$s_T^{(\text{GLO L1})}(t) = e_{L1I}(t) + je_{L1Q}(t), \quad (3.13)$$

with BPSK(5) and BPSK(0.5) modulations:

$$e_{L1I}(t) = \sum_{l=-\infty}^{\infty} D_{\text{GNAV}}[l]_{102200} \oplus C_{\text{HP}}[l]_{L_{\text{HP}}} p(t - lT_{c,\text{HP}}), \quad (3.14)$$

$$e_{L1Q}(t) = \sum_{l=-\infty}^{\infty} D_{\text{GNAV}}[l]_{10220} \oplus C_{\text{SP}}[l]_{511} p(t - lT_{c,\text{SP}}), \quad (3.15)$$

where $T_{c,\text{HP}} = \frac{1}{5.11} \mu\text{s}$, $T_{c,\text{SP}} = \frac{1}{0.511} \mu\text{s}$, and $L_{\text{HP}} = 3.3554 \cdot 10^7$. The navigation message D_{GNAV} is transmitted at 50 bps. Details of its content and structure, as well as the generation of the C_{SP} code, can be found at the [ICD](#). The usage of the HP signal should be agreed with the Russian Federation Defense Ministry, and no more details have been disclosed.

- **GLONASS L2.** Beginning with the second generation of satellites, called GLONASS-M and first launched in 2001, a second civil signal is available using the same SP code than the one in the L1 band.

The use of FDMA techniques, in which the same code is used to broadcast navigation signals on different frequencies, and the placement of civil GLONASS transmissions on frequencies close to 1600 MHz, well above the GPS L1 band, have complicated the design of combined GLONASS/GPS receivers, particularly low-cost equipment for mass-market applications. Future plans of modernization are intended to increase compatibility and interoperability with other GNSS, and include the addition of a code-division multiple access (CDMA) structure, and possibly binary offset carrier (BOC) modulation, beginning with the third civil signal in the L3 band (1197.648 – 1212.255 MHz). Russia is implementing the new signals on the next-generation GLONASS-K satellites, with a first prototype successfully launched into orbit on February 26, 2011.

3.1.3 Galileo signal in space

The nominal Galileo constellation comprises a total of 27 operational satellites (plus 3 active spares), that are evenly distributed among three orbital planes inclined at 56° relative to the equator. There are nine operational satellites per orbital plane, occupying evenly distributed orbital slots. Three additional spare satellites (one per orbital plane) complement the nominal constellation configuration. The Galileo satellites are placed in quasi-circular Earth orbits with a nominal semi-major axis of about 30,000 km and an approximate revolution period of 14 hours. The Control segment full infrastructure will be composed of 30 – 40 sensor stations, 3 control centers, 9 Mission Uplink stations, and 5 TT&C stations.

Galileo's Open Service is defined at [Signal In Space Interface Control Document](#). [Ref↔](#): [OS SIS ICD, Issue 1.3](#), where the following signal structures are specified:

- **Galileo E1.** This band, centered at $f_{\text{Gal E1}} = 1575.420$ MHz and with a reference bandwidth of 24.5520 MHz, uses the so-called composite binary offset carrier CBOC(6,1, $\frac{1}{11}$) modulation, defined in baseband as:

$$s_T^{(\text{Gal E1})}(t) = \frac{1}{\sqrt{2}} \left(e_{E1B}(t) (\alpha sc_A(t) + \beta sc_B(t)) + \right. \quad (3.16)$$

$$\left. - e_{E1C}(t) (\alpha sc_A(t) - \beta sc_B(t)) \right), \quad (3.17)$$

where the subcarriers $sc(t)$ are defined as

$$sc_A(t) = \text{sign} \left(\sin(2\pi f_{s,E1A} t) \right), \quad (3.18)$$

$$sc_B(t) = \text{sign} \left(\sin(2\pi f_{s,E1B} t) \right), \quad (3.19)$$

and $f_{s,E1A} = 1.023$ MHz, $f_{s,E1B} = 6.138$ MHz are the subcarrier rates, $\alpha = \sqrt{\frac{10}{11}}$, and $\beta = \sqrt{\frac{1}{11}}$. Channel B contains the I/NAV type of navigation message, $D_{I/NAV}$, intended for Safety-of-Life (SoL) services:

$$e_{E1B}(t) = \sum_{l=-\infty}^{+\infty} D_{I/NAV} \left[[l]_{4092} \right] \oplus C_{E1B} \left[[l]_{4092} \right] p(t - lT_{c,E1B}). \quad (3.20)$$

In case of channel C, it is a pilot (dataless) channel with a secondary code, forming a tiered code:

$$e_{E1C}(t) = \sum_{m=-\infty}^{+\infty} C_{E1Cs} \left[[m]_{25} \right] \oplus \sum_{l=1}^{4092} C_{E1Cp} \left[[l] \right] \cdot p(t - mT_{c,E1Cs} - lT_{c,E1Cp}), \quad (3.21)$$

with $T_{c,E1B} = T_{c,E1Cp} = \frac{1}{1.023} \mu\text{s}$ and $T_{c,E1Cs} = 4$ ms. The C_{E1B} and C_{E1Cp} primary codes are pseudorandom memory code sequences defined at Annex C.7 and C.8 of OS SIS ICD. The binary sequence of the secondary code C_{E1Cs} is 0011100000001010110110010. This band also contains another component, Galileo E1A, intended for the Public Regulated Service (PRS). It uses a BOC(15,2.5) modulation with cosine-shaped subcarrier $f_{s,E1A} = 15.345$ MHz and $T_{c,E1A} = \frac{1}{2.5575} \mu\text{s}$. The PRS spreading codes and the structure of the navigation message have not been made public.

- **Galileo E6.** Intended for the Commercial Service and centered at $f_{\text{Gal E6}} = 1278.750$ MHz, this band provides pilot and data components

$$s_T^{(\text{Gal E6})}(t) = \frac{1}{\sqrt{2}} (e_{E6B}(t) - e_{E6C}(t)), \quad (3.22)$$

$$e_{E6B}(t) = \sum_{m=-\infty}^{+\infty} D_{C/NAV} \left[[l]_{5115} \right] \oplus C_{E6B} \left[[l]_{L_{E6B}} \right] \cdot p(t - lT_{c,E6}), \quad (3.23)$$

$$e_{E6C}(t) = \sum_{m=-\infty}^{+\infty} C_{E6Cs} \left[[m]_{100} \right] \oplus \sum_{l=1}^{L_{E6C}} C_{E6Cp} \left[[l] \right] \cdot p(t - mT_{c,E6s} - lT_{c,E6p}), \quad (3.24)$$

where $D_{C/NAV}$ is the C/NAV navigation data stream, which is modulated with the encrypted ranging code C_{E6B} with chip period $T_{c,E6} = \frac{1}{5.115} \mu\text{s}$, thus being a BPSK(5) modulation. Codes C_{E6B} and primary codes C_{E6Cs} and their respective lengths, L_{E6B} and L_{E6C} , have not been published. The secondary codes for the pilot component, C_{E6Cp} , are available at the OS SIS ICD. The receiver reference bandwidth for this signal is 40.920 MHz. This band also contains another component, Galileo E6A, intended for PRS.

- **Galileo E5.** Centered at $f_{\text{Gal E5}} = 1191.795$ MHz and with a total bandwidth of 51.150 MHz, its signal structure deserves some analysis. The AltBOC modulation can be generically expressed as

$$s^{\text{AltBOC}}(t) = x_1(t)v^*(t) + x_2(t)v(t), \quad (3.25)$$

where $v(t) = \frac{1}{\sqrt{2}} (\text{sign}(\cos(2\pi f_s t)) + j \text{sign}(\sin(2\pi f_s t)))$ is the single side-band subcarrier, f_s is the subcarrier frequency, $(\cdot)^*$ stands for the conjugate operation, and $x_1(t)$ and $x_2(t)$ are QPSK signals. The resulting waveform does not exhibit constant envelope. In case of Galileo, the need for high efficiency of the satellites' onboard High Power Amplifier (HPA) has pushed a modification on the signal in order to make it envelope-constant and thus use the HPA at saturation. This can be done by adding some inter-modulation products to the expression above, coming up with the following definition:

$$s_T^{(\text{Gal E5})}(t) = e_{E5a}(t)ss_c^*(t) + e_{E5b}(t)ss_c(t) + \bar{e}_{E5a}(t)ss_c^*(t) + \bar{e}_{E5b}(t)ss_c(t), \quad (3.26)$$

where the single and product side-band signal subcarriers are

$$ssc_s(t) = sc_s(t) + jsc_s\left(t - \frac{T_s}{4}\right), \quad (3.27)$$

$$ssc_p(t) = sc_p(t) + jsc_p\left(t - \frac{T_s}{4}\right), \quad (3.28)$$

and

$$e_{E5a}(t) = e_{E5aI}(t) + je_{E5aQ}(t), \quad (3.29)$$

$$e_{E5b}(t) = e_{E5bI}(t) + je_{E5bQ}(t), \quad (3.30)$$

$$\bar{e}_{E5a}(t) = \bar{e}_{E5aI}(t) + j\bar{e}_{E5aQ}(t), \quad (3.31)$$

$$\bar{e}_{E5b}(t) = \bar{e}_{E5bI}(t) + j\bar{e}_{E5bQ}(t), \quad (3.32)$$

$$\bar{e}_{E5aI}(t) = e_{E5aQ}(t)e_{E5bI}(t)e_{E5bQ}(t), \quad (3.33)$$

$$\bar{e}_{E5aQ}(t) = e_{E5aI}(t)e_{E5bI}(t)e_{E5bQ}(t), \quad (3.34)$$

$$\bar{e}_{E5bI}(t) = e_{E5bQ}(t)e_{E5aI}(t)e_{E5aQ}(t), \quad (3.35)$$

$$\bar{e}_{E5bQ}(t) = e_{E5bI}(t)e_{E5aI}(t)e_{E5aQ}(t). \quad (3.36)$$

The signal components are defined as

$$e_{E5aI}(t) = \sum_{m=-\infty}^{+\infty} C_{E5aIs} \left[|m|_{20} \right] \oplus \sum_{l=1}^{10230} C_{E5aIp} \left[l \right] \oplus \quad (3.37)$$

$$\oplus D_{F/NAV} \left[[l]_{204600} \right] p(t - mT_{c,E5s} - lT_{c,E5p}), \quad (3.38)$$

$$e_{E5aQ}(t) = \sum_{m=-\infty}^{+\infty} C_{E5aQs} \left[|m|_{100} \right] \oplus \sum_{l=1}^{10230} C_{E5aQp} \left[l \right] \cdot \quad (3.39)$$

$$\cdot p(t - mT_{c,E5s} - lT_{c,E5p}), \quad (3.40)$$

$$e_{E5bI}(t) = \sum_{m=-\infty}^{+\infty} C_{E5bIs} \left[|m|_4 \right] \oplus \sum_{l=1}^{10230} C_{E5bIp} \left[l \right] \oplus \quad (3.41)$$

$$\oplus D_{I/NAV} \left[[l]_{40920} \right] p(t - mT_{c,E5s} - lT_{c,E5p}), \quad (3.42)$$

$$e_{E5bQ}(t) = \sum_{m=-\infty}^{+\infty} C_{E5bQs} \left[|m|_{100} \right] \oplus \sum_{l=1}^{10230} C_{E5bQp} \left[l \right] \cdot \quad (3.43)$$

$$\cdot p(t - mT_{c,E5s} - lT_{c,E5p}), \quad (3.44)$$

where $T_{c,E5s} = 1$ ms and $T_{c,E5p} = \frac{1}{10.23} \mu\text{s}$. **Channel A** contains the F/NAV type of navigation message, $D_{F/NAV}$, intended for the Open Service. The I/NAV message structures for the E5bI and E1B signals use the same page layout. Only page sequencing is different, with page swapping between both components in order to allow a fast reception of data by a dual frequency receiver. The single subcarrier $sc_s(t)$ and the product subcarrier $sc_p(t)$ are defined as:

$$sc_s(t) = \frac{\sqrt{2}}{4} \text{sign} \left(\cos \left(2\pi f_s t - \frac{\pi}{4} \right) \right) + \quad (3.45)$$

$$+ \frac{1}{2} \text{sign} \left(\cos (2\pi f_s t) \right) + \frac{\sqrt{2}}{4} \text{sign} \left(\cos \left(2\pi f_s t + \frac{\pi}{4} \right) \right), \quad (3.46)$$

$$sc_p(t) = -\frac{\sqrt{2}}{4} \text{sign} \left(\cos \left(2\pi f_s t - \frac{\pi}{4} \right) \right) + \quad (3.47)$$

$$+ \frac{1}{2} \text{sign} \left(\cos (2\pi f_s t) \right) - \frac{\sqrt{2}}{4} \text{sign} \left(\cos \left(2\pi f_s t + \frac{\pi}{4} \right) \right), \quad (3.48)$$

with a subcarrier frequency of $f_s = 15.345$ MHz, thus defining an AltBOC(15,10) modulation. The QPSK(10) signal $e_{E5a}(t)$ defined above is shifted to $f_{\text{Gal E5a}} \doteq f_{\text{Gal E5}} - f_s = 1176.450$ MHz, while $e_{E5b}(t)$ is shifted to $f_{\text{Gal E5b}} \doteq f_{\text{Gal E5}} + f_s = 1207.140$ MHz. Thus, we can bandpass filter around $f_{\text{Gal E5a}}$ and get a good approximation of a QPSK(10) signal, with very low energy components of $e_{E5b}(t)$, $\bar{e}_{E5a}(t)$, and $\bar{e}_{E5b}(t)$:

$$s_T^{(\text{Gal E5a})}(t) \simeq e_{E5aI}(t) + j e_{E5aQ}(t). \quad (3.49)$$

The same applies to $e_{E5b}(t)$, allowing an independent reception of two QPSK(10) signals and thus requiring considerably less bandwidth than the processing of the whole E5 band.

3.1.4 Reference

This text is an except of the following paper:

- C. Fernández-Prades, L. Lo Presti, E. Falletti, *Satellite Radiolocalization From GPS to GNSS and Beyond: Novel Technologies and Applications for Civil Mass↔Market*. Proceedings of the IEEE. Vol 99, No. 11, pp. 1882-1904. November, 2011. doi: 10.1109/JPR↔OC.2011.2158032

Chapter 4

Todo List

Member [ALPHA_0](#) ({69, 8})

read all pages of subframe 4

Member [T_OA](#) ({69, 8})

read all pages of subframe 5

Chapter 5

Module Index

5.1 Modules

Here is a list of all modules:

Common definitions	67
GPS_L2	68
Gps_cnav_decoder	69

Chapter 6

Hierarchical Index

6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Acq_Conf	71
Acquisition_Dump_Reader	72
Agnss_Ref_Location	77
Agnss_Ref_Time	78
alm_t	79
ambc_t	80
Bayesian_estimator	82
Beidou_Dnav_Almanac	88
Beidou_Dnav_Ephemeris	91
Beidou_Dnav_Iono	107
Beidou_Dnav_Navigation_Message	110
Beidou_Dnav_Utc_Model	114
block	
Acquisition_msg_rx	73
beidou_b1i_telemetry_decoder_gs	84
beidou_b3i_telemetry_decoder_gs	86
channel_msg_receiver_cc	137
channel_status_msg_receiver	138
direct_resampler_conditioner_cb	163
direct_resampler_conditioner_cc	164
direct_resampler_conditioner_cs	164
dll_pll_veml_tracking	169
dll_pll_veml_tracking_fpga	170
Galileo_E1_Tcp_Connector_Tracking_cc	203
galileo_e5a_noncoherentIQ_acquisition_caf_cc	204
galileo_pcps_8ms_acquisition_cc	228
galileo_telemetry_decoder_gs	234
glonass_l1_ca_dll_pll_c_aid_tracking_cc	346
glonass_l1_ca_dll_pll_c_aid_tracking_sc	347
Glonass_L1_Ca_Dll_Pll_Tracking_cc	348
glonass_l1_ca_telemetry_decoder_gs	349
glonass_l2_ca_dll_pll_c_aid_tracking_cc	351
glonass_l2_ca_dll_pll_c_aid_tracking_sc	352
Glonass_L2_Ca_Dll_Pll_Tracking_cc	352
glonass_l2_ca_telemetry_decoder_gs	353

gnss_sdr_fpga_sample_counter	383
gnss_sdr_time_counter	392
Gps_L1_Ca_Dll_Pll_Tracking_GPU_cc	466
Gps_L1_Ca_Kf_Tracking_cc	466
Gps_L1_Ca_Tcp_Connector_Tracking_cc	467
gps_l1_ca_telemetry_decoder_gs	468
gps_l2c_telemetry_decoder_gs	470
gps_l5_telemetry_decoder_gs	471
hybrid_observables_gs	558
labsat23_source	570
Notch	580
NotchLite	583
pcps_acquisition	591
pcps_acquisition_fine_doppler_cc	595
pcps_assisted_acquisition_cc	607
pcps_cccwsr_acquisition_cc	612
pcps_openc1_acquisition_cc	616
pcps_quicksync_acquisition_cc	622
pcps_tong_acquisition_cc	627
pulse_blanking_cc	636
sbas_l1_telemetry_decoder_gs	705
signal_generator_c	718
Channel_Event	137
ChannelFsm	140
cl_fft_plan	142
clFFT_Complex	142
clFFT_Dim3	143
clFFT_SplitComplex	143
cnav_msg_decoder_t	143
cnav_msg_t	144
cnav_v27_part_t	146
Command_Event	149
Concurrent_Map< Data >	151
Concurrent_Queue< Data >	152
Concurrent_Queue< pmt::pmt_t >	152
ConfigurationInterface	152
FileConfiguration	177
InMemoryConfiguration	564
ControlThread	155
Cpu_Multicorrelator	158
Cpu_Multicorrelator_16sc	158
Cpu_Multicorrelator_Real_Codes	159
CubatureFilter	160
cuda_multicorrelator	161
dgps_t	162
Dll_Pll_Conf	166
Dll_Pll_Conf_Fpga	167
eph_t	172
erp_t	173
erpd_t	173
std::exception	
std::runtime_error	
GnuplotException	415
Exponential_Smoother	174
exterr_t	176
fcbd_t	176
file_t	177
Fpga_Acquisition	183

Fpga_dynamic_bit_selection	187
Fpga_Multicorrelator_8sc	189
Fpga_Switch	194
FrontEndCal	196
ftp_t	198
Galileo_Almanac	198
Galileo_Almanac_Helper	202
Galileo_Ephemeris	209
Galileo_Fnav_Message	222
Galileo_Inav_Message	223
Galileo_Iono	224
Galileo_Utc_Model	236
GeoJSON_Printer	317
geph_t	318
Glonass_Gnav_Almanac	318
Glonass_Gnav_Ephemeris	325
Glonass_Gnav_Navigation_Message	339
Glonass_Gnav_Utc_Model	343
Gnss_circular_deque< T >	375
Gnss_circular_deque< Gnss_Synchro >	375
Gnss_Satellite	379
Gnss_Sdr_Supl_Client	384
Gnss_Signal	393
Gnss_Synchro	395
Gnss_Synchro_Udp_Sink	406
GNSSBlockFactory	406
GNSSBlockInterface	408
AcquisitionInterface	75
BeidouB1iPcpsAcquisition	119
BeidouB3iPcpsAcquisition	126
GalileoE1Pcps8msAmbiguousAcquisition	245
GalileoE1PcpsAmbiguousAcquisition	249
GalileoE1PcpsAmbiguousAcquisitionFpga	254
GalileoE1PcpsCccwsrAmbiguousAcquisition	260
GalileoE1PcpsQuickSyncAmbiguousAcquisition	264
GalileoE1PcpsTongAmbiguousAcquisition	268
GalileoE5aNoncoherentIQAcquisitionCaf	281
GalileoE5aPcpsAcquisition	285
GalileoE5aPcpsAcquisitionFpga	289
GalileoE5bPcpsAcquisition	300
GalileoE5bPcpsAcquisitionFpga	307
GlonassL1CaPcpsAcquisition	359
GlonassL2CaPcpsAcquisition	368
GpsL1CaPcpsAcquisition	489
GpsL1CaPcpsAcquisitionFineDoppler	493
GpsL1CaPcpsAcquisitionFpga	497
GpsL1CaPcpsAssistedAcquisition	504
GpsL1CaPcpsOpenCIAcquisition	507
GpsL1CaPcpsQuickSyncAcquisition	511
GpsL1CaPcpsTongAcquisition	516
GpsL2MPcpsAcquisition	528
GpsL2MPcpsAcquisitionFpga	532
GpsL5iPcpsAcquisition	543
GpsL5iPcpsAcquisitionFpga	547
Ad9361FpgaSignalSource	76
ArraySignalConditioner	80
BeamformerFilter	83
ByteToShort	132

ChannelInterface	141
Channel	133
CustomUDPSignalSource	161
DirectResamplerConditioner	165
FileSignalSource	178
FirFilter	179
FlexibandSignalSource	181
Fmcomms2SignalSource	182
FreqXlatingFirFilter	195
GenSignalSource	316
Gn3sSignalSource	373
GnMaxSignalSource	374
lbyteToCbyte	560
lbyteToComplex	561
lbyteToCshort	562
lshortToComplex	567
lshortToCshort	568
LabsatSignalSource	571
MmseResamplerConditioner	574
MultichannelFileSignalSource	577
NotchFilter	581
NotchFilterLite	582
NsrFileSignalSource	584
ObservablesInterface	587
HybridObservables	558
OsmosdrSignalSource	588
Pass_Through	589
PlutosdrSignalSource	634
PulseBlankingFilter	637
PvtInterface	644
Rtklib_Pvt	690
RawArraySignalSource	646
RtlTcpSignalSource	701
SignalConditioner	719
SignalGenerator	721
SpirFileSignalSource	726
SpirGSS6450FileSignalSource	727
TelemetryDecoderInterface	735
BeidouB1iTelemetryDecoder	123
BeidouB3iTelemetryDecoder	130
GalileoE1BTelemetryDecoder	238
GalileoE5aTelemetryDecoder	296
GalileoE5bTelemetryDecoder	314
GlonassL1CaTelemetryDecoder	363
GlonassL2CaTelemetryDecoder	372
GpsL1CaTelemetryDecoder	522
GpsL2CTelemetryDecoder	523
GpsL5TelemetryDecoder	554
SbasL1TelemetryDecoder	706
TrackingInterface	743
BeidouB1iDIIPIITracking	117
BeidouB3iDIIPIITracking	124
GalileoE1DIIPIIVemITracking	239
GalileoE1DIIPIIVemITrackingFpga	241
GalileoE1TcpConnectorTracking	273
GalileoE5aDIIPIITracking	275
GalileoE5aDIIPIITrackingFpga	277
GalileoE5bDIIPIITracking	297

GlonassL1CaDIIPIIAidTracking	355
GlonassL1CaDIIPIITracking	357
GlonassL2CaDIIPIIAidTracking	364
GlonassL2CaDIIPIITracking	366
GpsL1CaDIIPIITracking	479
GpsL1CaDIIPIITrackingFpga	481
GpsL1CaDIIPIITrackingGPU	485
GpsL1CaKfTracking	487
GpsL1CaTcpConnectorTracking	520
GpsL2MDIIPIITracking	524
GpsL2MDIIPIITrackingFpga	526
GpsL5DIIPIITracking	537
GpsL5DIIPIITrackingFpga	539
TwoBitCpxFileSignalSource	745
TwoBitPackedFileSignalSource	746
UhdSignalSource	747
GNSSFlowgraph	409
Gnuplot	413
Gps_Acq_Assist	416
Gps_Almanac	419
Gps_CNAV_Ephemeris	423
Gps_CNAV_Iono	438
Gps_CNAV_Navigation_Message	442
Gps_CNAV_Utc_Model	444
Gps_Ephemeris	447
Gps_Iono	462
Gps_Navigation_Message	472
Gps_Utc_Model	476
GPU_Complex	555
GPU_Complex_Short	555
Gpx_Printer	556
gtime_t	557
half_cyc_tag	557
INIReader	563
kernel_info_t	569
Kml_Printer	570
lex_t	572
lexeph_t	572
lexion_t	573
lexmsg_t	573
ModelFunction	574
Monitor_Pvt	575
Monitor_Pvt_Udp_Sink	576
msm_h_t	576
nav_t	578
Nmea_Printer	579
ntrip_t	585
Obs_Conf	585
obs_t	586
obsd_t	586
Observables_Dump_Reader	587
opt_t	588
pclk_t	590
pcps_acquisition_fpga	601
pcpsconf_fpga_t	632
pcv_t	632
pcvs_t	633
peph_t	633

pppcorr_t	635
prcopt_t	635
Pvt_Conf	638
Pvt_Solution	639
Rtklib_Solver	695
raw_t	645
Rinex_Printer	647
Rtcm	668
Rtcm_Printer	684
rtcm_t	689
rtk_t	690
Rtklib_Solver_Dump_Reader	698
rtksvr_t	698
Rtl_Tcp_Dongle_Info	699
Sbas_Ephemeris	702
sbs_t	707
sbsfcorr_t	708
sbsigp_t	708
sbsigpband_t	709
sbsion_t	709
sbslcorr_t	709
sbsmsg_t	710
sbssat_t	710
sbssatp_t	711
seph_t	711
Serdes_Gnss_Synchro	711
Serdes_Monitor_Pvt	714
serial_t	716
snrmask_t	722
sol_t	722
solbuf_t	722
soloft_t	723
solstat_t	723
solstatbuf_t	724
Spirent_Motion_Csv_Dump_Reader	724
ssat_t	728
ssr_t	728
sta_t	729
stec_t	729
stream_cfg	730
stream_t	730
StringConverter	731
sync_block	
beamformer	83
byte_x2_to_complex_byte	131
complex_byte_to_float_x2	150
complex_float_to_complex_byte	150
conjugate_cc	153
conjugate_ic	154
conjugate_sc	155
cshort_to_float_x2	160
Gnss_Sdr_Valve	393
gnss_synchro_monitor	405
Gr_Complex_Ip_Packet_Source	556
rtklib_pvt_gs	692
rtl_tcp_signal_source_c	700
short_x2_to_cshort	717
sync_decimator	

gnss_sdr_sample_counter	384
interleaved_byte_to_complex_byte	565
interleaved_byte_to_complex_short	566
interleaved_short_to_complex_short	566
sync_interpolator	
unpack_2bit_samples	748
unpack_byte_2bit_cpx_samples	749
unpack_byte_2bit_samples	750
unpack_byte_4bit_samples	750
unpack_intspir_1bit_samples	751
unpack_spir_gss6450_samples	752
Tcp_Communication	731
Tcp_Packet_Data	732
tcp_t	733
tcpcli_t	733
TcpCmdInterface	733
tcpsvr_t	734
tec_t	735
tfe_t	736
tled_t	736
Tlm_Dump_Reader	737
Tracking_2nd_DLL_filter	737
Tracking_2nd_PLL_filter	739
Tracking_Dump_Reader	740
Tracking_FLL_PLL_filter	740
Tracking_loop_filter	741
Tracking_True_Obs_Reader	742
trop_t	744
True_Observables_Reader	744
UnscentedFilter	752
url_t	753
v27_decision_t	753
v27_poly_t	753
v27_t	754
Viterbi_Decoder	754

Chapter 7

Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Acq_Conf	71
Acquisition_Dump_Reader	72
Acquisition_msg_rx	73
AcquisitionInterface	
This abstract class represents an interface to an acquisition GNSS block	75
Ad9361FpgaSignalSource	76
Agnss_Ref_Location	
Interface of an Assisted GNSS REFERENCE LOCATION storage	77
Agnss_Ref_Time	
Interface of an Assisted GNSS REFERENCE TIME storage	78
alm_t	79
ambc_t	80
ArraySignalConditioner	
This class wraps blocks to change data_type_adapter, input_filter and resampler to be applied to the input flow of sampled signal	80
Bayesian_estimator	
Bayesian_estimator is an estimator of noise characteristics (i.e. mean, covariance)	82
beamformer	
This class implements a real-time software-defined spatial filter using the CTTC GNSS experimental antenna array input and a set of dynamically reloadable weights	83
BeamformerFilter	
Interface of an adapter of a digital beamformer block to a GNSSBlockInterface	83
beidou_b1i_telemetry_decoder_gs	
This class implements a block that decodes the BeiDou DNAV data	84
beidou_b3i_telemetry_decoder_gs	
This class implements a block that decodes the BeiDou DNAV data	86
Beidou_Dnav_Almanac	
This class is a storage for the BeiDou D1 almanac	88
Beidou_Dnav_Ephemeris	
This class is a storage and orbital model functions for the GPS SV ephemeris data as described in BeiDou Navigation Satellite System Signal In Space Interface Control Document Open Service Signal B1I (Version 3.0)	91
Beidou_Dnav_Iono	
This class is a storage for the BEIDOU IONOSPHERIC data as described in ICD v2.1	107
Beidou_Dnav_Navigation_Message	
This class decodes a BeiDou D1 NAV Data message	110

Beidou_Dnav_Utc_Model	
This class is a storage for the BeiDou DNAV UTC Model	114
BeidouB1iDlIPllTracking	
This class implements a code DLL + carrier PLL tracking loop	117
BeidouB1iPcpsAcquisition	
This class adapts a PCPS acquisition block to an AcquisitionInterface for GPS L1 C/A signals	119
BeidouB1iTelemetryDecoder	
This class implements a NAV data decoder for BEIDOU B1I	123
BeidouB3iDlIPllTracking	
This class implements a code DLL + carrier PLL tracking loop	124
BeidouB3iPcpsAcquisition	
This class adapts a PCPS acquisition block to an AcquisitionInterface for BeiDou B3I signals	126
BeidouB3iTelemetryDecoder	
This class implements a NAV data decoder for BEIDOU B1I	130
byte_x2_to_complex_byte	
This class adapts two signed char streams into a std::complex<signed char> stream	131
ByteToShort	
Adapts an 8-bits sample stream (IF) to a short int stream (IF)	132
Channel	
This class represents a GNSS channel. It wraps an AcquisitionInterface , a Tracking Interface and a TelemetryDecoderInterface , and handles their interaction through a Finite State Machine	133
Channel_Event	137
channel_msg_receiver_cc	
GNU Radio block that receives asynchronous channel messages from acquisition and tracking blocks	137
channel_status_msg_receiver	
GNU Radio block that receives asynchronous channel messages from tlm blocks	138
ChannelFsm	
This class implements a State Machine for channel	140
ChannelInterface	
This abstract class represents an interface to a channel GNSS block	141
cl_fft_plan	142
clFFT_Complex	142
clFFT_Dim3	143
clFFT_SplitComplex	143
cnav_msg_decoder_t	143
cnav_msg_t	144
cnav_v27_part_t	146
Command_Event	149
complex_byte_to_float_x2	
This class adapts a std::complex<signed char> stream into two 16-bits (short) streams	150
complex_float_to_complex_byte	
This class adapts a gr_complex stream into a std::complex<signed char> stream	150
Concurrent_Map< Data >	
This class implements a thread-safe std::map	151
Concurrent_Queue< Data >	
This class implements a thread-safe std::queue	152
ConfigurationInterface	
This abstract class represents an interface to configuration parameters	152
conjugate_cc	
This class adapts a std::complex<short> stream into two 32-bits (float) streams	153
conjugate_ic	
This class adapts a std::complex<short> stream into two 32-bits (float) streams	154
conjugate_sc	
This class adapts a std::complex<short> stream into two 32-bits (float) streams	155

ControlThread	
This class represents the main thread of the application, so the name is ControlThread . This is the GNSS Receiver Control Plane: it connects the flowgraph, starts running it, and while it does not stop, reads the control messages generated by the blocks, processes them, and applies the corresponding actions	155
Cpu_Multicorrelator	
Class that implements carrier wipe-off and correlators	158
Cpu_Multicorrelator_16sc	
Class that implements carrier wipe-off and correlators	158
Cpu_Multicorrelator_Real_Codes	
Class that implements carrier wipe-off and correlators	159
cshort_to_float_x2	
This class adapts a <code>std::complex<short></code> stream into two 32-bits (float) streams	160
CubatureFilter	160
cuda_multicorrelator	
Class that implements carrier wipe-off and correlators using NVIDIA CUDA GPU accelerators	161
CustomUDPSignalSource	
This class reads from UDP packets, which streams interleaved I/Q samples over a network	161
dgps_t	162
direct_resampler_conditioner_cb	
This class implements a direct resampler conditioner for <code>std::complex<signed char></code>	163
direct_resampler_conditioner_cc	
This class implements a direct resampler conditioner for complex data	164
direct_resampler_conditioner_cs	
This class implements a direct resampler conditioner for <code>std::complex<short></code>	164
DirectResamplerConditioner	
Interface of an adapter of a direct resampler conditioner block to a <code>SignalConditionerInterface</code>	165
Dll_Pll_Conf	166
Dll_Pll_Conf_Fpga	167
dll_pll_veml_tracking	
This class implements a code DLL + carrier PLL tracking block	169
dll_pll_veml_tracking_fpga	
This class implements a code DLL + carrier PLL tracking block	170
eph_t	172
erp_t	173
erpd_t	173
Exponential_Smoother	
Class that implements a first-order exponential smoother	174
exterr_t	176
fcbd_t	176
file_t	177
FileConfiguration	
This class is an implementation of the interface ConfigurationInterface	177
FileSignalSource	
Class that reads signals samples from a file and adapts it to a <code>SignalSourceInterface</code>	178
FirFilter	
This class adapts a GNU Radio <code>gr_fir_filter</code> designed with <code>pm_remez</code>	179
FlexibandSignalSource	
This class configures and reads samples from Teleorbit Flexiband front-end. This software requires a Flexiband GNU Radio driver installed (not included with GNSS-SDR)	181
Fmcomms2SignalSource	182
Fpga_Acquisition	
Class that implements carrier wipe-off and correlators	183
Fpga_dynamic_bit_selection	
Class that controls the switch in the FPGA, which connects the FPGA acquisition and multicorrelator modules to either the DMA or the Analog Front-End	187
Fpga_Multicorrelator_8sc	
Class that implements carrier wipe-off and correlators	189

Fpga_Switch	
Class that controls the switch in the FPGA, which connects the FPGA acquisition and multicorrelator modules to either the DMA or the Analog Front-End	194
FreqXlatingFirFilter	
This class adapts a gnuradio gr_freq_xlating_fir_filter designed with pm_remez	195
FrontEndCal	196
ftp_t	198
Galileo_Almanac	
This class is a storage for the Galileo SV ALMANAC data	198
Galileo_Almanac_Helper	
This class is a storage for the GALILEO ALMANAC data as described in GALILEO ICD	202
Galileo_E1_Tcp_Connector_Tracking_cc	
This class implements a code DLL + carrier PLL VEML (Very Early Minus Late) tracking block for Galileo E1 signals	203
galileo_e5a_noncoherentIQ_acquisition_caf_cc	
This class implements a Parallel Code Phase Search Acquisition	204
Galileo_Ephemeris	
This class is a storage and orbital model functions for the Galileo SV ephemeris data as described in Galileo ICD paragraph 5.1.1 (See https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf)	209
Galileo_Fnav_Message	
This class handles the Galileo F/NAV Data message, as described in the Galileo Open Service Signal in Space Interface Control Document (OS SIS ICD), Issue 1.2 (Nov 2015). See https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf	222
Galileo_Inav_Message	
This class handles the Galileo I/NAV Data message, as described in the Galileo Open Service Signal in Space Interface Control Document (OS SIS ICD), Issue 1.2 (Nov 2015). See https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf	223
Galileo_Iono	
This class is a storage for the GALILEO IONOSPHERIC data as described in Galileo ICD paragraph 5.1.6	224
galileo_pcps_8ms_acquisition_cc	
This class implements a Parallel Code Phase Search Acquisition for Galileo E1 signals with coherent integration time = 8 ms (two codes)	228
galileo_telemetry_decoder_gs	
This class implements a block that decodes the INAV and FNAV data defined in Galileo ICD	234
Galileo_Utc_Model	
This class is a storage for the GALILEO UTC MODEL data as described in Galileo ICD https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf paragraph 5.1.7	236
GalileoE1BTelemetryDecoder	
This class implements a NAV data decoder for Galileo INAV frames in E1B radio link	238
GalileoE1DIIPIVemlTracking	
This class Adapts a DLL+PLL VEML (Very Early Minus Late) tracking loop block to a TrackingInterface for Galileo E1 signals	239
GalileoE1DIIPIVemlTrackingFpga	
This class Adapts a DLL+PLL VEML (Very Early Minus Late) tracking loop block to a TrackingInterface for Galileo E1 signals	241
GalileoE1Pcps8msAmbiguousAcquisition	
Adapts a PCPS 8ms acquisition block to an AcquisitionInterface for Galileo E1 Signals	245
GalileoE1PcpsAmbiguousAcquisition	
This class adapts a PCPS acquisition block to an AcquisitionInterface for Galileo E1 Signals	249
GalileoE1PcpsAmbiguousAcquisitionFpga	
This class adapts a PCPS acquisition block off-loaded on an FPGA to an AcquisitionInterface for Galileo E1 Signals	254

GalileoE1PcpsCccwsrAmbiguousAcquisition	
Adapts a PCPS CCCWSR acquisition block to an AcquisitionInterface for Galileo E1 Signals	260
GalileoE1PcpsQuickSyncAmbiguousAcquisition	
This class adapts a PCPS acquisition block to an AcquisitionInterface for Galileo E1 Signals	264
GalileoE1PcpsTongAmbiguousAcquisition	
Adapts a PCPS Tong acquisition block to an AcquisitionInterface for Galileo E1 Signals	268
GalileoE1TcpConnectorTracking	
This class implements a code DLL + carrier PLL tracking loop	273
GalileoE5aDIIPILTracking	
This class implements a code DLL + carrier PLL tracking loop	275
GalileoE5aDIIPILTrackingFpga	
This class implements a code DLL + carrier PLL tracking loop	277
GalileoE5aNoncoherentIQAcquisitionCaf	281
GalileoE5aPcpsAcquisition	285
GalileoE5aPcpsAcquisitionFpga	
This class adapts a PCPS acquisition block off-loaded on an FPGA to an AcquisitionInterface for Galileo E5a signals	289
GalileoE5aTelemetryDecoder	
This class implements a NAV data decoder for Galileo INAV frames in E1B radio link	296
GalileoE5bDIIPILTracking	
This class implements a code DLL + carrier PLL tracking loop	297
GalileoE5bPcpsAcquisition	300
GalileoE5bPcpsAcquisitionFpga	
This class adapts a PCPS acquisition block off-loaded on an FPGA to an AcquisitionInterface for Galileo E5b signals	307
GalileoE5bTelemetryDecoder	
This class implements a NAV data decoder for Galileo INAV frames in E5b radio link	314
GenSignalSource	
This class wraps blocks that generates synthesized GNSS signal and filters the signal	316
GeoJSON_Printer	
Prints PVT solutions in GeoJSON format file	317
geph_t	318
Glonass_Gnav_Almanac	
This class is a storage for the GLONASS SV ALMANAC data as described GLONASS ICD (Edition 5.1)	318
Glonass_Gnav_Ephemeris	
This class is a storage and orbital model functions for the GLONASS SV ephemeris data as described in GLONASS ICD (Edition 5.1)	325
Glonass_Gnav_Navigation_Message	
This class decodes a GLONASS GNAV Data message as described in GLONASS ICD (Edition 5.1)	339
Glonass_Gnav_Utc_Model	
This class is a storage for the GLONASS GNAV UTC MODEL data as described in GLONASS ICD (Edition 5.1)	343
glonass_l1_ca_dll_pll_c_aid_tracking_cc	
This class implements a DLL + PLL tracking loop block	346
glonass_l1_ca_dll_pll_c_aid_tracking_sc	
This class implements a DLL + PLL tracking loop block	347
Glonass_L1_Ca_Dll_Pll_Tracking_cc	
This class implements a DLL + PLL tracking loop block	348
glonass_l1_ca_telemetry_decoder_gs	
This class implements a block that decodes the GNAV data defined in GLONASS ICD v5.1	349
glonass_l2_ca_dll_pll_c_aid_tracking_cc	
This class implements a DLL + PLL tracking loop block	351
glonass_l2_ca_dll_pll_c_aid_tracking_sc	
This class implements a DLL + PLL tracking loop block	352
Glonass_L2_Ca_Dll_Pll_Tracking_cc	
This class implements a DLL + PLL tracking loop block	352

glonass_l2_ca_telemetry_decoder_gs	
This class implements a block that decodes the GNAV data defined in GLONASS ICD v5.1 . . .	353
GlonassL1CaDIPIICAidTracking	
This class implements a code DLL + carrier PLL tracking loop	355
GlonassL1CaDIPIITracking	
This class implements a code DLL + carrier PLL tracking loop	357
GlonassL1CaPcpsAcquisition	
This class adapts a PCPS acquisition block to an AcquisitionInterface for GPS L1 C/A signals .	359
GlonassL1CaTelemetryDecoder	
This class implements a NAV data decoder for GLONASS L1 C/A	363
GlonassL2CaDIPIICAidTracking	
This class implements a code DLL + carrier PLL tracking loop	364
GlonassL2CaDIPIITracking	
This class implements a code DLL + carrier PLL tracking loop	366
GlonassL2CaPcpsAcquisition	
This class adapts a PCPS acquisition block to an AcquisitionInterface for GLONASS L2 C/A signals	368
GlonassL2CaTelemetryDecoder	
This class implements a NAV data decoder for GLONASS L2 C/A	372
Gn3sSignalSource	
This class reads samples from a GN3S USB dongle, a RF front-end signal sampler	373
GnMaxSignalSource	
This class reads samples from a gnMAX2769 USB dongle, a RF front-end signal sampler . . .	374
Gnss_circular_deque< T >	375
Gnss_Satellite	
This class represents a GNSS satellite	379
gnss_sdr_fpga_sample_counter	383
gnss_sdr_sample_counter	384
Gnss_Sdr_Supl_Client	
Class that implements a C++ interface to external Secure User Location Protocol (SUPL) client library.	384
gnss_sdr_time_counter	392
Gnss_Sdr_Valve	
Implementation of a GNU Radio block that sends a STOP message to the control queue right after a specific number of samples have passed through it	393
Gnss_Signal	
This class represents a GNSS signal	393
Gnss_Synchro	
This is the class that contains the information that is shared by the processing blocks	395
gnss_synchro_monitor	
This class implements a monitoring block which allows sending a data stream with the receiver internal parameters (Gnss_Synchro objects) to local or remote clients over UDP	405
Gnss_Synchro_Udp_Sink	
This class sends serialized Gnss_Synchro objects over UDP to one or multiple endpoints . . .	406
GNSSBlockFactory	
Class that produces all kinds of GNSS blocks	406
GNSSBlockInterface	
This abstract class represents an interface to GNSS blocks	408
GNSSFlowgraph	
This class represents a GNSS flow graph	409
Gnuplot	413
GnuplotException	415
Gps_Acq_Assist	
This class is a storage for the GPS GSM RRLC acquisition assistance data as described in Digital cellular telecommunications system (Phase 2+); Location Services (LCS); Mobile Station (MS) - Serving Mobile Location Centre (SMLC) Radio Resource LCS Protocol (RRLP) (3GPP TS 44.031 version 5.12.0 Release 5)	416

Gps_Almanac	
This class is a storage for the GPS SV ALMANAC data as described in IS-GPS-200K	419
Gps_CNAV_Ephemeris	
This class is a storage and orbital model functions for the GPS SV ephemeris data as described in IS-GPS-200K	423
Gps_CNAV_Iono	
This class is a storage for the GPS IONOSPHERIC data as described in IS-GPS-200K	438
Gps_CNAV_Navigation_Message	
This class decodes a GPS CNAV Data message as described in IS-GPS-200K	442
Gps_CNAV_Utc_Model	
This class is a storage for the GPS UTC MODEL data as described in in IS-GPS-200K	444
Gps_Ephemeris	
This class is a storage and orbital model functions for the GPS SV ephemeris data as described in IS-GPS-200K	447
Gps_Iono	
This class is a storage for the GPS IONOSPHERIC data as described in IS-GPS-200K	462
Gps_L1_Ca_DLL_Pll_Tracking_GPU_cc	
This class implements a DLL + PLL tracking loop block	466
Gps_L1_Ca_Kf_Tracking_cc	
This class implements a DLL + PLL tracking loop block	466
Gps_L1_Ca_Tcp_Connector_Tracking_cc	
This class implements a DLL + PLL tracking loop block	467
gps_l1_ca_telemetry_decoder_gs	
This class implements a block that decodes the NAV data defined in IS-GPS-200K	468
gps_l2c_telemetry_decoder_gs	
This class implements a block that decodes CNAV data defined in IS-GPS-200K	470
gps_l5_telemetry_decoder_gs	
This class implements a GPS L5 Telemetry decoder	471
Gps_Navigation_Message	
This class decodes a GPS NAV Data message as described in IS-GPS-200K	472
Gps_Utc_Model	
This class is a storage for the GPS UTC MODEL data as described in IS-GPS-200K	476
GpsL1CaDIIPllTracking	
This class implements a code DLL + carrier PLL tracking loop	479
GpsL1CaDIIPllTrackingFpga	
This class implements a code DLL + carrier PLL tracking loop	481
GpsL1CaDIIPllTrackingGPU	
This class implements a code DLL + carrier PLL tracking loop using GPU accelerated functions	485
GpsL1CaKfTracking	
This class implements a code DLL + carrier PLL tracking loop	487
GpsL1CaPcpsAcquisition	
This class adapts a PCPS acquisition block to an AcquisitionInterface for GPS L1 C/A signals	489
GpsL1CaPcpsAcquisitionFineDoppler	
This class Adapts a PCPS acquisition block with fine Doppler estimation to an AcquisitionInterface for GPS L1 C/A signals	493
GpsL1CaPcpsAcquisitionFpga	
This class adapts a PCPS acquisition block off-loaded on an FPGA to an AcquisitionInterface for GPS L1 C/A signals	497
GpsL1CaPcpsAssistedAcquisition	
This class adapts a PCPS acquisition block to an AcquisitionInterface for GPS L1 C/A signals	504
GpsL1CaPcpsOpenCLAcquisition	
This class adapts an OpenCL PCPS acquisition block to an AcquisitionInterface for GPS L1 C/A signals	507
GpsL1CaPcpsQuickSyncAcquisition	
This class adapts a PCPS acquisition block to an AcquisitionInterface for GPS L1 C/A signals	511
GpsL1CaPcpsTongAcquisition	
This class adapts a PCPS Tong acquisition block to an AcquisitionInterface for GPS L1 C/A signals	516

GpsL1CaTcpConnectorTracking	
This class implements a code DLL + carrier PLL tracking loop	520
GpsL1CaTelemetryDecoder	
This class implements a NAV data decoder for GPS L1 C/A	522
GpsL2CTelemetryDecoder	
This class implements a NAV data decoder for GPS L2 M	523
GpsL2MDIIPITracking	
This class implements a code DLL + carrier PLL tracking loop	524
GpsL2MDIIPITrackingFpga	
This class implements a code DLL + carrier PLL tracking loop	526
GpsL2MPcpsAcquisition	
This class adapts a PCPS acquisition block to an AcquisitionInterface for GPS L2 M signals	528
GpsL2MPcpsAcquisitionFpga	
This class adapts a PCPS acquisition block off-loaded on an FPGA to an AcquisitionInterface for GPS L2 M signals	532
GpsL5DIIPITracking	
This class implements a code DLL + carrier PLL tracking loop	537
GpsL5DIIPITrackingFpga	
This class implements a code DLL + carrier PLL tracking loop	539
GpsL5iPcpsAcquisition	
This class adapts a PCPS acquisition block to an AcquisitionInterface for GPS L5i signals	543
GpsL5iPcpsAcquisitionFpga	
This class adapts a PCPS acquisition block off-loaded on an FPGA to an AcquisitionInterface for GPS L5i signals	547
GpsL5TelemetryDecoder	
This class implements a NAV data decoder for GPS L5	554
GPU_Complex	555
GPU_Complex_Short	555
Gpx_Printer	
Prints PVT information to GPX format file	556
Gr_Complex_Ip_Packet_Source	556
gtime_t	557
half_cyc_tag	557
hybrid_observables_gs	
This class implements a block that computes observables	558
HybridObservables	
This class implements an ObservablesInterface for observables of all kind of GNSS signals	558
lbyteToCbyte	560
lbyteToComplex	
Adapts an I/Q interleaved byte integer sample stream to a gr_complex (float) stream	561
lbyteToCshort	
Adapts a short integer (16 bits) interleaved sample stream into a std::complex<short> stream	562
INIReader	
Read an INI file into easy-to-access name/value pairs. (Note that I've gone for simplicity here rather than speed, but it should be pretty decent.)	563
InMemoryConfiguration	
This class is an implementation of the interface ConfigurationInterface	564
interleaved_byte_to_complex_byte	
This class adapts an 8-bits interleaved sample stream into a 16-bits complex stream (std::complex<unsigned char>)	565
interleaved_byte_to_complex_short	
This class adapts a short (16-bits) interleaved sample stream into a std::complex<short> stream	566
interleaved_short_to_complex_short	
This class adapts a short (16-bits) interleaved sample stream into a std::complex<short> stream	566
lshortToComplex	
Adapts an I/Q interleaved short integer sample stream to a gr_complex (float) stream	567
lshortToCshort	
Adapts a short integer (16 bits) interleaved sample stream into a std::complex<short> stream	568

kernel_info_t	569
Kml_Printer	
Prints PVT information to OGC KML format file (can be viewed with Google Earth)	570
labsat23_source	
This class implements conversion between Labsat2 and 3 format byte packet samples to gr↔ complex	570
LabsatSignalSource	
This class reads samples stored by a LabSat 2 or LabSat 3 device	571
lex_t	572
lexeph_t	572
lexion_t	573
lexmsg_t	573
MmseResamplerConditioner	
Interface of a MMSE resampler block adapter to a SignalConditionerInterface	574
ModelFunction	574
Monitor_Pvt	
This class contains parameters and outputs of the PVT block	575
Monitor_Pvt_Udp_Sink	576
msm_h_t	576
MultichannelFileSignalSource	
Class that reads signals samples from files at different frequency bands and adapts it to a SignalSourceInterface	577
nav_t	578
Nmea_Printer	
This class provides a implementation of a subset of the NMEA-0183 standard for interfacing marine electronic devices as defined by the National Marine Electronics Association (NMEA)	579
Notch	
This class implements a real-time software-defined multi state notch filter	580
NotchFilter	581
NotchFilterLite	582
NotchLite	
This class implements a real-time software-defined multi state notch filter light version	583
NsrFileSignalSource	
Class that reads signals samples from a file and adapts it to a SignalSourceInterface	584
ntrip_t	585
Obs_Conf	585
obs_t	586
obsd_t	586
Observables_Dump_Reader	587
ObservablesInterface	
This abstract class represents an interface to an observables block	587
opt_t	588
OsmosdrSignalSource	
This class reads samples OsmoSDR-compatible front-ends, such as HackRF or Re- altek's RTL2832U-based USB dongle DVB-T receivers (see https://osmocom.org/projects/rtl-sdr/wiki)	588
Pass_Through	
This class implements a block that connects input and output (does nothing)	589
pclk_t	590
pcps_acquisition	
This class implements a Parallel Code Phase Search Acquisition	591
pcps_acquisition_fine_doppler_cc	
This class implements a Parallel Code Phase Search Acquisition	595
pcps_acquisition_fpga	
This class implements a Parallel Code Phase Search Acquisition that uses the FPGA	601
pcps_assisted_acquisition_cc	
This class implements a Parallel Code Phase Search Acquisition	607

pcps_cccwsr_acquisition_cc	
This class implements a Parallel Code Phase Search Acquisition with Coherent Channel Combining With Sign Recovery scheme	612
pcps_opencl_acquisition_cc	
This class implements a Parallel Code Phase Search Acquisition	616
pcps_quicksync_acquisition_cc	
This class implements a Parallel Code Phase Search Acquisition with the implementation of the Sparse QuickSync Algorithm	622
pcps_tong_acquisition_cc	
This class implements a Parallel Code Phase Search Acquisition with Tong algorithm	627
pcpsconf_fpga_t	632
pcv_t	632
pcvs_t	633
peph_t	633
PlutosdrSignalSource	634
pppcorr_t	635
prcopt_t	635
pulse_blanking_cc	636
PulseBlankingFilter	637
Pvt_Conf	638
Pvt_Solution	
Base class for a PVT solution	639
PvtInterface	
This class represents an interface to a PVT block	644
raw_t	645
RawArraySignalSource	
This class reads samples from a GN3S USB dongle, a RF front-end signal sampler	646
Rinex_Printer	
Class that handles the generation of Receiver INdependent EXchange format (RINEX) files . .	647
Rtcm	
This class implements the generation and reading of some Message Types defined in the RTCM 3.2 Standard, plus some utilities to handle messages	668
Rtcm_Printer	
This class provides a implementation of a subset of the RTCM Standard 10403.2 messages .	684
rtcm_t	689
rtk_t	690
Rtklib_Pvt	
This class implements a PvtInterface for the RTKLIB PVT block	690
rtklib_pvt_gs	
This class implements a block that computes the PVT solution using the RTKLIB integrated library	692
Rtklib_Solver	
This class implements a PVT solution based on RTKLIB	695
Rtklib_Solver_Dump_Reader	698
rtksvr_t	698
Rtl_Tcp_Dongle_Info	
This class represents the dongle information which is sent by rtl_tcp	699
rtl_tcp_signal_source_c	
This class reads interleaved I/Q samples from an rtl_tcp server and outputs complex types . .	700
RtlTcpSignalSource	
This class reads from rtl_tcp, which streams interleaved I/Q samples over TCP. (see https://osmocom.org/projects/rtl-sdr/wiki)	701
Sbas_Ephemeris	
This class stores SBAS SV ephemeris data	702
sbas_l1_telemetry_decoder_gs	
This class implements a block that decodes the SBAS integrity and corrections data defined in RTCA MOPS DO-229	705

SbasL1TelemetryDecoder	
This class implements a NAV data decoder for SBAS frames in L1 radio link	706
sbs_t	707
sbsfcrr_t	708
sbsigp_t	708
sbsigpband_t	709
sbsion_t	709
sbslcorr_t	709
sbsmsg_t	710
sbssat_t	710
sbssatp_t	711
seph_t	711
Serdes_Gnss_Synchro	
This class implements serialization and deserialization of Gnss_Synchro objects using Protocol Buffers	711
Serdes_Monitor_Pvt	
This class implements serialization and deserialization of Monitor_Pvt objects using Protocol Buffers	714
serial_t	716
short_x2_to_cshort	
This class adapts two short streams into a <code>std::complex<short></code> stream	717
signal_generator_c	
This class generates synthesized GNSS signal	718
SignalConditioner	
This class wraps blocks to change <code>data_type_adapter</code> , <code>input_filter</code> and <code>resampler</code> to be applied to the input flow of sampled signal	719
SignalGenerator	
This class generates synthesized GNSS signal	721
snrmask_t	722
sol_t	722
solbuf_t	722
solopt_t	723
solstat_t	723
solstatbuf_t	724
Spirent_Motion_Csv_Dump_Reader	724
SpirFileSignalSource	
Class that reads signals samples from a file and adapts it to a <code>SignalSourceInterface</code>	726
SpirGSS6450FileSignalSource	
Class that reads signals samples from a file and adapts it to a <code>SignalSourceInterface</code>	727
ssat_t	728
ssr_t	728
sta_t	729
stec_t	729
stream_cfg	730
stream_t	730
StringConverter	
Class that interprets the contents of a string and converts it into different types	731
Tcp_Communication	
TCP communication class	731
Tcp_Packet_Data	
Class that implements a TCP data packet	732
tcp_t	733
tcpcli_t	733
TcpCmdInterface	733
tcpsvr_t	734
tec_t	735
TelemetryDecoderInterface	
This abstract class represents an interface to a navigation GNSS block	735

tle_t	736
tled_t	736
Tlm_Dump_Reader	737
Tracking_2nd_DLL_filter	
This class implements a 2nd order DLL filter for code tracking loop	737
Tracking_2nd_PLL_filter	
This class implements a 2nd order PLL filter for carrier tracking loop	739
Tracking_Dump_Reader	740
Tracking_FLL_PLL_filter	
This class implements a hybrid FLL and PLL filter for tracking carrier loop	740
Tracking_loop_filter	
This class implements a generic 1st, 2nd or 3rd order loop filter	741
Tracking_True_Obs_Reader	742
TrackingInterface	
This abstract class represents an interface to a tracking block	743
trop_t	744
True_Observables_Reader	744
TwoBitCpxFileSignalSource	
Class that reads signals samples from a file and adapts it to a SignalSourceInterface	745
TwoBitPackedFileSignalSource	
Class that reads signals samples from a file and adapts it to a SignalSourceInterface	746
UhdSignalSource	
This class reads samples from a UHD device (see http://code.ettus.com/redmine/ettus/projects/uhd)	747
unpack_2bit_samples	
This class takes 2 bit samples that have been packed into bytes or shorts as input and generates a byte for each sample. It generates eight times as much data as is input (every two bits become 16 bits)	748
unpack_byte_2bit_cpx_samples	
This class implements conversion between byte packet samples to 2bit_cpx samples 1 byte = 2 x complex 2bit I, + 2bit Q samples	749
unpack_byte_2bit_samples	
This class implements conversion between byte packet samples to 2bit samples 1 byte = 4 2bit samples	750
unpack_byte_4bit_samples	
This class implements conversion between byte packet samples to 4bit_cpx samples 1 byte = 1 x complex 4bit I, + 4bit Q samples	750
unpack_intspir_1bit_samples	
This class implements conversion between byte packet samples to 2bit samples 1 byte = 4 2bit samples	751
unpack_spir_gss6450_samples	752
UnscentedFilter	752
url_t	753
v27_decision_t	753
v27_poly_t	753
v27_t	754
Viterbi_Decoder	
Class that implements a Viterbi decoder	754

Chapter 8

File Index

8.1 File List

Here is a list of all documented files with brief descriptions:

acq_conf.h	Class that contains all the configuration parameters for generic acquisition block based on the PCPS algorithm	757
acquisition_dump_reader.h	Helper file for unit testing	757
acquisition_interface.h	Header file of the interface to an acquisition GNSS block	758
acquisition_msg_rx.h	This is a helper class to catch the asynchronous messages emitted by an acquisition block	759
ad9361_fpga_signal_source.h	Signal source for Analog Devices front-end AD9361 connected directly to FPGA accelerators. This source implements only the AD9361 control. It is NOT compatible with conventional SDR acquisition and tracking blocks. Please use the fmcomms2 source if conventional SDR acquisition and tracking is selected in the configuration file	759
ad9361_manager.h	An Analog Devices AD9361 front-end configuration library wrapper for configure some functions via iiod link	760
agnss_ref_location.h	Interface of an Assisted GNSS REFERENCE LOCATION storage	762
agnss_ref_time.h	Interface of an Assisted GNSS REFERENCE TIME storage	762
array_signal_conditioner.h	It wraps blocks to change data type, filter and resample input data, adapted to array receiver	763
bayesian_estimation.h	Interface of a library with Bayesian noise statistic estimation	763
beamformer.h	Simple spatial filter using RAW array input and beamforming coefficients	764
beamformer_filter.h	Interface of an adapter of a digital beamformer	765
Beidou_B1I.h	Defines system parameters for BeiDou B1I signal and DNAV data	766
beidou_b1i_dll_pll_tracking.h	Interface of an adapter of a DLL+PLL tracking loop block for Beidou B1I to a TrackingInterface	768
beidou_b1i_pcps_acquisition.h	Adapts a PCPS acquisition block to an AcquisitionInterface for Beidou B1I signals	768

beidou_b1i_signal_processing.h	
This class implements various functions for BeiDou B1I signals	769
beidou_b1i_telemetry_decoder.h	
Interface of an adapter of a Beidou B1I NAV data decoder block to a TelemetryDecoderInterface	771
beidou_b1i_telemetry_decoder_gs.h	
Implementation of a BEIDOU B1I DNAV data decoder block	772
Beidou_B3I.h	
Defines system parameters for BeiDou B3I signal and DNAV data	773
beidou_b3i_dll_pll_tracking.h	
Interface of an adapter of a DLL+PLL tracking loop block for Beidou B3I to a TrackingInterface	775
beidou_b3i_pcps_acquisition.h	
Adapts a PCPS acquisition block to an AcquisitionInterface for Beidou B3I signals	776
beidou_b3i_signal_processing.h	
This class implements various functions for BeiDou B3I signals	777
beidou_b3i_telemetry_decoder.h	
Interface of an adapter of a Beidou B3I NAV data decoder block to a TelemetryDecoderInterface	779
beidou_b3i_telemetry_decoder_gs.h	
Implementation of a BEIDOU B3I DNAV data decoder block	780
Beidou_DNAV.h	
Defines system parameters for BeiDou DNAV data processing	781
beidou_dnav_almanac.h	
Interface of a Beidou DNAV Almanac storage	785
beidou_dnav_ephemeris.h	
Interface of a BEIDOU EPHEMERIS storage	786
beidou_dnav_iono.h	
Interface of a BEIDOU IONOSPHERIC MODEL storage	787
beidou_dnav_navigation_message.h	
Interface of a BeiDou DNAV Data message decoder	787
beidou_dnav_utc_model.h	
Interface of a BeiDou UTC MODEL storage	788
bits.h	
Utilities for bit manipulation of the libswiftnav library	789
byte_to_short.h	
Adapts an 8-bits sample stream (IF) to a short int stream (IF)	789
byte_x2_to_complex_byte.h	
Adapts two signed char streams into a <code>std::complex<signed char></code> stream	790
channel.h	
Interface of a GNSS channel	791
channel_event.h	
Class that defines a channel event	792
channel_fsm.h	
Interface of the State Machine for channel	792
channel_interface.h	
This class represents an interface to a channel GNSS block	793
channel_msg_receiver_cc.h	
GNU Radio block that receives asynchronous channel messages from acquisition and tracking blocks	794
channel_status_msg_receiver.h	
GNU Radio block that receives asynchronous channel messages from acquisition and tracking blocks	794
cIFFT.h	
FFT in OpenCL	795
cnav_msg.h	
Utilities for CNAV message manipulation of the libswiftnav library	796
command_event.h	
Class that defines a receiver command event	797
complex_byte_to_float_x2.h	
Adapts a <code>std::complex<signed char></code> stream into two 16-bits (short) streams	798

complex_float_to_complex_byte.h	Adapts a gr_complex stream into a std::complex<signed char> stream	799
concurrent_map.h	Interface of a thread-safe std::map	800
concurrent_queue.h	Interface of a thread-safe std::queue	800
configuration_interface.h	This class represents an interface to configuration parameters	801
conjugate_cc.h	Conjugate a stream of gr_complex	802
conjugate_ic.h	Conjugate a stream of lv_8sc_t (std::complex<char>)	802
conjugate_sc.h	Conjugate a stream of lv_16sc_t (std::complex<short>)	803
control_thread.h	Interface of the receiver control plane	804
convolutional.h	General functions used to implement convolutional encoding	805
cpu_multicorrelator.h	High optimized CPU vector multiTAP correlator class	809
cpu_multicorrelator_16sc.h	Highly optimized CPU vector multiTAP correlator class for lv_16sc_t (short int complex)	809
cpu_multicorrelator_real_codes.h	Highly optimized CPU vector multiTAP correlator class using real-valued local codes	810
cshort_to_float_x2.h	Adapts a std::complex<short> stream into two float streams	811
cuda_multicorrelator.h	Highly optimized CUDA GPU vector multiTAP correlator class	811
custom_udp_signal_source.h	Receives ip frames containing samples in UDP frame encapsulation using a high performance packet capture library (libpcap)	812
direct_resampler_conditioner.h	Interface of an adapter of a direct resampler conditioner block to a SignalConditionerInterface	813
direct_resampler_conditioner_cb.h	Nearest neighborhood resampler with std::complex<signed char> input and std::complex<signed char> output	814
direct_resampler_conditioner_cc.h	Nearest neighborhood resampler with gr_complex input and gr_complex output	814
direct_resampler_conditioner_cs.h	Nearest neighborhood resampler with std::complex<short> input and std::complex<short> output	815
display.h	Defines useful display constants	816
dll_pll_conf.h	Class that contains all the configuration parameters for generic tracking block based on a DLL and a PLL	817
dll_pll_conf_fpga.h	Class that contains all the configuration parameters for generic tracking block based on a DLL and a PLL for the FPGA	818
dll_pll_veml_tracking.h	Implementation of a code DLL + carrier PLL tracking block	818
dll_pll_veml_tracking_fpga.h	Implementation of a code DLL + carrier PLL tracking block using an FPGA	819
edc.h	Utilities for CRC computation of the libswiftnav library	820
exponential_smoother.h	Class that implements an exponential smoother	821

fec.h	Utilities for the convolutional encoder of the libswiftnav library	822
fft_base_kernels.h	FFT base kernels for OpenCL	822
fft_internal.h	Internals of FFT for OpenCL	823
file_configuration.h	A ConfigurationInterface that reads the configuration from a file	824
file_signal_source.h	Interface of a class that reads signals samples from a file and adapts it to a SignalSourceInterface	824
fir_filter.h	Adapts a gnuradio <code>gr_fir_filter</code> designed with <code>pm_remez</code>	825
flexiband_signal_source.h	Ignal Source adapter for the Teleorbit Flexiband front-end device. This adapter requires a Flexiband GNU Radio driver installed (not included with GNSS-SDR)	826
fmcomms2_signal_source.h	Interface to use SDR hardware based in FMCOMMS2 driver from analog devices, for example FMCOMMS4 and ADALM-PLUTO (PlutoSdr)	827
fpga_acquisition.h	Highly optimized FPGA vector correlator class	827
fpga_dynamic_bit_selection.h	Dynamic bit selection in the received signal	828
fpga_multicorrelator.h	FPGA vector correlator class	829
fpga_switch.h	Switch that connects the HW accelerator queues to the analog front end or the DMA	829
freq_xlating_fir_filter.h	Adapts a gnuradio <code>gr_freq_xlating_fir_filter</code> designed with <code>gr_remez</code>	830
front_end_cal.h	Interface of the Front-end calibration program	831
galileo_almanac.h	Interface of a Galileo ALMANAC storage	831
galileo_almanac_helper.h	Interface of a Galileo ALMANAC storage helper	832
Galileo_E1.h	Defines system parameters for Galileo E1 signal and NAV data	832
galileo_e1_dll_pll_veml_tracking.h	Adapts a DLL+PLL VEML (Very Early Minus Late) tracking loop block to a TrackingInterface for Galileo E1 signals	836
galileo_e1_dll_pll_veml_tracking_fpga.h	Adapts a DLL+PLL VEML (Very Early Minus Late) tracking loop block to a TrackingInterface for Galileo E1 signals for the FPGA	837
galileo_e1_pcps_8ms_ambiguous_acquisition.h	Adapts a PCPS 8ms acquisition block to an AcquisitionInterface for Galileo E1 Signals	838
galileo_e1_pcps_ambiguous_acquisition.h	Adapts a PCPS acquisition block to an AcquisitionInterface for Galileo E1 Signals	838
galileo_e1_pcps_ambiguous_acquisition_fpga.h	Adapts a PCPS acquisition block to an AcquisitionInterface for Galileo E1 Signals for the FPGA	839
galileo_e1_pcps_cccwsr_ambiguous_acquisition.h	Adapts a PCPS CCCWSR acquisition block to an AcquisitionInterface for Galileo E1 Signals	840
galileo_e1_pcps_quicksync_ambiguous_acquisition.h	Adapts a PCPS acquisition block to an AcquisitionInterface for Galileo E1 Signals	840
galileo_e1_pcps_tong_ambiguous_acquisition.h	Adapts a PCPS Tong acquisition block to an AcquisitionInterface for Galileo E1 Signals	841
galileo_e1_signal_processing.h	This library implements various functions for Galileo E1 signals	842

galileo_e1_tcp_connector_tracking.h	Interface of an adapter of a TCP connector block based on code DLL + carrier PLL for Galileo E1 to a TrackingInterface	844
galileo_e1_tcp_connector_tracking_cc.h	Interface of a TCP connector block based on code DLL + carrier PLL VEML (Very Early Minus Late) tracking block for Galileo E1 signals	845
galileo_e1b_telemetry_decoder.h	Interface of an adapter of a GALILEO E1B NAV data decoder block to a TelemetryDecoderInterface	846
galileo_e5_signal_processing.h	This library implements various functions for Galileo E5 signals such as replica code generation	846
Galileo_E5a.h	Defines system parameters for Galileo E5a signal and NAV data	848
galileo_e5a_dll_pll_tracking.h	Adapts a code DLL + carrier PLL tracking block to a TrackingInterface for Galileo E5a signals .	852
galileo_e5a_dll_pll_tracking_fpga.h	Adapts a code DLL + carrier PLL tracking block to a TrackingInterface for Galileo E5a signals for the FPGA	853
galileo_e5a_noncoherent_iq_acquisition_caf.h	Adapts a PCPS acquisition block to an AcquisitionInterface for Galileo E5a data and pilot Signals	854
galileo_e5a_noncoherent_iq_acquisition_caf_cc.h	Adapts a PCPS acquisition block to an AcquisitionInterface for Galileo E5a data and pilot Signals	854
galileo_e5a_pcps_acquisition.h	Adapts a PCPS acquisition block to an AcquisitionInterface for Galileo E5a data and pilot Signals	855
galileo_e5a_pcps_acquisition_fpga.h	Adapts a PCPS acquisition block to an AcquisitionInterface for Galileo E5a data and pilot Signals for the FPGA	856
galileo_e5a_telemetry_decoder.h	Interface of an adapter of a GALILEO E5a FNAV data decoder block to a TelemetryDecoderInterface	857
Galileo_E5b.h	Defines system parameters for Galileo E5b signal and NAV data	857
galileo_e5b_dll_pll_tracking.h	Adapts a code DLL + carrier PLL tracking block to a TrackingInterface for Galileo E5b signals .	861
galileo_e5b_pcps_acquisition.h	Adapts a PCPS acquisition block to an AcquisitionInterface for Galileo E5b data and pilot Signals	861
galileo_e5b_pcps_acquisition_fpga.h	Adapts a PCPS acquisition block to an AcquisitionInterface for Galileo E5b data and pilot Signals for the FPGA	862
galileo_e5b_telemetry_decoder.h	Interface of an adapter of a GALILEO E5B NAV data decoder block to a TelemetryDecoderInterface	863
galileo_ephemeris.h	Interface of a Galileo EPHEMERIS storage	864
Galileo_FNAV.h	Galileo FNAV message constants	864
galileo_fnav_message.h	Implementation of a Galileo F/NAV Data message as described in Galileo OS SIS ICD Issue 1.2 (Nov. 2015)	868
Galileo_INAV.h	Galileo INAV message constants	868
galileo_inav_message.h	Implementation of a Galileo I/NAV Data message as described in Galileo OS SIS ICD Issue 1.2 (Nov. 2015)	873
galileo_iono.h	Interface of a Galileo Ionospheric Model storage	874

galileo_pcps_8ms_acquisition_cc.h	
This class implements a Parallel Code Phase Search Acquisition for Galileo E1 signals with coherent integration time = 8 ms (two codes)	875
galileo_telemetry_decoder_gs.h	
Implementation of a Galileo unified INAV and FNAV message demodulator block	876
galileo_utc_model.h	
Interface of a Galileo UTC MODEL storage	877
gen_signal_source.h	
It wraps blocks that generates synthesized GNSS signal and filters it	878
geofunctions.h	
A set of coordinate transformations functions and helpers, some of them migrated from MATLAB, for geographic information systems	878
geojson_printer.h	
Interface of a class that prints PVT solutions in GeoJSON format	883
glonass_gnav_almanac.h	
Interface of a GLONASS GNAV ALMANAC storage	884
glonass_gnav_ephemeris.h	
Interface of a GLONASS EPHEMERIS storage	885
glonass_gnav_navigation_message.h	
Interface of a GLONASS GNAV Data message decoder as described in GLONASS ICD (Edition 5.1)	885
glonass_gnav_utc_model.h	
Interface of a GLONASS GNAV UTC MODEL storage	886
glonass_l1_ca_dll_pll_c_aid_tracking.h	
Interface of an adapter of a DLL+PLL tracking loop block for Glonass L1 C/A to a TrackingInterface	887
glonass_l1_ca_dll_pll_c_aid_tracking_cc.h	
Implementation of a code DLL + carrier PLL tracking block	888
glonass_l1_ca_dll_pll_c_aid_tracking_sc.h	
Implementation of a code DLL + carrier PLL tracking block	889
glonass_l1_ca_dll_pll_tracking.h	
Interface of an adapter of a DLL+PLL tracking loop block for Glonass L1 C/A to a TrackingInterface	890
glonass_l1_ca_dll_pll_tracking_cc.h	
Implementation of a code DLL + carrier PLL tracking block	890
glonass_l1_ca_pcps_acquisition.h	
Adapts a PCPS acquisition block to an AcquisitionInterface for Glonass L1 C/A signals	891
glonass_l1_ca_telemetry_decoder.h	
Interface of an adapter of a GLONASS L1 C/A NAV data decoder block to a TelemetryDecoderInterface	892
glonass_l1_ca_telemetry_decoder_gs.h	
Implementation of a GLONASS L1 C/A NAV data decoder block	893
GLONASS_L1_L2_CA.h	
Defines system parameters for GLONASS L1 C/A signal and NAV data	894
glonass_l1_signal_processing.h	
This class implements various functions for GLONASS L1 CA signals	908
glonass_l2_ca_dll_pll_c_aid_tracking.h	
Interface of an adapter of a DLL+PLL tracking loop block for Glonass L2 C/A to a TrackingInterface	909
glonass_l2_ca_dll_pll_c_aid_tracking_cc.h	
Implementation of a code DLL + carrier PLL tracking block	910
glonass_l2_ca_dll_pll_c_aid_tracking_sc.h	
Implementation of a code DLL + carrier PLL tracking block	911
glonass_l2_ca_dll_pll_tracking.h	
Interface of an adapter of a DLL+PLL tracking loop block for Glonass L2 C/A to a TrackingInterface	912
glonass_l2_ca_dll_pll_tracking_cc.h	
Implementation of a code DLL + carrier PLL tracking block	913

glonass_l2_ca_pcps_acquisition.h	Adapts a PCPS acquisition block to an AcquisitionInterface for Glonass L2 C/A signals	914
glonass_l2_ca_telemetry_decoder.h	Interface of an adapter of a GLONASS L2 C/A NAV data decoder block to a TelemetryDecoderInterface	914
glonass_l2_ca_telemetry_decoder_gs.h	Implementation of a GLONASS L2 C/A NAV data decoder block	915
glonass_l2_signal_processing.h	This class implements various functions for GLONASS L2 CA signals	916
gn3s_signal_source.h	GN3S USB dongle GPS RF front-end signal sampler driver	917
gnmax_signal_source.h	GnMAX2769 USB dongle GPS RF front-end signal sampler driver	918
gnss_block_factory.h	Interface of a factory that returns smart pointers to GNSS blocks	919
gnss_block_interface.h	This interface represents a GNSS block	919
gnss_circular_deque.h	This class implements a circular deque for Gnss_Synchro	920
gnss_flowgraph.h	Interface of a GNSS receiver flow graph	921
gnss_frequencies.h	GNSS Frequencies	921
gnss_obs_codes.h	GNSS Observable codes	926
gnss_satellite.h	Interface of the Gnss_Satellite class	940
gnss_sdr_create_directory.h	Create a directory	940
gnss_sdr_flags.h	Helper file for gnss-sdr cmdline flags	941
gnss_sdr_fpga_sample_counter.h	Simple block to report the current receiver time based on the output of the tracking or telemetry blocks	945
gnss_sdr_make_unique.h	This file implements std::make_unique for C++11	946
gnss_sdr_sample_counter.h	Simple block to report the current receiver time based on the output of the tracking or telemetry blocks	947
gnss_sdr_supl_client.h	Class that implements a C++ interface to external Secure User Location Protocol (SUPL) client library	948
gnss_sdr_time_counter.h	Simple block to report the current receiver time based on the output of the tracking or telemetry blocks	949
gnss_sdr_valve.h	Interface of a GNU Radio block that sends a STOP message to the control queue right after a specific number of samples have passed through it	949
gnss_signal.h	Implementation of the Gnss_Signal class	950
gnss_signal_processing.h	This library gathers a few functions used by the algorithms of gnss-sdr, regardless of system used	951
gnss_synchro.h	Interface of the Gnss_Synchro class	953
gnss_synchro_monitor.h	Interface of a receiver monitoring block which allows sending a data stream with the receiver internal parameters (Gnss_Synchro objects) to local or remote clients over UDP	953

gnss_synchro_udp_sink.h	Interface of a class that sends serialized Gnss_Synchro objects over udp to one or multiple endpoints	954
gnuplot_i.h	A C++ interface to gnuplot	955
gps_acq_assist.h	Interface of a GPS RRLL ACQUISITION ASSISTANCE storage	956
gps_almanac.h	Interface of a GPS ALMANAC storage	957
GPS_CNAV.h	Defines parameters for GPS CNAV	957
gps_cnav_ephemeris.h	Interface of a GPS CNAV EPHEMERIS storage	960
gps_cnav_iono.h	Interface of a GPS CNAV IONOSPHERIC MODEL storage	961
gps_cnav_navigation_message.h	Interface of a GPS CNAV Data message decoder	961
gps_cnav_utc_model.h	Interface of a GPS CNAV UTC MODEL storage	962
gps_ephemeris.h	Interface of a GPS EPHEMERIS storage	963
gps_iono.h	Interface of a GPS IONOSPHERIC MODEL storage	963
GPS_L1_CA.h	Defines system parameters for GPS L1 C/A signal and NAV data	964
gps_l1_ca_dll_pll_tracking.h	Interface of an adapter of a DLL+PLL tracking loop block for GPS L1 C/A to a TrackingInterface	972
gps_l1_ca_dll_pll_tracking_fpga.h	Interface of an adapter of a DLL+PLL tracking loop block for GPS L1 C/A to a TrackingInterface for the FPGA	972
gps_l1_ca_dll_pll_tracking_gpu.h	Implementation of an adapter of a DLL+PLL tracking loop block using GPU accelerated functions for GPS L1 C/A to a TrackingInterface	973
gps_l1_ca_dll_pll_tracking_gpu_cc.h	Implementation of a code DLL + carrier PLL tracking block, GPU ACCELERATED	974
gps_l1_ca_kf_tracking.h	Interface of an adapter of a DLL + Kalman carrier tracking loop block for GPS L1 C/A signals	975
gps_l1_ca_kf_tracking_cc.h	Interface of a processing block of a DLL + Kalman carrier tracking loop for GPS L1 C/A signals	975
gps_l1_ca_pcps_acquisition.h	Adapts a PCPS acquisition block to an AcquisitionInterface for GPS L1 C/A signals	976
gps_l1_ca_pcps_acquisition_fine_doppler.h	Adapts a PCPS acquisition block with fine Doppler estimation to an AcquisitionInterface for GPS L1 C/A signals	977
gps_l1_ca_pcps_acquisition_fpga.h	Adapts a PCPS acquisition block to an AcquisitionInterface for GPS L1 C/A signals for the FPGA	978
gps_l1_ca_pcps_assisted_acquisition.h	Adapts a PCPS Assisted acquisition block to an AcquisitionInterface for GPS L1 C/A signals	979
gps_l1_ca_pcps_openccl_acquisition.h	Adapts an OpenCL PCPS acquisition block to an AcquisitionInterface for GPS L1 C/A signals	979
gps_l1_ca_pcps_quicksync_acquisition.h	Adapts a PCPS acquisition block to an AcquisitionInterface for GPS L1 C/A signals implementing the QuickSync Algorithm	980
gps_l1_ca_pcps_tong_acquisition.h	Adapts a PCPS Tong acquisition block to an AcquisitionInterface for GPS L1 C/A signals	981
gps_l1_ca_tcp_connector_tracking.h	Interface of an adapter of a TCP connector block based on code DLL + carrier PLL for GPS L1 C/A to a TrackingInterface	981

gps_l1_ca_tcp_connector_tracking_cc.h	Interface of a TCP connector block based on code DLL + carrier PLL	982
gps_l1_ca_telemetry_decoder.h	Interface of an adapter of a GPS L1 C/A NAV data decoder block to a TelemetryDecoderInterface	983
gps_l1_ca_telemetry_decoder_gs.h	Interface of a NAV message demodulator block based on Kay Borre book MATLAB-based GPS receiver	984
gps_l2_m_dll_pll_tracking.h	Interface of an adapter of a DLL+PLL tracking loop block for GPS L1 C/A to a TrackingInterface	985
gps_l2_m_dll_pll_tracking_fpga.h	Interface of an adapter of a DLL+PLL tracking loop block for GPS L2C to a TrackingInterface for the FPGA	985
gps_l2_m_pcps_acquisition.h	Adapts a PCPS acquisition block to an AcquisitionInterface for GPS L2 M signals	986
gps_l2_m_pcps_acquisition_fpga.h	Adapts an FPGA-offloaded PCPS acquisition block to an AcquisitionInterface for GPS L2 M signals	987
GPS_L2C.h	Defines system parameters for GPS L2C signal	987
gps_l2c_signal.h	This class implements signal generators for the GPS L2C signals	991
gps_l2c_telemetry_decoder.h	Interface of an adapter of a GPS L2C (CNAV) data decoder block to a TelemetryDecoderInterface	992
gps_l2c_telemetry_decoder_gs.h	Interface of a CNAV message demodulator block	993
GPS_L5.h	Defines system parameters for GPS L5 signal	993
gps_l5_dll_pll_tracking.h	Interface of an adapter of a DLL+PLL tracking loop block for GPS L5 to a TrackingInterface . .	997
gps_l5_dll_pll_tracking_fpga.h	Interface of an adapter of a DLL+PLL tracking loop block for GPS L5 to a TrackingInterface for the FPGA	998
gps_l5_signal.h	This class implements signal generators for the GPS L5 signals	998
gps_l5_telemetry_decoder.h	Interface of an adapter of a GPS L5 (CNAV) data decoder block to a TelemetryDecoderInterface	1001
gps_l5_telemetry_decoder_gs.h	Interface of a CNAV message demodulator block	1001
gps_l5i_pcps_acquisition.h	Adapts a PCPS acquisition block to an AcquisitionInterface for GPS L5i signals	1002
gps_l5i_pcps_acquisition_fpga.h	Adapts a PCPS acquisition block to an AcquisitionInterface for GPS L5i signals for the FPGA .	1003
gps_navigation_message.h	Interface of a GPS NAV Data message decoder	1004
gps_sdr_signal_processing.h	This class implements various functions for GPS L1 CA signals	1004
gps_utc_model.h	Interface of a GPS UTC MODEL storage	1006
gpx_printer.h	Interface of a class that prints PVT information to a gpx file	1007
gr_complex_ip_packet_source.h	Receives ip frames containing samples in UDP frame encapsulation using a high performance packet capture library (libpcap)	1008
hybrid_observables.h	Implementation of an adapter of an observables block accepting all kind of signals to a ObservablesInterface	1008
hybrid_observables_gs.h	Interface of the observables computation block	1009

ibyte_to_cbyte.h	Adapts an I/Q interleaved byte (unsigned char) sample stream into a <code>std::complex<unsigned char></code> stream	1010
ibyte_to_complex.h	Adapts an I/Q interleaved byte integer sample stream to a <code>gr_complex</code> (float) stream	1011
ibyte_to_cshort.h	Adapts a short interleaved sample stream into a <code>std::complex<short></code> stream	1011
in_memory_configuration.h	A ConfigurationInterface for testing purposes	1012
ini.h	This function parses an INI file into easy-to-access name/value pairs	1013
INIReader.h	This class reads an INI file into easy-to-access name/value pairs	1013
interleaved_byte_to_complex_byte.h	Adapts an 8-bits interleaved sample stream into a 16-bits complex stream	1014
interleaved_byte_to_complex_short.h	Adapts a byte (8-bits) interleaved sample stream into a <code>std::complex<short></code> stream	1015
interleaved_short_to_complex_short.h	Adapts a short (16-bits) interleaved sample stream into a <code>std::complex<short></code> stream	1016
ishort_to_complex.h	Adapts an I/Q interleaved short integer sample stream to a <code>gr_complex</code> (float) stream	1017
ishort_to_cshort.h	Adapts a short interleaved sample stream into a <code>std::complex<short></code> stream	1017
item_type_helpers.h	Utility functions for converting between item types	1018
kml_printer.h	Interface of a class that prints PVT information to a kml file	1020
labsat23_source.h	Unpacks the Labsat 2 (ls2) and (ls3) capture files	1021
labsat_signal_source.h	Labsat 2 and 3 front-end signal sampler driver	1021
lock_detectors.h	Interface of a library with a set of code and carrier phase lock detectors	1022
MATH_CONSTANTS.h	Defines useful mathematical constants and their scaled versions	1024
mmse_resampler_conditioner.h	Interface of an adapter of a mmse resampler conditioner block to a <code>SignalConditionerInterface</code>	1043
monitor_pvt.h	Interface of the Monitor_Pvt class	1044
monitor_pvt_udp_sink.h	Interface of a class that sends serialized Monitor_Pvt objects over udp to one or multiple end-points	1044
multichannel_file_signal_source.h	Implementation of a class that reads signals samples from files at different frequency band and adapts it to a <code>SignalSourceInterface</code>	1045
nmea_printer.h	Interface of a NMEA 2.1 printer for GNSS-SDR This class provides a implementation of a subset of the NMEA-0183 standard for interfacing marine electronic devices as defined by the National Marine Electronics Association (NMEA). See https://www.nmea.org/ for the NMEA 183 standard	1046
nonlinear_tracking.h	Interface of a library for nonlinear tracking algorithms	1046
notch_cc.h	Implements a notch filter algorithm	1047
notch_filter.h	Adapter of a multistate Notch filter	1048
notch_filter_lite.h	Adapts a light version of a multistate notch filter	1049

notch_lite_cc.h	Implements a notch filter light algorithm	1049
nsr_file_signal_source.h	Implementation of a class that reads signals samples from a NSR 2 bits sampler front-end file and adapts it to a SignalSourceInterface. More information about the front-end here http://www.ifen.com/products/sx-scientific-gnss-solutions/nsr-software-receiver.html	1050
obs_conf.h	Class that contains all the configuration parameters for generic observables block	1051
obsdiff_flags.h	Helper file for unit testing	1052
observable_tests_flags.h	Helper file for unit testing	1052
observables_dump_reader.h	Helper file for unit testing	1053
observables_interface.h	This class represents an interface to an Observables block	1054
osmosdr_signal_source.h	Signal source wrapper for OsmoSDR-compatible front-ends, such as HackRF or Realtek's RTL2832U-based USB dongle DVB-T receivers (see https://osmocom.org/projects/rtl-sdr/wiki for more information)	1054
pass_through.h	Interface of a block that just puts its input in its output	1055
pcps_acquisition.h	This class implements a Parallel Code Phase Search Acquisition	1056
pcps_acquisition_fine_doppler_cc.h	This class implements a Parallel Code Phase Search Acquisition with multi-dwells and fine Doppler estimation for GPS L1 C/A signal	1057
pcps_acquisition_fpga.h	This class implements a Parallel Code Phase Search Acquisition for the FPGA	1058
pcps_assisted_acquisition_cc.h	This class implements a Parallel Code Phase Search Acquisition with assistance and multi-dwells	1059
pcps_cccwsr_acquisition_cc.h	This class implements a Parallel Code Phase Search acquisition with Coherent Channel Combining With Sign Recovery scheme	1060
pcps_opencl_acquisition_cc.h	This class implements a Parallel Code Phase Search Acquisition using OpenCL to offload some functions to the GPU	1061
pcps_quicksync_acquisition_cc.h	This class implements a Parallel Code Phase Search Acquisition with the QuickSync Algorithm	1063
pcps_tong_acquisition_cc.h	This class implements a Parallel Code Phase Search Acquisition with Tong algorithm	1064
plutosdr_signal_source.h	Signal source for PlutoSDR	1066
position_test_flags.h	Helper file for unit testing	1066
pulse_blanking_cc.h	Implements a pulse blanking algorithm	1067
pulse_blanking_filter.h	Instantiates the GNSS-SDR pulse blanking filter	1068
pvt_conf.h	Class that contains all the configuration parameters for the PVT block	1069
pvt_interface.h	This class represents an interface to a PVT block	1070
pvt_solution.h	Interface of a base class for a PVT solution	1070

raw_array_signal_source.h	
CTTC Experimental GNSS 8 channels array signal source	1071
rinex_printer.h	
Interface of a RINEX 2.11 / 3.01 printer See ftp://igs.org/pub/data/format/rinex301.pdf	1072
rtcm.h	
Interface for the RTCM 3.2 Standard	1073
rtcm_printer.h	
Interface of a RTCM 3.2 printer for GNSS-SDR This class provides a implementation of a subset of the RTCM Standard 10403.2 for Differential GNSS Services	1074
rtklib.h	
Main header file for the rtklib library	1074
rtklib_conversions.h	
GNSS-SDR to RTKLIB data structures conversion functions	1113
rtklib_ephemeris.h	
Satellite ephemeris and clock functions	1114
rtklib_ionex.h	
Ionex functions	1115
rtklib_lambda.h	
Integer ambiguity resolution	1116
rtklib_pntpos.h	
Standard code-based positioning	1118
rtklib_ppp.h	
Precise Point Positioning	1121
rtklib_preceph.h	
Precise ephemeris and clock functions	1123
rtklib_pvt.h	
Interface of a Position Velocity and Time computation block	1125
rtklib_pvt_gs.h	
Interface of a Position Velocity and Time computation block	1125
rtklib_rtcn.h	
RTCM functions headers	1126
rtklib_rtcn2.h	
RTCM v2 functions headers	1127
rtklib_rtcn3.h	
RTCM v3 functions headers	1128
rtklib_rtkcmn.h	
Rtklib common functions	1133
rtklib_rtkpos.h	
Rtklib ppp-related functions	1137
rtklib_rtksvr.h	
Rtk server functions	1139
rtklib_sbas.h	
Sbas functions	1141
rtklib_solution.h	
Solution functions headers	1144
rtklib_solver.h	
PVT solver based on rtklib library functions adapted to the GNSS-SDR data flow and structures	1146
rtklib_solver_dump_reader.h	
Helper file for unit testing	1147
rtklib_stream.h	
Streaming functions	1147
rtklib_tides.h	
Tidal displacement corrections	1150
rtl_tcp_commands.h	
Defines structures and constants for communicating with rtl_tcp	1151
rtl_tcp_dongle_info.h	
Interface for a structure sent by rtl_tcp defining the hardware	1152

rtl_tcp_signal_source.h	Signal source which reads from rtl_tcp. (see https://osmocom.org/projects/rtl-sdr/wiki for more information)	1153
rtl_tcp_signal_source_c.h	Interface of an rtl_tcp signal source reader	1154
sbas_ephemeris.h	Interface of a SBAS REFERENCE LOCATION storage	1155
sbas_l1_telemetry_decoder.h	Interface of an adapter of a SBAS telemetry data decoder block to a TelemetryDecoderInterface	1155
sbas_l1_telemetry_decoder_gs.h	Interface of a SBAS telemetry data decoder block	1156
serdes_gnss_synchro.h	Serialization / Deserialization of Gnss_Synchro objects using Protocol Buffers	1157
serdes_monitor_pvt.h	Serialization / Deserialization of Monitor_Pvt objects using Protocol Buffers	1158
short_x2_to_cshort.h	Adapts two short streams into a std::complex<short> stream	1158
signal_conditioner.h	It wraps blocks to change data type, filter and resample input data	1159
signal_generator.h	Adapter of a class that generates synthesized GNSS signal	1160
signal_generator_c.h	GNU Radio source block that generates synthesized GNSS signal	1160
signal_generator_flags.h	Helper file for unit testing	1162
spir_file_signal_source.h	Implementation of a class that reads signals samples from a SPIR file and adapts it to a SignalSourceInterface	1163
spir_gss6450_file_signal_source.h	Implementation of a class that reads signals samples from a SPIR file and adapts it to a SignalSourceInterface	1163
spirent_motion_csv_dump_reader.h	Helper file for unit testing	1164
string_converter.h	Interface of a class that interprets the contents of a string and converts it into different types	1165
swift_common.h	Common definitions used throughout the libswiftnav library	1166
tcp_cmd_interface.h	Class that implements a TCP/IP telecommand command line interface for GNSS-SDR	1166
tcp_communication.h	Interface of the TCP communication class	1167
tcp_packet_data.h	Interface of the TCP data packet class	1168
telemetry_decoder_interface.h	This class represents an interface to a telemetry decoder block	1168
test_flags.h	Helper file for unit testing	1169
tlm_dump_reader.h	Helper file for unit testing	1170
tracking_2nd_DLL_filter.h	Interface of a 2nd order DLL filter for code tracking loop	1170
tracking_2nd_PLL_filter.h	Interface of a 2nd order PLL filter for carrier tracking loop	1171
tracking_discriminators.h	Interface of a library with a set of code tracking and carrier tracking discriminators	1171
tracking_dump_reader.h	Helper file for unit testing	1174

tracking_FLL_PLL_filter.h	Interface of a hybrid FLL and PLL filter for tracking carrier loop	1175
tracking_interface.h	This class represents an interface to a tracking block	1175
tracking_loop_filter.h	Generic 1st to 3rd order loop filter implementation	1176
tracking_tests_flags.h	Helper file for unit testing	1177
tracking_true_obs_reader.h	Helper file for unit testing	1178
true_observables_reader.h	Helper file for unit testing	1179
two_bit_cpx_file_signal_source.h	Interface of a class that reads signals samples from a 2 bit complex sampler front-end file and adapts it to a SignalSourceInterface	1179
two_bit_packed_file_signal_source.h	Interface of a class that reads signals samples from a file. Each sample is two bits, which are packed into bytes or shorts	1180
uhd_signal_source.h	Interface for the Universal Hardware Driver signal source	1181
unpack_2bit_samples.h	Unpacks 2 bit samples samples may be packed in any of the following ways: 1) Into bytes [item == byte] 1a) Big endian ordering within the byte 1b) Little endian ordering within the byte 2) Into shorts [item == short] 2a) Big endian ordering of bytes, big endian within the byte 2b) Big endian ordering of bytes, little endian within the byte 2c) Little endian ordering of bytes, big endian within the byte 2d) Little endian ordering of bytes, little endian within the byte	1182
unpack_byte_2bit_cpx_samples.h	Unpacks byte samples to 2 bits complex samples. Packing Order Most Significant Nibble - Sample n Least Significant Nibble - Sample n+1 Packing order in Nibble Q1 Q0 I1 I0	1183
unpack_byte_2bit_samples.h	Unpacks byte samples to NSR 2 bits samples	1184
unpack_byte_4bit_samples.h	Unpacks byte samples to 4 bits samples. Packing Order Packing order in Nibble I0 I1 I2 I3 I0 I1 I2 I3	1184
unpack_intspir_1bit_samples.h	Unpacks SPIR int samples to NSR 1 bit samples	1185
unpack_spir_gss6450_samples.h	Unpacks SPIR int samples	1186
viterbi_decoder.h	Interface of a Viterbi decoder class based on the Iterative Solutions Coded Modulation Library by Matthew C. Valenti	1187

Chapter 9

Module Documentation

9.1 Common definitions

Macros

- `#define ABS(x) ((x) < 0 ? -(x) : (x))`
- `#define MIN(x, y) (((x) < (y)) ? (x) : (y))`
- `#define MAX(x, y) (((x) > (y)) ? (x) : (y))`
- `#define CLAMP_DIFF(a, b) (MAX((a), (b)) - (b))`

9.1.1 Detailed Description

Common definitions used throughout the library.

9.2 GPS_L2

Modules

- [Gps_cnav_decoder](#)

9.2.1 Detailed Description

9.3 Gps_cnav_decoder

Classes

- struct [cnav_msg_t](#)
- struct [cnav_v27_part_t](#)
- struct [cnav_msg_decoder_t](#)

Macros

- `#define GPS_L2_V27_HISTORY_LENGTH_BITS 64`
- `#define GPS_L2C_V27_INIT_BITS (32)`
- `#define GPS_L2C_V27_DECODE_BITS (32)`
- `#define GPS_L2C_V27_DELAY_BITS (32)`

Functions

- `const v27_poly_t * cnav_msg_decoder_get_poly (void)`
- `void cnav_msg_decoder_init (cnav_msg_decoder_t *dec)`
- `bool cnav_msg_decoder_add_symbol (cnav_msg_decoder_t *dec, unsigned char symbol, cnav_msg_t *msg, uint32_t *delay)`

9.3.1 Detailed Description

9.3.2 Macro Definition Documentation

9.3.2.1 GPS_L2_V27_HISTORY_LENGTH_BITS

```
#define GPS_L2_V27_HISTORY_LENGTH_BITS 64
```

Size of the Viterbi decoder history.

Definition at line 41 of file `cnav_msg.h`.

9.3.2.2 GPS_L2C_V27_DECODE_BITS

```
#define GPS_L2C_V27_DECODE_BITS (32)
```

Bits to decode at a time.

Definition at line 45 of file `cnav_msg.h`.

9.3.2.3 GPS_L2C_V27_DELAY_BITS

```
#define GPS_L2C_V27_DELAY_BITS (32)
```

Bits in decoder tail. We ignore them.

Definition at line 47 of file cnav_msg.h.

9.3.2.4 GPS_L2C_V27_INIT_BITS

```
#define GPS_L2C_V27_INIT_BITS (32)
```

Bits to accumulate before decoding starts.

Definition at line 43 of file cnav_msg.h.

Chapter 10

Class Documentation

10.1 Acq_Conf Class Reference

Public Member Functions

- void **SetFromConfiguration** (const [ConfigurationInterface](#) *configuration, const std::string &role, double chip_rate, double opt_freq)

Public Attributes

- std::string **item_type**
- std::string **dump_filename**
- int64_t **fs_in**
- int64_t **resampled_fs**
- size_t **it_size**
- float **doppler_step**
- float **samples_per_ms**
- float **doppler_step2**
- float **pfa**
- float **pfa2**
- float **samples_per_code**
- float **resampler_ratio**
- uint32_t **sampled_ms**
- uint32_t **ms_per_code**
- uint32_t **samples_per_chip**
- uint32_t **chips_per_second**
- uint32_t **max_dwells**
- uint32_t **num_doppler_bins_step2**
- uint32_t **resampler_latency_samples**
- uint32_t **dump_channel**
- int32_t **doppler_max**
- int32_t **doppler_min**
- bool **bit_transition_flag**
- bool **use_CFAR_algorithm_flag**
- bool **dump**
- bool **blocking**
- bool **blocking_on_standby**
- bool **make_2_steps**
- bool **use_automatic_resampler**

10.1.1 Detailed Description

Definition at line 28 of file `acq_conf.h`.

The documentation for this class was generated from the following file:

- [acq_conf.h](#)

10.2 Acquisition_Dump_Reader Class Reference

Public Member Functions

- **Acquisition_Dump_Reader** (const std::string &basename, unsigned int sat, unsigned int doppler_max, unsigned int doppler_step, unsigned int samples_per_code, int channel=0, int execution=1)
- **Acquisition_Dump_Reader** (const std::string &basename, int channel=0, int execution=1)
- **Acquisition_Dump_Reader** (const [Acquisition_Dump_Reader](#) &other) noexcept
Copy constructor.
- **Acquisition_Dump_Reader** & **operator=** (const [Acquisition_Dump_Reader](#) &)
Copy assignment operator.
- **Acquisition_Dump_Reader** ([Acquisition_Dump_Reader](#) &&other) noexcept
Move constructor.
- **Acquisition_Dump_Reader** & **operator=** ([Acquisition_Dump_Reader](#) &&other) noexcept
Move assignment operator.
- bool **read_binary_acq** ()

Public Attributes

- std::vector< int > **doppler**
- std::vector< unsigned int > **samples**
- std::vector< std::vector< float > > **mag**
- float **acq_doppler_hz** {}
- float **acq_delay_samples** {}
- float **test_statistic** {}
- float **input_power** {}
- float **threshold** {}
- int **positive_acq** {}
- unsigned int **PRN** {}
- unsigned int **num_dwells** {}
- uint64_t **sample_counter** {}

10.2.1 Detailed Description

Definition at line 28 of file `acquisition_dump_reader.h`.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 Acquisition_Dump_Reader() [1/2]

```
Acquisition_Dump_Reader::Acquisition_Dump_Reader (
    const Acquisition_Dump_Reader & other ) [noexcept]
```

Copy constructor.

10.2.2.2 Acquisition_Dump_Reader() [2/2]

```
Acquisition_Dump_Reader::Acquisition_Dump_Reader (
    Acquisition_Dump_Reader && other ) [noexcept]
```

Move constructor.

10.2.3 Member Function Documentation

10.2.3.1 operator=() [1/2]

```
Acquisition_Dump_Reader& Acquisition_Dump_Reader::operator= (
    const Acquisition_Dump_Reader & )
```

Copy assignment operator.

10.2.3.2 operator=() [2/2]

```
Acquisition_Dump_Reader& Acquisition_Dump_Reader::operator= (
    Acquisition_Dump_Reader && other ) [noexcept]
```

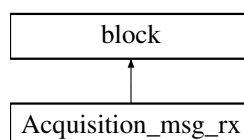
Move assignment operator.

The documentation for this class was generated from the following file:

- [acquisition_dump_reader.h](#)

10.3 Acquisition_msg_rx Class Reference

Inheritance diagram for Acquisition_msg_rx:



Public Member Functions

- [~Acquisition_msg_rx\(\)](#)

Default destructor.

Public Attributes

- int **rx_message**
- gr::top_block_sptr **top_block**

Friends

- Acquisition_msg_rx_sptr **Acquisition_msg_rx_make()**

10.3.1 Detailed Description

Definition at line 47 of file acquisition_msg_rx.h.

10.3.2 Constructor & Destructor Documentation

10.3.2.1 ~Acquisition_msg_rx()

```
Acquisition_msg_rx::~Acquisition_msg_rx ( )
```

Default destructor.

The documentation for this class was generated from the following file:

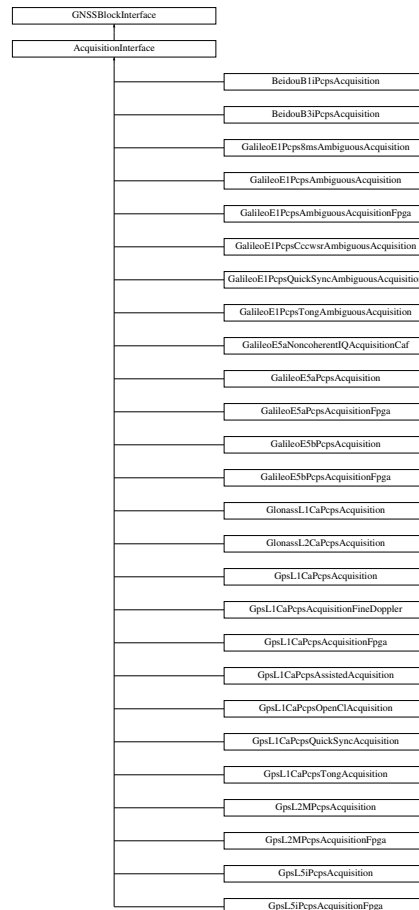
- [acquisition_msg_rx.h](#)

10.4 AcquisitionInterface Class Reference

This abstract class represents an interface to an acquisition GNSS block.

```
#include <acquisition_interface.h>
```

Inheritance diagram for AcquisitionInterface:



Public Member Functions

- virtual void **set_gnss_synchro** ([Gnss_Synchro](#) *gnss_synchro)=0
- virtual void **set_channel** (unsigned int channel_id)=0
- virtual void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm)=0
- virtual void **set_threshold** (float threshold)=0
- virtual void **set_doppler_max** (unsigned int doppler_max)=0
- virtual void **set_doppler_step** (unsigned int doppler_step)=0
- virtual void **set_doppler_center** (int doppler_center __attribute__((unused)))=0
- virtual void **init** ()=0
- virtual void **set_local_code** ()=0
- virtual void **set_state** (int state)=0
- virtual signed int **mag** ()=0
- virtual void **reset** ()=0
- virtual void **stop_acquisition** ()=0
- virtual void **set_resampler_latency** (uint32_t latency_samples)=0

10.4.1 Detailed Description

This abstract class represents an interface to an acquisition GNSS block.

Abstract class for acquisition algorithms. Since all its methods are virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

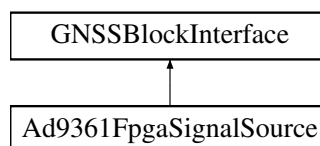
Definition at line 46 of file `acquisition_interface.h`.

The documentation for this class was generated from the following file:

- [acquisition_interface.h](#)

10.5 Ad9361FpgaSignalSource Class Reference

Inheritance diagram for Ad9361FpgaSignalSource:



Public Member Functions

- **Ad9361FpgaSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_stream, unsigned int out_stream, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
Returns "Ad9361_Fpga_Signal_Source".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.5.1 Detailed Description

Definition at line 37 of file `ad9361_fpga_signal_source.h`.

10.5.2 Member Function Documentation

10.5.2.1 implementation()

```
std::string Ad9361FpgaSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "Ad9361_Fpga_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 54 of file `ad9361_fpga_signal_source.h`.

The documentation for this class was generated from the following file:

- [ad9361_fpga_signal_source.h](#)

10.6 Agnss_Ref_Location Class Reference

Interface of an Assisted GNSS REFERENCE LOCATION storage.

```
#include <agnss_ref_location.h>
```

Public Member Functions

- [Agnss_Ref_Location](#) ()=default
- `template<class Archive >`
void [serialize](#) (Archive &archive, const unsigned int version)
Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the Ref location on disk file.

Public Attributes

- double **lat** {}
- double **lon** {}
- double **uncertainty** {}
- bool **valid** {}

10.6.1 Detailed Description

Interface of an Assisted GNSS REFERENCE LOCATION storage.

Definition at line 31 of file `agnss_ref_location.h`.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 Agnss_Ref_Location()

```
Agnss_Ref_Location::Agnss_Ref_Location ( ) [default]
```

Default constructor

10.6.3 Member Function Documentation

10.6.3.1 serialize()

```
template<class Archive >
void Agnss_Ref_Location::serialize (
    Archive & archive,
    const unsigned int version ) [inline]
```

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the Ref location on disk file.

Definition at line 49 of file agnss_ref_location.h.

The documentation for this class was generated from the following file:

- [agnss_ref_location.h](#)

10.7 Agnss_Ref_Time Class Reference

Interface of an Assisted GNSS REFERENCE TIME storage.

```
#include <agnss_ref_time.h>
```

Public Member Functions

- [Agnss_Ref_Time](#) ()=default
- template<class Archive >
void [serialize](#) (Archive &archive, const unsigned int version)
Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ref time data on disk file.

Public Attributes

- double **d_TOW** {}
- double **d_Week** {}
- double **d_tv_sec** {}
- double **d_tv_usec** {}
- bool **valid** {}

10.7.1 Detailed Description

Interface of an Assisted GNSS REFERENCE TIME storage.

Definition at line 31 of file agnss_ref_time.h.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 Agnss_Ref_Time()

```
Agnss_Ref_Time::Agnss_Ref_Time ( ) [default]
```

Default constructor

10.7.3 Member Function Documentation

10.7.3.1 serialize()

```
template<class Archive >
void Agnss_Ref_Time::serialize (
    Archive & archive,
    const unsigned int version ) [inline]
```

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ref time data on disk file.

Definition at line 50 of file agnss_ref_time.h.

The documentation for this class was generated from the following file:

- [agnss_ref_time.h](#)

10.8 alm_t Struct Reference

Public Attributes

- int **sat**
- int **svh**
- int **svconf**
- int **week**
- [gtime_t](#) **toa**
- double **A**
- double **e**
- double **i0**
- double **OMG0**
- double **omg**
- double **M0**
- double **OMGd**
- double **toas**
- double **f0**
- double **f1**

10.8.1 Detailed Description

Definition at line 417 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.9 ambc_t Struct Reference

Public Attributes

- [gtime_t](#) **epoch** [4]
- int **n** [4]
- double **LC** [4]
- double **LCv** [4]
- int **fixcnt**
- char **flags** [MAXSAT]

10.9.1 Detailed Description

Definition at line 1048 of file rtklib.h.

The documentation for this struct was generated from the following file:

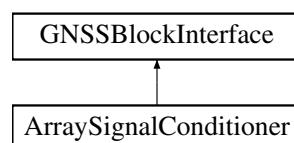
- [rtklib.h](#)

10.10 ArraySignalConditioner Class Reference

This class wraps blocks to change `data_type_adapter`, `input_filter` and `resampler` to be applied to the input flow of sampled signal.

```
#include <array_signal_conditioner.h>
```

Inheritance diagram for ArraySignalConditioner:



Public Member Functions

- [ArraySignalConditioner](#) (const [ConfigurationInterface](#) *configuration, std::shared_ptr< [GNSSBlockInterface](#) > data_type_adapt, std::shared_ptr< [GNSSBlockInterface](#) > in_filt, std::shared_ptr< [GNSSBlockInterface](#) > res, std::string role, std::string [implementation](#))
Constructor.
- [~ArraySignalConditioner](#) ()=default
Destructor.
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- std::string **role** () override
- std::string [implementation](#) () override
Returns "Array_Signal_Conditioner".
- size_t **item_size** () override
- std::shared_ptr< [GNSSBlockInterface](#) > **data_type_adapter** ()
- std::shared_ptr< [GNSSBlockInterface](#) > **input_filter** ()
- std::shared_ptr< [GNSSBlockInterface](#) > **resampler** ()

10.10.1 Detailed Description

This class wraps blocks to change data_type_adapter, input_filter and resampler to be applied to the input flow of sampled signal.

Definition at line 39 of file array_signal_conditioner.h.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 ArraySignalConditioner()

```
ArraySignalConditioner::ArraySignalConditioner (
    const ConfigurationInterface * configuration,
    std::shared_ptr< GNSSBlockInterface > data_type_adapt,
    std::shared_ptr< GNSSBlockInterface > in_filt,
    std::shared_ptr< GNSSBlockInterface > res,
    std::string role,
    std::string implementation )
```

Constructor.

10.10.2.2 ~ArraySignalConditioner()

```
ArraySignalConditioner::~~ArraySignalConditioner ( ) [default]
```

Destructor.

10.10.3 Member Function Documentation

10.10.3.1 implementation()

```
std::string ArraySignalConditioner::implementation ( ) [inline], [override], [virtual]
```

Returns "Array_Signal_Conditioner".

Implements [GNSSBlockInterface](#).

Definition at line 57 of file array_signal_conditioner.h.

The documentation for this class was generated from the following file:

- [array_signal_conditioner.h](#)

10.11 Bayesian_estimator Class Reference

[Bayesian_estimator](#) is an estimator of noise characteristics (i.e. mean, covariance)

```
#include <bayesian_estimation.h>
```

Public Member Functions

- **Bayesian_estimator** (int ny)
- **Bayesian_estimator** (const arma::vec &mu_prior_0, int kappa_prior_0, int nu_prior_0, const arma::mat &Psi_prior_0)
- void **init** (const arma::mat &mu_prior_0, int kappa_prior_0, int nu_prior_0, const arma::mat &Psi_prior_0)
- void **update_sequential** (const arma::vec &data)
- void **update_sequential** (const arma::vec &data, const arma::vec &mu_prior_0, int kappa_prior_0, int nu←_prior_0, const arma::mat &Psi_prior_0)
- arma::mat **get_mu_est** () const
- arma::mat **get_Psi_est** () const

10.11.1 Detailed Description

[Bayesian_estimator](#) is an estimator of noise characteristics (i.e. mean, covariance)

[Bayesian_estimator](#) is an estimator which performs estimation of noise characteristics from a sequence of identically and independently distributed (IID) samples of a stationary stochastic process by way of Bayesian inference using conjugate priors. The posterior distribution is assumed to be Gaussian with mean {} and covariance {{C}}, which has a conjugate prior given by a normal-inverse-Wishart distribution with paramemters {{0}}, {0}, {0}, and {}.

[1] TODO: Ref1

Definition at line 57 of file bayesian_estimation.h.

The documentation for this class was generated from the following file:

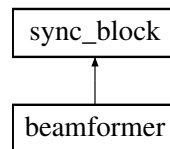
- [bayesian_estimation.h](#)

10.12 beamformer Class Reference

This class implements a real-time software-defined spatial filter using the CTTC GNSS experimental antenna array input and a set of dynamically reloadable weights.

```
#include <beamformer.h>
```

Inheritance diagram for beamformer:



Public Member Functions

- `int work (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)`

Friends

- `beamformer_sptr make_beamformer_sptr ()`

10.12.1 Detailed Description

This class implements a real-time software-defined spatial filter using the CTTC GNSS experimental antenna array input and a set of dynamically reloadable weights.

Definition at line 46 of file `beamformer.h`.

The documentation for this class was generated from the following file:

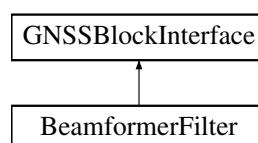
- [beamformer.h](#)

10.13 BeamformerFilter Class Reference

Interface of an adapter of a digital beamformer block to a [GNSSBlockInterface](#).

```
#include <beamformer_filter.h>
```

Inheritance diagram for BeamformerFilter:



Public Member Functions

- **BeamformerFilter** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_stream, unsigned int out_stream)
- std::string **role** () override
- std::string **implementation** () override
returns "Beamformer_Filte"
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.13.1 Detailed Description

Interface of an adapter of a digital beamformer block to a [GNSSBlockInterface](#).

Definition at line 35 of file beamformer_filter.h.

10.13.2 Member Function Documentation

10.13.2.1 implementation()

```
std::string BeamformerFilter::implementation ( ) [inline], [override], [virtual]
```

returns "Beamformer_Filte"

Implements [GNSSBlockInterface](#).

Definition at line 50 of file beamformer_filter.h.

The documentation for this class was generated from the following file:

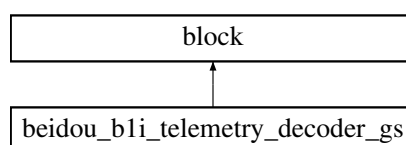
- [beamformer_filter.h](#)

10.14 beidou_b1i_telemetry_decoder_gs Class Reference

This class implements a block that decodes the BeiDou DNAV data.

```
#include <beidou_b1i_telemetry_decoder_gs.h>
```

Inheritance diagram for beidou_b1i_telemetry_decoder_gs:



Public Member Functions

- [~beidou_b1i_telemetry_decoder_gs](#) ()
Class destructor.
- void [set_satellite](#) (const [Gnss_Satellite](#) &satellite)
Set satellite PRN.
- void [set_channel](#) (int channel)
Set receiver's channel.
- void **reset** ()
- int [general_work](#) (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
This is where all signal processing takes place.

Friends

- beidou_b1i_telemetry_decoder_gs_sptr **beidou_b1i_make_telemetry_decoder_gs** (const [Gnss_Satellite](#) &satellite, bool dump)

10.14.1 Detailed Description

This class implements a block that decodes the BeiDou DNAV data.

Note

Code added as part of GSoC 2018 program

Definition at line 59 of file beidou_b1i_telemetry_decoder_gs.h.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 ~beidou_b1i_telemetry_decoder_gs()

```
beidou_b1i_telemetry_decoder_gs::~beidou_b1i_telemetry_decoder_gs ( )
```

Class destructor.

10.14.3 Member Function Documentation

10.14.3.1 `general_work()`

```
int beidou_bli_telemetry_decoder_gs::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

This is where all signal processing takes place.

10.14.3.2 `set_channel()`

```
void beidou_bli_telemetry_decoder_gs::set_channel (
    int channel )
```

Set receiver's channel.

10.14.3.3 `set_satellite()`

```
void beidou_bli_telemetry_decoder_gs::set_satellite (
    const Gnss_Satellite & satellite )
```

Set satellite PRN.

The documentation for this class was generated from the following file:

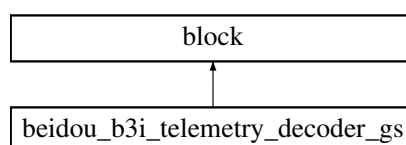
- [beidou_b1i_telemetry_decoder_gs.h](#)

10.15 `beidou_b3i_telemetry_decoder_gs` Class Reference

This class implements a block that decodes the BeiDou DNAV data.

```
#include <beidou_b3i_telemetry_decoder_gs.h>
```

Inheritance diagram for `beidou_b3i_telemetry_decoder_gs`:



Public Member Functions

- [~beidou_b3i_telemetry_decoder_gs](#) ()
Class destructor.
- void [set_satellite](#) (const [Gnss_Satellite](#) &satellite)
Set satellite PRN.
- void [set_channel](#) (int channel)
Set receiver's channel.
- void **reset** ()
- int [general_work](#) (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
This is where all signal processing takes place.

Friends

- beidou_b3i_telemetry_decoder_gs_sptr **beidou_b3i_make_telemetry_decoder_gs** (const [Gnss_Satellite](#) &satellite, bool dump)

10.15.1 Detailed Description

This class implements a block that decodes the BeiDou DNAV data.

Definition at line 55 of file beidou_b3i_telemetry_decoder_gs.h.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 ~beidou_b3i_telemetry_decoder_gs()

```
beidou_b3i_telemetry_decoder_gs::~~beidou_b3i_telemetry_decoder_gs ( )
```

Class destructor.

10.15.3 Member Function Documentation

10.15.3.1 general_work()

```
int beidou_b3i_telemetry_decoder_gs::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

This is where all signal processing takes place.

10.15.3.2 set_channel()

```
void beidou_b3i_telemetry_decoder_gs::set_channel (
    int channel )
```

Set receiver's channel.

10.15.3.3 set_satellite()

```
void beidou_b3i_telemetry_decoder_gs::set_satellite (
    const Gnss_Satellite & satellite )
```

Set satellite PRN.

The documentation for this class was generated from the following file:

- [beidou_b3i_telemetry_decoder_gs.h](#)

10.16 Beidou_Dnav_Almanac Class Reference

This class is a storage for the BeiDou D1 almanac.

```
#include <beidou_dnav_almanac.h>
```

Public Member Functions

- [Beidou_Dnav_Almanac](#) ()=default
- `template<class Archive >`
void **serialize** (Archive &ar, const unsigned int version)

Public Attributes

- unsigned int [i_satellite_PRN](#) {}
SV PRN NUMBER.
- double [d_Delta_i](#) {}
- double [d_Toa](#) {}
Almanac data reference time of week [s].
- double [d_M_0](#) {}
Mean Anomaly at Reference Time [semi-circles].
- double [d_e_eccentricity](#) {}
Eccentricity [dimensionless].
- double [d_sqrt_A](#) {}
Square Root of the Semi-Major Axis [sqrt(m)].
- double [d_OMEGA0](#) {}
Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-circles].
- double [d_OMEGA](#) {}
Argument of Perigee [semi-circles].
- double [d_OMEGA_DOT](#) {}
Rate of Right Ascension [semi-circles/s].
- int [i_SV_health](#) {}
SV Health.
- double [d_A_f0](#) {}
Coefficient 0 of code phase offset model [s].
- double [d_A_f1](#) {}
Coefficient 1 of code phase offset model [s/s].

10.16.1 Detailed Description

This class is a storage for the BeiDou D1 almanac.

Definition at line 31 of file beidou_dnav_almanac.h.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 Beidou_Dnav_Almanac()

```
Beidou_Dnav_Almanac::Beidou_Dnav_Almanac ( ) [default]
```

Default constructor

10.16.3 Member Data Documentation

10.16.3.1 d_A_f0

```
double Beidou_Dnav_Almanac::d_A_f0 {}
```

Coefficient 0 of code phase offset model [s].

Definition at line 49 of file beidou_dnav_almanac.h.

10.16.3.2 d_A_f1

```
double Beidou_Dnav_Almanac::d_A_f1 {}
```

Coefficient 1 of code phase offset model [s/s].

Definition at line 50 of file beidou_dnav_almanac.h.

10.16.3.3 d_e_eccentricity

```
double Beidou_Dnav_Almanac::d_e_eccentricity {}
```

Eccentricity [dimensionless].

Definition at line 43 of file beidou_dnav_almanac.h.

10.16.3.4 d_M_0

```
double Beidou_Dnav_Almanac::d_M_0 {}
```

Mean Anomaly at Reference Time [semi-circles].

Definition at line 42 of file beidou_dnav_almanac.h.

10.16.3.5 d_OMEGA

```
double Beidou_Dnav_Almanac::d_OMEGA {}
```

Argument of Perigee [semi-circles].

Definition at line 46 of file beidou_dnav_almanac.h.

10.16.3.6 d_OMEGA0

```
double Beidou_Dnav_Almanac::d_OMEGA0 {}
```

Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-circles].

Definition at line 45 of file beidou_dnav_almanac.h.

10.16.3.7 d_OMEGA_DOT

```
double Beidou_Dnav_Almanac::d_OMEGA_DOT {}
```

Rate of Right Ascension [semi-circles/s].

Definition at line 47 of file beidou_dnav_almanac.h.

10.16.3.8 d_sqrt_A

```
double Beidou_Dnav_Almanac::d_sqrt_A {}
```

Square Root of the Semi-Major Axis [sqrt(m)].

Definition at line 44 of file beidou_dnav_almanac.h.

10.16.3.9 d_Toa

```
double Beidou_Dnav_Almanac::d_Toa {}
```

Almanac data reference time of week [s].

Definition at line 41 of file beidou_dnav_almanac.h.

10.16.3.10 i_satellite_PRN

```
unsigned int Beidou_Dnav_Almanac::i_satellite_PRN {}
```

SV PRN NUMBER.

Definition at line 39 of file beidou_dnav_almanac.h.

10.16.3.11 i_SV_health

```
int Beidou_Dnav_Almanac::i_SV_health {}
```

SV Health.

Definition at line 48 of file beidou_dnav_almanac.h.

The documentation for this class was generated from the following file:

- [beidou_dnav_almanac.h](#)

10.17 Beidou_Dnav_Ephemeris Class Reference

This class is a storage and orbital model functions for the GPS SV ephemeris data as described in BeiDou Navigation Satellite System Signal In Space Interface Control Document Open Service Signal B1I (Version 3.0)

```
#include <beidou_dnav_ephemeris.h>
```

Public Member Functions

- [Beidou_Dnav_Ephemeris](#) ()
- double [satellitePosition](#) (double transmitTime)
Compute the ECEF SV coordinates and ECEF velocity Implementation of Table 20-IV (IS-GPS-200K) and compute the clock bias term including relativistic effect (return value)
- double [sv_clock_drift](#) (double transmitTime)
Sets (d_satClkDrift) and returns the clock drift in seconds according to the User Algorithm for SV Clock Correction (IS-GPS-200K, 20.3.3.3.3.1)
- double [sv_clock_relativistic_term](#) (double transmitTime)
Sets (d_dtr) and returns the clock relativistic correction term in seconds according to the User Algorithm for SV Clock Correction (IS-GPS-200K, 20.3.3.3.3.1)
- template<class Archive >
void [serialize](#) (Archive &archive, const unsigned int version)
Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

Public Attributes

- unsigned int [i_satellite_PRN](#) {}
SV PRN NUMBER.
- double [d_TOW](#) {}
Time of BEIDOU Week of the ephemeris set (taken from subframes TOW) [s].
- double [d_Crs](#) {}
Amplitude of the Sine Harmonic Correction Term to the Orbit Radius [m].
- double [d_Delta_n](#) {}
Mean Motion Difference From Computed Value [semi-circles/s].
- double [d_M_0](#) {}
Mean Anomaly at Reference Time [semi-circles].
- double [d_Cuc](#) {}
Amplitude of the Cosine Harmonic Correction Term to the Argument of Latitude [rad].
- double [d_eccentricity](#) {}
Eccentricity [dimensionless].
- double [d_Cus](#) {}
Amplitude of the Sine Harmonic Correction Term to the Argument of Latitude [rad].
- double [d_sqrt_A](#) {}
Square Root of the Semi-Major Axis [sqrt(m)].
- double [d_Toe](#) {}
Ephemeris data reference time of week (Ref. 20.3.3.4.3 IS-GPS-200K) [s].
- double [d_Toc](#) {}
clock data reference time (Ref. 20.3.3.3.1 IS-GPS-200K) [s]
- double [d_Cic](#) {}
Amplitude of the Cosine Harmonic Correction Term to the Angle of Inclination [rad].
- double [d_OMEGA0](#) {}
Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-circles].
- double [d_Cis](#) {}
Amplitude of the Sine Harmonic Correction Term to the Angle of Inclination [rad].
- double [d_i_0](#) {}
Inclination Angle at Reference Time [semi-circles].
- double [d_Crc](#) {}
Amplitude of the Cosine Harmonic Correction Term to the Orbit Radius [m].
- double [d_OMEGA](#) {}
Argument of Perigee [semi-circles].
- double [d_OMEGA_DOT](#) {}
Rate of Right Ascension [semi-circles/s].
- double [d_IDOT](#) {}
Rate of Inclination Angle [semi-circles/s].
- int [i_BEIDOU_week](#) {}
BEIDOU week number, aka WN [week].
- int [i_SV_accuracy](#) {}
User Range Accuracy (URA) index of the SV (reference paragraph 6.2.1) for the standard positioning service user (Ref 20.3.3.3.1.3 IS-GPS-200K)
- int [i_SV_health](#) {}
- double [d_TGD1](#) {}
Estimated Group Delay Differential on B1I [s].
- double [d_TGD2](#) {}
Estimated Group Delay Differential on B2I [s].
- double [d_AODC](#) {}

- Age of Data, Clock.*

 - double [d_AODE](#) {}
- Age of Data, Ephemeris.*

 - int [i_AODO](#) {}

Age of Data Offset (AODO) term for the navigation message correction table (NMCT) contained in subframe 4 (reference paragraph 20.3.3.5.1.9) [s].
- int [i_sig_type](#) {}

*BDS: data source (0:unknown, 1:B1I, 2:B1Q, 3:B2I, 4:B2Q, 5:B3I, 6:B3Q) */.*
- int [i_nav_type](#) {}

*BDS: nav type (0:unknown, 1:IGSO/MEO, 2:GEO) */.*
- bool [b_fit_interval_flag](#) {}

indicates the curve-fit interval used by the CS (Block II/IIA/IIR/IIR-M/IIF) and SS (Block IIIA) in determining the ephemeris parameters, as follows: 0 = 4 hours, 1 = greater than 4 hours.
- double [d_spare1](#) {}
- double [d_spare2](#) {}
- double [d_A_f0](#) {}

Coefficient 0 of code phase offset model [s].
- double [d_A_f1](#) {}

Coefficient 1 of code phase offset model [s/s].
- double [d_A_f2](#) {}

Coefficient 2 of code phase offset model [s/s²].
- bool [b_integrity_status_flag](#) {}

If true, enhanced level of integrity assurance.
- bool [b_alert_flag](#) {}

If true, indicates that the SV URA may be worse than indicated in d_SV_accuracy, use that SV at our own risk.
- bool [b_antispoofing_flag](#) {}

If true, the AntiSpoofing mode is ON in that SV.
- double [d_satClkDrift](#) {}

GPS clock error.
- double [d_dtr](#) {}

relativistic clock correction term
- double [d_satpos_X](#) {}

Earth-fixed coordinate x of the satellite [m]. Intersection of the IERS Reference Meridian (IRM) and the plane passing through the origin and normal to the Z-axis.
- double [d_satpos_Y](#) {}

Earth-fixed coordinate y of the satellite [m]. Completes a right-handed, Earth-Centered, Earth-Fixed orthogonal coordinate system.
- double [d_satpos_Z](#) {}

Earth-fixed coordinate z of the satellite [m]. The direction of the IERS (International Earth Rotation and Reference Systems Service) Reference Pole (IRP).
- double [d_satvel_X](#) {}

Earth-fixed velocity coordinate x of the satellite [m].
- double [d_satvel_Y](#) {}

Earth-fixed velocity coordinate y of the satellite [m].
- double [d_satvel_Z](#) {}

Earth-fixed velocity coordinate z of the satellite [m].
- std::map< int, std::string > [satelliteBlock](#)

Map that stores to which block the PRN belongs.

10.17.1 Detailed Description

This class is a storage and orbital model functions for the GPS SV ephemeris data as described in BeiDou Navigation Satellite System Signal In Space Interface Control Document Open Service Signal B1I (Version 3.0)

See <http://en.beidou.gov.cn/SYSTEMS/Officialdocument/201902/P020190227601370045731.pdf>.

Definition at line 36 of file beidou_dnav_ephemeris.h.

10.17.2 Constructor & Destructor Documentation

10.17.2.1 Beidou_Dnav_Ephemeris()

```
Beidou_Dnav_Ephemeris::Beidou_Dnav_Ephemeris ( )
```

Default constructor

10.17.3 Member Function Documentation

10.17.3.1 satellitePosition()

```
double Beidou_Dnav_Ephemeris::satellitePosition (
    double transmitTime )
```

Compute the ECEF SV coordinates and ECEF velocity Implementation of Table 20-IV (IS-GPS-200K) and compute the clock bias term including relativistic effect (return value)

10.17.3.2 `serialize()`

```
template<class Archive >
void Beidou_Dnav_Ephemeris::serialize (
    Archive & archive,
    const unsigned int version ) [inline]
```

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

- < Time of GPS Week of the ephemeris set (taken from subframes TOW) [s]
- < Amplitude of the Sine Harmonic Correction Term to the Orbit Radius [m]
- < Mean Motion Difference From Computed Value [semi-circles/s]
- < Mean Anomaly at Reference Time [semi-circles]
- < Amplitude of the Cosine Harmonic Correction Term to the Argument of Latitude [rad]
- < Eccentricity [dimensionless]
- < Amplitude of the Sine Harmonic Correction Term to the Argument of Latitude [rad]
- < Square Root of the Semi-Major Axis [sqrt(m)]
- < Ephemeris data reference time of week (Ref. 20.3.3.4.3 IS-GPS-200K) [s]
- < clock data reference time (Ref. 20.3.3.3.1 IS-GPS-200K) [s]
- < Amplitude of the Cosine Harmonic Correction Term to the Angle of Inclination [rad]
- < Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-circles]
- < Amplitude of the Sine Harmonic Correction Term to the Angle of Inclination [rad]
- < Inclination Angle at Reference Time [semi-circles]
- < Amplitude of the Cosine Harmonic Correction Term to the Orbit Radius [m]
- < Argument of Perigee [semi-circles]
- < Rate of Right Ascension [semi-circles/s]
- < Rate of Inclination Angle [semi-circles/s]
- < GPS week number, aka WN [week]
- < User Range Accuracy (URA) index of the SV (reference paragraph 6.2.1) for the standard positioning service user (Ref 20.3.3.1.3 IS-GPS-200K)
- < Issue of Data, Clock
- < Estimated Group Delay Differential: L1-L2 correction term only for the benefit of "L1 P(Y)" or "L2 P(Y)" s users [s]
- < Estimated Group Delay Differential: L1-L2 correction term only for the benefit of "L1 P(Y)" or "L2 P(Y)" s users [s]
- < Age of Data Offset (AODO) term for the navigation message correction table (NMCT) contained in subframe 4 (reference paragraph 20.3.3.5.1.9) [s]

< indicates the curve-fit interval used by the CS (Block II/IIA/IIR/IIR-M/IIF) and SS (Block IIIA) in determining the ephemeris parameters, as follows: 0 = 4 hours, 1 = greater than 4 hours.

< Coefficient 0 of code phase offset model [s]

< Coefficient 1 of code phase offset model [s/s]

< Coefficient 2 of code phase offset model [s/s²]

< If true, indicates that the SV URA may be worse than indicated in d_SV_accuracy, use that SV at our own risk.

< If true, the AntiSpoofing mode is ON in that SV

Definition at line 138 of file beidou_dnav_ephemeris.h.

References b_alert_flag, b_antispoofing_flag, b_fit_interval_flag, b_integrity_status_flag, d_A_f0, d_A_f1, d_A_f2, d_AODC, d_AODE, d_Cic, d_Cis, d_Crc, d_Crs, d_Cuc, d_Cus, d_Delta_n, d_eccentricity, d_i_0, d_IDOT, d_M_0, d_OMEGA, d_OMEGA0, d_OMEGA_DOT, d_sqrt_A, d_TGD1, d_TGD2, d_Toe, d_TOW, i_AODO, i_BEIDOU_↔ week, i_satellite_PRN, and i_SV_accuracy.

10.17.3.3 sv_clock_drift()

```
double Beidou_Dnav_Ephemeris::sv_clock_drift (
    double transmitTime )
```

Sets (*d_satClkDrift*) and returns the clock drift in seconds according to the User Algorithm for SV Clock Correction (IS-GPS-200K, 20.3.3.3.3.1)

10.17.3.4 sv_clock_relativistic_term()

```
double Beidou_Dnav_Ephemeris::sv_clock_relativistic_term (
    double transmitTime )
```

Sets (*d_dtr*) and returns the clock relativistic correction term in seconds according to the User Algorithm for SV Clock Correction (IS-GPS-200K, 20.3.3.3.3.1)

10.17.4 Member Data Documentation

10.17.4.1 b_alert_flag

```
bool Beidou_Dnav_Ephemeris::b_alert_flag {}
```

If true, indicates that the SV URA may be worse than indicated in d_SV_accuracy, use that SV at our own risk.

Definition at line 113 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.2 b_antispoofing_flag

```
bool Beidou_Dnav_Ephemeris::b_antispoofing_flag {}
```

If true, the AntiSpoofing mode is ON in that SV.

Definition at line 114 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.3 b_fit_interval_flag

```
bool Beidou_Dnav_Ephemeris::b_fit_interval_flag {}
```

indicates the curve-fit interval used by the CS (Block II/IIA/IIR/IIR-M/IIF) and SS (Block IIIA) in determining the ephemeris parameters, as follows: 0 = 4 hours, 1 = greater than 4 hours.

Definition at line 94 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.4 b_integrity_status_flag

```
bool Beidou_Dnav_Ephemeris::b_integrity_status_flag {}
```

If true, enhanced level of integrity assurance.

If false, indicates that the conveying signal is provided with the legacy level of integrity assurance. That is, the probability that the instantaneous URE of the conveying signal exceeds 4.42 times the upper bound value of the current broadcast URA index, for more than 5.2 seconds, without an accompanying alert, is less than 1E-5 per hour. If true, indicates that the conveying signal is provided with an enhanced level of integrity assurance. That is, the probability that the instantaneous URE of the conveying signal exceeds 5.73 times the upper bound value of the current broadcast URA index, for more than 5.2 seconds, without an accompanying alert, is less than 1E-8 per hour.

Definition at line 112 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.5 d_A_f0

```
double Beidou_Dnav_Ephemeris::d_A_f0 {}
```

Coefficient 0 of code phase offset model [s].

Definition at line 98 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.6 d_A_f1

```
double Beidou_Dnav_Ephemeris::d_A_f1 {}
```

Coefficient 1 of code phase offset model [s/s].

Definition at line 99 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.7 d_A_f2

```
double Beidou_Dnav_Ephemeris::d_A_f2 {}
```

Coefficient 2 of code phase offset model [s/s²].

Definition at line 100 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.8 d_AODC

```
double Beidou_Dnav_Ephemeris::d_AODC {}
```

Age of Data, Clock.

Definition at line 87 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.9 d_AODE

```
double Beidou_Dnav_Ephemeris::d_AODE {}
```

Age of Data, Ephemeris.

Definition at line 88 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.10 d_Cic

```
double Beidou_Dnav_Ephemeris::d_Cic {}
```

Amplitude of the Cosine Harmonic Correction Term to the Angle of Inclination [rad].

Definition at line 74 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.11 d_Cis

```
double Beidou_Dnav_Ephemeris::d_Cis {}
```

Amplitude of the Sine Harmonic Correction Term to the Angle of Inclination [rad].

Definition at line 76 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.12 d_Crc

```
double Beidou_Dnav_Ephemeris::d_Crc {}
```

Amplitude of the Cosine Harmonic Correction Term to the Orbit Radius [m].

Definition at line 78 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.13 d_Crs

```
double Beidou_Dnav_Ephemeris::d_Crs {}
```

Amplitude of the Sine Harmonic Correction Term to the Orbit Radius [m].

Definition at line 65 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.14 d_Cuc

```
double Beidou_Dnav_Ephemeris::d_Cuc {}
```

Amplitude of the Cosine Harmonic Correction Term to the Argument of Latitude [rad].

Definition at line 68 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.15 d_Cus

```
double Beidou_Dnav_Ephemeris::d_Cus {}
```

Amplitude of the Sine Harmonic Correction Term to the Argument of Latitude [rad].

Definition at line 70 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.16 d_Delta_n

```
double Beidou_Dnav_Ephemeris::d_Delta_n {}
```

Mean Motion Difference From Computed Value [semi-circles/s].

Definition at line 66 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.17 d_dtr

```
double Beidou_Dnav_Ephemeris::d_dtr {}
```

relativistic clock correction term

Definition at line 118 of file beidou_dnav_ephemeris.h.

10.17.4.18 d_eccentricity

```
double Beidou_Dnav_Ephemeris::d_eccentricity {}
```

Eccentricity [dimensionless].

Definition at line 69 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.19 d_i_0

```
double Beidou_Dnav_Ephemeris::d_i_0 {}
```

Inclination Angle at Reference Time [semi-circles].

Definition at line 77 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.20 d_IDOT

```
double Beidou_Dnav_Ephemeris::d_IDOT {}
```

Rate of Inclination Angle [semi-circles/s].

Definition at line 81 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.21 d_M_0

```
double Beidou_Dnav_Ephemeris::d_M_0 {}
```

Mean Anomaly at Reference Time [semi-circles].

Definition at line 67 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.22 d_OMEGA

```
double Beidou_Dnav_Ephemeris::d_OMEGA {}
```

Argument of Perigee [semi-circles].

Definition at line 79 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.23 d_OMEGA0

```
double Beidou_Dnav_Ephemeris::d_OMEGA0 {}
```

Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-circles].

Definition at line 75 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.24 d_OMEGA_DOT

```
double Beidou_Dnav_Ephemeris::d_OMEGA_DOT {}
```

Rate of Right Ascension [semi-circles/s].

Definition at line 80 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.25 d_satClkDrift

```
double Beidou_Dnav_Ephemeris::d_satClkDrift {}
```

GPS clock error.

Definition at line 117 of file beidou_dnav_ephemeris.h.

10.17.4.26 d_satpos_X

```
double Beidou_Dnav_Ephemeris::d_satpos_X {}
```

Earth-fixed coordinate x of the satellite [m]. Intersection of the IERS Reference Meridian (IRM) and the plane passing through the origin and normal to the Z-axis.

Definition at line 121 of file beidou_dnav_ephemeris.h.

10.17.4.27 d_satpos_Y

```
double Beidou_Dnav_Ephemeris::d_satpos_Y {}
```

Earth-fixed coordinate y of the satellite [m]. Completes a right-handed, Earth-Centered, Earth-Fixed orthogonal coordinate system.

Definition at line 122 of file beidou_dnav_ephemeris.h.

10.17.4.28 d_satpos_Z

```
double Beidou_Dnav_Ephemeris::d_satpos_Z {}
```

Earth-fixed coordinate z of the satellite [m]. The direction of the IERS (International Earth Rotation and Reference Systems Service) Reference Pole (IRP).

Definition at line 123 of file beidou_dnav_ephemeris.h.

10.17.4.29 d_satvel_X

```
double Beidou_Dnav_Ephemeris::d_satvel_X {}
```

Earth-fixed velocity coordinate x of the satellite [m].

Definition at line 126 of file beidou_dnav_ephemeris.h.

10.17.4.30 d_satvel_Y

```
double Beidou_Dnav_Ephemeris::d_satvel_Y {}
```

Earth-fixed velocity coordinate y of the satellite [m].

Definition at line 127 of file beidou_dnav_ephemeris.h.

10.17.4.31 d_satvel_Z

```
double Beidou_Dnav_Ephemeris::d_satvel_Z {}
```

Earth-fixed velocity coordinate z of the satellite [m].

Definition at line 128 of file beidou_dnav_ephemeris.h.

10.17.4.32 d_sqrt_A

```
double Beidou_Dnav_Ephemeris::d_sqrt_A {}
```

Square Root of the Semi-Major Axis [sqrt(m)].

Definition at line 71 of file beidou_dnav_ephemeris.h.

Referenced by serialize().

10.17.4.33 d_TGD1

```
double Beidou_Dnav_Ephemeris::d_TGD1 {}
```

Estimated Group Delay Differential on B1I [s].

Definition at line 85 of file beidou_dnav_ephemeris.h.

Referenced by serialize().

10.17.4.34 d_TGD2

```
double Beidou_Dnav_Ephemeris::d_TGD2 {}
```

Estimated Group Delay Differential on B2I [s].

Definition at line 86 of file beidou_dnav_ephemeris.h.

Referenced by serialize().

10.17.4.35 d_Toc

```
double Beidou_Dnav_Ephemeris::d_Toc {}
```

clock data reference time (Ref. 20.3.3.3.1 IS-GPS-200K) [s]

Definition at line 73 of file beidou_dnav_ephemeris.h.

10.17.4.36 d_Toe

```
double Beidou_Dnav_Ephemeris::d_Toe {}
```

Ephemeris data reference time of week (Ref. 20.3.3.4.3 IS-GPS-200K) [s].

Definition at line 72 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.37 d_TOW

```
double Beidou_Dnav_Ephemeris::d_TOW {}
```

Time of BEIDOU Week of the ephemeris set (taken from subframes TOW) [s].

Definition at line 64 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.38 i_AODO

```
int Beidou_Dnav_Ephemeris::i_AODO {}
```

Age of Data Offset (AODO) term for the navigation message correction table (NMCT) contained in subframe 4 (reference paragraph 20.3.3.5.1.9) [s].

Definition at line 89 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.39 i_BEIDOU_week

```
int Beidou_Dnav_Ephemeris::i_BEIDOU_week {}
```

BEIDOU week number, aka WN [week].

Definition at line 82 of file beidou_dnav_ephemeris.h.

Referenced by `serialize()`.

10.17.4.40 i_nav_type

```
int Beidou_Dnav_Ephemeris::i_nav_type {}
```

BDS: nav type (0:unknown,1:IGSO/MEO,2:GEO) */.

Definition at line 92 of file beidou_dnav_ephemeris.h.

10.17.4.41 i_satellite_PRN

```
unsigned int Beidou_Dnav_Ephemeris::i_satellite_PRN {}
```

SV PRN NUMBER.

Definition at line 63 of file beidou_dnav_ephemeris.h.

Referenced by serialize().

10.17.4.42 i_sig_type

```
int Beidou_Dnav_Ephemeris::i_sig_type {}
```

BDS: data source (0:unknown,1:B1I,2:B1Q,3:B2I,4:B2Q,5:B3I,6:B3Q) */.

Definition at line 91 of file beidou_dnav_ephemeris.h.

10.17.4.43 i_SV_accuracy

```
int Beidou_Dnav_Ephemeris::i_SV_accuracy {}
```

User Range Accuracy (URA) index of the SV (reference paragraph 6.2.1) for the standard positioning service user (Ref 20.3.3.3.1.3 IS-GPS-200K)

Definition at line 83 of file beidou_dnav_ephemeris.h.

Referenced by serialize().

10.17.4.44 satelliteBlock

```
std::map<int, std::string> Beidou_Dnav_Ephemeris::satelliteBlock
```

Map that stores to which block the PRN belongs.

Definition at line 130 of file beidou_dnav_ephemeris.h.

The documentation for this class was generated from the following file:

- [beidou_dnav_ephemeris.h](#)

10.18 Beidou_Dnav_Iono Class Reference

This class is a storage for the BEIDOU IONOSPHERIC data as described in ICD v2.1.

```
#include <beidou_dnav_iono.h>
```

Public Member Functions

- [Beidou_Dnav_Iono](#) ()=default
Default constructor.
- `template<class Archive >`
`void serialize (Archive &archive, const unsigned int version)`
Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

Public Attributes

- double [d_alpha0](#) {}
Coefficient 0 of a cubic equation representing the amplitude of the vertical delay [s].
- double [d_alpha1](#) {}
Coefficient 1 of a cubic equation representing the amplitude of the vertical delay [s/semi-circle].
- double [d_alpha2](#) {}
Coefficient 2 of a cubic equation representing the amplitude of the vertical delay [s(semi-circle)²].
- double [d_alpha3](#) {}
Coefficient 3 of a cubic equation representing the amplitude of the vertical delay [s(semi-circle)³].
- double [d_beta0](#) {}
Coefficient 0 of a cubic equation representing the period of the model [s].
- double [d_beta1](#) {}
Coefficient 1 of a cubic equation representing the period of the model [s/semi-circle].
- double [d_beta2](#) {}
Coefficient 2 of a cubic equation representing the period of the model [s(semi-circle)²].
- double [d_beta3](#) {}
Coefficient 3 of a cubic equation representing the period of the model [s(semi-circle)³].
- bool [valid](#) {}
Valid flag.

10.18.1 Detailed Description

This class is a storage for the BEIDOU IONOSPHERIC data as described in ICD v2.1.

Definition at line 31 of file beidou_dnav_iono.h.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 Beidou_Dnav_Iono()

```
Beidou_Dnav_Iono::Beidou_Dnav_Iono ( ) [default]
```

Default constructor.

10.18.3 Member Function Documentation

10.18.3.1 serialize()

```
template<class Archive >
void Beidou_Dnav_Iono::serialize (
    Archive & archive,
    const unsigned int version ) [inline]
```

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

Definition at line 53 of file beidou_dnav_iono.h.

References `d_alpha0`, `d_alpha1`, `d_alpha2`, `d_alpha3`, `d_beta0`, `d_beta1`, `d_beta2`, and `d_beta3`.

10.18.4 Member Data Documentation

10.18.4.1 d_alpha0

```
double Beidou_Dnav_Iono::d_alpha0 {}
```

Coefficient 0 of a cubic equation representing the amplitude of the vertical delay [s].

Definition at line 37 of file beidou_dnav_iono.h.

Referenced by `serialize()`.

10.18.4.2 d_alpha1

```
double Beidou_Dnav_Iono::d_alpha1 {}
```

Coefficient 1 of a cubic equation representing the amplitude of the vertical delay [s/semi-circle].

Definition at line 38 of file beidou_dnav_iono.h.

Referenced by `serialize()`.

10.18.4.3 d_alpha2

```
double Beidou_Dnav_Iono::d_alpha2 {}
```

Coefficient 2 of a cubic equation representing the amplitude of the vertical delay $[s(\text{semi-circle})^2]$.

Definition at line 39 of file beidou_dnav_iono.h.

Referenced by `serialize()`.

10.18.4.4 d_alpha3

```
double Beidou_Dnav_Iono::d_alpha3 {}
```

Coefficient 3 of a cubic equation representing the amplitude of the vertical delay $[s(\text{semi-circle})^3]$.

Definition at line 40 of file beidou_dnav_iono.h.

Referenced by `serialize()`.

10.18.4.5 d_beta0

```
double Beidou_Dnav_Iono::d_beta0 {}
```

Coefficient 0 of a cubic equation representing the period of the model [s].

Definition at line 41 of file beidou_dnav_iono.h.

Referenced by `serialize()`.

10.18.4.6 d_beta1

```
double Beidou_Dnav_Iono::d_beta1 {}
```

Coefficient 1 of a cubic equation representing the period of the model $[s/\text{semi-circle}]$.

Definition at line 42 of file beidou_dnav_iono.h.

Referenced by `serialize()`.

10.18.4.7 d_beta2

```
double Beidou_Dnav_Iono::d_beta2 {}
```

Coefficient 2 of a cubic equation representing the period of the model [s(semi-circle)²].

Definition at line 43 of file beidou_dnav_iono.h.

Referenced by `serialize()`.

10.18.4.8 d_beta3

```
double Beidou_Dnav_Iono::d_beta3 {}
```

Coefficient 3 of a cubic equation representing the period of the model [s(semi-circle)³].

Definition at line 44 of file beidou_dnav_iono.h.

Referenced by `serialize()`.

10.18.4.9 valid

```
bool Beidou_Dnav_Iono::valid {}
```

Valid flag.

Definition at line 46 of file beidou_dnav_iono.h.

The documentation for this class was generated from the following file:

- [beidou_dnav_iono.h](#)

10.19 Beidou_Dnav_Navigation_Message Class Reference

This class decodes a BeiDou D1 NAV Data message.

```
#include <beidou_dnav_navigation_message.h>
```

Public Member Functions

- [Beidou_Dnav_Navigation_Message](#) ()
- [Beidou_Dnav_Ephemeris](#) [get_ephemeris](#) () const
Obtain a BDS SV Ephemeris class filled with current SV data.
- [Beidou_Dnav_Iono](#) [get_iono](#) ()
Obtain a BDS ionospheric correction parameters class filled with current SV data.
- [Beidou_Dnav_Utc_Model](#) [get_utc_model](#) ()
Obtain a BDS UTC model parameters class filled with current SV data.
- [int32_t](#) [d1_subframe_decoder](#) (std::string const &subframe)
Decodes the BDS D1 NAV message.
- [int32_t](#) [d2_subframe_decoder](#) (std::string const &subframe)
Decodes the BDS D2 NAV message.
- void [satellitePosition](#) (double transmitTime)
Computes the position of the satellite.
- double [sv_clock_correction](#) (double transmitTime)
Sets (d_satClkCorr) according to the User Algorithm for SV Clock Correction and returns the corrected clock.
- double [utc_time](#) (const double beidoutime_corrected) const
Computes the Coordinated Universal Time (UTC) and returns it in [s].
- bool [satellite_validation](#) ()
- bool [have_new_ephemeris](#) ()
Returns true if new Ephemeris has arrived. The flag is set to false when the function is executed.
- bool [have_new_iono](#) ()
Returns true if new Iono model has arrived. The flag is set to false when the function is executed.
- bool [have_new_utc_model](#) ()
Returns true if new UTC model has arrived. The flag is set to false when the function is executed.
- bool [have_new_almanac](#) ()
Returns true if new UTC model has arrived. The flag is set to false when the function is executed.
- void [set_satellite_PRN](#) (uint32_t prn)
Sets satellite PRN number.
- void [set_signal_type](#) (int32_t signal_type)
- bool [get_flag_CRC_test](#) () const
- bool [get_flag_new_SOW_available](#) () const
- void [set_flag_new_SOW_available](#) (bool new_SOW_available)
- double [get_SOW](#) () const

10.19.1 Detailed Description

This class decodes a BeiDou D1 NAV Data message.

Definition at line 44 of file `beidou_dnav_navigation_message.h`.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 Beidou_Dnav_Navigation_Message()

```
Beidou_Dnav_Navigation_Message::Beidou_Dnav_Navigation_Message ( )
```

Default constructor

10.19.3 Member Function Documentation

10.19.3.1 d1_subframe_decoder()

```
int32_t Beidou_Dnav_Navigation_Message::d1_subframe_decoder (
    std::string const & subframe )
```

Decodes the BDS D1 NAV message.

10.19.3.2 d2_subframe_decoder()

```
int32_t Beidou_Dnav_Navigation_Message::d2_subframe_decoder (
    std::string const & subframe )
```

Decodes the BDS D2 NAV message.

10.19.3.3 get_ephemeris()

```
Beidou_Dnav_Ephemeris Beidou_Dnav_Navigation_Message::get_ephemeris ( ) const
```

Obtain a BDS SV Ephemeris class filled with current SV data.

10.19.3.4 get_iono()

```
Beidou_Dnav_Iono Beidou_Dnav_Navigation_Message::get_iono ( )
```

Obtain a BDS ionospheric correction parameters class filled with current SV data.

10.19.3.5 get_utc_model()

```
Beidou_Dnav_Utc_Model Beidou_Dnav_Navigation_Message::get_utc_model ( )
```

Obtain a BDS UTC model parameters class filled with current SV data.

10.19.3.6 have_new_almanac()

```
bool Beidou_Dnav_Navigation_Message::have_new_almanac ( )
```

Returns true if new UTC model has arrived. The flag is set to false when the function is executed.

10.19.3.7 have_new_ephemeris()

```
bool Beidou_Dnav_Navigation_Message::have_new_ephemeris ( )
```

Returns true if new Ephemeris has arrived. The flag is set to false when the function is executed.

10.19.3.8 have_new_iono()

```
bool Beidou_Dnav_Navigation_Message::have_new_iono ( )
```

Returns true if new Iono model has arrived. The flag is set to false when the function is executed.

10.19.3.9 have_new_utc_model()

```
bool Beidou_Dnav_Navigation_Message::have_new_utc_model ( )
```

Returns true if new UTC model has arrived. The flag is set to false when the function is executed.

10.19.3.10 satellitePosition()

```
void Beidou_Dnav_Navigation_Message::satellitePosition (
    double transmitTime )
```

Computes the position of the satellite.

10.19.3.11 set_satellite_PRN()

```
void Beidou_Dnav_Navigation_Message::set_satellite_PRN (
    uint32_t prn ) [inline]
```

Sets satellite PRN number.

Definition at line 119 of file beidou_dnav_navigation_message.h.

10.19.3.12 sv_clock_correction()

```
double Beidou_Dnav_Navigation_Message::sv_clock_correction (
    double transmitTime )
```

Sets (*d_satClkCorr*) according to the User Algorithm for SV Clock Correction and returns the corrected clock.

10.19.3.13 utc_time()

```
double Beidou_Dnav_Navigation_Message::utc_time (
    const double beidoutime_corrected ) const
```

Computes the Coordinated Universal Time (UTC) and returns it in [s].

The documentation for this class was generated from the following file:

- [beidou_dnav_navigation_message.h](#)

10.20 Beidou_Dnav_Utc_Model Class Reference

This class is a storage for the BeiDou DNAV UTC Model.

```
#include <beidou_dnav_utc_model.h>
```

Public Member Functions

- `template<class Archive >`
void **serialize** (Archive &archive, const unsigned int version)

Public Attributes

- double [d_A0_UTC](#) {}
BDT clock bias relative to UTC [s].
- double [d_A1_UTC](#) {}
BDT clock rate relative to UTC [s/s].
- int [i_DeltaT_LS](#) {}
Delta time due to leap seconds before the new leap second effective.
- int [i_WN_LSF](#) {}
Week number of the new leap second.
- int [i_DN](#) {}
Day number of week of the new leap second.
- double [d_DeltaT_LSF](#) {}
Delta time due to leap seconds after the new leap second effective [s].
- double [d_A0_GPS](#) {}
BDT clock bias relative to GPS time [s].
- double [d_A1_GPS](#) {}
BDT clock rate relative to GPS time [s/s].
- double [d_A0_GAL](#) {}
BDT clock bias relative to GAL time [s].
- double [d_A1_GAL](#) {}
BDT clock rate relative to GAL time [s/s].
- double [d_A0_GLO](#) {}
BDT clock bias relative to GLO time [s].
- double [d_A1_GLO](#) {}
BDT clock rate relative to GLO time [s/s].
- bool **valid** {}

10.20.1 Detailed Description

This class is a storage for the BeiDou DNAV UTC Model.

Implementation follows the interface described in the Open Service Signal (Version 2.1)

Definition at line 33 of file beidou_dnav_utc_model.h.

10.20.2 Member Data Documentation

10.20.2.1 d_A0_GAL

```
double Beidou_Dnav_Utc_Model::d_A0_GAL {}
```

BDT clock bias relative to GAL time [s].

Definition at line 51 of file beidou_dnav_utc_model.h.

10.20.2.2 d_A0_GLO

```
double Beidou_Dnav_Utc_Model::d_A0_GLO {}
```

BDT clock bias relative to GLO time [s].

Definition at line 55 of file beidou_dnav_utc_model.h.

10.20.2.3 d_A0_GPS

```
double Beidou_Dnav_Utc_Model::d_A0_GPS {}
```

BDT clock bias relative to GPS time [s].

Definition at line 47 of file beidou_dnav_utc_model.h.

10.20.2.4 d_A0_UTC

```
double Beidou_Dnav_Utc_Model::d_A0_UTC {}
```

BDT clock bias relative to UTC [s].

Definition at line 39 of file beidou_dnav_utc_model.h.

10.20.2.5 d_A1_GAL

```
double Beidou_Dnav_Utc_Model::d_A1_GAL {}
```

BDT clock rate relative to GAL time [s/s].

Definition at line 52 of file beidou_dnav_utc_model.h.

10.20.2.6 d_A1_GLO

```
double Beidou_Dnav_Utc_Model::d_A1_GLO {}
```

BDT clock rate relative to GLO time [s/s].

Definition at line 56 of file beidou_dnav_utc_model.h.

10.20.2.7 d_A1_GPS

```
double Beidou_Dnav_Utc_Model::d_A1_GPS {}
```

BDT clock rate relative to GPS time [s/s].

Definition at line 48 of file beidou_dnav_utc_model.h.

10.20.2.8 d_A1_UTC

```
double Beidou_Dnav_Utc_Model::d_A1_UTC {}
```

BDT clock rate relative to UTC [s/s].

Definition at line 40 of file beidou_dnav_utc_model.h.

10.20.2.9 d_DeltaT_LSF

```
double Beidou_Dnav_Utc_Model::d_DeltaT_LSF {}
```

Delta time due to leap seconds after the new leap second effective [s].

Definition at line 44 of file beidou_dnav_utc_model.h.

10.20.2.10 i_DeltaT_LS

```
int Beidou_Dnav_Utc_Model::i_DeltaT_LS {}
```

Delta time due to leap seconds before the new leap second effective.

Definition at line 41 of file beidou_dnav_utc_model.h.

10.20.2.11 i_DN

```
int Beidou_Dnav_Utc_Model::i_DN {}
```

Day number of week of the new leap second.

Definition at line 43 of file beidou_dnav_utc_model.h.

10.20.2.12 i_WN_LSF

```
int Beidou_Dnav_Utc_Model::i_WN_LSF {}
```

Week number of the new leap second.

Definition at line 42 of file beidou_dnav_utc_model.h.

The documentation for this class was generated from the following file:

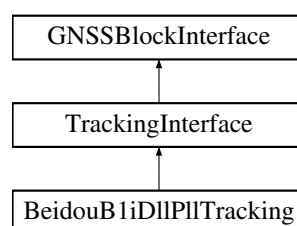
- [beidou_dnav_utc_model.h](#)

10.21 BeidouB1iDIIPITracking Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <beidou_b1i_dll_pll_tracking.h>
```

Inheritance diagram for BeidouB1iDIIPITracking:



Public Member Functions

- **BeidouB1iDllPllTracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override

Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override

Stop running tracking.

10.21.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 38 of file beidou_b1i_dll_pll_tracking.h.

10.21.2 Member Function Documentation

10.21.2.1 set_channel()

```
void BeidouB1iDllPllTracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.21.2.2 set_gnss_synchro()

```
void BeidouB1iDllPllTracking::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.21.2.3 stop_tracking()

```
void BeidouB1iDllPllTracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

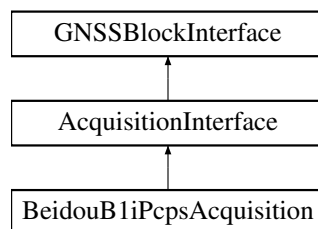
- [beidou_b1i_dll_pll_tracking.h](#)

10.22 BeidouB1iPcpsAcquisition Class Reference

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

```
#include <beidou_b1i_pcps_acquisition.h>
```

Inheritance diagram for BeidouB1iPcpsAcquisition:



Public Member Functions

- **BeidouB1iPcpsAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override

Returns "BEIDOU_B1I_PCPS_Acquisition".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **set_channel** (unsigned int channel) override

Set acquisition channel unique ID.
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override

Set channel fsm associated to this acquisition instance.
- void **set_threshold** (float threshold) override

Set statistics threshold of PCPS algorithm.
- void **set_doppler_max** (uint32_t doppler_max) override

- Set maximum Doppler off grid search.*
- void [set_doppler_step](#) (uint32_t doppler_step) override
 - Set Doppler steps for the grid search.*
- void [init](#) () override
 - Initializes acquisition algorithm.*
- void [set_local_code](#) () override
 - Sets local code for GPS L1/CA PCPS acquisition algorithm.*
- signed int [mag](#) () override
 - Returns the maximum peak of grid search.*
- void [reset](#) () override
 - Restart acquisition algorithm.*
- void [set_state](#) (int state) override
 - If state = 1, it forces the block to start acquiring from the first sample.*
- void [stop_acquisition](#) () override
 - Stop running acquisition.*
- void [set_resampler_latency](#) (uint32_t latency_samples) override
 - Sets the resampler latency to account it in the acquisition code delay estimation.*

10.22.1 Detailed Description

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Definition at line 44 of file beidou_b1i_pcps_acquisition.h.

10.22.2 Member Function Documentation

10.22.2.1 implementation()

```
std::string BeidouB1iPcpsAcquisition::implementation ( ) [inline], [override], [virtual]
```

Returns "BEIDOU_B1I_PCPS_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 61 of file beidou_b1i_pcps_acquisition.h.

10.22.2.2 init()

```
void BeidouB1iPcpsAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.22.2.3 mag()

```
signed int BeidouBliPcpsAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.22.2.4 reset()

```
void BeidouBliPcpsAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.22.2.5 set_channel()

```
void BeidouBliPcpsAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 86 of file beidou_b1i_pcps_acquisition.h.

10.22.2.6 set_channel_fsm()

```
void BeidouBliPcpsAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 95 of file beidou_b1i_pcps_acquisition.h.

10.22.2.7 set_doppler_max()

```
void BeidouBliPcpsAcquisition::set_doppler_max (
    uint32_t doppler_max ) [override]
```

Set maximum Doppler off grid search.

10.22.2.8 set_doppler_step()

```
void BeidouBliPcpsAcquisition::set_doppler_step (
    uint32_t doppler_step ) [override]
```

Set Doppler steps for the grid search.

10.22.2.9 set_gnss_synchro()

```
void BeidouBliPcpsAcquisition::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.22.2.10 set_local_code()

```
void BeidouBliPcpsAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local code for GPS L1/CA PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.22.2.11 set_resampler_latency()

```
void BeidouBliPcpsAcquisition::set_resampler_latency (
    uint32_t latency_samples ) [override], [virtual]
```

Sets the resampler latency to account it in the acquisition code delay estimation.

Implements [AcquisitionInterface](#).

10.22.2.12 set_state()

```
void BeidouBliPcpsAcquisition::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.22.2.13 `set_threshold()`

```
void BeidouB1iPcpsAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.22.2.14 `stop_acquisition()`

```
void BeidouB1iPcpsAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

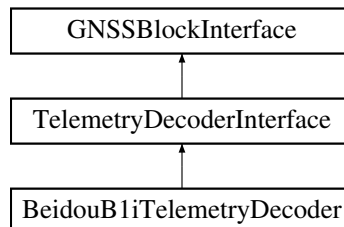
- [beidou_b1i_pcps_acquisition.h](#)

10.23 BeidouB1iTelemetryDecoder Class Reference

This class implements a NAV data decoder for BEIDOU B1I.

```
#include <beidou_b1i_telemetry_decoder.h>
```

Inheritance diagram for BeidouB1iTelemetryDecoder:



Public Member Functions

- **BeidouB1iTelemetryDecoder** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_satellite** (const [Gnss_Satellite](#) &satellite) override
- std::string **role** () override
- std::string **implementation** () override
- *Returns "BEIDOU_B1I_Telemetry_Decoder".*
- void **set_channel** (int channel) override
- void **reset** () override
- size_t **item_size** () override

10.23.1 Detailed Description

This class implements a NAV data decoder for BEIDOU B1I.

Definition at line 39 of file `beidou_b1i_telemetry_decoder.h`.

10.23.2 Member Function Documentation

10.23.2.1 `implementation()`

```
std::string BeidouB1iTelemetryDecoder::implementation ( ) [inline], [override], [virtual]
```

Returns "BEIDOU_B1I_Telemetry_Decoder".

Implements [GNSSBlockInterface](#).

Definition at line 63 of file `beidou_b1i_telemetry_decoder.h`.

The documentation for this class was generated from the following file:

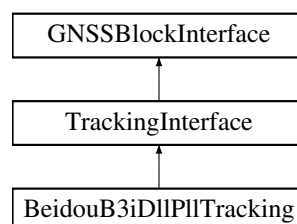
- [beidou_b1i_telemetry_decoder.h](#)

10.24 BeidouB3iDIIPITracking Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <beidou_b3i_dll_pll_tracking.h>
```

Inheritance diagram for BeidouB3iDIIPITracking:



Public Member Functions

- **BeidouB3iDlIPllTracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override

Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override

Stop running tracking.

10.24.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 38 of file beidou_b3i_dll_pll_tracking.h.

10.24.2 Member Function Documentation

10.24.2.1 set_channel()

```
void BeidouB3iDlIPllTracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.24.2.2 set_gnss_synchro()

```
void BeidouB3iDlIPllTracking::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.24.2.3 stop_tracking()

```
void BeidouB3iDllPllTracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

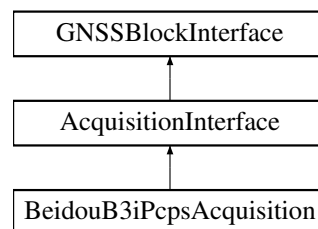
- [beidou_b3i_dll_pll_tracking.h](#)

10.25 BeidouB3iPcpsAcquisition Class Reference

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for BeiDou B3I signals.

```
#include <beidou_b3i_pcps_acquisition.h>
```

Inheritance diagram for BeidouB3iPcpsAcquisition:



Public Member Functions

- **BeidouB3iPcpsAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "BEIDOU_B1I_PCPS_Acquisition".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **set_channel** (unsigned int channel) override
Set acquisition channel unique ID.
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
Set channel fsm associated to this acquisition instance.
- void **set_threshold** (float threshold) override
Set statistics threshold of PCPS algorithm.
- void **set_doppler_max** (unsigned int doppler_max) override

- Set maximum Doppler off grid search.*
- void [set_doppler_step](#) (unsigned int doppler_step) override
 - Set Doppler steps for the grid search.*
- void [init](#) () override
 - Initializes acquisition algorithm.*
- void [set_local_code](#) () override
 - Sets local code for GPS L1/CA PCPS acquisition algorithm.*
- signed int [mag](#) () override
 - Returns the maximum peak of grid search.*
- void [reset](#) () override
 - Restart acquisition algorithm.*
- void [set_state](#) (int state) override
 - If state = 1, it forces the block to start acquiring from the first sample.*
- void [stop_acquisition](#) () override
 - Stop running acquisition.*
- void [set_resampler_latency](#) (uint32_t latency_samples) override
 - Sets the resampler latency to account it in the acquisition code delay estimation.*

10.25.1 Detailed Description

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for BeiDou B3I signals.

Definition at line 43 of file `beidou_b3i_pcps_acquisition.h`.

10.25.2 Member Function Documentation

10.25.2.1 implementation()

```
std::string BeidouB3iPcpsAcquisition::implementation ( ) [inline], [override], [virtual]
```

Returns "BEIDOU_B1I_PCPS_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 60 of file `beidou_b3i_pcps_acquisition.h`.

10.25.2.2 init()

```
void BeidouB3iPcpsAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.25.2.3 mag()

```
signed int BeidouB3iPcpsAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.25.2.4 reset()

```
void BeidouB3iPcpsAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.25.2.5 set_channel()

```
void BeidouB3iPcpsAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 85 of file beidou_b3i_pcps_acquisition.h.

10.25.2.6 set_channel_fsm()

```
void BeidouB3iPcpsAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 94 of file beidou_b3i_pcps_acquisition.h.

10.25.2.7 set_doppler_max()

```
void BeidouB3iPcpsAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.25.2.8 set_doppler_step()

```
void BeidouB3iPcpsAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.25.2.9 set_gnss_synchro()

```
void BeidouB3iPcpsAcquisition::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.25.2.10 set_local_code()

```
void BeidouB3iPcpsAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local code for GPS L1/CA PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.25.2.11 set_resampler_latency()

```
void BeidouB3iPcpsAcquisition::set_resampler_latency (
    uint32_t latency_samples ) [override], [virtual]
```

Sets the resampler latency to account it in the acquisition code delay estimation.

Implements [AcquisitionInterface](#).

10.25.2.12 set_state()

```
void BeidouB3iPcpsAcquisition::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.25.2.13 set_threshold()

```
void BeidouB3iPcpsAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.25.2.14 stop_acquisition()

```
void BeidouB3iPcpsAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

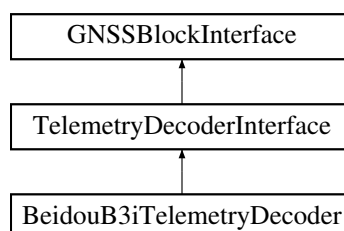
- [beidou_b3i_pcps_acquisition.h](#)

10.26 BeidouB3iTelemetryDecoder Class Reference

This class implements a NAV data decoder for BEIDOU B1I.

```
#include <beidou_b3i_telemetry_decoder.h>
```

Inheritance diagram for BeidouB3iTelemetryDecoder:



Public Member Functions

- **BeidouB3iTelemetryDecoder** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_satellite** (const [Gnss_Satellite](#) &satellite) override
- std::string **role** () override
- std::string **implementation** () override
- *Returns "BEIDOU_B3I_Telemetry_Decoder".*
- void **set_channel** (int channel) override
- void **reset** () override
- size_t **item_size** () override

10.26.1 Detailed Description

This class implements a NAV data decoder for BEIDOU B1I.

Definition at line 37 of file beidou_b3i_telemetry_decoder.h.

10.26.2 Member Function Documentation

10.26.2.1 implementation()

```
std::string BeidouB3iTelemetryDecoder::implementation ( ) [inline], [override], [virtual]
```

Returns "BEIDOU_B3I_Telemetry_Decoder".

Implements [GNSSBlockInterface](#).

Definition at line 57 of file beidou_b3i_telemetry_decoder.h.

The documentation for this class was generated from the following file:

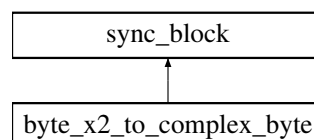
- [beidou_b3i_telemetry_decoder.h](#)

10.27 byte_x2_to_complex_byte Class Reference

This class adapts two signed char streams into a std::complex<signed char> stream.

```
#include <byte_x2_to_complex_byte.h>
```

Inheritance diagram for byte_x2_to_complex_byte:



Public Member Functions

- int **work** (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- byte_x2_to_complex_byte_sptr **make_byte_x2_to_complex_byte** ()

10.27.1 Detailed Description

This class adapts two signed char streams into a `std::complex<signed char>` stream.

Definition at line 47 of file `byte_x2_to_complex_byte.h`.

The documentation for this class was generated from the following file:

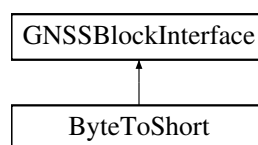
- [byte_x2_to_complex_byte.h](#)

10.28 ByteToShort Class Reference

Adapts an 8-bits sample stream (IF) to a short int stream (IF)

```
#include <byte_to_short.h>
```

Inheritance diagram for ByteToShort:



Public Member Functions

- **ByteToShort** (const [ConfigurationInterface](#) *configuration, std::string role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string [implementation](#) () override
 - Returns "Byte_To_Short".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.28.1 Detailed Description

Adapts an 8-bits sample stream (IF) to a short int stream (IF)

Definition at line 35 of file `byte_to_short.h`.

10.28.2 Member Function Documentation

10.28.2.1 `implementation()`

```
std::string ByteToShort::implementation ( ) [inline], [override], [virtual]
```

Returns "Byte_To_Short".

Implements [GNSSBlockInterface](#).

Definition at line 50 of file `byte_to_short.h`.

The documentation for this class was generated from the following file:

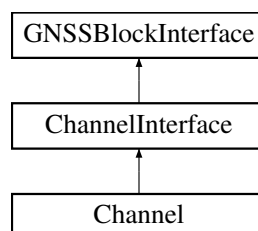
- [byte_to_short.h](#)

10.29 Channel Class Reference

This class represents a GNSS channel. It wraps an [AcquisitionInterface](#), a Tracking Interface and a [TelemetryDecoderInterface](#), and handles their interaction through a Finite State Machine.

```
#include <channel.h>
```

Inheritance diagram for Channel:



Public Member Functions

- [Channel](#) (const [ConfigurationInterface](#) *configuration, uint32_t channel, std::shared_ptr< [AcquisitionInterface](#) > acq, std::shared_ptr< [TrackingInterface](#) > trk, std::shared_ptr< [TelemetryDecoderInterface](#) > nav, const std::string &role, const std::string &signal_str, [Concurrent_Queue](#)< pmt::pmt_t > *queue)

Constructor.

- [~Channel](#) ()=default

Destructor.

- void [connect](#) (gr::top_block_sptr top_block) override
connects the tracking block to the top_block and to the telemetry
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr [get_left_block](#) () override
gets the gnuradio tracking block pointer
- gr::basic_block_sptr [get_left_block_trk](#) () override
gets the gnuradio tracking block pointer
- gr::basic_block_sptr [get_left_block_acq](#) () override
gets the gnuradio tracking block pointer
- gr::basic_block_sptr **get_right_block** () override
- std::string **role** () override
- std::string [implementation](#) () override
Returns "Channel".
- size_t **item_size** () override
- [Gnss_Signal](#) **get_signal** () const override
- void [start_acquisition](#) () override
Start the State Machine.
- void [stop_channel](#) () override
Stop the State Machine.
- void [set_signal](#) (const [Gnss_Signal](#) &gnss_signal_) override
Sets the channel GNSS signal.
- void **assist_acquisition_doppler** (double Carrier_Doppler_hz) override
- std::shared_ptr< [AcquisitionInterface](#) > **acquisition** () const
- std::shared_ptr< [TrackingInterface](#) > **tracking** () const
- std::shared_ptr< [TelemetryDecoderInterface](#) > **telemetry** () const
- void **msg_handler_events** (pmt::pmt_t msg)

10.29.1 Detailed Description

This class represents a GNSS channel. It wraps an [AcquisitionInterface](#), a Tracking Interface and a [TelemetryDecoderInterface](#), and handles their interaction through a Finite State Machine.

Definition at line 53 of file channel.h.

10.29.2 Constructor & Destructor Documentation

10.29.2.1 Channel()

```
Channel::Channel (
    const ConfigurationInterface * configuration,
    uint32_t channel,
    std::shared_ptr< AcquisitionInterface > acq,
    std::shared_ptr< TrackingInterface > trk,
    std::shared_ptr< TelemetryDecoderInterface > nav,
    const std::string & role,
    const std::string & signal_str,
    Concurrent\_Queue< pmt::pmt_t > * queue )
```

Constructor.

10.29.2.2 ~Channel()

```
Channel::~~Channel ( ) [default]
```

Destructor.

10.29.3 Member Function Documentation

10.29.3.1 connect()

```
void Channel::connect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

connects the tracking block to the top_block and to the telemetry

Implements [GNSSBlockInterface](#).

10.29.3.2 get_left_block()

```
gr::basic_block_sptr Channel::get_left_block ( ) [override], [virtual]
```

gets the gnuradio tracking block pointer

Implements [ChannelInterface](#).

10.29.3.3 `get_left_block_acq()`

```
gr::basic_block_sptr Channel::get_left_block_acq ( ) [override], [virtual]
```

gets the gnuradio tracking block pointer

Implements [ChannelInterface](#).

10.29.3.4 `get_left_block_trk()`

```
gr::basic_block_sptr Channel::get_left_block_trk ( ) [override], [virtual]
```

gets the gnuradio tracking block pointer

Implements [ChannelInterface](#).

10.29.3.5 `implementation()`

```
std::string Channel::implementation ( ) [inline], [override], [virtual]
```

Returns "Channel".

Implements [GNSSBlockInterface](#).

Definition at line 72 of file channel.h.

10.29.3.6 `set_signal()`

```
void Channel::set_signal (
    const Gnss\_Signal & gnss_signal_ ) [override], [virtual]
```

Sets the channel GNSS signal.

Implements [ChannelInterface](#).

10.29.3.7 `start_acquisition()`

```
void Channel::start_acquisition ( ) [override], [virtual]
```

Start the State Machine.

Implements [ChannelInterface](#).

10.29.3.8 stop_channel()

```
void Channel::stop_channel ( ) [override], [virtual]
```

Stop the State Machine.

Implements [ChannelInterface](#).

The documentation for this class was generated from the following file:

- [channel.h](#)

10.30 Channel_Event Class Reference

Public Attributes

- int **channel_id**
- int **event_type**

Friends

- channel_event_sptr **channel_event_make** (int channel_id, int event_type)

10.30.1 Detailed Description

Definition at line 31 of file channel_event.h.

The documentation for this class was generated from the following file:

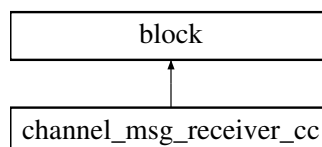
- [channel_event.h](#)

10.31 channel_msg_receiver_cc Class Reference

GNU Radio block that receives asynchronous channel messages from acquisition and tracking blocks.

```
#include <channel_msg_receiver_cc.h>
```

Inheritance diagram for channel_msg_receiver_cc:



Public Member Functions

- [~channel_msg_receiver_cc](#) ()=default

Default destructor.

Friends

- `channel_msg_receiver_cc_sptr` **channel_msg_receiver_make_cc** (std::shared_ptr< [ChannelFsm](#) > channel_fsm, bool repeat)

10.31.1 Detailed Description

GNU Radio block that receives asynchronous channel messages from acquisition and tracking blocks.

Definition at line 45 of file `channel_msg_receiver_cc.h`.

10.31.2 Constructor & Destructor Documentation

10.31.2.1 ~channel_msg_receiver_cc()

```
channel_msg_receiver_cc::~channel_msg_receiver_cc ( ) [default]
```

Default destructor.

The documentation for this class was generated from the following file:

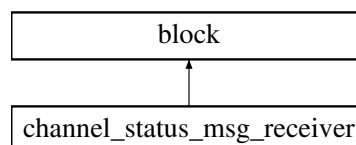
- [channel_msg_receiver_cc.h](#)

10.32 channel_status_msg_receiver Class Reference

GNU Radio block that receives asynchronous channel messages from tlm blocks.

```
#include <channel_status_msg_receiver.h>
```

Inheritance diagram for `channel_status_msg_receiver`:



Public Member Functions

- [~channel_status_msg_receiver](#) ()=default
Default destructor.
- `std::map< int, std::shared_ptr< Gnss_Synchro > > get_current_status_map ()`
return the current status map of all channels with valid telemetry
- [Monitor_Pvt get_current_status_pvt](#) ()
return the current receiver PVT

Friends

- `channel_status_msg_receiver_sptr channel_status_msg_receiver_make ()`

10.32.1 Detailed Description

GNU Radio block that receives asynchronous channel messages from tlm blocks.

Definition at line 47 of file `channel_status_msg_receiver.h`.

10.32.2 Constructor & Destructor Documentation

10.32.2.1 ~channel_status_msg_receiver()

```
channel_status_msg_receiver::~channel_status_msg_receiver ( ) [default]
```

Default destructor.

10.32.3 Member Function Documentation

10.32.3.1 get_current_status_map()

```
std::map<int, std::shared_ptr<Gnss\_Synchro> > channel_status_msg_receiver::get_current_↔  
status_map ( )
```

return the current status map of all channels with valid telemetry

10.32.3.2 `get_current_status_pvt()`

```
Monitor_Pvt channel_status_msg_receiver::get_current_status_pvt ( )
```

return the current receiver PVT

The documentation for this class was generated from the following file:

- [channel_status_msg_receiver.h](#)

10.33 ChannelFsm Class Reference

This class implements a State Machine for channel.

```
#include <channel_fsm.h>
```

Public Member Functions

- **ChannelFsm** (std::shared_ptr< [AcquisitionInterface](#) > acquisition)
- void **set_acquisition** (std::shared_ptr< [AcquisitionInterface](#) > acquisition)
- void **set_tracking** (std::shared_ptr< [TrackingInterface](#) > tracking)
- void **set_telemetry** (std::shared_ptr< [TelemetryDecoderInterface](#) > telemetry)
- void **set_queue** ([Concurrent_Queue](#)< pmt::pmt_t > *queue)
- void **set_channel** (uint32_t channel)
- void **start_acquisition** ()
- bool **Event_start_acquisition** ()
- bool **Event_start_acquisition_fpga** ()
- bool **Event_stop_channel** ()
- bool **Event_failed_tracking_standby** ()
- virtual bool **Event_valid_acquisition** ()
- virtual bool **Event_failed_acquisition_repeat** ()
- virtual bool **Event_failed_acquisition_no_repeat** ()

10.33.1 Detailed Description

This class implements a State Machine for channel.

Definition at line 38 of file `channel_fsm.h`.

The documentation for this class was generated from the following file:

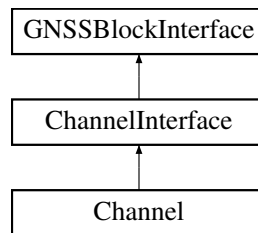
- [channel_fsm.h](#)

10.34 ChannelInterface Class Reference

This abstract class represents an interface to a channel GNSS block.

```
#include <channel_interface.h>
```

Inheritance diagram for ChannelInterface:



Public Member Functions

- virtual gr::basic_block_sptr **get_left_block_trk** ()=0
- virtual gr::basic_block_sptr **get_left_block_acq** ()=0
- virtual gr::basic_block_sptr **get_left_block** ()=0
- virtual gr::basic_block_sptr **get_right_block** ()=0
- virtual [Gnss_Signal](#) **get_signal** () const =0
- virtual void **start_acquisition** ()=0
- virtual void **assist_acquisition_doppler** (double Carrier_Doppler_hz)=0
- virtual void **stop_channel** ()=0
- virtual void **set_signal** (const [Gnss_Signal](#) &)=0

10.34.1 Detailed Description

This abstract class represents an interface to a channel GNSS block.

Abstract class for channel blocks. Since all its methods are pure virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

Definition at line 40 of file [channel_interface.h](#).

The documentation for this class was generated from the following file:

- [channel_interface.h](#)

10.35 `cl_fft_plan` Struct Reference

Public Attributes

- `cl_context` **context**
- `clFFT_Dim3` **n**
- `clFFT_Dimension` **dim**
- `clFFT_DataFormat` **format**
- `string *` **kernel_string**
- `cl_program` **program**
- `cl_fft_kernel_info *` **kernel_info**
- `int` **num_kernels**
- `cl_kernel` **twist_kernel**
- `cl_int` **temp_buffer_needed**
- `unsigned` **last_batch_size**
- `cl_mem` **tempmemobj**
- `cl_mem` **tempmemobj_real**
- `cl_mem` **tempmemobj_imag**
- `unsigned` **max_localmem_fft_size**
- `unsigned` **max_work_item_per_workgroup**
- `unsigned` **max_radix**
- `unsigned` **min_mem_coalesce_width**
- `unsigned` **num_local_mem_banks**

10.35.1 Detailed Description

Definition at line 45 of file `fft_internal.h`.

The documentation for this struct was generated from the following file:

- [fft_internal.h](#)

10.36 `clFFT_Complex` Struct Reference

Public Attributes

- `float` **real**
- `float` **imag**

10.36.1 Detailed Description

Definition at line 65 of file `clFFT.h`.

The documentation for this struct was generated from the following file:

- [clFFT.h](#)

10.37 cIFFT_Dim3 Struct Reference

Public Attributes

- unsigned int **x**
- unsigned int **y**
- unsigned int **z**

10.37.1 Detailed Description

Definition at line 52 of file cIFFT.h.

The documentation for this struct was generated from the following file:

- [cIFFT.h](#)

10.38 cIFFT_SplitComplex Struct Reference

Public Attributes

- float * **real**
- float * **imag**

10.38.1 Detailed Description

Definition at line 59 of file cIFFT.h.

The documentation for this struct was generated from the following file:

- [cIFFT.h](#)

10.39 cnav_msg_decoder_t Struct Reference

```
#include <cnav_msg.h>
```

Public Attributes

- [cnav_v27_part_t](#) part1
- [cnav_v27_part_t](#) part2

10.39.1 Detailed Description

GPS CNAV message lock and decoder object.

Decoder uses two Viterbi decoder objects to ensure the lock is acquired when the input symbol phase is not known.

Definition at line 96 of file `cnav_msg.h`.

10.39.2 Member Data Documentation

10.39.2.1 `part1`

`cnav_v27_part_t` `cnav_msg_decoder_t::part1`

Decoder for odd symbol pairs

Definition at line 98 of file `cnav_msg.h`.

10.39.2.2 `part2`

`cnav_v27_part_t` `cnav_msg_decoder_t::part2`

Decoder for even symbol pairs

Definition at line 99 of file `cnav_msg.h`.

The documentation for this struct was generated from the following file:

- [cnav_msg.h](#)

10.40 `cnav_msg_t` Struct Reference

```
#include <cnav_msg.h>
```

Public Attributes

- `uint8_t` `prn`
- `uint8_t` `msg_id`
- `uint32_t` `tow`
- `bool` `alert`
- `uint8_t` `raw_msg` [`GPS_L2C_V27_DECODE_BITS+GPS_L2C_V27_DELAY_BITS`]

10.40.1 Detailed Description

GPS CNAV message container.

See also

`cnav_msg_decoder_add_symbol`

Definition at line 53 of file `cnav_msg.h`.

10.40.2 Member Data Documentation

10.40.2.1 alert

```
bool cnav_msg_t::alert
```

CNAV message alert flag

Definition at line 58 of file `cnav_msg.h`.

10.40.2.2 msg_id

```
uint8_t cnav_msg_t::msg_id
```

Message id. 0..31

Definition at line 56 of file `cnav_msg.h`.

10.40.2.3 prn

```
uint8_t cnav_msg_t::prn
```

SV PRN. 0..31

Definition at line 55 of file `cnav_msg.h`.

10.40.2.4 raw_msg

```
uint8_t cnav_msg_t::raw_msg[GPS_L2C_V27_DECODE_BITS+GPS_L2C_V27_DELAY_BITS]
```

RAW MSG for GNSS-SDR

Definition at line 59 of file cnav_msg.h.

10.40.2.5 tow

```
uint32_t cnav_msg_t::tow
```

GPS ToW in 6-second units. Multiply to 6 to get seconds.

Definition at line 57 of file cnav_msg.h.

The documentation for this struct was generated from the following file:

- [cnav_msg.h](#)

10.41 cnav_v27_part_t Struct Reference

```
#include <cnav_msg.h>
```

Public Attributes

- [v27_t dec](#)
- [v27_decision_t decisions](#) [GPS_L2_V27_HISTORY_LENGTH_BITS]
- unsigned char [symbols](#) [(GPS_L2C_V27_INIT_BITS+GPS_L2C_V27_DECODE_BITS) *2]
- size_t [n_symbols](#)
- unsigned char [decoded](#) [GPS_L2C_V27_DECODE_BITS+GPS_L2C_V27_DELAY_BITS]
- size_t [n_decoded](#)
- bool [preamble_seen](#)
- bool [invert](#)
- bool [message_lock](#)
- bool [crc_ok](#)
- size_t [n_crc_fail](#)
- bool [init](#)

10.41.1 Detailed Description

GPS CNAV decoder component. This component controls symbol decoding string.

See also

[cnav_msg_decoder_t](#)

Definition at line 68 of file cnav_msg.h.

10.41.2 Member Data Documentation

10.41.2.1 crc_ok

```
bool cnav_v27_part_t::crc_ok
```

Flag that the last message had good CRC

Definition at line 84 of file cnav_msg.h.

10.41.2.2 dec

```
v27_t cnav_v27_part_t::dec
```

Viterbi block decoder object

Definition at line 70 of file cnav_msg.h.

10.41.2.3 decisions

```
v27_decision_t cnav_v27_part_t::decisions[GPS_L2_V27_HISTORY_LENGTH_BITS]
```

Decision graph

Definition at line 71 of file cnav_msg.h.

10.41.2.4 decoded

```
unsigned char cnav_v27_part_t::decoded[GPS_L2C_V27_DECODE_BITS+GPS_L2C_V27_DELAY_BITS]
```

Decode buffer

Definition at line 76 of file cnav_msg.h.

10.41.2.5 init

```
bool cnav_v27_part_t::init
```

Initial state flag. When true, initial bits do not produce output.

Definition at line 86 of file cnav_msg.h.

10.41.2.6 invert

```
bool cnav_v27_part_t::invert
```

When true, indicates the bits are inverted

Definition at line 81 of file cnav_msg.h.

10.41.2.7 message_lock

```
bool cnav_v27_part_t::message_lock
```

When true, indicates the message boundary is found.

Definition at line 82 of file cnav_msg.h.

10.41.2.8 n_crc_fail

```
size_t cnav_v27_part_t::n_crc_fail
```

Counter for CRC failures

Definition at line 85 of file cnav_msg.h.

10.41.2.9 n_decoded

```
size_t cnav_v27_part_t::n_decoded
```

Number of bits in the decode buffer

Definition at line 78 of file cnav_msg.h.

10.41.2.10 n_symbols

```
size_t cnav_v27_part_t::n_symbols
```

Count of symbols in the symbol buffer

Definition at line 75 of file cnav_msg.h.

10.41.2.11 preamble_seen

```
bool cnav_v27_part_t::preamble_seen
```

When true, the decode buffer is aligned on preamble.

Definition at line 79 of file `cnav_msg.h`.

10.41.2.12 symbols

```
unsigned char cnav_v27_part_t::symbols[(GPS_L2C_V27_INIT_BITS+GPS_L2C_V27_DECODE_BITS) *2]
```

Symbol buffer

Definition at line 73 of file `cnav_msg.h`.

The documentation for this struct was generated from the following file:

- [cnav_msg.h](#)

10.42 Command_Event Class Reference

Public Attributes

- int **command_id**
- int **event_type**

Friends

- command_event_sptr **command_event_make** (int command_id, int event_type)

10.42.1 Detailed Description

Definition at line 31 of file `command_event.h`.

The documentation for this class was generated from the following file:

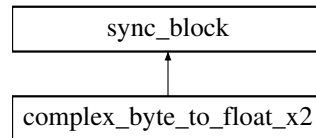
- [command_event.h](#)

10.43 `complex_byte_to_float_x2` Class Reference

This class adapts a `std::complex<signed char>` stream into two 16-bits (short) streams.

```
#include <complex_byte_to_float_x2.h>
```

Inheritance diagram for `complex_byte_to_float_x2`:



Public Member Functions

- `int work` (`int` noutput_items, `gr_vector_const_void_star` &input_items, `gr_vector_void_star` &output_items)

Friends

- `complex_byte_to_float_x2_sptr make_complex_byte_to_float_x2` ()

10.43.1 Detailed Description

This class adapts a `std::complex<signed char>` stream into two 16-bits (short) streams.

Definition at line 47 of file `complex_byte_to_float_x2.h`.

The documentation for this class was generated from the following file:

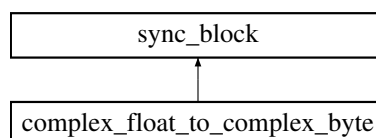
- [complex_byte_to_float_x2.h](#)

10.44 `complex_float_to_complex_byte` Class Reference

This class adapts a `gr_complex` stream into a `std::complex<signed char>` stream.

```
#include <complex_float_to_complex_byte.h>
```

Inheritance diagram for `complex_float_to_complex_byte`:



Public Member Functions

- int **work** (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- complex_float_to_complex_byte_sptr **make_complex_float_to_complex_byte** ()

10.44.1 Detailed Description

This class adapts a gr_complex stream into a std::complex<signed char> stream.

Definition at line 46 of file complex_float_to_complex_byte.h.

The documentation for this class was generated from the following file:

- [complex_float_to_complex_byte.h](#)

10.45 Concurrent_Map< Data > Class Template Reference

This class implements a thread-safe std::map.

```
#include <concurrent_map.h>
```

Public Member Functions

- void **write** (int key, Data const &data)
- std::map< int, Data > **get_map_copy** ()
- size_t **size** ()
- bool **read** (int key, Data &p_data)

10.45.1 Detailed Description

```
template<typename Data>  
class Concurrent_Map< Data >
```

This class implements a thread-safe std::map.

Definition at line 35 of file concurrent_map.h.

The documentation for this class was generated from the following file:

- [concurrent_map.h](#)

10.46 Concurrent_Queue< Data > Class Template Reference

This class implements a thread-safe std::queue.

```
#include <concurrent_queue.h>
```

Public Member Functions

- void **push** (Data const &data)
- bool **empty** () const
- bool **try_pop** (Data &popped_value)
- void **wait_and_pop** (Data &popped_value)
- bool **timed_wait_and_pop** (Data &popped_value, int wait_ms)

10.46.1 Detailed Description

```
template<typename Data>  
class Concurrent_Queue< Data >
```

This class implements a thread-safe std::queue.

Thread-safe object queue which uses the library boost_thread to perform MUTEX based on the code available at <https://www.justsoftwaresolutions.co.uk/threading/implementing-a-thread-safe-queue-using-boost-mutex.html>

Definition at line 35 of file acquisition_interface.h.

The documentation for this class was generated from the following files:

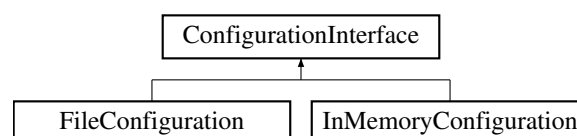
- [acquisition_interface.h](#)
- [concurrent_queue.h](#)

10.47 ConfigurationInterface Class Reference

This abstract class represents an interface to configuration parameters.

```
#include <configuration_interface.h>
```

Inheritance diagram for ConfigurationInterface:



Public Member Functions

- virtual std::string **property** (std::string property_name, std::string default_value) const =0
- virtual bool **property** (std::string property_name, bool default_value) const =0
- virtual int64_t **property** (std::string property_name, int64_t default_value) const =0
- virtual uint64_t **property** (std::string property_name, uint64_t default_value) const =0
- virtual int32_t **property** (std::string property_name, int32_t default_value) const =0
- virtual uint32_t **property** (std::string property_name, uint32_t default_value) const =0
- virtual int16_t **property** (std::string property_name, int16_t default_value) const =0
- virtual uint16_t **property** (std::string property_name, uint16_t default_value) const =0
- virtual float **property** (std::string property_name, float default_value) const =0
- virtual double **property** (std::string property_name, double default_value) const =0
- virtual void **set_property** (std::string property_name, std::string value)=0

10.47.1 Detailed Description

This abstract class represents an interface to configuration parameters.

The interface defines an accessor method that gets a parameter name as input and returns the value of this parameter, a string, as output. Property names are defined here. This is an abstract class for interfaces. Since all its methods are virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

Definition at line 41 of file configuration_interface.h.

The documentation for this class was generated from the following file:

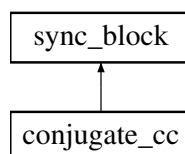
- [configuration_interface.h](#)

10.48 conjugate_cc Class Reference

This class adapts a std::complex<short> stream into two 32-bits (float) streams.

```
#include <conjugate_cc.h>
```

Inheritance diagram for conjugate_cc:



Public Member Functions

- int **work** (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- conjugate_cc_sptr **make_conjugate_cc** ()

10.48.1 Detailed Description

This class adapts a `std::complex<short>` stream into two 32-bits (float) streams.

Definition at line 46 of file `conjugate_cc.h`.

The documentation for this class was generated from the following file:

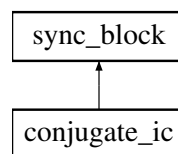
- [conjugate_cc.h](#)

10.49 conjugate_ic Class Reference

This class adapts a `std::complex<short>` stream into two 32-bits (float) streams.

```
#include <conjugate_ic.h>
```

Inheritance diagram for `conjugate_ic`:



Public Member Functions

- `int` **work** (`int` noutput_items, `gr_vector_const_void_star` &input_items, `gr_vector_void_star` &output_items)

Friends

- `conjugate_ic_sptr` **make_conjugate_ic** ()

10.49.1 Detailed Description

This class adapts a `std::complex<short>` stream into two 32-bits (float) streams.

Definition at line 46 of file `conjugate_ic.h`.

The documentation for this class was generated from the following file:

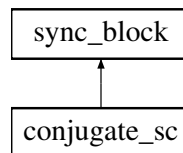
- [conjugate_ic.h](#)

10.50 conjugate_sc Class Reference

This class adapts a `std::complex<short>` stream into two 32-bits (float) streams.

```
#include <conjugate_sc.h>
```

Inheritance diagram for `conjugate_sc`:



Public Member Functions

- `int work (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)`

Friends

- `conjugate_sc_sptr make_conjugate_sc ()`

10.50.1 Detailed Description

This class adapts a `std::complex<short>` stream into two 32-bits (float) streams.

Definition at line 46 of file `conjugate_sc.h`.

The documentation for this class was generated from the following file:

- [conjugate_sc.h](#)

10.51 ControlThread Class Reference

This class represents the main thread of the application, so the name is [ControlThread](#). This is the GNSS Receiver Control Plane: it connects the flowgraph, starts running it, and while it does not stop, reads the control messages generated by the blocks, processes them, and applies the corresponding actions.

```
#include <control_thread.h>
```

Public Member Functions

- [ControlThread](#) ()
Default constructor.
- [ControlThread](#) (std::shared_ptr< [ConfigurationInterface](#) > configuration)
Constructor that initializes the class with parameters.
- [~ControlThread](#) ()
Destructor.
- int [run](#) ()
Runs the control thread.
- void [set_control_queue](#) (std::shared_ptr< [Concurrent_Queue](#)< pmt::pmt_t >> control_queue)
Sets the control_queue.
- unsigned int [processed_control_messages](#) () const
- unsigned int [applied_actions](#) () const
- std::shared_ptr< [GNSSFlowgraph](#) > [flowgraph](#) ()
Instantiates a flowgraph.

10.51.1 Detailed Description

This class represents the main thread of the application, so the name is [ControlThread](#). This is the GNSS Receiver Control Plane: it connects the flowgraph, starts running it, and while it does not stop, reads the control messages generated by the blocks, processes them, and applies the corresponding actions.

Definition at line 59 of file `control_thread.h`.

10.51.2 Constructor & Destructor Documentation

10.51.2.1 [ControlThread](#)() [1/2]

```
ControlThread::ControlThread ( )
```

Default constructor.

10.51.2.2 [ControlThread](#)() [2/2]

```
ControlThread::ControlThread (
    std::shared_ptr< ConfigurationInterface > configuration ) [explicit]
```

Constructor that initializes the class with parameters.

Parameters

in	<i>configuration</i>	Pointer to a ConfigurationInterface
----	----------------------	---

10.51.2.3 ~ControlThread()

```
ControlThread::~~ControlThread ( )
```

Destructor.

10.51.3 Member Function Documentation

10.51.3.1 flowgraph()

```
std::shared_ptr<GNSSFlowgraph> ControlThread::flowgraph ( ) [inline]
```

Instantiates a flowgraph.

Returns

Returns a smart pointer to a flowgraph object

Definition at line 115 of file control_thread.h.

10.51.3.2 run()

```
int ControlThread::run ( )
```

Runs the control thread.

This is the main loop that reads and process the control messages:

- Connect the GNSS receiver flowgraph;
- Start the GNSS receiver flowgraph;

```
while (flowgraph_ ->running() && !stop_){
```

- Read control messages and process them; }

10.51.3.3 set_control_queue()

```
void ControlThread::set_control_queue (
    std::shared_ptr< Concurrent_Queue< pmt::pmt_t >> control_queue )
```

Sets the control_queue.

Parameters

in	<code>std::shared_ptr<Concurrent_Queue<pmt::pmt_t>></code>	<code>control_queue</code>
----	--	----------------------------

The documentation for this class was generated from the following file:

- [control_thread.h](#)

10.52 Cpu_Multicorrelator Class Reference

Class that implements carrier wipe-off and correlators.

```
#include <cpu_multicorrelator.h>
```

Public Member Functions

- **bool** **init** (int max_signal_length_samples, int n_correlators)
- **bool** **set_local_code_and_taps** (int code_length_chips, const std::complex< float > *local_code_in, float *shifts_chips)
- **bool** **set_input_output_vectors** (std::complex< float > *corr_out, const std::complex< float > *sig_in)
- **void** **update_local_code** (int correlator_length_samples, float rem_code_phase_chips, float code_phase_step_chips)
- **bool** **Carrier_wipeoff_multicorrelator_resampler** (float rem_carrier_phase_in_rad, float phase_step_rad, float rem_code_phase_chips, float code_phase_step_chips, int signal_length_samples)
- **bool** **free** ()

10.52.1 Detailed Description

Class that implements carrier wipe-off and correlators.

Definition at line 33 of file `cpu_multicorrelator.h`.

The documentation for this class was generated from the following file:

- [cpu_multicorrelator.h](#)

10.53 Cpu_Multicorrelator_16sc Class Reference

Class that implements carrier wipe-off and correlators.

```
#include <cpu_multicorrelator_16sc.h>
```

Public Member Functions

- **bool init** (int max_signal_length_samples, int n_correlators)
- **bool set_local_code_and_taps** (int code_length_chips, const lv_16sc_t *local_code_in, float *shifts_chips)
- **bool set_input_output_vectors** (lv_16sc_t *corr_out, const lv_16sc_t *sig_in)
- **void update_local_code** (int correlator_length_samples, float rem_code_phase_chips, float code_phase↵_step_chips)
- **bool Carrier_wipeoff_multicorrelator_resampler** (float rem_carrier_phase_in_rad, float phase_step_rad, float rem_code_phase_chips, float code_phase_step_chips, int signal_length_samples)
- **bool free** ()

10.53.1 Detailed Description

Class that implements carrier wipe-off and correlators.

Definition at line 33 of file `cpu_multicorrelator_16sc.h`.

The documentation for this class was generated from the following file:

- [cpu_multicorrelator_16sc.h](#)

10.54 Cpu_Multicorrelator_Real_Codes Class Reference

Class that implements carrier wipe-off and correlators.

```
#include <cpu_multicorrelator_real_codes.h>
```

Public Member Functions

- **void set_high_dynamics_resampler** (bool use_high_dynamics_resampler)
- **bool init** (int max_signal_length_samples, int n_correlators)
- **bool set_local_code_and_taps** (int code_length_chips, const float *local_code_in, float *shifts_chips)
- **bool set_input_output_vectors** (std::complex< float > *corr_out, const std::complex< float > *sig_in)
- **void update_local_code** (int correlator_length_samples, float rem_code_phase_chips, float code_phase↵_step_chips, float code_phase_rate_step_chips=0.0)
- **bool Carrier_wipeoff_multicorrelator_resampler** (float rem_carrier_phase_in_rad, float phase_step_rad, float phase_rate_step_rad, float rem_code_phase_chips, float code_phase_step_chips, float code_phase↵_rate_step_chips, int signal_length_samples)
- **bool Carrier_wipeoff_multicorrelator_resampler** (float rem_carrier_phase_in_rad, float phase_step_rad, float rem_code_phase_chips, float code_phase_step_chips, float code_phase_rate_step_chips, int signal↵_length_samples)
- **bool free** ()

10.54.1 Detailed Description

Class that implements carrier wipe-off and correlators.

Definition at line 34 of file `cpu_multicorrelator_real_codes.h`.

The documentation for this class was generated from the following file:

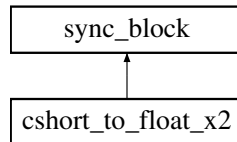
- [cpu_multicorrelator_real_codes.h](#)

10.55 cshort_to_float_x2 Class Reference

This class adapts a `std::complex<short>` stream into two 32-bits (float) streams.

```
#include <csort_to_float_x2.h>
```

Inheritance diagram for `csort_to_float_x2`:



Public Member Functions

- `int work (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)`

Friends

- `csort_to_float_x2_sptr make_csort_to_float_x2 ()`

10.55.1 Detailed Description

This class adapts a `std::complex<short>` stream into two 32-bits (float) streams.

Definition at line 47 of file `csort_to_float_x2.h`.

The documentation for this class was generated from the following file:

- [csort_to_float_x2.h](#)

10.56 CubatureFilter Class Reference

Public Member Functions

- `CubatureFilter (int nx)`
- `CubatureFilter (const arma::vec &x_pred_0, const arma::mat &P_x_pred_0)`
- `void initialize (const arma::mat &x_pred_0, const arma::mat &P_x_pred_0)`
- `void predict_sequential (const arma::vec &x_post, const arma::mat &P_x_post, ModelFunction *transition_fcn, const arma::mat &noise_covariance)`
- `void update_sequential (const arma::vec &z_upd, const arma::vec &x_pred, const arma::mat &P_x_pred, ModelFunction *measurement_fcn, const arma::mat &noise_covariance)`
- `arma::mat get_x_pred () const`
- `arma::mat get_P_x_pred () const`
- `arma::mat get_x_est () const`
- `arma::mat get_P_x_est () const`

10.56.1 Detailed Description

Definition at line 51 of file `nonlinear_tracking.h`.

The documentation for this class was generated from the following file:

- [nonlinear_tracking.h](#)

10.57 `cuda_multicorrelator` Class Reference

Class that implements carrier wipe-off and correlators using NVIDIA CUDA GPU accelerators.

```
#include <cuda_multicorrelator.h>
```

Public Member Functions

- `bool init_cuda_integrated_resampler` (int signal_length_samples, int code_length_chips, int n_correlators)
- `bool set_local_code_and_taps` (int code_length_chips, const std::complex< float > *local_codes_in, float *shifts_chips, int n_correlators)
- `bool set_input_output_vectors` (std::complex< float > *corr_out, std::complex< float > *sig_in)
- `bool free_cuda` ()
- `bool Carrier_wipeoff_multicorrelator_resampler_cuda` (float rem_carrier_phase_in_rad, float phase_step_rad, float code_phase_step_chips, float rem_code_phase_chips, int signal_length_samples, int n_correlators)

10.57.1 Detailed Description

Class that implements carrier wipe-off and correlators using NVIDIA CUDA GPU accelerators.

Definition at line 108 of file `cuda_multicorrelator.h`.

The documentation for this class was generated from the following file:

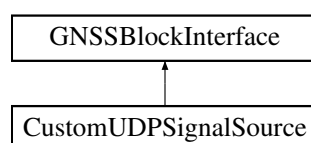
- [cuda_multicorrelator.h](#)

10.58 CustomUDPSignalSource Class Reference

This class reads from UDP packets, which streams interleaved I/Q samples over a network.

```
#include <custom_udp_signal_source.h>
```

Inheritance diagram for CustomUDPSignalSource:



Public Member Functions

- **CustomUDPSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_stream, unsigned int out_stream, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
Returns "Custom_UDP_Signal_Source".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- gr::basic_block_sptr **get_right_block** (int RF_channel) override

10.58.1 Detailed Description

This class reads from UDP packets, which streams interleaved I/Q samples over a network.

Definition at line 46 of file custom_udp_signal_source.h.

10.58.2 Member Function Documentation

10.58.2.1 implementation()

```
std::string CustomUDPSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "Custom_UDP_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 63 of file custom_udp_signal_source.h.

The documentation for this class was generated from the following file:

- [custom_udp_signal_source.h](#)

10.59 dgps_t Struct Reference

Public Attributes

- [gtime_t](#) **t0**
- double **prc**
- double **rrc**
- int **iod**
- double **udre**

10.59.1 Detailed Description

Definition at line 639 of file rtklib.h.

The documentation for this struct was generated from the following file:

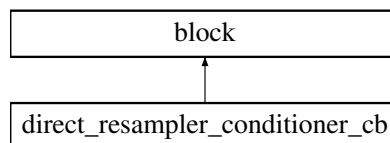
- [rtklib.h](#)

10.60 direct_resampler_conditioner_cb Class Reference

This class implements a direct resampler conditioner for `std::complex<signed char>`

```
#include <direct_resampler_conditioner_cb.h>
```

Inheritance diagram for `direct_resampler_conditioner_cb`:



Public Member Functions

- unsigned int **sample_freq_in** () const
- unsigned int **sample_freq_out** () const
- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- `direct_resampler_conditioner_cb_sptr` **direct_resampler_make_conditioner_cb** (double sample_freq_in, double sample_freq_out)

10.60.1 Detailed Description

This class implements a direct resampler conditioner for `std::complex<signed char>`

Direct resampling without interpolation

Definition at line 50 of file `direct_resampler_conditioner_cb.h`.

The documentation for this class was generated from the following file:

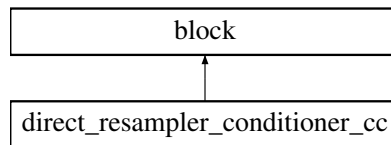
- [direct_resampler_conditioner_cb.h](#)

10.61 `direct_resampler_conditioner_cc` Class Reference

This class implements a direct resampler conditioner for complex data.

```
#include <direct_resampler_conditioner_cc.h>
```

Inheritance diagram for `direct_resampler_conditioner_cc`:



Public Member Functions

- unsigned int **sample_freq_in** () const
- unsigned int **sample_freq_out** () const
- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- `direct_resampler_conditioner_cc_sptr` **direct_resampler_make_conditioner_cc** (double sample_freq_in, double sample_freq_out)

10.61.1 Detailed Description

This class implements a direct resampler conditioner for complex data.

Direct resampling without interpolation

Definition at line 56 of file `direct_resampler_conditioner_cc.h`.

The documentation for this class was generated from the following file:

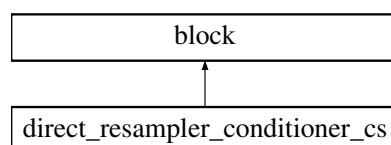
- [direct_resampler_conditioner_cc.h](#)

10.62 `direct_resampler_conditioner_cs` Class Reference

This class implements a direct resampler conditioner for `std::complex<short>`

```
#include <direct_resampler_conditioner_cs.h>
```

Inheritance diagram for `direct_resampler_conditioner_cs`:



Public Member Functions

- unsigned int **sample_freq_in** () const
- unsigned int **sample_freq_out** () const
- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- direct_resampler_conditioner_cs_sptr **direct_resampler_make_conditioner_cs** (double sample_freq_in, double sample_freq_out)

10.62.1 Detailed Description

This class implements a direct resampler conditioner for `std::complex<short>`

Direct resampling without interpolation

Definition at line 49 of file `direct_resampler_conditioner_cs.h`.

The documentation for this class was generated from the following file:

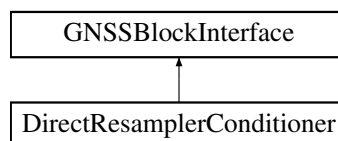
- [direct_resampler_conditioner_cs.h](#)

10.63 DirectResamplerConditioner Class Reference

Interface of an adapter of a direct resampler conditioner block to a SignalConditionerInterface.

```
#include <direct_resampler_conditioner.h>
```

Inheritance diagram for DirectResamplerConditioner:



Public Member Functions

- **DirectResamplerConditioner** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_stream, unsigned int out_stream)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "Direct_Resampler".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.63.1 Detailed Description

Interface of an adapter of a direct resampler conditioner block to a `SignalConditionerInterface`.

Definition at line 35 of file `direct_resampler_conditioner.h`.

10.63.2 Member Function Documentation

10.63.2.1 implementation()

```
std::string DirectResamplerConditioner::implementation ( ) [inline], [override], [virtual]
```

Returns "Direct_Resampler".

Implements [GNSSBlockInterface](#).

Definition at line 50 of file `direct_resampler_conditioner.h`.

The documentation for this class was generated from the following file:

- [direct_resampler_conditioner.h](#)

10.64 Dll_Pll_Conf Class Reference

Public Member Functions

- void **SetFromConfiguration** (const [ConfigurationInterface](#) *configuration, const std::string &role)

Public Attributes

- std::string **item_type**
- std::string **dump_filename**
- double **fs_in**
- double **carrier_lock_th**
- float **pll_pull_in_bw_hz**
- float **dll_pull_in_bw_hz**
- float **fil_bw_hz**
- float **pll_bw_hz**
- float **dll_bw_hz**
- float **pll_bw_narrow_hz**
- float **dll_bw_narrow_hz**
- float **early_late_space_chips**
- float **very_early_late_space_chips**
- float **early_late_space_narrow_chips**
- float **very_early_late_space_narrow_chips**
- float **slope**

- float **spc**
- float **y_intercept**
- float **cn0_smoother_alpha**
- float **carrier_lock_test_smoother_alpha**
- uint32_t **pull_in_time_s**
- uint32_t **bit_synchronization_time_limit_s**
- uint32_t **vector_length**
- uint32_t **smoother_length**
- int32_t **fil_filter_order**
- int32_t **pll_filter_order**
- int32_t **dll_filter_order**
- int32_t **extend_correlation_symbols**
- int32_t **cn0_samples**
- int32_t **cn0_smoother_samples**
- int32_t **carrier_lock_test_smoother_samples**
- int32_t **cn0_min**
- int32_t **max_code_lock_fail**
- int32_t **max_carrier_lock_fail**
- char **signal** [3] {}
- char **system**
- bool **enable_fil_pull_in**
- bool **enable_fil_steady_state**
- bool **track_pilot**
- bool **enable_doppler_correction**
- bool **carrier_aiding**
- bool **high_dyn**
- bool **dump**
- bool **dump_mat**

10.64.1 Detailed Description

Definition at line 29 of file `dll_pll_conf.h`.

The documentation for this class was generated from the following file:

- [dll_pll_conf.h](#)

10.65 Dll_Pll_Conf_Fpga Class Reference

Public Member Functions

- void **SetFromConfiguration** (const [ConfigurationInterface](#) *configuration, const std::string &role)

Public Attributes

- std::string **device_name**
- std::string **dump_filename**
- double **fs_in**
- double **carrier_lock_th**
- float **pll_pull_in_bw_hz**
- float **dll_pull_in_bw_hz**
- float **fil_bw_hz**
- float **pll_bw_hz**
- float **dll_bw_hz**
- float **pll_bw_narrow_hz**
- float **dll_bw_narrow_hz**
- float **early_late_space_chips**
- float **very_early_late_space_chips**
- float **early_late_space_narrow_chips**
- float **very_early_late_space_narrow_chips**
- float **slope**
- float **spc**
- float **y_intercept**
- float **cn0_smoother_alpha**
- float **carrier_lock_test_smoother_alpha**
- uint32_t **pull_in_time_s**
- uint32_t **bit_synchronization_time_limit_s**
- uint32_t **vector_length**
- uint32_t **smoother_length**
- uint32_t **dev_file_num**
- uint32_t **num_prev_assigned_ch**
- uint32_t **code_length_chips**
- uint32_t **code_samples_per_chip**
- uint32_t **extend_fpga_integration_periods**
- uint32_t **fpga_integration_period**
- int32_t **fil_filter_order**
- int32_t **pll_filter_order**
- int32_t **dll_filter_order**
- int32_t **extend_correlation_symbols**
- int32_t **cn0_samples**
- int32_t **cn0_min**
- int32_t **max_code_lock_fail**
- int32_t **max_carrier_lock_fail**
- int32_t **cn0_smoother_samples**
- int32_t **carrier_lock_test_smoother_samples**
- int32_t * **ca_codes**
- int32_t * **data_codes**
- char **signal** [3]
- char **system**
- bool **extended_correlation_in_fpga**
- bool **track_pilot**
- bool **enable_doppler_correction**
- bool **enable_fil_pull_in**
- bool **enable_fil_steady_state**
- bool **carrier_aiding**
- bool **high_dyn**
- bool **dump**
- bool **dump_mat**

10.65.1 Detailed Description

Definition at line 31 of file `dll_pll_conf_fpga.h`.

The documentation for this class was generated from the following file:

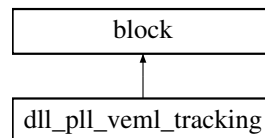
- [dll_pll_conf_fpga.h](#)

10.66 dll_pll_veml_tracking Class Reference

This class implements a code DLL + carrier PLL tracking block.

```
#include <dll_pll_veml_tracking.h>
```

Inheritance diagram for `dll_pll_veml_tracking`:



Public Member Functions

- void **set_channel** (uint32_t channel)
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro)
- void **start_tracking** ()
- void **stop_tracking** ()
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)

Friends

- `dll_pll_veml_tracking_sptr` **dll_pll_veml_make_tracking** (const [Dll_Pll_Conf](#) &conf_)

10.66.1 Detailed Description

This class implements a code DLL + carrier PLL tracking block.

Definition at line 62 of file `dll_pll_veml_tracking.h`.

The documentation for this class was generated from the following file:

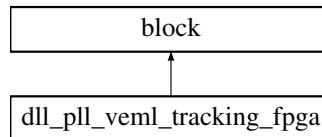
- [dll_pll_veml_tracking.h](#)

10.67 dll_pll_veml_tracking_fpga Class Reference

This class implements a code DLL + carrier PLL tracking block.

```
#include <dll_pll_veml_tracking_fpga.h>
```

Inheritance diagram for `dll_pll_veml_tracking_fpga`:



Public Member Functions

- `~dll_pll_veml_tracking_fpga ()`
Destructor.
- void `set_channel` (uint32_t channel)
Set the channel number and configure some multicorrelator parameters.
- void `set_gnss_synchro` (Gnss_Synchro *p_gnss_synchro)
This function is used with two purposes: 1 -> To set the gnss_synchro 2 -> A set_gnss_synchro command with a valid PRN is received when the system is going to run acquisition with that PRN. We can use this command to pre-initialize tracking parameters and variables before the actual acquisition process takes place. In this way we minimize the latency between acquisition and tracking once the acquisition has been made.
- void `start_tracking` ()
This function starts the tracking process.
- void `stop_tracking` ()
This function sets a flag that makes general_work to stop in order to finish the tracking process.
- int `general_work` (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
General Work.
- void `reset` ()
This function disables the HW multicorrelator in the FPGA in order to stop the tracking process.

Friends

- `dll_pll_veml_tracking_fpga_sptr dll_pll_veml_make_tracking_fpga` (const `Dll_Pll_Conf_Fpga` &conf_)

10.67.1 Detailed Description

This class implements a code DLL + carrier PLL tracking block.

Definition at line 62 of file `dll_pll_veml_tracking_fpga.h`.

10.67.2 Constructor & Destructor Documentation

10.67.2.1 ~dll_pll_veml_tracking_fpga()

```
dll_pll_veml_tracking_fpga::~dll_pll_veml_tracking_fpga ( )
```

Destructor.

10.67.3 Member Function Documentation

10.67.3.1 general_work()

```
int dll_pll_veml_tracking_fpga::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

General Work.

10.67.3.2 reset()

```
void dll_pll_veml_tracking_fpga::reset ( )
```

This function disables the HW multicorrelator in the FPGA in order to stop the tracking process.

10.67.3.3 set_channel()

```
void dll_pll_veml_tracking_fpga::set_channel (
    uint32_t channel )
```

Set the channel number and configure some multicorrelator parameters.

10.67.3.4 set_gnss_synchro()

```
void dll_pll_veml_tracking_fpga::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro )
```

This function is used with two purposes: 1 -> To set the gnss_synchro 2 -> A set_gnss_synchro command with a valid PRN is received when the system is going to run acquisition with that PRN. We can use this command to pre-initialize tracking parameters and variables before the actual acquisition process takes place. In this way we minimize the latency between acquisition and tracking once the acquisition has been made.

10.67.3.5 start_tracking()

```
void dll_pll_veml_tracking_fpga::start_tracking ( )
```

This function starts the tracking process.

10.67.3.6 stop_tracking()

```
void dll_pll_veml_tracking_fpga::stop_tracking ( )
```

This function sets a flag that makes general_work to stop in order to finish the tracking process.

The documentation for this class was generated from the following file:

- [dll_pll_veml_tracking_fpga.h](#)

10.68 eph_t Struct Reference

Public Attributes

- int **sat**
- int **iode**
- int **iodc**
- int **sva**
- int **svh**
- int **week**
- int **code**
- int **flag**
- [gtime_t](#) **toe**
- [gtime_t](#) **toc**
- [gtime_t](#) **ttr**
- double **A**
- double **e**
- double **i0**
- double **OMG0**
- double **omg**
- double **M0**
- double **deln**
- double **OMGd**
- double **idot**
- double **crc**
- double **crs**
- double **cuc**
- double **cus**
- double **cic**
- double **cis**
- double **toes**
- double **fit**
- double **f0**
- double **f1**
- double **f2**
- double **tgd** [4]
- double **isc** [4]
- double **Adot**
- double **ndot**

10.68.1 Detailed Description

Definition at line 431 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.69 `erp_t` Struct Reference

Public Attributes

- `int n`
- `int nmax`
- `erp_t * data`

10.69.1 Detailed Description

Definition at line 391 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.70 `erpd_t` Struct Reference

Public Attributes

- `double mjd`
- `double xp`
- `double yp`
- `double xpr`
- `double ypr`
- `double ut1_utc`
- `double lod`

10.70.1 Detailed Description

Definition at line 381 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.71 Exponential_Smoother Class Reference

Class that implements a first-order exponential smoother.

```
#include <exponential_smoother.h>
```

Public Member Functions

- [Exponential_Smoother](#) ()
Constructor.
- [~Exponential_Smoother](#) ()=default
Destructor.
- [Exponential_Smoother](#) ([Exponential_Smoother](#) &&)=default
Move operator.
- [Exponential_Smoother](#) & [operator=](#) ([Exponential_Smoother](#) &&)=default
Move assignment operator.
- void [set_alpha](#) (float alpha)
 $0 < \alpha < 1$. The higher, the most responsive, but more variance. Default value: 0.001
- void [set_samples_for_initialization](#) (int num_samples)
Number of samples averaged for initialization. Default value: 200.
- void [reset](#) ()
- void [set_min_value](#) (float value)
- void [set_offset](#) (float offset)
- float [smooth](#) (float raw)
- double [smooth](#) (double raw)

10.71.1 Detailed Description

Class that implements a first-order exponential smoother.

$$\text{smoothed_value}[k] = \alpha * \text{raw} + (1-\alpha) * \text{smoothed_value}[k-1]$$

The length of the initialization can be controlled with [set_samples_for_initialization\(int num_samples\)](#)

Definition at line 36 of file exponential_smoother.h.

10.71.2 Constructor & Destructor Documentation

10.71.2.1 Exponential_Smoother() [1/2]

```
Exponential_Smoother::Exponential_Smoother ( )
```

Constructor.

10.71.2.2 ~Exponential_Smoother()

```
Exponential_Smoother::~~Exponential_Smoother ( ) [default]
```

Destructor.

10.71.2.3 Exponential_Smoother() [2/2]

```
Exponential_Smoother::Exponential_Smoother (
    Exponential_Smoother && ) [default]
```

Move operator.

10.71.3 Member Function Documentation

10.71.3.1 operator=()

```
Exponential_Smoother& Exponential_Smoother::operator= (
    Exponential_Smoother && ) [default]
```

Move assignment operator.

10.71.3.2 set_alpha()

```
void Exponential_Smoother::set_alpha (
    float alpha )
```

$0 < \alpha < 1$. The higher, the most responsive, but more variance. Default value: 0.001

10.71.3.3 set_samples_for_initialization()

```
void Exponential_Smoother::set_samples_for_initialization (
    int num_samples )
```

Number of samples averaged for initialization. Default value: 200.

The documentation for this class was generated from the following file:

- [exponential_smoother.h](#)

10.72 exterr_t Struct Reference

Public Attributes

- int **ena** [4]
- double **cerr** [4][[NFREQ](#) *2]
- double **perr** [4][[NFREQ](#) *2]
- double **gpsglob** [[NFREQ](#)]
- double **gloicb** [[NFREQ](#)]

10.72.1 Detailed Description

Definition at line 919 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.73 fcbd_t Struct Reference

Public Attributes

- [gtime_t](#) **ts**
- [gtime_t](#) **te**
- double **bias** [MAXSAT][3]
- double **std** [MAXSAT][3]

10.73.1 Detailed Description

Definition at line 551 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.74 `file_t` Struct Reference

Public Attributes

- `FILE * fp`
- `FILE * fp_tag`
- `FILE * fp_tmp`
- `FILE * fp_tag_tmp`
- `char path [MAXSTRPATH]`
- `char openpath [MAXSTRPATH]`
- `int mode`
- `int timetag`
- `int repmode`
- `int offset`
- `gtime_t time`
- `gtime_t wtime`
- `unsigned int tick`
- `unsigned int tick_f`
- `unsigned int fpos`
- `double start`
- `double speed`
- `double swapintv`
- `lock_t lock`

10.74.1 Detailed Description

Definition at line 1110 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

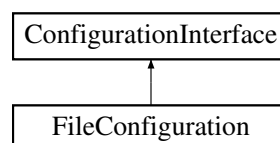
- [rtklib.h](#)

10.75 FileConfiguration Class Reference

This class is an implementation of the interface [ConfigurationInterface](#).

```
#include <file_configuration.h>
```

Inheritance diagram for FileConfiguration:



Public Member Functions

- **FileConfiguration** (std::string filename)
- std::string **property** (std::string property_name, std::string default_value) const override
- bool **property** (std::string property_name, bool default_value) const override
- int64_t **property** (std::string property_name, int64_t default_value) const override
- uint64_t **property** (std::string property_name, uint64_t default_value) const override
- int32_t **property** (std::string property_name, int32_t default_value) const override
- uint32_t **property** (std::string property_name, uint32_t default_value) const override
- int16_t **property** (std::string property_name, int16_t default_value) const override
- uint16_t **property** (std::string property_name, uint16_t default_value) const override
- float **property** (std::string property_name, float default_value) const override
- double **property** (std::string property_name, double default_value) const override
- void **set_property** (std::string property_name, std::string value) override
- bool **is_present** (const std::string &property_name) const

10.75.1 Detailed Description

This class is an implementation of the interface [ConfigurationInterface](#).

Derived from [ConfigurationInterface](#), this class implements an interface to a configuration file. This implementation has a text file as the source for the values of the parameters. The file is in the INI format, containing sections and pairs of names and values. For more information about the INI format, see https://en.wikipedia.org/wiki/INI_file

Definition at line 45 of file file_configuration.h.

The documentation for this class was generated from the following file:

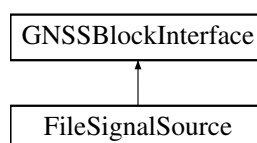
- [file_configuration.h](#)

10.76 FileSignalSource Class Reference

Class that reads signals samples from a file and adapts it to a SignalSourceInterface.

```
#include <file_signal_source.h>
```

Inheritance diagram for FileSignalSource:



Public Member Functions

- **FileSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
Returns "File_Signal_Source".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- std::string **filename** () const
- std::string **item_type** () const
- bool **repeat** () const
- int64_t **sampling_frequency** () const
- uint64_t **samples** () const

10.76.1 Detailed Description

Class that reads signals samples from a file and adapts it to a [SignalSourceInterface](#).

Definition at line 48 of file `file_signal_source.h`.

10.76.2 Member Function Documentation

10.76.2.1 implementation()

```
std::string FileSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "File_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 65 of file `file_signal_source.h`.

The documentation for this class was generated from the following file:

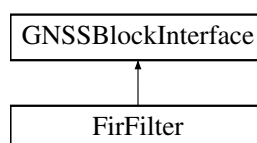
- [file_signal_source.h](#)

10.77 FirFilter Class Reference

This class adapts a GNU Radio `gr_fir_filter` designed with `pm_remez`.

```
#include <fir_filter.h>
```

Inheritance diagram for `FirFilter`:



Public Member Functions

- **FirFilter** (const [ConfigurationInterface](#) *configuration, std::string role, unsigned int in_streams, unsigned int out_streams)
Constructor.
- **~FirFilter** ()=default
Destructor.
- std::string **role** () override
- std::string **implementation** () override
Returns "Fir_Filter".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.77.1 Detailed Description

This class adapts a GNU Radio `gr_fir_filter` designed with `pm_remez`.

See Parks-McClellan FIR filter design, https://en.wikipedia.org/wiki/Parks-McClellan_filter_design_algorithm Calculates the optimal (in the Chebyshev/minimax sense) FIR filter impulse response given a set of band edges, the desired response on those bands, and the weight given to the error in those bands.

Definition at line 54 of file `fir_filter.h`.

10.77.2 Constructor & Destructor Documentation

10.77.2.1 FirFilter()

```
FirFilter::FirFilter (
    const ConfigurationInterface * configuration,
    std::string role,
    unsigned int in_streams,
    unsigned int out_streams )
```

Constructor.

10.77.2.2 ~FirFilter()

```
FirFilter::~~FirFilter ( ) [default]
```

Destructor.

10.77.3 Member Function Documentation

10.77.3.1 implementation()

```
std::string FirFilter::implementation ( ) [inline], [override], [virtual]
```

Returns "Fir_Filter".

Implements [GNSSBlockInterface](#).

Definition at line 72 of file fir_filter.h.

The documentation for this class was generated from the following file:

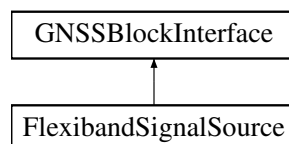
- [fir_filter.h](#)

10.78 FlexibandSignalSource Class Reference

This class configures and reads samples from Teleorbit Flexiband front-end. This software requires a Flexiband GNU Radio driver installed (not included with GNSS-SDR).

```
#include <flexiband_signal_source.h>
```

Inheritance diagram for FlexibandSignalSource:



Public Member Functions

- **FlexibandSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_stream, unsigned int out_stream, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
Returns "Flexiband_Signal_Source".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- gr::basic_block_sptr **get_right_block** (int RF_channel) override

10.78.1 Detailed Description

This class configures and reads samples from Teleorbit Flexiband front-end. This software requires a Flexiband GNU Radio driver installed (not included with GNSS-SDR).

Definition at line 45 of file flexiband_signal_source.h.

10.78.2 Member Function Documentation

10.78.2.1 implementation()

```
std::string FlexibandSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "Flexiband_Signal_Source".

Implements [GNSSBlockInterface](#).

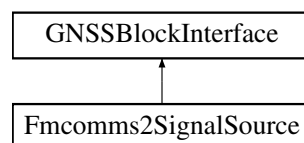
Definition at line 62 of file flexiband_signal_source.h.

The documentation for this class was generated from the following file:

- [flexiband_signal_source.h](#)

10.79 Fmcomms2SignalSource Class Reference

Inheritance diagram for Fmcomms2SignalSource:



Public Member Functions

- **Fmcomms2SignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_stream, unsigned int out_stream, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "Fmcomms2_Signal_Source".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.79.1 Detailed Description

Definition at line 45 of file fmcomms2_signal_source.h.

10.79.2 Member Function Documentation

10.79.2.1 implementation()

```
std::string Fmcomms2SignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "Fmcomms2_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 62 of file fmcomms2_signal_source.h.

The documentation for this class was generated from the following file:

- [fmcomms2_signal_source.h](#)

10.80 Fpga_Acquisition Class Reference

Class that implements carrier wipe-off and correlators.

```
#include <fpga_acquisition.h>
```

Public Member Functions

- [Fpga_Acquisition](#) (std::string device_name, uint32_t nsamples, uint32_t doppler_max, uint32_t nsamples_↵ total, int64_t fs_in, uint32_t select_queue, uint32_t *all_fft_codes, uint32_t excludelimit)
Constructor.
- [~Fpga_Acquisition](#) ()=default
Destructor.
- bool [set_local_code](#) (uint32_t PRN)
Select the code with the chosen PRN.
- void [set_doppler_sweep](#) (uint32_t num_sweeps, uint32_t doppler_step, int32_t doppler_min)
Configure the doppler sweep parameters in the FPGA.
- void [run_acquisition](#) ()
Run the acquisition process in the FPGA.
- void [read_acquisition_results](#) (uint32_t *max_index, float *firstpeak, float *secondpeak, uint64_t *initial_↵ sample, float *power_sum, uint32_t *doppler_index, uint32_t *total_blk_exp)
Read the results of the acquisition process.
- void [set_doppler_max](#) (uint32_t doppler_max)
Set maximum Doppler grid search.
- void [set_doppler_step](#) (uint32_t doppler_step)

- *Set Doppler steps for the grid search.*
- void `reset_acquisition` ()
- *Reset the FPGA PL.*
- void `read_fpga_total_scale_factor` (uint32_t *total_scale_factor, uint32_t *fw_scale_factor)
- *Read the scaling factor that has been used by the FFT-IFFT.*
- void `set_block_exp` (uint32_t total_block_exp)
- *Set the block exponent of the FFT in the FPGA.*
- void `write_local_code` (void)
- *Write the PRN code in the FPGA.*
- void `configure_acquisition` (void)
- *Write the acquisition parameters into the FPGA.*
- void `open_device` ()
- *Open the device driver.*
- void `close_device` ()
- *Close the device driver.*

10.80.1 Detailed Description

Class that implements carrier wipe-off and correlators.

Definition at line 34 of file `fpga_acquisition.h`.

10.80.2 Constructor & Destructor Documentation

10.80.2.1 Fpga_Acquisition()

```
Fpga_Acquisition::Fpga_Acquisition (
    std::string device_name,
    uint32_t nsamples,
    uint32_t doppler_max,
    uint32_t nsamples_total,
    int64_t fs_in,
    uint32_t select_queue,
    uint32_t * all_fft_codes,
    uint32_t excludelimit )
```

Constructor.

10.80.2.2 ~Fpga_Acquisition()

```
Fpga_Acquisition::~~Fpga_Acquisition ( ) [default]
```

Destructor.

10.80.3 Member Function Documentation

10.80.3.1 close_device()

```
void Fpga_Acquisition::close_device ( )
```

Close the device driver.

10.80.3.2 configure_acquisition()

```
void Fpga_Acquisition::configure_acquisition (
    void )
```

Write the acquisition parameters into the FPGA.

10.80.3.3 open_device()

```
void Fpga_Acquisition::open_device ( )
```

Open the device driver.

10.80.3.4 read_acquisition_results()

```
void Fpga_Acquisition::read_acquisition_results (
    uint32_t * max_index,
    float * firstpeak,
    float * secondpeak,
    uint64_t * initial_sample,
    float * power_sum,
    uint32_t * doppler_index,
    uint32_t * total_blk_exp )
```

Read the results of the acquisition process.

10.80.3.5 read_fpga_total_scale_factor()

```
void Fpga_Acquisition::read_fpga_total_scale_factor (
    uint32_t * total_scale_factor,
    uint32_t * fw_scale_factor )
```

Read the scaling factor that has been used by the FFT-IFFT.

10.80.3.6 reset_acquisition()

```
void Fpga_Acquisition::reset_acquisition ( )
```

Reset the FPGA PL.

10.80.3.7 run_acquisition()

```
void Fpga_Acquisition::run_acquisition ( )
```

Run the acquisition process in the FPGA.

10.80.3.8 set_block_exp()

```
void Fpga_Acquisition::set_block_exp (
    uint32_t total_block_exp )
```

Set the block exponent of the FFT in the FPGA.

10.80.3.9 set_doppler_max()

```
void Fpga_Acquisition::set_doppler_max (
    uint32_t doppler_max ) [inline]
```

Set maximum Doppler grid search.

Parameters

<i>doppler_max</i>	- Maximum Doppler shift considered in the grid search [Hz].
--------------------	---

Definition at line 86 of file fpga_acquisition.h.

10.80.3.10 set_doppler_step()

```
void Fpga_Acquisition::set_doppler_step (
    uint32_t doppler_step ) [inline]
```

Set Doppler steps for the grid search.

Parameters

<i>doppler_step</i>	- Frequency bin of the search grid [Hz].
---------------------	--

Definition at line 95 of file `fpga_acquisition.h`.

10.80.3.11 set_doppler_sweep()

```
void Fpga_Acquisition::set_doppler_sweep (
    uint32_t num_sweeps,
    uint32_t doppler_step,
    int32_t doppler_min )
```

Configure the doppler sweep parameters in the FPGA.

10.80.3.12 set_local_code()

```
bool Fpga_Acquisition::set_local_code (
    uint32_t PRN )
```

Select the code with the chosen PRN.

10.80.3.13 write_local_code()

```
void Fpga_Acquisition::write_local_code (
    void )
```

Write the PRN code in the FPGA.

The documentation for this class was generated from the following file:

- [fpga_acquisition.h](#)

10.81 Fpga_dynamic_bit_selection Class Reference

Class that controls the switch in the FPGA, which connects the FPGA acquisition and multicorrelator modules to either the DMA or the Analog Front-End.

```
#include <fpga_dynamic_bit_selection.h>
```

Public Member Functions

- [Fpga_dynamic_bit_selection](#) (const std::string &device_name1, const std::string &device_name2)
Constructor.
- [~Fpga_dynamic_bit_selection](#) ()
Destructor.
- void [bit_selection](#) (void)
This function configures the switch in the eFPGA.

10.81.1 Detailed Description

Class that controls the switch in the FPGA, which connects the FPGA acquisition and multicorrelator modules to either the DMA or the Analog Front-End.

Definition at line 36 of file `fpga_dynamic_bit_selection.h`.

10.81.2 Constructor & Destructor Documentation

10.81.2.1 Fpga_dynamic_bit_selection()

```
Fpga_dynamic_bit_selection::Fpga_dynamic_bit_selection (
    const std::string & device_name1,
    const std::string & device_name2 ) [explicit]
```

Constructor.

10.81.2.2 ~Fpga_dynamic_bit_selection()

```
Fpga_dynamic_bit_selection::~~Fpga_dynamic_bit_selection ( )
```

Destructor.

10.81.3 Member Function Documentation

10.81.3.1 bit_selection()

```
void Fpga_dynamic_bit_selection::bit_selection (
    void )
```

This function configures the switch in the eFPGA.

The documentation for this class was generated from the following file:

- [fpga_dynamic_bit_selection.h](#)

10.82 Fpga_Multicorrelator_8sc Class Reference

Class that implements carrier wipe-off and correlators.

```
#include <fpga_multicorrelator.h>
```

Public Member Functions

- [Fpga_Multicorrelator_8sc](#) (int32_t n_correlators, const std::string &device_name, uint32_t dev_file_num, uint32_t num_prev_assigned_ch, int32_t *ca_codes, int32_t *data_codes, uint32_t code_length_chips, bool track_pilot, uint32_t code_samples_per_chip)
Constructor.
- [~Fpga_Multicorrelator_8sc](#) ()
Destructor.
- void [set_output_vectors](#) (gr_complex *corr_out, gr_complex *Prompt_Data)
Configure pointers to the FPGA multicorrelator results.
- void [set_local_code_and_taps](#) (float *shifts_chips, float *prompt_data_shift, int32_t PRN)
Configure the local code in the FPGA multicorrelator.
- void [update_local_code](#) ()
Configure code phase and code rate parameters in the FPGA.
- void [Carrier_wipeoff_multicorrelator_resampler](#) (float rem_carrier_phase_in_rad, float phase_step_rad, float carrier_phase_rate_step_rad, float rem_code_phase_chips, float code_phase_step_chips, float code_phase_rate_step_chips, int32_t signal_length_samples)
Perform a multicorrelation.
- bool [free](#) ()
Stop the correlation process in the FPGA and free code phase and code rate parameters.
- void [set_channel](#) (uint32_t channel)
Set channel number and open the FPGA device driver.
- void [set_initial_sample](#) (uint64_t samples_offset)
Set the initial sample number where the tracking process begins.
- uint64_t [read_sample_counter](#) ()
Read the sample counter in the FPGA.
- void [lock_channel](#) ()
Start the tracking process in the FPGA.
- void [unlock_channel](#) ()
finish the tracking process in the FPGA
- void [set_secondary_code_lengths](#) (uint32_t secondary_code_0_length, uint32_t secondary_code_1_length)
Set the secondary code length in the FPGA. This is only used when extended coherent integration is enabled in the FPGA. If tracking the pilot is enabled then secondary_code_0_length is the length of the pilot secondary code and secondary_code_1_length is the length of the data secondary code. If tracking the pilot is disabled then secondary_code_0_length is the length of the data secondary code, and secondary_code_1_length must be set to zero.
- void [initialize_secondary_code](#) (uint32_t secondary_code, std::string *secondary_code_string)
Initialize the secondary code in the FPGA. If tracking the pilot is enabled then the pilot secondary code is configured when secondary_code = 0 and the data secondary code is configured when secondary_code = 1. If tracking the pilot is disabled then the data secondary code is configured when secondary_code = 0.
- void [update_prn_code_length](#) (uint32_t first_prn_length, uint32_t next_prn_length)
Set the PRN length in the FPGA in number of samples. This function is only used then extended coherent integration is enabled in the FPGA. The FPGA allows for the configuration of two PRN lengths. When the length of the extended coherent integration is bigger than the length of the PRN code, the FPGA uses the first_length_secondary_code as the length of the PRN code immediately following the beginning of the extended coherent integration, and the next_length_secondary_code as the length of the remaining PRN codes. The purpose of this is to have the option to allow the FPGA to compensate for a possible deviation between the nominal value of the PRN code length and the measured PRN code length in the PRN immediately following the start of the coherent integration only. If this option is not used then write the same value to first_length_secondary_code and next_length_secondary_code.

- void [enable_secondary_codes](#) ()
Enable the use of secondary codes in the FPGA.
- void [disable_secondary_codes](#) ()
Disable the use of secondary codes in the FPGA.

10.82.1 Detailed Description

Class that implements carrier wipe-off and correlators.

Definition at line 38 of file `fpga_multicorrelator.h`.

10.82.2 Constructor & Destructor Documentation

10.82.2.1 Fpga_Multicorrelator_8sc()

```
Fpga_Multicorrelator_8sc::Fpga_Multicorrelator_8sc (
    int32_t n_correlators,
    const std::string & device_name,
    uint32_t dev_file_num,
    uint32_t num_prev_assigned_ch,
    int32_t * ca_codes,
    int32_t * data_codes,
    uint32_t code_length_chips,
    bool track_pilot,
    uint32_t code_samples_per_chip )
```

Constructor.

10.82.2.2 ~Fpga_Multicorrelator_8sc()

```
Fpga_Multicorrelator_8sc::~Fpga_Multicorrelator_8sc ( )
```

Destructor.

10.82.3 Member Function Documentation

10.82.3.1 Carrier_wipeoff_multicorrelator_resampler()

```
void Fpga_Multicorrelator_8sc::Carrier_wipeoff_multicorrelator_resampler (
    float rem_carrier_phase_in_rad,
    float phase_step_rad,
    float carrier_phase_rate_step_rad,
    float rem_code_phase_chips,
    float code_phase_step_chips,
    float code_phase_rate_step_chips,
    int32_t signal_length_samples )
```

Perform a multicorrelation.

10.82.3.2 disable_secondary_codes()

```
void Fpga_Multicorrelator_8sc::disable_secondary_codes ( )
```

Disable the use of secondary codes in the FPGA.

10.82.3.3 enable_secondary_codes()

```
void Fpga_Multicorrelator_8sc::enable_secondary_codes ( )
```

Enable the use of secondary codes in the FPGA.

10.82.3.4 free()

```
bool Fpga_Multicorrelator_8sc::free ( )
```

Stop the correlation process in the FPGA and free code phase and code rate parameters.

10.82.3.5 initialize_secondary_code()

```
void Fpga_Multicorrelator_8sc::initialize_secondary_code (
    uint32_t secondary_code,
    std::string * secondary_code_string )
```

Initialize the secondary code in the FPGA. If tracking the pilot is enabled then the pilot secondary code is configured when `secondary_code = 0` and the data secondary code is configured when `secondary_code = 1`. If tracking the pilot is disabled then the data secondary code is configured when `secondary_code = 0`.

10.82.3.6 lock_channel()

```
void Fpga_Multicorrelator_8sc::lock_channel ( )
```

Start the tracking process in the FPGA.

10.82.3.7 read_sample_counter()

```
uint64_t Fpga_Multicorrelator_8sc::read_sample_counter ( )
```

Read the sample counter in the FPGA.

10.82.3.8 set_channel()

```
void Fpga_Multicorrelator_8sc::set_channel (
    uint32_t channel )
```

Set channel number and open the FPGA device driver.

10.82.3.9 set_initial_sample()

```
void Fpga_Multicorrelator_8sc::set_initial_sample (
    uint64_t samples_offset )
```

Set the initial sample number where the tracking process begins.

10.82.3.10 set_local_code_and_taps()

```
void Fpga_Multicorrelator_8sc::set_local_code_and_taps (
    float * shifts_chips,
    float * prompt_data_shift,
    int32_t PRN )
```

Configure the local code in the FPGA multicorrelator.

10.82.3.11 set_output_vectors()

```
void Fpga_Multicorrelator_8sc::set_output_vectors (
    gr_complex * corr_out,
    gr_complex * Prompt_Data )
```

Configure pointers to the FPGA multicorrelator results.

10.82.3.12 set_secondary_code_lengths()

```
void Fpga_Multicorrelator_8sc::set_secondary_code_lengths (
    uint32_t secondary_code_0_length,
    uint32_t secondary_code_1_length )
```

Set the secondary code length in the FPGA. This is only used when extended coherent integration is enabled in the FPGA. If tracking the pilot is enabled then `secondary_code_0_length` is the length of the pilot secondary code and `secondary_code_1_length` is the length of the data secondary code. If tracking the pilot is disabled then `secondary_code_0_length` is the length of the data secondary code, and `secondary_code_1_length` must be set to zero.

10.82.3.13 unlock_channel()

```
void Fpga_Multicorrelator_8sc::unlock_channel ( )
```

finish the tracking process in the FPGA

10.82.3.14 update_local_code()

```
void Fpga_Multicorrelator_8sc::update_local_code ( )
```

Configure code phase and code rate parameters in the FPGA.

10.82.3.15 update_prn_code_length()

```
void Fpga_Multicorrelator_8sc::update_prn_code_length (
    uint32_t first_prn_length,
    uint32_t next_prn_length )
```

Set the PRN length in the FPGA in number of samples. This function is only used then extended coherent integration is enabled in the FPGA. The FPGA allows for the configuration of two PRN lengths. When the length of the extended coherent integration is bigger than the length of the PRN code, the FPGA uses the `first_length_secondary_code` as the length of the PRN code immediately following the beginning of the extended coherent integration, and the `next_length_secondary_code` as the length of the remaining PRN codes. The purpose of this is to have the option to allow the FPGA to compensate for a possible deviation between the nominal value of the PRN code length and the measured PRN code length in the PRN immediately following the start of the coherent integration only. If this option is not used then write the same value to `first_length_secondary_code` and `next_length_secondary_code`.

The documentation for this class was generated from the following file:

- [fpga_multicorrelator.h](#)

10.83 Fpga_Switch Class Reference

Class that controls the switch in the FPGA, which connects the FPGA acquisition and multicorrelator modules to either the DMA or the Analog Front-End.

```
#include <fpga_switch.h>
```

Public Member Functions

- [Fpga_Switch](#) (const std::string &device_name)
Constructor.
- [~Fpga_Switch](#) ()
Destructor.
- void [set_switch_position](#) (int32_t switch_position)
This function configures the switch in the FPGA.

10.83.1 Detailed Description

Class that controls the switch in the FPGA, which connects the FPGA acquisition and multicorrelator modules to either the DMA or the Analog Front-End.

Definition at line 35 of file fpga_switch.h.

10.83.2 Constructor & Destructor Documentation

10.83.2.1 Fpga_Switch()

```
Fpga_Switch::Fpga_Switch (
    const std::string & device_name ) [explicit]
```

Constructor.

10.83.2.2 ~Fpga_Switch()

```
Fpga_Switch::~~Fpga_Switch ( )
```

Destructor.

10.83.3 Member Function Documentation

10.83.3.1 set_switch_position()

```
void Fpga_Switch::set_switch_position (
    int32_t switch_position )
```

This function configures the switch in the eFPGA.

The documentation for this class was generated from the following file:

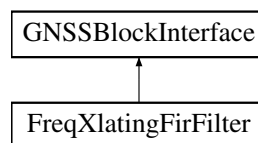
- [fpga_switch.h](#)

10.84 FreqXlatingFirFilter Class Reference

This class adapts a gnuradio `gr_freq_xlating_fir_filter` designed with `pm_remez`.

```
#include <freq_xlating_fir_filter.h>
```

Inheritance diagram for FreqXlatingFirFilter:



Public Member Functions

- **FreqXlatingFirFilter** (const [ConfigurationInterface](#) *configuration, std::string role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "Freq_Xlating_Fir_Filter".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.84.1 Detailed Description

This class adapts a gnuradio `gr_freq_xlating_fir_filter` designed with `pm_remez`.

Construct a FIR filter with the given taps and a composite frequency translation that shifts `intermediate_freq` down to zero Hz. The frequency translation logically comes before the filtering operation.

See Parks-McClellan FIR filter design, https://en.wikipedia.org/wiki/Parks-McClellan_filter_design_algorithm Calculates the optimal (in the Chebyshev/minimax sense) FIR filter impulse response given a set of band edges, the desired response on those bands, and the weight given to the error in those bands.

Definition at line 55 of file `freq_xlating_fir_filter.h`.

10.84.2 Member Function Documentation

10.84.2.1 implementation()

```
std::string FreqXlatingFirFilter::implementation ( ) [inline], [override], [virtual]
```

Returns "Freq_Xlating_Fir_Filter".

Implements [GNSSBlockInterface](#).

Definition at line 70 of file `freq_xlating_fir_filter.h`.

The documentation for this class was generated from the following file:

- [freq_xlating_fir_filter.h](#)

10.85 FrontEndCal Class Reference

Public Member Functions

- void [set_configuration](#) (std::shared_ptr< [ConfigurationInterface](#) > configuration)
Sets the configuration data required by `get_ephemeris` function.
- bool [get_ephemeris](#) ()
This function connects to a Secure User Location Protocol (SUPL) server to obtain the current GPS ephemeris and GPS assistance data. It requires the configuration parameters set by `set_configuration` function.
- double [estimate_doppler_from_eph](#) (unsigned int PRN, double tow, double lat, double lon, double height) noexcept(false)
This function estimates the GPS L1 satellite Doppler frequency [Hz] using the following data: 1- Orbital model from the ephemeris 2- Approximate GPS Time of Week (TOW) 3- Approximate receiver Latitude and Longitude (WGS-84)
- void [GPS_L1_front_end_model_E4000](#) (double `f_bb_true_Hz`, double `f_bb_meas_Hz`, double `fs_nominal`, double `*estimated_fs_Hz`, double `*estimated_f_if_Hz`, double `*f_osc_err_ppm`)
This function models the Elonics E4000 + RTL2832 front-end Inputs: `f_bb_true_Hz` - Ideal output frequency in baseband [Hz] `f_in_bb_meas_Hz` - measured output frequency in baseband [Hz] Outputs: `estimated_fs_Hz` - Sampling frequency estimation based on the measurements and the front-end model `estimated_f_if_bb_Hz` - Equivalent bb if frequency estimation based on the measurements and the front-end model Front-end TUNER Elonics E4000 + RTL2832 sampler For GPS L1 1575.42 MHz.

10.85.1 Detailed Description

Definition at line 30 of file `front_end_cal.h`.

10.85.2 Member Function Documentation

10.85.2.1 estimate_doppler_from_eph()

```
double FrontEndCal::estimate_doppler_from_eph (
    unsigned int PRN,
    double tow,
    double lat,
    double lon,
    double height ) [noexcept]
```

This function estimates the GPS L1 satellite Doppler frequency [Hz] using the following data: 1- Orbital model from the ephemeris 2- Approximate GPS Time of Week (TOW) 3- Approximate receiver Latitude and Longitude (WGS-84)

10.85.2.2 get_ephemeris()

```
bool FrontEndCal::get_ephemeris ( )
```

This function connects to a Secure User Location Protocol (SUPL) server to obtain the current GPS ephemeris and GPS assistance data. It requires the configuration parameters set by set_configuration function.

10.85.2.3 GPS_L1_front_end_model_E4000()

```
void FrontEndCal::GPS_L1_front_end_model_E4000 (
    double f_bb_true_Hz,
    double f_bb_meas_Hz,
    double fs_nominal_hz,
    double * estimated_fs_Hz,
    double * estimated_f_if_Hz,
    double * f_osc_err_ppm )
```

This function models the Elonics E4000 + RTL2832 front-end Inputs: f_bb_true_Hz - Ideal output frequency in baseband [Hz] f_in_bb_meas_Hz - measured output frequency in baseband [Hz] Outputs: estimated_fs_Hz - Sampling frequency estimation based on the measurements and the front-end model estimated_f_if_bb_Hz - Equivalent bb if frequency estimation based on the measurements and the front-end model Front-end TUNER Elonics E4000 + RTL2832 sampler For GPS L1 1575.42 MHz.

10.85.2.4 set_configuration()

```
void FrontEndCal::set_configuration (
    std::shared_ptr< ConfigurationInterface > configuration )
```

Sets the configuration data required by get_ephemeris function.

The documentation for this class was generated from the following file:

- [front_end_cal.h](#)

10.86 ftp_t Struct Reference

Public Attributes

- int **state**
- int **proto**
- int **error**
- char **addr** [1024]
- char **file** [1024]
- char **user** [256]
- char **passwd** [256]
- char **local** [1024]
- int **topts** [4]
- [gtime_t](#) **tnext**
- [pthread_t](#) **thread**

10.86.1 Detailed Description

Definition at line 1177 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.87 Galileo_Almanac Class Reference

This class is a storage for the Galileo SV ALMANAC data.

```
#include <galileo_almanac.h>
```

Public Member Functions

- [Galileo_Almanac](#) ()=default
- [template<class Archive >](#)
void **serialize** (Archive &ar, const unsigned int version)

Public Attributes

- uint32_t [i_satellite_PRN](#) {}
SV PRN NUMBER.
- int32_t [i_Toa](#) {}
- int32_t [i_WNa](#) {}
- int32_t [i_IODa](#) {}
- double [d_Delta_i](#) {}
Inclination at reference time relative to $i_0 = 56^\circ$ [semi-circles].
- double [d_M_0](#) {}
Mean Anomaly at Reference Time [semi-circles].
- double [d_e_eccentricity](#) {}
Eccentricity [dimensionless].
- double [d_Delta_sqrt_A](#) {}
Square Root of the Semi-Major Axis [sqrt(m)].
- double [d_OMEGA0](#) {}
Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-circles].
- double [d_OMEGA](#) {}
Argument of Perigee [semi-circles].
- double [d_OMEGA_DOT](#) {}
Rate of Right Ascension [semi-circles/s].
- double [d_A_f0](#) {}
Coefficient 0 of code phase offset model [s].
- double [d_A_f1](#) {}
Coefficient 1 of code phase offset model [s/s].
- int32_t [E5b_HS](#) {}
- int32_t [E1B_HS](#) {}
- int32_t [E5a_HS](#) {}

10.87.1 Detailed Description

This class is a storage for the Galileo SV ALMANAC data.

Definition at line 30 of file galileo_almanac.h.

10.87.2 Constructor & Destructor Documentation

10.87.2.1 Galileo_Almanac()

```
Galileo_Almanac::Galileo_Almanac ( ) [default]
```

Default constructor

10.87.3 Member Data Documentation

10.87.3.1 d_A_f0

```
double Galileo_Almanac::d_A_f0 {}
```

Coefficient 0 of code phase offset model [s].

Definition at line 49 of file galileo_almanac.h.

10.87.3.2 d_A_f1

```
double Galileo_Almanac::d_A_f1 {}
```

Coefficient 1 of code phase offset model [s/s].

Definition at line 50 of file galileo_almanac.h.

10.87.3.3 d_Delta_i

```
double Galileo_Almanac::d_Delta_i {}
```

Inclination at reference time relative to $i_0 = 56^\circ$ [semi-circles].

Definition at line 42 of file galileo_almanac.h.

10.87.3.4 d_Delta_sqrt_A

```
double Galileo_Almanac::d_Delta_sqrt_A {}
```

Square Root of the Semi-Major Axis [sqrt(m)].

Definition at line 45 of file galileo_almanac.h.

10.87.3.5 d_e_eccentricity

```
double Galileo_Almanac::d_e_eccentricity {}
```

Eccentricity [dimensionless].

Definition at line 44 of file galileo_almanac.h.

10.87.3.6 d_M_0

```
double Galileo_Almanac::d_M_0 {}
```

Mean Anomaly at Reference Time [semi-circles].

Definition at line 43 of file galileo_almanac.h.

10.87.3.7 d_OMEGA

```
double Galileo_Almanac::d_OMEGA {}
```

Argument of Perigee [semi-circles].

Definition at line 47 of file galileo_almanac.h.

10.87.3.8 d_OMEGA0

```
double Galileo_Almanac::d_OMEGA0 {}
```

Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-circles].

Definition at line 46 of file galileo_almanac.h.

10.87.3.9 d_OMEGA_DOT

```
double Galileo_Almanac::d_OMEGA_DOT {}
```

Rate of Right Ascension [semi-circles/s].

Definition at line 48 of file galileo_almanac.h.

10.87.3.10 i_satellite_PRN

```
uint32_t Galileo_Almanac::i_satellite_PRN {}
```

SV PRN NUMBER.

Definition at line 38 of file galileo_almanac.h.

The documentation for this class was generated from the following file:

- [galileo_almanac.h](#)

10.88 Galileo_Almanac_Helper Class Reference

This class is a storage for the GALILEO ALMANAC data as described in GALILEO ICD.

```
#include <galileo_almanac_helper.h>
```

Public Member Functions

- [Galileo_Almanac_Helper](#) ()=default
Default constructor.
- [Galileo_Almanac](#) **get_almanac** (int i)

Public Attributes

- int32_t **IOD_a_7** {}
- int32_t **WN_a_7** {}
- int32_t **t0a_7** {}
- int32_t **SVID1_7** {}
- double **DELTA_A_7** {}
- double **e_7** {}
- double **omega_7** {}
- double **delta_i_7** {}
- double **Omega0_7** {}
- double **Omega_dot_7** {}
- double **M0_7** {}
- int32_t **IOD_a_8** {}
- double **af0_8** {}
- double **af1_8** {}
- int32_t **E5b_HS_8** {}
- int32_t **E1B_HS_8** {}
- int32_t **E5a_HS_8** {}
- int32_t **SVID2_8** {}
- double **DELTA_A_8** {}
- double **e_8** {}
- double **omega_8** {}
- double **delta_i_8** {}
- double **Omega0_8** {}
- double **Omega_dot_8** {}
- int32_t **IOD_a_9** {}
- int32_t **WN_a_9** {}
- int32_t **t0a_9** {}
- double **M0_9** {}
- double **af0_9** {}
- double **af1_9** {}
- int32_t **E5b_HS_9** {}
- int32_t **E1B_HS_9** {}
- int32_t **E5a_HS_9** {}
- int32_t **SVID3_9** {}
- double **DELTA_A_9** {}
- double **e_9** {}
- double **omega_9** {}
- double **delta_i_9** {}

- `int32_t IOD_a_10 {}`
- `double Omega0_10 {}`
- `double Omega_dot_10 {}`
- `double M0_10 {}`
- `double af0_10 {}`
- `double af1_10 {}`
- `int32_t E5b_HS_10 {}`
- `int32_t E1B_HS_10 {}`
- `int32_t E5a_HS_10 {}`

10.88.1 Detailed Description

This class is a storage for the GALILEO ALMANAC data as described in GALILEO ICD.

See <https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-O+S-SIS-ICD.pdf> paragraph 5.1.10

Definition at line 31 of file `galileo_almanac_helper.h`.

10.88.2 Constructor & Destructor Documentation

10.88.2.1 Galileo_Almanac_Helper()

```
Galileo_Almanac_Helper::Galileo_Almanac_Helper ( ) [default]
```

Default constructor.

The documentation for this class was generated from the following file:

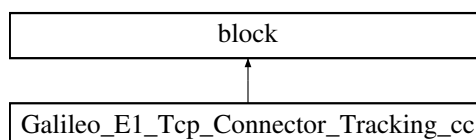
- [galileo_almanac_helper.h](#)

10.89 Galileo_E1_Tcp_Connector_Tracking_cc Class Reference

This class implements a code DLL + carrier PLL VEML (Very Early Minus Late) tracking block for Galileo E1 signals.

```
#include <galileo_e1_tcp_connector_tracking_cc.h>
```

Inheritance diagram for `Galileo_E1_Tcp_Connector_Tracking_cc`:



Public Member Functions

- void **set_channel** (uint32_t channel)
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro)
- void **start_tracking** ()
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)

Friends

- galileo_e1_tcp_connector_tracking_cc_sptr **galileo_e1_tcp_connector_make_tracking_cc** (int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float pll_bw_hz, float dll_bw_hz, float early_late_space_chips, float very_early_late_space_chips, size_t port_ch0)

10.89.1 Detailed Description

This class implements a code DLL + carrier PLL VEML (Very Early Minus Late) tracking block for Galileo E1 signals.

Definition at line 69 of file `galileo_e1_tcp_connector_tracking_cc.h`.

The documentation for this class was generated from the following file:

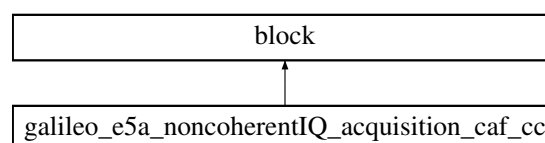
- [galileo_e1_tcp_connector_tracking_cc.h](#)

10.90 galileo_e5a_noncoherentIQ_acquisition_caf_cc Class Reference

This class implements a Parallel Code Phase Search Acquisition.

```
#include <galileo_e5a_noncoherent_iq_acquisition_caf_cc.h>
```

Inheritance diagram for `galileo_e5a_noncoherentIQ_acquisition_caf_cc`:



Public Member Functions

- [~galileo_e5a_noncoherentIQ_acquisition_caf_cc](#) ()
Default destructor.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro)
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.
- unsigned int [mag](#) () const
Returns the maximum peak of grid search.
- void [init](#) ()
Initializes acquisition algorithm.
- void [set_local_code](#) (std::complex< float > *code, std::complex< float > *codeQ)
Sets local code for PCPS acquisition algorithm.
- void [set_active](#) (bool active)
Starts acquisition algorithm, turning from standby mode to active mode.
- void [set_state](#) (int state)
If set to 1, ensures that acquisition starts at the first available sample.
- void [set_channel](#) (unsigned int channel)
Set acquisition channel unique ID.
- void [set_channel_fsm](#) (std::weak_ptr< [ChannelFsm](#) > channel_fsm)
Set channel fsm associated to this acquisition instance.
- void [set_threshold](#) (float threshold)
Set statistics threshold of PCPS algorithm.
- void [set_doppler_max](#) (unsigned int doppler_max)
Set maximum Doppler grid search.
- void [set_doppler_step](#) (unsigned int doppler_step)
Set Doppler steps for the grid search.
- int [general_work](#) (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
Parallel Code Phase Search Acquisition signal processing.

Friends

- galileo_e5a_noncoherentIQ_acquisition_caf_cc_sptr [galileo_e5a_noncoherentIQ_make_acquisition_caf_cc](#) (unsigned int sampled_ms, unsigned int max_dwells, unsigned int doppler_max, int64_t fs_in, int samples_per_ms, int samples_per_code, bool bit_transition_flag, bool dump, const std::string &dump_filename, bool both_signal_components_, int CAF_window_hz_, int Zero_padding_)

10.90.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition.

Check [An Open Source Galileo E1 Software Receiver](#), Algorithm 1, for a pseudocode description of this implementation.

Definition at line 71 of file galileo_e5a_noncoherent_iq_acquisition_caf_cc.h.

10.90.2 Constructor & Destructor Documentation

10.90.2.1 `~galileo_e5a_noncoherentIQ_acquisition_caf_cc()`

```
galileo_e5a_noncoherentIQ_acquisition_caf_cc::~~galileo_e5a_noncoherentIQ_acquisition_caf_cc (
)
```

Default destructor.

10.90.3 Member Function Documentation

10.90.3.1 `general_work()`

```
int galileo_e5a_noncoherentIQ_acquisition_caf_cc::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

Parallel Code Phase Search Acquisition signal processing.

10.90.3.2 `init()`

```
void galileo_e5a_noncoherentIQ_acquisition_caf_cc::init ( )
```

Initializes acquisition algorithm.

10.90.3.3 `mag()`

```
unsigned int galileo_e5a_noncoherentIQ_acquisition_caf_cc::mag ( ) const [inline]
```

Returns the maximum peak of grid search.

Definition at line 92 of file `galileo_e5a_noncoherent_iq_acquisition_caf_cc.h`.

10.90.3.4 `set_active()`

```
void galileo_e5a_noncoherentIQ_acquisition_caf_cc::set_active (
    bool active ) [inline]
```

Starts acquisition algorithm, turning from standby mode to active mode.

Parameters

<i>active</i>	- bool that activates/deactivates the block.
---------------	--

Definition at line 113 of file galileo_e5a_noncoherent_iq_acquisition_caf_cc.h.

10.90.3.5 set_channel()

```
void galileo_e5a_noncoherentIQ_acquisition_caf_cc::set_channel (
    unsigned int channel ) [inline]
```

Set acquisition channel unique ID.

Parameters

<i>channel</i>	- receiver channel.
----------------	---------------------

Definition at line 129 of file galileo_e5a_noncoherent_iq_acquisition_caf_cc.h.

10.90.3.6 set_channel_fsm()

```
void galileo_e5a_noncoherentIQ_acquisition_caf_cc::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline]
```

Set channel fsm associated to this acquisition instance.

Definition at line 137 of file galileo_e5a_noncoherent_iq_acquisition_caf_cc.h.

10.90.3.7 set_doppler_max()

```
void galileo_e5a_noncoherentIQ_acquisition_caf_cc::set_doppler_max (
    unsigned int doppler_max ) [inline]
```

Set maximum Doppler grid search.

Parameters

<i>doppler_max</i>	- Maximum Doppler shift considered in the grid search [Hz].
--------------------	---

Definition at line 156 of file galileo_e5a_noncoherent_iq_acquisition_caf_cc.h.

10.90.3.8 set_doppler_step()

```
void galileo_e5a_noncoherentIQ_acquisition_caf_cc::set_doppler_step (
    unsigned int doppler_step ) [inline]
```

Set Doppler steps for the grid search.

Parameters

<i>doppler_step</i>	- Frequency bin of the search grid [Hz].
---------------------	--

Definition at line 165 of file galileo_e5a_noncoherent_iq_acquisition_caf_cc.h.

10.90.3.9 set_gnss_synchro()

```
void galileo_e5a_noncoherentIQ_acquisition_caf_cc::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [inline]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.

Parameters

<i>p_gnss_synchro</i>	Satellite information shared by the processing blocks.
-----------------------	--

Definition at line 84 of file galileo_e5a_noncoherent_iq_acquisition_caf_cc.h.

10.90.3.10 set_local_code()

```
void galileo_e5a_noncoherentIQ_acquisition_caf_cc::set_local_code (
    std::complex< float > * code,
    std::complex< float > * codeQ )
```

Sets local code for PCPS acquisition algorithm.

Parameters

<i>code</i>	- Pointer to the PRN code.
-------------	----------------------------

10.90.3.11 set_state()

```
void galileo_e5a_noncoherentIQ_acquisition_caf_cc::set_state (
    int state )
```

If set to 1, ensures that acquisition starts at the first available sample.

Parameters

<i>state</i>	- int=1 forces start of acquisition
--------------	-------------------------------------

10.90.3.12 set_threshold()

```
void galileo_e5a_noncoherentIQ_acquisition_caf_cc::set_threshold (
    float threshold ) [inline]
```

Set statistics threshold of PCPS algorithm.

Parameters

<i>threshold</i>	- Threshold for signal detection (check Navitec2012 , Algorithm 1, for a definition of this threshold).
------------------	---

Definition at line 147 of file galileo_e5a_noncoherent_iq_acquisition_caf_cc.h.

The documentation for this class was generated from the following file:

- [galileo_e5a_noncoherent_iq_acquisition_caf_cc.h](#)

10.91 Galileo_Ephemeris Class Reference

This class is a storage and orbital model functions for the Galileo SV ephemeris data as described in Galileo ICD paragraph 5.1.1 (See <https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf>)

```
#include <galileo_ephemeris.h>
```

Public Member Functions

- void [satellitePosition](#) (double transmitTime)
Computes the ECEF SV coordinates and ECEF velocity.
- double [Galileo_System_Time](#) (double WN, double TOW)
Galileo System Time (GST), ICD paragraph 5.1.2.
- double [sv_clock_drift](#) (double transmitTime)
Satellite Time Correction Algorithm, ICD 5.1.4.
- double [sv_clock_relativistic_term](#) (double transmitTime)
Satellite Time Correction Algorithm, ICD 5.1.4.
- template<class Archive >
void [serialize](#) (Archive &archive, const uint32_t version)
Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

Public Attributes

- `int32_t IOD_ephemeris {}`
- `int32_t IOD_nav_1 {}`
- `int32_t SV_ID_PRN_4 {}`
- `double M0_1 {}`
Mean anomaly at reference time [semi-circles].
- `double delta_n_3 {}`
Mean motion difference from computed value [semi-circles/sec].
- `double e_1 {}`
Eccentricity.
- `double A_1 {}`
Square root of the semi-major axis [meters^{1/2}].
- `double OMEGA_0_2 {}`
Longitude of ascending node of orbital plane at weekly epoch [semi-circles].
- `double i_0_2 {}`
Inclination angle at reference time [semi-circles].
- `double omega_2 {}`
Argument of perigee [semi-circles].
- `double OMEGA_dot_3 {}`
Rate of right ascension [semi-circles/sec].
- `double iDot_2 {}`
Rate of inclination angle [semi-circles/sec].
- `double C_uc_3 {}`
Amplitude of the cosine harmonic correction term to the argument of latitude [radians].
- `double C_us_3 {}`
Amplitude of the sine harmonic correction term to the argument of latitude [radians].
- `double C_rc_3 {}`
Amplitude of the cosine harmonic correction term to the orbit radius [meters].
- `double C_rs_3 {}`
Amplitude of the sine harmonic correction term to the orbit radius [meters].
- `double C_ic_4 {}`
Amplitude of the cosine harmonic correction term to the angle of inclination [radians].
- `double C_is_4 {}`
Amplitude of the sine harmonic correction term to the angle of inclination [radians].
- `int32_t t0e_1 {}`
Ephemeris reference time [s].
- `int32_t t0c_4 {}`
Clock correction data reference Time of Week [sec].
- `double af0_4 {}`
SV clock bias correction coefficient [s].
- `double af1_4 {}`
SV clock drift correction coefficient [s/s].
- `double af2_4 {}`
SV clock drift rate correction coefficient [s/s²].
- `int32_t WN_5 {}`
Week number.
- `int32_t TOW_5 {}`
Time of Week.
- `double Galileo_satClkDrift {}`
- `double Galileo_dtr {}`

- relativistic clock correction term*
- int32_t **SISA_3** {}
- int32_t **E5a_HS** {}
 - E5a Signal Health Status.*
- int32_t **E5b_HS_5** {}
 - E5b Signal Health Status.*
- int32_t **E1B_HS_5** {}
 - E1B Signal Health Status.*
- bool **E5a_DVS** {}
 - E5a Data Validity Status.*
- bool **E5b_DVS_5** {}
 - E5b Data Validity Status.*
- bool **E1B_DVS_5** {}
 - E1B Data Validity Status.*
- double **BGD_E1E5a_5** {}
 - E1-E5a Broadcast Group Delay [s].*
- double **BGD_E1E5b_5** {}
 - E1-E5b Broadcast Group Delay [s].*
- double **d_satpos_X** {}
 - Earth-fixed coordinate x of the satellite [m]. Intersection of the IERS Reference Meridian (IRM) and the plane passing through the origin and normal to the Z-axis.*
- double **d_satpos_Y** {}
 - Earth-fixed coordinate y of the satellite [m]. Completes a right-handed, Earth-Centered, Earth-Fixed orthogonal coordinate system.*
- double **d_satpos_Z** {}
 - Earth-fixed coordinate z of the satellite [m]. The direction of the IERS (International Earth Rotation and Reference Systems Service) Reference Pole (IRP).*
- double **d_satvel_X** {}
 - Earth-fixed velocity coordinate x of the satellite [m].*
- double **d_satvel_Y** {}
 - Earth-fixed velocity coordinate y of the satellite [m].*
- double **d_satvel_Z** {}
 - Earth-fixed velocity coordinate z of the satellite [m].*
- uint32_t **i_satellite_PRN** {}
 - SV PRN NUMBER.*
- bool **flag_all_ephemeris** {}

10.91.1 Detailed Description

This class is a storage and orbital model functions for the Galileo SV ephemeris data as described in Galileo ICD paragraph 5.1.1 (See <https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf>)

Definition at line 34 of file galileo_ephemeris.h.

10.91.2 Member Function Documentation

10.91.2.1 Galileo_System_Time()

```
double Galileo_Ephemeris::Galileo_System_Time (
    double WN,
    double TOW )
```

Galileo System Time (GST), ICD paragraph 5.1.2.

10.91.2.2 satellitePosition()

```
void Galileo_Ephemeris::satellitePosition (
    double transmitTime )
```

Computes the ECEF SV coordinates and ECEF velocity.

10.91.2.3 serialize()

```
template<class Archive >
void Galileo_Ephemeris::serialize (
    Archive & archive,
    const uint32_t version ) [inline]
```

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

Definition at line 109 of file `galileo_ephemeris.h`.

References `A_1`, `af0_4`, `af1_4`, `af2_4`, `BGD_E1E5a_5`, `BGD_E1E5b_5`, `C_ic_4`, `C_is_4`, `C_rc_3`, `C_rs_3`, `C_uc_3`, `C_us_3`, `delta_n_3`, `E1B_DVS_5`, `E1B_HS_5`, `E5a_DVS`, `E5a_HS`, `E5b_DVS_5`, `E5b_HS_5`, `e_1`, `Galileo_dtr`, `i_0_2`, `i_satellite_PRN`, `iDot_2`, `M0_1`, `OMEGA_0_2`, `omega_2`, `OMEGA_dot_3`, `t0c_4`, `t0e_1`, `TOW_5`, and `WN_5`.

10.91.2.4 sv_clock_drift()

```
double Galileo_Ephemeris::sv_clock_drift (
    double transmitTime )
```

Satellite Time Correction Algorithm, ICD 5.1.4.

10.91.2.5 sv_clock_relativistic_term()

```
double Galileo_Ephemeris::sv_clock_relativistic_term (
    double transmitTime )
```

Satellite Time Correction Algorithm, ICD 5.1.4.

10.91.3 Member Data Documentation

10.91.3.1 A_1

```
double Galileo_Ephemeris::A_1 {}
```

Square root of the semi-major axis [meters^{1/2}].

Definition at line 52 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.2 af0_4

```
double Galileo_Ephemeris::af0_4 {}
```

SV clock bias correction coefficient [s].

Definition at line 68 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.3 af1_4

```
double Galileo_Ephemeris::af1_4 {}
```

SV clock drift correction coefficient [s/s].

Definition at line 69 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.4 af2_4

```
double Galileo_Ephemeris::af2_4 {}
```

SV clock drift rate correction coefficient [s/s²].

Definition at line 70 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.5 BGD_E1E5a_5

```
double Galileo_Ephemeris::BGD_E1E5a_5 {}
```

E1-E5a Broadcast Group Delay [s].

Definition at line 87 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.6 BGD_E1E5b_5

```
double Galileo_Ephemeris::BGD_E1E5b_5 {}
```

E1-E5b Broadcast Group Delay [s].

Definition at line 88 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.7 C_ic_4

```
double Galileo_Ephemeris::C_ic_4 {}
```

Amplitude of the cosine harmonic correction term to the angle of inclination [radians].

Definition at line 62 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.8 C_is_4

```
double Galileo_Ephemeris::C_is_4 {}
```

Amplitude of the sine harmonic correction term to the angle of inclination [radians].

Definition at line 63 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.9 C_rc_3

```
double Galileo_Ephemeris::C_rc_3 {}
```

Amplitude of the cosine harmonic correction term to the orbit radius [meters].

Definition at line 60 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.10 C_rs_3

```
double Galileo_Ephemeris::C_rs_3 {}
```

Amplitude of the sine harmonic correction term to the orbit radius [meters].

Definition at line 61 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.11 C_uc_3

```
double Galileo_Ephemeris::C_uc_3 {}
```

Amplitude of the cosine harmonic correction term to the argument of latitude [radians].

Definition at line 58 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.12 C_us_3

```
double Galileo_Ephemeris::C_us_3 {}
```

Amplitude of the sine harmonic correction term to the argument of latitude [radians].

Definition at line 59 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.13 d_satpos_X

```
double Galileo_Ephemeris::d_satpos_X {}
```

Earth-fixed coordinate x of the satellite [m]. Intersection of the IERS Reference Meridian (IRM) and the plane passing through the origin and normal to the Z-axis.

Definition at line 91 of file galileo_ephemeris.h.

10.91.3.14 d_satpos_Y

```
double Galileo_Ephemeris::d_satpos_Y {}
```

Earth-fixed coordinate y of the satellite [m]. Completes a right-handed, Earth-Centered, Earth-Fixed orthogonal coordinate system.

Definition at line 92 of file galileo_ephemeris.h.

10.91.3.15 d_satpos_Z

```
double Galileo_Ephemeris::d_satpos_Z {}
```

Earth-fixed coordinate z of the satellite [m]. The direction of the IERS (International Earth Rotation and Reference Systems Service) Reference Pole (IRP).

Definition at line 93 of file galileo_ephemeris.h.

10.91.3.16 d_satvel_X

```
double Galileo_Ephemeris::d_satvel_X {}
```

Earth-fixed velocity coordinate x of the satellite [m].

Definition at line 96 of file galileo_ephemeris.h.

10.91.3.17 d_satvel_Y

```
double Galileo_Ephemeris::d_satvel_Y {}
```

Earth-fixed velocity coordinate y of the satellite [m].

Definition at line 97 of file galileo_ephemeris.h.

10.91.3.18 d_satvel_Z

```
double Galileo_Ephemeris::d_satvel_Z {}
```

Earth-fixed velocity coordinate z of the satellite [m].

Definition at line 98 of file galileo_ephemeris.h.

10.91.3.19 delta_n_3

```
double Galileo_Ephemeris::delta_n_3 {}
```

Mean motion difference from computed value [semi-circles/sec].

Definition at line 50 of file galileo_ephemeris.h.

Referenced by serialize().

10.91.3.20 E1B_DVS_5

```
bool Galileo_Ephemeris::E1B_DVS_5 {}
```

E1B Data Validity Status.

Definition at line 86 of file galileo_ephemeris.h.

Referenced by serialize().

10.91.3.21 E1B_HS_5

```
int32_t Galileo_Ephemeris::E1B_HS_5 {}
```

E1B Signal Health Status.

Definition at line 83 of file galileo_ephemeris.h.

Referenced by serialize().

10.91.3.22 E5a_DVS

```
bool Galileo_Ephemeris::E5a_DVS {}
```

E5a Data Validity Status.

Definition at line 84 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.23 E5a_HS

```
int32_t Galileo_Ephemeris::E5a_HS {}
```

E5a Signal Health Status.

Definition at line 81 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.24 E5b_DVS_5

```
bool Galileo_Ephemeris::E5b_DVS_5 {}
```

E5b Data Validity Status.

Definition at line 85 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.25 E5b_HS_5

```
int32_t Galileo_Ephemeris::E5b_HS_5 {}
```

E5b Signal Health Status.

Definition at line 82 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.26 e_1

```
double Galileo_Ephemeris::e_1 {}
```

Eccentricity.

Definition at line 51 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.27 Galileo_dtr

```
double Galileo_Ephemeris::Galileo_dtr {}
```

relativistic clock correction term

Definition at line 77 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.28 i_0_2

```
double Galileo_Ephemeris::i_0_2 {}
```

Inclination angle at reference time [semi-circles].

Definition at line 54 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.29 i_satellite_PRN

```
uint32_t Galileo_Ephemeris::i_satellite_PRN {}
```

SV PRN NUMBER.

Definition at line 100 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.30 iDot_2

```
double Galileo_Ephemeris::iDot_2 {}
```

Rate of inclination angle [semi-circles/sec].

Definition at line 57 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.31 M0_1

```
double Galileo_Ephemeris::M0_1 {}
```

Mean anomaly at reference time [semi-circles].

Definition at line 49 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.32 OMEGA_0_2

```
double Galileo_Ephemeris::OMEGA_0_2 {}
```

Longitude of ascending node of orbital plane at weekly epoch [semi-circles].

Definition at line 53 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.33 omega_2

```
double Galileo_Ephemeris::omega_2 {}
```

Argument of perigee [semi-circles].

Definition at line 55 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.34 OMEGA_dot_3

```
double Galileo_Ephemeris::OMEGA_dot_3 {}
```

Rate of right ascension [semi-circles/sec].

Definition at line 56 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.35 t0c_4

```
int32_t Galileo_Ephemeris::t0c_4 {}
```

Clock correction data reference Time of Week [sec].

Definition at line 67 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.36 t0e_1

```
int32_t Galileo_Ephemeris::t0e_1 {}
```

Ephemeris reference time [s].

Definition at line 64 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.37 TOW_5

```
int32_t Galileo_Ephemeris::TOW_5 {}
```

Time of Week.

Definition at line 75 of file galileo_ephemeris.h.

Referenced by `serialize()`.

10.91.3.38 WN_5

```
int32_t Galileo_Ephemeris::WN_5 {}
```

Week number.

Definition at line 74 of file galileo_ephemeris.h.

Referenced by `serialize()`.

The documentation for this class was generated from the following file:

- [galileo_ephemeris.h](#)

10.92 Galileo_Fnav_Message Class Reference

This class handles the Galileo F/NAV Data message, as described in the Galileo Open Service Signal in Space Interface Control Document (OS SIS ICD), Issue 1.2 (Nov 2015). See <https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf>.

```
#include <galileo_fnav_message.h>
```

Public Member Functions

- void **split_page** (const std::string &page_string)
- bool **have_new_ephemeris** ()
- bool **have_new_iono_and_GST** ()
- bool **have_new_utc_model** ()
- bool **have_new_almanac** ()
- [Galileo_Ephemeris](#) **get_ephemeris** () const
- [Galileo_Iono](#) **get_iono** () const
- [Galileo_Utc_Model](#) **get_utc_model** () const
- [Galileo_Almanac_Helper](#) **get_almanac** () const
- int32_t **get_TOW1** () const
- int32_t **get_TOW2** () const
- int32_t **get_TOW3** () const
- int32_t **get_TOW4** () const
- bool **get_flag_CRC_test** () const
- bool **get_flag_TOW_set** () const
- void **set_flag_TOW_set** (bool flag_tow)
- bool **is_TOW_set** () const
- bool **is_TOW1_set** () const
- void **set_TOW1_flag** (bool flag_tow1)
- bool **is_TOW2_set** () const
- void **set_TOW2_flag** (bool flag_tow2)
- bool **is_TOW3_set** () const
- void **set_TOW3_flag** (bool flag_tow3)
- bool **is_TOW4_set** () const
- void **set_TOW4_flag** (bool flag_tow4)

10.92.1 Detailed Description

This class handles the Galileo F/NAV Data message, as described in the Galileo Open Service Signal in Space Interface Control Document (OS SIS ICD), Issue 1.2 (Nov 2015). See <https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf>.

Definition at line 47 of file `galileo_fnav_message.h`.

The documentation for this class was generated from the following file:

- [galileo_fnav_message.h](#)

10.93 Galileo_Inav_Message Class Reference

This class handles the Galileo I/NAV Data message, as described in the Galileo Open Service Signal in Space Interface Control Document (OS SIS ICD), Issue 1.2 (Nov 2015). See <https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf>.

```
#include <galileo_inav_message.h>
```

Public Member Functions

- void **split_page** (std::string page_string, int32_t flag_even_word)
- int32_t **page_jk_decoder** (const char *data_jk)
- bool **have_new_ephemeris** ()
- bool **have_new_iono_and_GST** ()
- bool **have_new_utc_model** ()
- bool **have_new_almanac** ()
- [Galileo_Ephemeris](#) **get_ephemeris** () const
- [Galileo_Iono](#) **get_iono** () const
- [Galileo_Utc_Model](#) **get_utc_model** () const
- [Galileo_Almanac_Helper](#) **get_almanac** () const
- bool **get_flag_CRC_test** () const
- bool **get_flag_TOW_set** () const
- void **set_flag_TOW_set** (bool flag_tow)
- int32_t **get_Galileo_week** () const
- bool **is_TOW_set** () const
- int32_t **get_TOW5** () const
- int32_t **get_TOW6** () const
- bool **is_TOW5_set** () const
- void **set_TOW5_flag** (bool flag_tow5)
- bool **is_TOW6_set** () const
- void **set_TOW6_flag** (bool flag_tow6)
- bool **get_flag_GGTO** () const
- double **get_A0G** () const
- double **get_A1G** () const
- double **get_t0G** () const
- double **get_WN0G** () const

10.93.1 Detailed Description

This class handles the Galileo I/NAV Data message, as described in the Galileo Open Service Signal in Space Interface Control Document (OS SIS ICD), Issue 1.2 (Nov 2015). See <https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf>.

Definition at line 42 of file `galileo_inav_message.h`.

The documentation for this class was generated from the following file:

- [galileo_inav_message.h](#)

10.94 Galileo_Iono Class Reference

This class is a storage for the GALILEO IONOSPHERIC data as described in Galileo ICD paragraph 5.1.6.

```
#include <galileo_iono.h>
```

Public Member Functions

- [Galileo_Iono](#) ()=default
- `template<class Archive >`
void [serialize](#) (Archive &archive, const unsigned int version)
Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the iono data on disk file.

Public Attributes

- double [ai0_5](#) {}
Effective Ionisation Level 1st order parameter [sfu].
- double [ai1_5](#) {}
Effective Ionisation Level 2st order parameter [sfu/degree].
- double [ai2_5](#) {}
Effective Ionisation Level 3st order parameter [sfu/degree].
- int32_t [TOW_5](#) {}
UTC data reference Time of Week [s].
- int32_t [WN_5](#) {}
UTC data reference Week number [week].
- bool [Region1_flag_5](#) {}
Ionospheric Disturbance Flag for region 1.
- bool [Region2_flag_5](#) {}
Ionospheric Disturbance Flag for region 2.
- bool [Region3_flag_5](#) {}
Ionospheric Disturbance Flag for region 3.
- bool [Region4_flag_5](#) {}
Ionospheric Disturbance Flag for region 4.
- bool [Region5_flag_5](#) {}
Ionospheric Disturbance Flag for region 5.

10.94.1 Detailed Description

This class is a storage for the GALILEO IONOSPHERIC data as described in Galileo ICD paragraph 5.1.6.

See <https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OSS-SIS-ICD.pdf>

Definition at line 32 of file galileo_iono.h.

10.94.2 Constructor & Destructor Documentation

10.94.2.1 Galileo_Iono()

```
Galileo_Iono::Galileo_Iono ( ) [default]
```

Default constructor

10.94.3 Member Function Documentation

10.94.3.1 serialize()

```
template<class Archive >
void Galileo_Iono::serialize (
    Archive & archive,
    const unsigned int version ) [inline]
```

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the iono data on disk file.

Definition at line 62 of file galileo_iono.h.

References `ai0_5`, `ai1_5`, `ai2_5`, `Region1_flag_5`, `Region2_flag_5`, `Region3_flag_5`, `Region4_flag_5`, `Region5_flag_5`, `TOW_5`, and `WN_5`.

10.94.4 Member Data Documentation

10.94.4.1 ai0_5

```
double Galileo_Iono::ai0_5 {}
```

Effective Ionisation Level 1st order parameter [sfu].

Definition at line 41 of file galileo_iono.h.

Referenced by `serialize()`.

10.94.4.2 ai1_5

```
double Galileo_Iono::ai1_5 {}
```

Effective Ionisation Level 2st order parameter [sfu/degree].

Definition at line 42 of file galileo_iono.h.

Referenced by `serialize()`.

10.94.4.3 ai2_5

```
double Galileo_Iono::ai2_5 {}
```

Effective Ionisation Level 3st order parameter [sfu/degree].

Definition at line 43 of file galileo_iono.h.

Referenced by `serialize()`.

10.94.4.4 Region1_flag_5

```
bool Galileo_Iono::Region1_flag_5 {}
```

Ionospheric Disturbance Flag for region 1.

Definition at line 50 of file galileo_iono.h.

Referenced by `serialize()`.

10.94.4.5 Region2_flag_5

```
bool Galileo_Iono::Region2_flag_5 {}
```

Ionospheric Disturbance Flag for region 2.

Definition at line 51 of file galileo_iono.h.

Referenced by `serialize()`.

10.94.4.6 Region3_flag_5

```
bool Galileo_Iono::Region3_flag_5 {}
```

Ionospheric Disturbance Flag for region 3.

Definition at line 52 of file galileo_iono.h.

Referenced by `serialize()`.

10.94.4.7 Region4_flag_5

```
bool Galileo_Iono::Region4_flag_5 {}
```

Ionospheric Disturbance Flag for region 4.

Definition at line 53 of file galileo_iono.h.

Referenced by `serialize()`.

10.94.4.8 Region5_flag_5

```
bool Galileo_Iono::Region5_flag_5 {}
```

Ionospheric Disturbance Flag for region 5.

Definition at line 54 of file galileo_iono.h.

Referenced by `serialize()`.

10.94.4.9 TOW_5

```
int32_t Galileo_Iono::TOW_5 {}
```

UTC data reference Time of Week [s].

Definition at line 46 of file galileo_iono.h.

Referenced by `serialize()`.

10.94.4.10 WN_5

```
int32_t Galileo_Iono::WN_5 {}
```

UTC data reference Week number [week].

Definition at line 47 of file galileo_iono.h.

Referenced by `serialize()`.

The documentation for this class was generated from the following file:

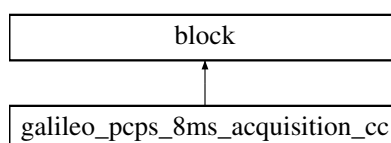
- [galileo_iono.h](#)

10.95 galileo_pcps_8ms_acquisition_cc Class Reference

This class implements a Parallel Code Phase Search Acquisition for Galileo E1 signals with coherent integration time = 8 ms (two codes)

```
#include <galileo_pcps_8ms_acquisition_cc.h>
```

Inheritance diagram for `galileo_pcps_8ms_acquisition_cc`:



Public Member Functions

- [~galileo_pcps_8ms_acquisition_cc](#) ()
Default destructor.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro)
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.
- [uint32_t mag](#) () const
Returns the maximum peak of grid search.
- void [init](#) ()
Initializes acquisition algorithm.
- void [set_local_code](#) (std::complex< float > *code)
Sets local code for PCPS acquisition algorithm.
- void [set_active](#) (bool active)
Starts acquisition algorithm, turning from standby mode to active mode.
- void [set_state](#) (int32_t state)
If set to 1, ensures that acquisition starts at the first available sample.
- void [set_channel](#) (uint32_t channel)
Set acquisition channel unique ID.
- void [set_channel_fsm](#) (std::weak_ptr< [ChannelFsm](#) > channel_fsm)
Set channel fsm associated to this acquisition instance.
- void [set_threshold](#) (float threshold)
Set statistics threshold of PCPS algorithm.
- void [set_doppler_max](#) (uint32_t doppler_max)
Set maximum Doppler grid search.
- void [set_doppler_step](#) (uint32_t doppler_step)
Set Doppler steps for the grid search.
- int [general_work](#) (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
Parallel Code Phase Search Acquisition signal processing.

Friends

- [galileo_pcps_8ms_acquisition_cc_sptr](#) [galileo_pcps_8ms_make_acquisition_cc](#) (uint32_t sampled_ms, uint32_t max_dwells, uint32_t doppler_max, int64_t fs_in, int32_t samples_per_ms, int32_t samples_per_code, bool dump, const std::string &dump_filename)

10.95.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition for Galileo E1 signals with coherent integration time = 8 ms (two codes)

Definition at line 61 of file [galileo_pcps_8ms_acquisition_cc.h](#).

10.95.2 Constructor & Destructor Documentation

10.95.2.1 `~galileo_pcps_8ms_acquisition_cc()`

```
galileo_pcps_8ms_acquisition_cc::~~galileo_pcps_8ms_acquisition_cc ( )
```

Default destructor.

10.95.3 Member Function Documentation

10.95.3.1 `general_work()`

```
int galileo_pcps_8ms_acquisition_cc::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

Parallel Code Phase Search Acquisition signal processing.

10.95.3.2 `init()`

```
void galileo_pcps_8ms_acquisition_cc::init ( )
```

Initializes acquisition algorithm.

10.95.3.3 `mag()`

```
uint32_t galileo_pcps_8ms_acquisition_cc::mag ( ) const [inline]
```

Returns the maximum peak of grid search.

Definition at line 82 of file `galileo_pcps_8ms_acquisition_cc.h`.

10.95.3.4 `set_active()`

```
void galileo_pcps_8ms_acquisition_cc::set_active (
    bool active ) [inline]
```

Starts acquisition algorithm, turning from standby mode to active mode.

Parameters

<i>active</i>	- bool that activates/deactivates the block.
---------------	--

Definition at line 103 of file galileo_pcps_8ms_acquisition_cc.h.

10.95.3.5 set_channel()

```
void galileo_pcps_8ms_acquisition_cc::set_channel (
    uint32_t channel ) [inline]
```

Set acquisition channel unique ID.

Parameters

<i>channel</i>	- receiver channel.
----------------	---------------------

Definition at line 119 of file galileo_pcps_8ms_acquisition_cc.h.

10.95.3.6 set_channel_fsm()

```
void galileo_pcps_8ms_acquisition_cc::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline]
```

Set channel fsm associated to this acquisition instance.

Definition at line 127 of file galileo_pcps_8ms_acquisition_cc.h.

10.95.3.7 set_doppler_max()

```
void galileo_pcps_8ms_acquisition_cc::set_doppler_max (
    uint32_t doppler_max ) [inline]
```

Set maximum Doppler grid search.

Parameters

<i>doppler_max</i>	- Maximum Doppler shift considered in the grid search [Hz].
--------------------	---

Definition at line 146 of file galileo_pcps_8ms_acquisition_cc.h.

10.95.3.8 set_doppler_step()

```
void galileo_pcps_8ms_acquisition_cc::set_doppler_step (
    uint32_t doppler_step ) [inline]
```

Set Doppler steps for the grid search.

Parameters

<i>doppler_step</i>	- Frequency bin of the search grid [Hz].
---------------------	--

Definition at line 155 of file galileo_pcps_8ms_acquisition_cc.h.

10.95.3.9 set_gnss_synchro()

```
void galileo_pcps_8ms_acquisition_cc::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [inline]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.

Parameters

<i>p_gnss_synchro</i>	Satellite information shared by the processing blocks.
-----------------------	--

Definition at line 74 of file galileo_pcps_8ms_acquisition_cc.h.

10.95.3.10 set_local_code()

```
void galileo_pcps_8ms_acquisition_cc::set_local_code (
    std::complex< float > * code )
```

Sets local code for PCPS acquisition algorithm.

Parameters

<i>code</i>	- Pointer to the PRN code.
-------------	----------------------------

10.95.3.11 set_state()

```
void galileo_pcps_8ms_acquisition_cc::set_state (
    int32_t state )
```

If set to 1, ensures that acquisition starts at the first available sample.

Parameters

<i>state</i>	- int=1 forces start of acquisition
--------------	-------------------------------------

10.95.3.12 `set_threshold()`

```
void galileo_pcps_8ms_acquisition_cc::set_threshold (
    float threshold ) [inline]
```

Set statistics threshold of PCPS algorithm.

Parameters

<i>threshold</i>	- Threshold for signal detection (check Navitec2012 , Algorithm 1, for a definition of this threshold).
------------------	---

Definition at line 137 of file `galileo_pcps_8ms_acquisition_cc.h`.

The documentation for this class was generated from the following file:

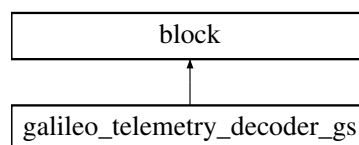
- [galileo_pcps_8ms_acquisition_cc.h](#)

10.96 `galileo_telemetry_decoder_gs` Class Reference

This class implements a block that decodes the INAV and FNAV data defined in Galileo ICD.

```
#include <galileo_telemetry_decoder_gs.h>
```

Inheritance diagram for `galileo_telemetry_decoder_gs`:



Public Member Functions

- void `set_satellite` (const [Gnss_Satellite](#) &satellite)
Set satellite PRN.
- void `set_channel` (int32_t channel)
Set receiver's channel.
- void `reset` ()
- int `general_work` (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
This is where all signal processing takes place.

Public Attributes

- `int32_t flag_even_word_arrived`

Friends

- `galileo_telemetry_decoder_gs_sptr galileo_make_telemetry_decoder_gs` (const [Gnss_Satellite](#) &satellite, int frame_type, bool dump)

10.96.1 Detailed Description

This class implements a block that decodes the INAV and FNAV data defined in Galileo ICD.

Definition at line 60 of file `galileo_telemetry_decoder_gs.h`.

10.96.2 Member Function Documentation

10.96.2.1 `general_work()`

```
int galileo_telemetry_decoder_gs::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

This is where all signal processing takes place.

10.96.2.2 `set_channel()`

```
void galileo_telemetry_decoder_gs::set_channel (
    int32_t channel )
```

Set receiver's channel.

10.96.2.3 `set_satellite()`

```
void galileo_telemetry_decoder_gs::set_satellite (
    const Gnss\_Satellite & satellite )
```

Set satellite PRN.

The documentation for this class was generated from the following file:

- [galileo_telemetry_decoder_gs.h](#)

10.97 Galileo_Utc_Model Class Reference

This class is a storage for the GALILEO UTC MODEL data as described in Galileo ICD <https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf> paragraph 5.1.7.

```
#include <galileo_utc_model.h>
```

Public Member Functions

- [Galileo_Utc_Model](#) ()=default
- double [GST_to_UTC_time](#) (double t_e, int32_t WN)
GST-UTC Conversion Algorithm and Parameters.
- template<class Archive >
void [serialize](#) (Archive &archive, const unsigned int version)
Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the UTC data on disk file.

Public Attributes

- double **A0_6** {}
- double **A1_6** {}
- int32_t **Delta_tLS_6** {}
- int32_t [t0t_6](#) {}
UTC data reference Time of Week [s].
- int32_t [WNNot_6](#) {}
UTC data reference Week number [week].
- int32_t **WN_LSF_6** {}
- int32_t **DN_6** {}
- int32_t **Delta_tLSF_6** {}
- double **A_0G_10** {}
- double **A_1G_10** {}
- int32_t **t_0G_10** {}
- int32_t **WN_0G_10** {}
- bool **flag_utc_model** {}

10.97.1 Detailed Description

This class is a storage for the GALILEO UTC MODEL data as described in Galileo ICD <https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf> paragraph 5.1.7.

Definition at line 33 of file galileo_utc_model.h.

10.97.2 Constructor & Destructor Documentation

10.97.2.1 Galileo_Utc_Model()

```
Galileo_Utc_Model::Galileo_Utc_Model ( ) [default]
```

Default constructor

10.97.3 Member Function Documentation

10.97.3.1 GST_to_UTC_time()

```
double Galileo_Utc_Model::GST_to_UTC_time (
    double t_e,
    int32_t WN )
```

GST-UTC Conversion Algorithm and Parameters.

10.97.3.2 serialize()

```
template<class Archive >
void Galileo_Utc_Model::serialize (
    Archive & archive,
    const unsigned int version ) [inline]
```

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the UTC data on disk file.

Definition at line 68 of file galileo_utc_model.h.

References `t0t_6`, and `WNot_6`.

10.97.4 Member Data Documentation

10.97.4.1 t0t_6

```
int32_t Galileo_Utc_Model::t0t_6 {}
```

UTC data reference Time of Week [s].

Definition at line 48 of file galileo_utc_model.h.

Referenced by `serialize()`.

10.97.4.2 WNot_6

```
int32_t Galileo_Utc_Model::WNot_6 {}
```

UTC data reference Week number [week].

Definition at line 49 of file galileo_utc_model.h.

Referenced by `serialize()`.

The documentation for this class was generated from the following file:

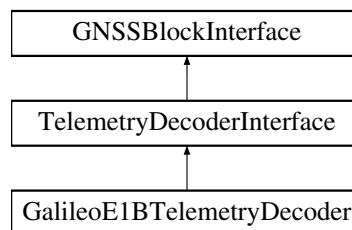
- [galileo_utc_model.h](#)

10.98 GalileoE1BTelemetryDecoder Class Reference

This class implements a NAV data decoder for Galileo INAV frames in E1B radio link.

```
#include <galileo_e1b_telemetry_decoder.h>
```

Inheritance diagram for GalileoE1BTelemetryDecoder:



Public Member Functions

- **GalileoE1BTelemetryDecoder** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_satellite** (const [Gnss_Satellite](#) &satellite) override
- std::string **role** () override
- std::string **implementation** () override
Returns "Galileo_E1B_Telemetry_Decoder".
- void **set_channel** (int channel) override
- void **reset** () override
- size_t **item_size** () override

10.98.1 Detailed Description

This class implements a NAV data decoder for Galileo INAV frames in E1B radio link.

Definition at line 41 of file galileo_e1b_telemetry_decoder.h.

10.98.2 Member Function Documentation

10.98.2.1 implementation()

```
std::string GalileoE1BTelemetryDecoder::implementation ( ) [inline], [override], [virtual]
```

Returns "Galileo_E1B_Telemetry_Decoder".

Implements [GNSSBlockInterface](#).

Definition at line 67 of file `galileo_e1b_telemetry_decoder.h`.

The documentation for this class was generated from the following file:

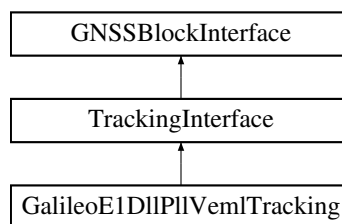
- [galileo_e1b_telemetry_decoder.h](#)

10.99 GalileoE1DIPIIVemlTracking Class Reference

This class Adapts a DLL+PLL VEML (Very Early Minus Late) tracking loop block to a [TrackingInterface](#) for Galileo E1 signals.

```
#include <galileo_e1_dll_pll_veml_tracking.h>
```

Inheritance diagram for GalileoE1DIPIIVemlTracking:



Public Member Functions

- **GalileoE1DIPIIVemlTracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "Galileo_E1_DLL_PLL_VEML_Tracking".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override
 - Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override
 - Stop running tracking.

10.99.1 Detailed Description

This class Adapts a DLL+PLL VEML (Very Early Minus Late) tracking loop block to a [TrackingInterface](#) for Galileo E1 signals.

Definition at line 40 of file `galileo_e1_dll_pll_veml_tracking.h`.

10.99.2 Member Function Documentation

10.99.2.1 `implementation()`

```
std::string GalileoE1DllPllVemlTracking::implementation ( ) [inline], [override], [virtual]
```

Returns "Galileo_E1_DLL_PLL_VEML_Tracking".

Implements [GNSSBlockInterface](#).

Definition at line 57 of file `galileo_e1_dll_pll_veml_tracking.h`.

10.99.2.2 `set_channel()`

```
void GalileoE1DllPllVemlTracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.99.2.3 `set_gnss_synchro()`

```
void GalileoE1DllPllVemlTracking::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.99.2.4 stop_tracking()

```
void GalileoE1DllPllVemlTracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

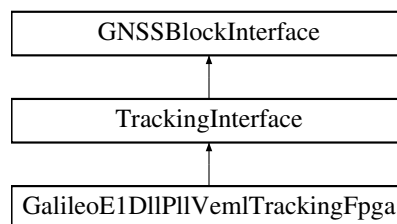
- [galileo_e1_dll_pll_veml_tracking.h](#)

10.100 GalileoE1DIIPIIVemlTrackingFpga Class Reference

This class Adapts a DLL+PLL VEML (Very Early Minus Late) tracking loop block to a [TrackingInterface](#) for Galileo E1 signals.

```
#include <galileo_e1_dll_pll_veml_tracking_fpga.h>
```

Inheritance diagram for GalileoE1DIIPIIVemlTrackingFpga:



Public Member Functions

- [GalileoE1DIIPIIVemlTrackingFpga](#) (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
Constructor.
- virtual [~GalileoE1DIIPIIVemlTrackingFpga](#) ()
Destructor.
- std::string [role](#) () override
Role.
- std::string [implementation](#) () override
Returns "Galileo_E1_DLL_PLL_VEML_Tracking_Fpga".
- size_t [item_size](#) () override
Returns size of lv_16sc_t.
- void [connect](#) (gr::top_block_sptr top_block) override
Connect.
- void [disconnect](#) (gr::top_block_sptr top_block) override
Disconnect.
- gr::basic_block_sptr [get_left_block](#) () override
Get left block.
- gr::basic_block_sptr [get_right_block](#) () override
Get right block.

- void [set_channel](#) (unsigned int channel) override
Set tracking channel unique ID.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void [start_tracking](#) () override
Start the tracking process in the FPGA.
- void [stop_tracking](#) () override
Stop the tracking process in the FPGA.

10.100.1 Detailed Description

This class Adapts a DLL+PLL VEML (Very Early Minus Late) tracking loop block to a [TrackingInterface](#) for Galileo E1 signals.

Definition at line 40 of file `galileo_e1_dll_pll_veml_tracking_fpga.h`.

10.100.2 Constructor & Destructor Documentation

10.100.2.1 GalileoE1DllPllVemlTrackingFpga()

```
GalileoE1DllPllVemlTrackingFpga::GalileoE1DllPllVemlTrackingFpga (
    const ConfigurationInterface * configuration,
    const std::string & role,
    unsigned int in_streams,
    unsigned int out_streams )
```

Constructor.

10.100.2.2 ~GalileoE1DllPllVemlTrackingFpga()

```
virtual GalileoE1DllPllVemlTrackingFpga::~GalileoE1DllPllVemlTrackingFpga ( ) [virtual]
```

Destructor.

10.100.3 Member Function Documentation

10.100.3.1 connect()

```
void GalileoE1DllPllVemlTrackingFpga::connect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Connect.

Implements [GNSSBlockInterface](#).

10.100.3.2 disconnect()

```
void GalileoE1DllPllVemlTrackingFpga::disconnect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Disconnect.

Implements [GNSSBlockInterface](#).

10.100.3.3 get_left_block()

```
gr::basic_block_sptr GalileoE1DllPllVemlTrackingFpga::get_left_block ( ) [override], [virtual]
```

Get left block.

Implements [GNSSBlockInterface](#).

10.100.3.4 get_right_block()

```
gr::basic_block_sptr GalileoE1DllPllVemlTrackingFpga::get_right_block ( ) [override], [virtual]
```

Get right block.

Implements [GNSSBlockInterface](#).

10.100.3.5 implementation()

```
std::string GalileoE1DllPllVemlTrackingFpga::implementation ( ) [inline], [override], [virtual]
```

Returns "Galileo_E1_DLL_PLL_VEML_Tracking_Fpga".

Implements [GNSSBlockInterface](#).

Definition at line 68 of file `galileo_e1_dll_pll_veml_tracking_fpga.h`.

10.100.3.6 item_size()

```
size_t GalileoE1D1lP1lVemlTrackingFpga::item_size ( ) [inline], [override], [virtual]
```

Returns size of lv_16sc_t.

Implements [GNSSBlockInterface](#).

Definition at line 76 of file galileo_e1_dll_pll_veml_tracking_fpga.h.

10.100.3.7 role()

```
std::string GalileoE1D1lP1lVemlTrackingFpga::role ( ) [inline], [override], [virtual]
```

Role.

Implements [GNSSBlockInterface](#).

Definition at line 60 of file galileo_e1_dll_pll_veml_tracking_fpga.h.

10.100.3.8 set_channel()

```
void GalileoE1D1lP1lVemlTrackingFpga::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.100.3.9 set_gnss_synchro()

```
void GalileoE1D1lP1lVemlTrackingFpga::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.100.3.10 start_tracking()

```
void GalileoE1DllPllVemlTrackingFpga::start_tracking ( ) [override], [virtual]
```

Start the tracking process in the FPGA.

Implements [TrackingInterface](#).

10.100.3.11 stop_tracking()

```
void GalileoE1DllPllVemlTrackingFpga::stop_tracking ( ) [override], [virtual]
```

Stop the tracking process in the FPGA.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

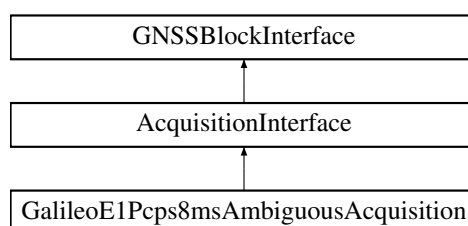
- [galileo_e1_dll_pll_veml_tracking_fpga.h](#)

10.101 GalileoE1Pcps8msAmbiguousAcquisition Class Reference

Adapts a PCPS 8ms acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

```
#include <galileo_e1_pcps_8ms_ambiguous_acquisition.h>
```

Inheritance diagram for GalileoE1Pcps8msAmbiguousAcquisition:



Public Member Functions

- **GalileoE1Pcps8msAmbiguousAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "Galileo_E1_PCPS_8ms_Ambiguous_Acquisition".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.*
- void **set_channel** (unsigned int channel) override
 - Set acquisition channel unique ID.*
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
 - Set channel fsm associated to this acquisition instance.*
- void **set_threshold** (float threshold) override
 - Set statistics threshold of PCPS algorithm.*
- void **set_doppler_max** (unsigned int doppler_max) override
 - Set maximum Doppler off grid search.*
- void **set_doppler_step** (unsigned int doppler_step) override
 - Set Doppler steps for the grid search.*
- void **init** () override
 - Initializes acquisition algorithm.*
- void **set_local_code** () override
 - Sets local code for Galileo E1 PCPS acquisition algorithm.*
- signed int **mag** () override
 - Returns the maximum peak of grid search.*
- void **reset** () override
 - Restart acquisition algorithm.*
- void **stop_acquisition** () override
 - Stop running acquisition.*
- void **set_state** (int state __attribute__((unused))) override
- void **set_resampler_latency** (uint32_t latency_samples __attribute__((unused))) override

10.101.1 Detailed Description

Adapts a PCPS 8ms acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

Definition at line 38 of file `galileo_e1_pcps_8ms_ambiguous_acquisition.h`.

10.101.2 Member Function Documentation

10.101.2.1 implementation()

```
std::string GalileoE1Pcps8msAmbiguousAcquisition::implementation ( ) [inline], [override],  
[virtual]
```

Returns "Galileo_E1_PCPS_8ms_Ambiguous_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 56 of file `galileo_e1_pcps_8ms_ambiguous_acquisition.h`.

10.101.2.2 init()

```
void GalileoE1Pcps8msAmbiguousAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.101.2.3 mag()

```
signed int GalileoE1Pcps8msAmbiguousAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.101.2.4 reset()

```
void GalileoE1Pcps8msAmbiguousAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.101.2.5 set_channel()

```
void GalileoE1Pcps8msAmbiguousAcquisition::set_channel (  
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 81 of file `galileo_e1_pcps_8ms_ambiguous_acquisition.h`.

10.101.2.6 set_channel_fsm()

```
void GalileoE1Pcps8msAmbiguousAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 90 of file galileo_e1_pcps_8ms_ambiguous_acquisition.h.

10.101.2.7 set_doppler_max()

```
void GalileoE1Pcps8msAmbiguousAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.101.2.8 set_doppler_step()

```
void GalileoE1Pcps8msAmbiguousAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.101.2.9 set_gnss_synchro()

```
void GalileoE1Pcps8msAmbiguousAcquisition::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.101.2.10 set_local_code()

```
void GalileoE1Pcps8msAmbiguousAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local code for Galileo E1 PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.101.2.11 set_threshold()

```
void GalileoE1Pcps8msAmbiguousAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.101.2.12 stop_acquisition()

```
void GalileoE1Pcps8msAmbiguousAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

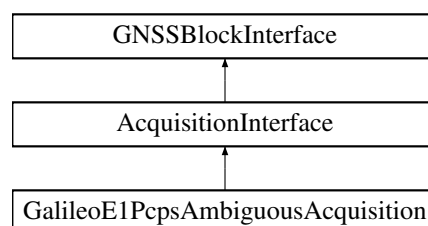
- [galileo_e1_pcps_8ms_ambiguous_acquisition.h](#)

10.102 GalileoE1PcpsAmbiguousAcquisition Class Reference

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

```
#include <galileo_e1_pcps_ambiguous_acquisition.h>
```

Inheritance diagram for GalileoE1PcpsAmbiguousAcquisition:



Public Member Functions

- **GalileoE1PcpsAmbiguousAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "Galileo_E1_PCPS_Ambiguous_Acquisition".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.*
- void **set_channel** (unsigned int channel) override
 - Set acquisition channel unique ID.*
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
 - Set channel fsm associated to this acquisition instance.*
- void **set_threshold** (float threshold) override
 - Set statistics threshold of PCPS algorithm.*
- void **set_doppler_max** (unsigned int doppler_max) override
 - Set maximum Doppler off grid search.*
- void **set_doppler_step** (unsigned int doppler_step) override
 - Set Doppler steps for the grid search.*
- void **set_doppler_center** (int doppler_center) override
 - Set Doppler center for the grid search.*
- void **init** () override
 - Initializes acquisition algorithm.*
- void **set_local_code** () override
 - Sets local code for Galileo E1 PCPS acquisition algorithm.*
- signed int **mag** () override
 - Returns the maximum peak of grid search.*
- void **reset** () override
 - Restart acquisition algorithm.*
- void **set_state** (int state) override
 - If state = 1, it forces the block to start acquiring from the first sample.*
- void **stop_acquisition** () override
 - Stop running acquisition.*
- void **set_resampler_latency** (uint32_t latency_samples) override
 - Sets the resampler latency to account it in the acquisition code delay estimation.*

10.102.1 Detailed Description

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

Definition at line 41 of file `galileo_e1_pcps_ambiguous_acquisition.h`.

10.102.2 Member Function Documentation

10.102.2.1 implementation()

```
std::string GalileoE1PcpsAmbiguousAcquisition::implementation ( ) [inline], [override], [virtual]
```

Returns "Galileo_E1_PCPS_Ambiguous_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 60 of file `galileo_e1_pcps_ambiguous_acquisition.h`.

10.102.2.2 init()

```
void GalileoE1PcpsAmbiguousAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.102.2.3 mag()

```
signed int GalileoE1PcpsAmbiguousAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.102.2.4 reset()

```
void GalileoE1PcpsAmbiguousAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.102.2.5 set_channel()

```
void GalileoE1PcpsAmbiguousAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 85 of file `galileo_e1_pcps_ambiguous_acquisition.h`.

10.102.2.6 set_channel_fsm()

```
void GalileoElPcpsAmbiguousAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 94 of file `galileo_e1_pcps_ambiguous_acquisition.h`.

10.102.2.7 set_doppler_center()

```
void GalileoElPcpsAmbiguousAcquisition::set_doppler_center (
    int doppler_center ) [override]
```

Set Doppler center for the grid search.

10.102.2.8 set_doppler_max()

```
void GalileoElPcpsAmbiguousAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.102.2.9 set_doppler_step()

```
void GalileoElPcpsAmbiguousAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.102.2.10 set_gnss_synchro()

```
void GalileoElPcpsAmbiguousAcquisition::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.102.2.11 set_local_code()

```
void GalileoE1PcpsAmbiguousAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local code for Galileo E1 PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.102.2.12 set_resampler_latency()

```
void GalileoE1PcpsAmbiguousAcquisition::set_resampler_latency (
    uint32_t latency_samples ) [override], [virtual]
```

Sets the resampler latency to account it in the acquisition code delay estimation.

Implements [AcquisitionInterface](#).

10.102.2.13 set_state()

```
void GalileoE1PcpsAmbiguousAcquisition::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.102.2.14 set_threshold()

```
void GalileoE1PcpsAmbiguousAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.102.2.15 stop_acquisition()

```
void GalileoE1PcpsAmbiguousAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

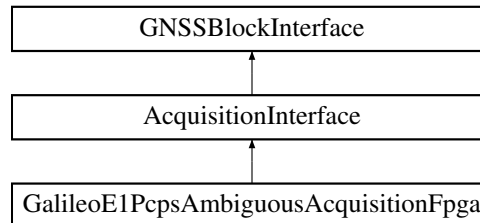
- [galileo_e1_pcps_ambiguous_acquisition.h](#)

10.103 GalileoE1PcpsAmbiguousAcquisitionFpga Class Reference

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for Galileo E1 Signals.

```
#include <galileo_e1_pcps_ambiguous_acquisition_fpga.h>
```

Inheritance diagram for GalileoE1PcpsAmbiguousAcquisitionFpga:



Public Member Functions

- [GalileoE1PcpsAmbiguousAcquisitionFpga](#) (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
Constructor.
- [~GalileoE1PcpsAmbiguousAcquisitionFpga](#) ()=default
Destructor.
- std::string [role](#) () override
Role.
- std::string [implementation](#) () override
Returns "Galileo_E1_PCPS_Ambiguous_Acquisition_Fpga".
- size_t [item_size](#) () override
Returns size of lv_16sc_t.
- void [connect](#) (gr::top_block_sptr top_block) override
Connect.
- void [disconnect](#) (gr::top_block_sptr top_block) override
Disconnect.
- gr::basic_block_sptr [get_left_block](#) () override
Get left block.
- gr::basic_block_sptr [get_right_block](#) () override
Get right block.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common Gnss_Synchro object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void [set_channel](#) (unsigned int channel) override
Set acquisition channel unique ID.
- void [set_channel_fsm](#) (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
Set channel fsm associated to this acquisition instance.
- void [set_threshold](#) (float threshold) override
Set statistics threshold of PCPS algorithm.
- void [set_doppler_max](#) (unsigned int doppler_max) override
Set maximum Doppler off grid search.
- void [set_doppler_step](#) (unsigned int doppler_step) override
Set Doppler steps for the grid search.

- void [set_doppler_center](#) (int doppler_center) override
Set Doppler center for the grid search.
- void [init](#) () override
Initializes acquisition algorithm.
- void [set_local_code](#) () override
Sets local code for Galileo E1 PCPS acquisition algorithm.
- signed int [mag](#) () override
Returns the maximum peak of grid search.
- void [reset](#) () override
Restart acquisition algorithm.
- void [set_state](#) (int state) override
If state = 1, it forces the block to start acquiring from the first sample.
- void [stop_acquisition](#) () override
Stop running acquisition.
- void [set_resampler_latency](#) (uint32_t latency_samples __attribute__((unused))) override
Set resampler latency.

10.103.1 Detailed Description

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for Galileo E1 Signals.

Definition at line 39 of file `galileo_e1_pcps_ambiguous_acquisition_fpga.h`.

10.103.2 Constructor & Destructor Documentation

10.103.2.1 GalileoE1PcpsAmbiguousAcquisitionFpga()

```
GalileoE1PcpsAmbiguousAcquisitionFpga::GalileoE1PcpsAmbiguousAcquisitionFpga (
    const ConfigurationInterface * configuration,
    const std::string & role,
    unsigned int in_streams,
    unsigned int out_streams )
```

Constructor.

10.103.2.2 ~GalileoE1PcpsAmbiguousAcquisitionFpga()

```
GalileoE1PcpsAmbiguousAcquisitionFpga::~GalileoE1PcpsAmbiguousAcquisitionFpga ( ) [default]
```

Destructor.

10.103.3 Member Function Documentation

10.103.3.1 connect()

```
void GalileoE1PcpsAmbiguousAcquisitionFpga::connect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Connect.

Implements [GNSSBlockInterface](#).

10.103.3.2 disconnect()

```
void GalileoE1PcpsAmbiguousAcquisitionFpga::disconnect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Disconnect.

Implements [GNSSBlockInterface](#).

10.103.3.3 get_left_block()

```
gr::basic_block_sptr GalileoE1PcpsAmbiguousAcquisitionFpga::get_left_block ( ) [override],
[virtual]
```

Get left block.

Implements [GNSSBlockInterface](#).

10.103.3.4 get_right_block()

```
gr::basic_block_sptr GalileoE1PcpsAmbiguousAcquisitionFpga::get_right_block ( ) [override],
[virtual]
```

Get right block.

Implements [GNSSBlockInterface](#).

10.103.3.5 implementation()

```
std::string GalileoE1PcpsAmbiguousAcquisitionFpga::implementation ( ) [inline], [override],
[virtual]
```

Returns "Galileo_E1_PCPS_Ambiguous_Acquisition_Fpga".

Implements [GNSSBlockInterface](#).

Definition at line 67 of file `galileo_e1_pcps_ambiguous_acquisition_fpga.h`.

10.103.3.6 init()

```
void GalileoE1PcpsAmbiguousAcquisitionFpga::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.103.3.7 item_size()

```
size_t GalileoE1PcpsAmbiguousAcquisitionFpga::item_size ( ) [inline], [override], [virtual]
```

Returns size of lv_16sc_t.

Implements [GNSSBlockInterface](#).

Definition at line 75 of file galileo_e1_pcps_ambiguous_acquisition_fpga.h.

10.103.3.8 mag()

```
signed int GalileoE1PcpsAmbiguousAcquisitionFpga::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.103.3.9 reset()

```
void GalileoE1PcpsAmbiguousAcquisitionFpga::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.103.3.10 role()

```
std::string GalileoE1PcpsAmbiguousAcquisitionFpga::role ( ) [inline], [override], [virtual]
```

Role.

Implements [GNSSBlockInterface](#).

Definition at line 59 of file galileo_e1_pcps_ambiguous_acquisition_fpga.h.

10.103.3.11 `set_channel()`

```
void GalileoElPcpsAmbiguousAcquisitionFpga::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 110 of file `galileo_e1_pcps_ambiguous_acquisition_fpga.h`.

10.103.3.12 `set_channel_fsm()`

```
void GalileoElPcpsAmbiguousAcquisitionFpga::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 119 of file `galileo_e1_pcps_ambiguous_acquisition_fpga.h`.

10.103.3.13 `set_doppler_center()`

```
void GalileoElPcpsAmbiguousAcquisitionFpga::set_doppler_center (
    int doppler_center ) [override]
```

Set Doppler center for the grid search.

10.103.3.14 `set_doppler_max()`

```
void GalileoElPcpsAmbiguousAcquisitionFpga::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.103.3.15 `set_doppler_step()`

```
void GalileoElPcpsAmbiguousAcquisitionFpga::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.103.3.16 set_gnss_synchro()

```
void GalileoE1PcpsAmbiguousAcquisitionFpga::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.103.3.17 set_local_code()

```
void GalileoE1PcpsAmbiguousAcquisitionFpga::set_local_code ( ) [override], [virtual]
```

Sets local code for Galileo E1 PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.103.3.18 set_resampler_latency()

```
void GalileoE1PcpsAmbiguousAcquisitionFpga::set_resampler_latency (
    uint32_t latency_samples __attribute__((unused)) ) [inline], [override]
```

Set resampler latency.

Definition at line 178 of file `galileo_e1_pcps_ambiguous_acquisition_fpga.h`.

10.103.3.19 set_state()

```
void GalileoE1PcpsAmbiguousAcquisitionFpga::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.103.3.20 set_threshold()

```
void GalileoE1PcpsAmbiguousAcquisitionFpga::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.103.3.21 stop_acquisition()

```
void GalileoE1PcpsAmbiguousAcquisitionFpga::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

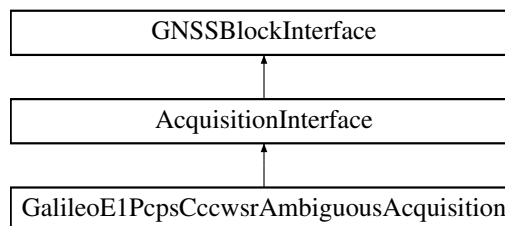
- [galileo_e1_pcps_ambiguous_acquisition_fpga.h](#)

10.104 GalileoE1PcpsCccwsrAmbiguousAcquisition Class Reference

Adapts a PCPS CCCWSR acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

```
#include <galileo_e1_pcps_cccwsr_ambiguous_acquisition.h>
```

Inheritance diagram for GalileoE1PcpsCccwsrAmbiguousAcquisition:



Public Member Functions

- **GalileoE1PcpsCccwsrAmbiguousAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override

Returns "Galileo_E1_PCPS_CCCWSR_Ambiguous_Acquisition".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **set_channel** (unsigned int channel) override

Set acquisition channel unique ID.
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override

Set channel fsm associated to this acquisition instance.
- void **set_threshold** (float threshold) override

Set statistics threshold of CCCWSR algorithm.
- void **set_doppler_max** (unsigned int doppler_max) override

- Set maximum Doppler off grid search.*
- void [set_doppler_step](#) (unsigned int doppler_step) override
- Set Doppler steps for the grid search.*
- void [init](#) () override
- Initializes acquisition algorithm.*
- void [set_local_code](#) () override
- signed int [mag](#) () override
- Returns the maximum peak of grid search.*
- void [reset](#) () override
- Restart acquisition algorithm.*
- void [set_state](#) (int state) override
- If state = 1, it forces the block to start acquiring from the first sample.*
- void [stop_acquisition](#) () override
- Stop running acquisition.*
- void [set_resampler_latency](#) (uint32_t latency_samples __attribute__((unused))) override

10.104.1 Detailed Description

Adapts a PCPS CCCWSR acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

Definition at line 38 of file `galileo_e1_pcps_cccwsr_ambiguous_acquisition.h`.

10.104.2 Member Function Documentation

10.104.2.1 implementation()

```
std::string GalileoE1PcpsCccwsrAmbiguousAcquisition::implementation ( ) [inline], [override], [virtual]
```

Returns "Galileo_E1_PCPS_CCCWSR_Ambiguous_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 57 of file `galileo_e1_pcps_cccwsr_ambiguous_acquisition.h`.

10.104.2.2 init()

```
void GalileoE1PcpsCccwsrAmbiguousAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.104.2.3 mag()

```
signed int GalileoElPcpsCccwsrAmbiguousAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.104.2.4 reset()

```
void GalileoElPcpsCccwsrAmbiguousAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.104.2.5 set_channel()

```
void GalileoElPcpsCccwsrAmbiguousAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 82 of file `galileo_e1_pcps_cccwsr_ambiguous_acquisition.h`.

10.104.2.6 set_channel_fsm()

```
void GalileoElPcpsCccwsrAmbiguousAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 91 of file `galileo_e1_pcps_cccwsr_ambiguous_acquisition.h`.

10.104.2.7 set_doppler_max()

```
void GalileoE1PcpsCccwsrAmbiguousAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.104.2.8 set_doppler_step()

```
void GalileoE1PcpsCccwsrAmbiguousAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.104.2.9 set_gnss_synchro()

```
void GalileoE1PcpsCccwsrAmbiguousAcquisition::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.104.2.10 set_state()

```
void GalileoE1PcpsCccwsrAmbiguousAcquisition::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.104.2.11 set_threshold()

```
void GalileoE1PcpsCccwsrAmbiguousAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of CCCWSR algorithm.

Implements [AcquisitionInterface](#).

10.104.2.12 stop_acquisition()

```
void GalileoE1PcpsCccwsrAmbiguousAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

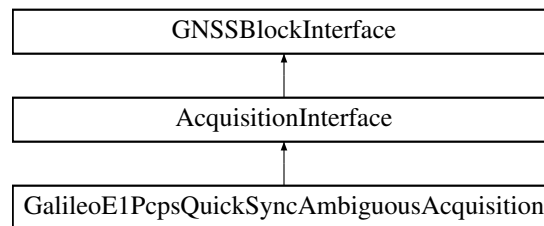
- [galileo_e1_pcps_cccwsr_ambiguous_acquisition.h](#)

10.105 GalileoE1PcpsQuickSyncAmbiguousAcquisition Class Reference

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

```
#include <galileo_e1_pcps_quicksync_ambiguous_acquisition.h>
```

Inheritance diagram for GalileoE1PcpsQuickSyncAmbiguousAcquisition:



Public Member Functions

- **GalileoE1PcpsQuickSyncAmbiguousAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override

Returns "Galileo_E1_PCPS_Ambiguous_Acquisition".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **set_channel** (unsigned int channel) override

Set acquisition channel unique ID.
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override

Set channel fsm associated to this acquisition instance.
- void **set_threshold** (float threshold) override

Set statistics threshold of PCPS algorithm.
- void **set_doppler_max** (unsigned int doppler_max) override

- *Set maximum Doppler off grid search.*
void [set_doppler_step](#) (unsigned int doppler_step) override
- *Set Doppler steps for the grid search.*
void [init](#) () override
- *Initializes acquisition algorithm.*
void [set_local_code](#) () override
- *Sets local code for Galileo E1 PCPS acquisition algorithm.*
signed int [mag](#) () override
- *Returns the maximum peak of grid search.*
void [reset](#) () override
- *Restart acquisition algorithm.*
void [set_state](#) (int state) override
- *If state = 1, it forces the block to start acquiring from the first sample.*
void [stop_acquisition](#) () override
- *Stop running acquisition.*
void [set_resampler_latency](#) (uint32_t latency_samples __attribute__((unused))) override

10.105.1 Detailed Description

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

Definition at line 39 of file `galileo_e1_pcps_quicksync_ambiguous_acquisition.h`.

10.105.2 Member Function Documentation

10.105.2.1 implementation()

```
std::string GalileoE1PcpsQuickSyncAmbiguousAcquisition::implementation ( ) [inline], [override],
[virtual]
```

Returns "Galileo_E1_PCPS_Ambiguous_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 58 of file `galileo_e1_pcps_quicksync_ambiguous_acquisition.h`.

10.105.2.2 init()

```
void GalileoE1PcpsQuickSyncAmbiguousAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.105.2.3 mag()

```
signed int GalileoE1PcpsQuickSyncAmbiguousAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.105.2.4 reset()

```
void GalileoE1PcpsQuickSyncAmbiguousAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.105.2.5 set_channel()

```
void GalileoE1PcpsQuickSyncAmbiguousAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 83 of file galileo_e1_pcps_quicksync_ambiguous_acquisition.h.

10.105.2.6 set_channel_fsm()

```
void GalileoE1PcpsQuickSyncAmbiguousAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 92 of file galileo_e1_pcps_quicksync_ambiguous_acquisition.h.

10.105.2.7 set_doppler_max()

```
void GalileoE1PcpsQuickSyncAmbiguousAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.105.2.8 set_doppler_step()

```
void GalileoE1PcpsQuickSyncAmbiguousAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.105.2.9 set_gnss_synchro()

```
void GalileoE1PcpsQuickSyncAmbiguousAcquisition::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.105.2.10 set_local_code()

```
void GalileoE1PcpsQuickSyncAmbiguousAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local code for Galileo E1 PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.105.2.11 set_state()

```
void GalileoE1PcpsQuickSyncAmbiguousAcquisition::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.105.2.12 `set_threshold()`

```
void GalileoE1PcpsQuickSyncAmbiguousAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.105.2.13 `stop_acquisition()`

```
void GalileoE1PcpsQuickSyncAmbiguousAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

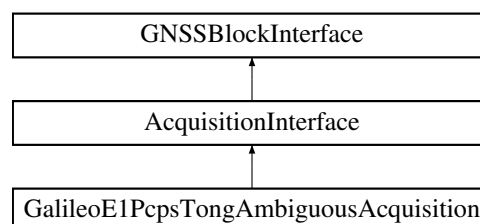
- [galileo_e1_pcps_quicksync_ambiguous_acquisition.h](#)

10.106 GalileoE1PcpsTongAmbiguousAcquisition Class Reference

Adapts a PCPS Tong acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

```
#include <galileo_e1_pcps_tong_ambiguous_acquisition.h>
```

Inheritance diagram for GalileoE1PcpsTongAmbiguousAcquisition:



Public Member Functions

- **GalileoE1PcpsTongAmbiguousAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "Galileo_E1_PCPS_Tong_Ambiguous_Acquisition".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.*
- void **set_channel** (unsigned int channel) override
 - Set acquisition channel unique ID.*
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
 - Set channel fsm associated to this acquisition instance.*
- void **set_threshold** (float threshold) override
 - Set statistics threshold of TONG algorithm.*
- void **set_doppler_max** (unsigned int doppler_max) override
 - Set maximum Doppler off grid search.*
- void **set_doppler_step** (unsigned int doppler_step) override
 - Set Doppler steps for the grid search.*
- void **init** () override
 - Initializes acquisition algorithm.*
- void **set_local_code** () override
 - Sets local code for Galileo E1 TONG acquisition algorithm.*
- signed int **mag** () override
 - Returns the maximum peak of grid search.*
- void **reset** () override
 - Restart acquisition algorithm.*
- void **set_state** (int state) override
 - If state = 1, it forces the block to start acquiring from the first sample.*
- void **stop_acquisition** () override
 - Stop running acquisition.*
- void **set_resampler_latency** (uint32_t latency_samples __attribute__((unused))) override

10.106.1 Detailed Description

Adapts a PCPS Tong acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

Definition at line 38 of file `galileo_e1_pcps_tong_ambiguous_acquisition.h`.

10.106.2 Member Function Documentation

10.106.2.1 implementation()

```
std::string GalileoE1PcpsTongAmbiguousAcquisition::implementation ( ) [inline], [override],  
[virtual]
```

Returns "Galileo_E1_PCPS_Tong_Ambiguous_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 57 of file galileo_e1_pcps_tong_ambiguous_acquisition.h.

10.106.2.2 init()

```
void GalileoE1PcpsTongAmbiguousAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.106.2.3 mag()

```
signed int GalileoE1PcpsTongAmbiguousAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.106.2.4 reset()

```
void GalileoE1PcpsTongAmbiguousAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.106.2.5 set_channel()

```
void GalileoE1PcpsTongAmbiguousAcquisition::set_channel (   
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 82 of file galileo_e1_pcps_tong_ambiguous_acquisition.h.

10.106.2.6 set_channel_fsm()

```
void GalileoE1PcpsTongAmbiguousAcquisition::set_channel_fsm (  
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 91 of file galileo_e1_pcps_tong_ambiguous_acquisition.h.

10.106.2.7 set_doppler_max()

```
void GalileoE1PcpsTongAmbiguousAcquisition::set_doppler_max (  
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.106.2.8 set_doppler_step()

```
void GalileoE1PcpsTongAmbiguousAcquisition::set_doppler_step (  
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.106.2.9 set_gnss_synchro()

```
void GalileoE1PcpsTongAmbiguousAcquisition::set_gnss_synchro (  
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.106.2.10 `set_local_code()`

```
void GalileoE1PcpsTongAmbiguousAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local code for Galileo E1 TONG acquisition algorithm.

Implements [AcquisitionInterface](#).

10.106.2.11 `set_state()`

```
void GalileoE1PcpsTongAmbiguousAcquisition::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.106.2.12 `set_threshold()`

```
void GalileoE1PcpsTongAmbiguousAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of TONG algorithm.

Implements [AcquisitionInterface](#).

10.106.2.13 `stop_acquisition()`

```
void GalileoE1PcpsTongAmbiguousAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

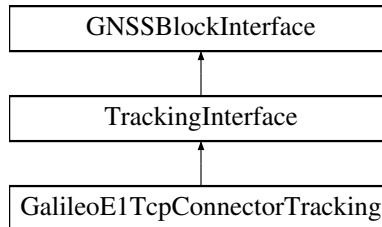
- [galileo_e1_pcps_tong_ambiguous_acquisition.h](#)

10.107 GalileoE1TcpConnectorTracking Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <galileo_e1_tcp_connector_tracking.h>
```

Inheritance diagram for GalileoE1TcpConnectorTracking:



Public Member Functions

- **GalileoE1TcpConnectorTracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "Galileo_E1_TCP_CONNECTOR_Tracking".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override
Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override
Stop running tracking.

10.107.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 41 of file galileo_e1_tcp_connector_tracking.h.

10.107.2 Member Function Documentation

10.107.2.1 implementation()

```
std::string GalileoE1TcpConnectorTracking::implementation ( ) [inline], [override], [virtual]
```

Returns "Galileo_E1_TCP_CONNECTOR_Tracking".

Implements [GNSSBlockInterface](#).

Definition at line 58 of file `galileo_e1_tcp_connector_tracking.h`.

10.107.2.2 set_channel()

```
void GalileoE1TcpConnectorTracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.107.2.3 set_gnss_synchro()

```
void GalileoE1TcpConnectorTracking::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.107.2.4 stop_tracking()

```
void GalileoE1TcpConnectorTracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

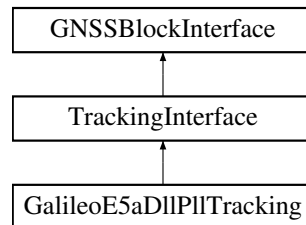
- [galileo_e1_tcp_connector_tracking.h](#)

10.108 GalileoE5aDIPIITracking Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <galileo_e5a_dll_pll_tracking.h>
```

Inheritance diagram for GalileoE5aDIPIITracking:



Public Member Functions

- **GalileoE5aDIPIITracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "Galileo_E5a_DLL_PLL_Tracking".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override
Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override
Stop running tracking.

10.108.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 38 of file galileo_e5a_dll_pll_tracking.h.

10.108.2 Member Function Documentation

10.108.2.1 implementation()

```
std::string GalileoE5aDllPllTracking::implementation ( ) [inline], [override], [virtual]
```

Returns "Galileo_E5a_DLL_PLL_Tracking".

Implements [GNSSBlockInterface](#).

Definition at line 55 of file `galileo_e5a_dll_pll_tracking.h`.

10.108.2.2 set_channel()

```
void GalileoE5aDllPllTracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.108.2.3 set_gnss_synchro()

```
void GalileoE5aDllPllTracking::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.108.2.4 stop_tracking()

```
void GalileoE5aDllPllTracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

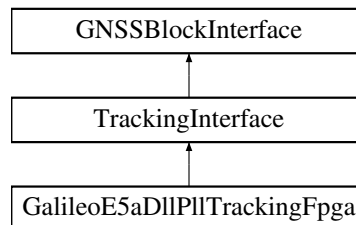
- [galileo_e5a_dll_pll_tracking.h](#)

10.109 GalileoE5aDIIPITrackingFpga Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <galileo_e5a_dll_pll_tracking_fpga.h>
```

Inheritance diagram for GalileoE5aDIIPITrackingFpga:



Public Member Functions

- [GalileoE5aDIIPITrackingFpga](#) (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
Constructor.
- virtual [~GalileoE5aDIIPITrackingFpga](#) ()
Destructor.
- std::string [role](#) () override
Role.
- std::string [implementation](#) () override
Returns "Galileo_E5a_DLL_PLL_Tracking_Fpga".
- size_t [item_size](#) () override
Returns size of lv_16sc_t.
- void [connect](#) (gr::top_block_sptr top_block) override
Connect.
- void [disconnect](#) (gr::top_block_sptr top_block) override
Disconnect.
- gr::basic_block_sptr [get_left_block](#) () override
Get left block.
- gr::basic_block_sptr [get_right_block](#) () override
Get right block.
- void [set_channel](#) (unsigned int channel) override
Set tracking channel unique ID.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void [start_tracking](#) () override
Start the tracking process in the FPGA.
- void [stop_tracking](#) () override
Stop the tracking process in the FPGA.

10.109.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 33 of file galileo_e5a_dll_pll_tracking_fpga.h.

10.109.2 Constructor & Destructor Documentation

10.109.2.1 GalileoE5aDlIPllTrackingFpga()

```
GalileoE5aDlIPllTrackingFpga::GalileoE5aDlIPllTrackingFpga (
    const ConfigurationInterface * configuration,
    const std::string & role,
    unsigned int in_streams,
    unsigned int out_streams )
```

Constructor.

10.109.2.2 ~GalileoE5aDlIPllTrackingFpga()

```
virtual GalileoE5aDlIPllTrackingFpga::~~GalileoE5aDlIPllTrackingFpga ( ) [virtual]
```

Destructor.

10.109.3 Member Function Documentation

10.109.3.1 connect()

```
void GalileoE5aDlIPllTrackingFpga::connect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Connect.

Implements [GNSSBlockInterface](#).

10.109.3.2 disconnect()

```
void GalileoE5aDlIPllTrackingFpga::disconnect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Disconnect.

Implements [GNSSBlockInterface](#).

10.109.3.3 get_left_block()

```
gr::basic_block_sptr GalileoE5aDllPllTrackingFpga::get_left_block ( ) [override], [virtual]
```

Get left block.

Implements [GNSSBlockInterface](#).

10.109.3.4 get_right_block()

```
gr::basic_block_sptr GalileoE5aDllPllTrackingFpga::get_right_block ( ) [override], [virtual]
```

Get right block.

Implements [GNSSBlockInterface](#).

10.109.3.5 implementation()

```
std::string GalileoE5aDllPllTrackingFpga::implementation ( ) [inline], [override], [virtual]
```

Returns "Galileo_E5a_DLL_PLL_Tracking_Fpga".

Implements [GNSSBlockInterface](#).

Definition at line 61 of file `galileo_e5a_dll_pll_tracking_fpga.h`.

10.109.3.6 item_size()

```
size_t GalileoE5aDllPllTrackingFpga::item_size ( ) [inline], [override], [virtual]
```

Returns size of `lv_16sc_t`.

Implements [GNSSBlockInterface](#).

Definition at line 69 of file `galileo_e5a_dll_pll_tracking_fpga.h`.

10.109.3.7 role()

```
std::string GalileoE5aDllPllTrackingFpga::role ( ) [inline], [override], [virtual]
```

Role.

Implements [GNSSBlockInterface](#).

Definition at line 53 of file `galileo_e5a_dll_pll_tracking_fpga.h`.

10.109.3.8 set_channel()

```
void GalileoE5aDllPllTrackingFpga::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.109.3.9 set_gnss_synchro()

```
void GalileoE5aDllPllTrackingFpga::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.109.3.10 start_tracking()

```
void GalileoE5aDllPllTrackingFpga::start_tracking ( ) [override], [virtual]
```

Start the tracking process in the FPGA.

Implements [TrackingInterface](#).

10.109.3.11 stop_tracking()

```
void GalileoE5aDllPllTrackingFpga::stop_tracking ( ) [override], [virtual]
```

Stop the tracking process in the FPGA.

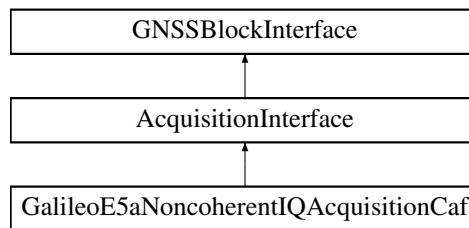
Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

- [galileo_e5a_dll_pll_tracking_fpga.h](#)

10.110 GalileoE5aNoncoherentIQAcquisitionCaf Class Reference

Inheritance diagram for GalileoE5aNoncoherentIQAcquisitionCaf:



Public Member Functions

- **GalileoE5aNoncoherentIQAcquisitionCaf** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "Galileo_E5a_Noncoherent_IQ_Acquisition_CAF".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.*
- void **set_channel** (unsigned int channel) override
 - Set acquisition channel unique ID.*
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
 - Set channel fsm associated to this acquisition instance.*
- void **set_threshold** (float threshold) override
 - Set statistics threshold of PCPS algorithm.*
- void **set_doppler_max** (unsigned int doppler_max) override
 - Set maximum Doppler off grid search.*
- void **set_doppler_step** (unsigned int doppler_step) override
 - Set Doppler steps for the grid search.*
- void **init** () override
 - Initializes acquisition algorithm.*
- void **set_local_code** () override
 - Sets local Galileo E5a code for PCPS acquisition algorithm.*
- signed int **mag** () override
 - Returns the maximum peak of grid search.*
- void **reset** () override
 - Restart acquisition algorithm.*
- void **set_state** (int state) override
 - If set to 1, ensures that acquisition starts at the first available sample.*
- void **stop_acquisition** () override
 - Stop running acquisition.*
- void **set_resampler_latency** (uint32_t latency_samples __attribute__((unused))) override

10.110.1 Detailed Description

Definition at line 39 of file `galileo_e5a_noncoherent_iq_acquisition_caf.h`.

10.110.2 Member Function Documentation

10.110.2.1 `implementation()`

```
std::string GalileoE5aNoncoherentIQAcquisitionCaf::implementation ( ) [inline], [override],  
[virtual]
```

Returns "Galileo_E5a_Noncoherent_IQ_Acquisition_CAF".

Implements [GNSSBlockInterface](#).

Definition at line 57 of file `galileo_e5a_noncoherent_iq_acquisition_caf.h`.

10.110.2.2 `init()`

```
void GalileoE5aNoncoherentIQAcquisitionCaf::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.110.2.3 `mag()`

```
signed int GalileoE5aNoncoherentIQAcquisitionCaf::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.110.2.4 `reset()`

```
void GalileoE5aNoncoherentIQAcquisitionCaf::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.110.2.5 set_channel()

```
void GalileoE5aNoncoherentIQAcquisitionCaf::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 82 of file galileo_e5a_noncoherent_iq_acquisition_caf.h.

10.110.2.6 set_channel_fsm()

```
void GalileoE5aNoncoherentIQAcquisitionCaf::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 91 of file galileo_e5a_noncoherent_iq_acquisition_caf.h.

10.110.2.7 set_doppler_max()

```
void GalileoE5aNoncoherentIQAcquisitionCaf::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.110.2.8 set_doppler_step()

```
void GalileoE5aNoncoherentIQAcquisitionCaf::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.110.2.9 set_gnss_synchro()

```
void GalileoE5aNoncoherentIQAcquisitionCaf::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.110.2.10 set_local_code()

```
void GalileoE5aNoncoherentIQAcquisitionCaf::set_local_code ( ) [override], [virtual]
```

Sets local Galileo E5a code for PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.110.2.11 set_state()

```
void GalileoE5aNoncoherentIQAcquisitionCaf::set_state (
    int state ) [override], [virtual]
```

If set to 1, ensures that acquisition starts at the first available sample.

Parameters

<i>state</i>	- int=1 forces start of acquisition
--------------	-------------------------------------

Implements [AcquisitionInterface](#).

10.110.2.12 set_threshold()

```
void GalileoE5aNoncoherentIQAcquisitionCaf::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.110.2.13 stop_acquisition()

```
void GalileoE5aNoncoherentIQAcquisitionCaf::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

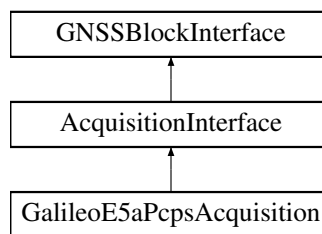
Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

- [galileo_e5a_noncoherent_iq_acquisition_caf.h](#)

10.111 GalileoE5aPcpsAcquisition Class Reference

Inheritance diagram for GalileoE5aPcpsAcquisition:



Public Member Functions

- **GalileoE5aPcpsAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **set_channel** (unsigned int channel) override
Set acquisition channel unique ID.
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
Set channel fsm associated to this acquisition instance.
- void **set_threshold** (float threshold) override
Set statistics threshold of PCPS algorithm.
- void **set_doppler_max** (unsigned int doppler_max) override
Set maximum Doppler off grid search.
- void **set_doppler_step** (unsigned int doppler_step) override
Set Doppler steps for the grid search.
- void **set_doppler_center** (int doppler_center) override
Set Doppler center for the grid search.

- void `init()` override
Initializes acquisition algorithm.
- void `set_local_code()` override
Sets local Galileo E5a code for PCPS acquisition algorithm.
- signed int `mag()` override
Returns the maximum peak of grid search.
- void `reset()` override
Restart acquisition algorithm.
- void `set_state(int state)` override
If set to 1, ensures that acquisition starts at the first available sample.
- void `stop_acquisition()` override
Stop running acquisition.
- void `set_resampler_latency(uint32_t latency_samples)` override
Sets the resampler latency to account it in the acquisition code delay estimation.

10.111.1 Detailed Description

Definition at line 33 of file `galileo_e5a_pcps_acquisition.h`.

10.111.2 Member Function Documentation

10.111.2.1 `init()`

```
void GalileoE5aPcpsAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.111.2.2 `mag()`

```
signed int GalileoE5aPcpsAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.111.2.3 reset()

```
void GalileoE5aPcpsAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.111.2.4 set_channel()

```
void GalileoE5aPcpsAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 74 of file `galileo_e5a_pcps_acquisition.h`.

10.111.2.5 set_channel_fsm()

```
void GalileoE5aPcpsAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 83 of file `galileo_e5a_pcps_acquisition.h`.

10.111.2.6 set_doppler_center()

```
void GalileoE5aPcpsAcquisition::set_doppler_center (
    int doppler_center ) [override]
```

Set Doppler center for the grid search.

10.111.2.7 set_doppler_max()

```
void GalileoE5aPcpsAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.111.2.8 `set_doppler_step()`

```
void GalileoE5aPcpsAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.111.2.9 `set_gnss_synchro()`

```
void GalileoE5aPcpsAcquisition::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.111.2.10 `set_local_code()`

```
void GalileoE5aPcpsAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local Galileo E5a code for PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.111.2.11 `set_resampler_latency()`

```
void GalileoE5aPcpsAcquisition::set_resampler_latency (
    uint32_t latency_samples ) [override], [virtual]
```

Sets the resampler latency to account it in the acquisition code delay estimation.

Implements [AcquisitionInterface](#).

10.111.2.12 `set_state()`

```
void GalileoE5aPcpsAcquisition::set_state (
    int state ) [override], [virtual]
```

If set to 1, ensures that acquisition starts at the first available sample.

Parameters

<i>state</i>	- int=1 forces start of acquisition
--------------	-------------------------------------

Implements [AcquisitionInterface](#).

10.111.2.13 `set_threshold()`

```
void GalileoE5aPcpsAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.111.2.14 `stop_acquisition()`

```
void GalileoE5aPcpsAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

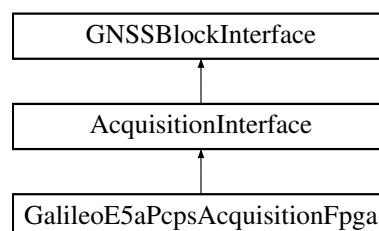
- [galileo_e5a_pcps_acquisition.h](#)

10.112 GalileoE5aPcpsAcquisitionFpga Class Reference

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for Galileo E5a signals.

```
#include <galileo_e5a_pcps_acquisition_fpga.h>
```

Inheritance diagram for GalileoE5aPcpsAcquisitionFpga:



Public Member Functions

- [GalileoE5aPcpsAcquisitionFpga](#) (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
Constructor.
- [~GalileoE5aPcpsAcquisitionFpga](#) ()=default
Destructor.
- std::string [role](#) () override
Role.
- std::string [implementation](#) () override
Returns "Galileo_E5a_Pcps_Acquisition_Fpga".
- size_t [item_size](#) () override
Returns size of lv_16sc_t.
- void [connect](#) (gr::top_block_sptr top_block) override
Connect.
- void [disconnect](#) (gr::top_block_sptr top_block) override
Disconnect.
- gr::basic_block_sptr [get_left_block](#) () override
Get left block.
- gr::basic_block_sptr [get_right_block](#) () override
Get right block.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void [set_channel](#) (unsigned int channel) override
Set acquisition channel unique ID.
- void [set_channel_fsm](#) (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
Set channel fsm associated to this acquisition instance.
- void [set_threshold](#) (float threshold) override
Set statistics threshold of PCPS algorithm.
- void [set_doppler_max](#) (unsigned int doppler_max) override
Set maximum Doppler off grid search.
- void [set_doppler_step](#) (unsigned int doppler_step) override
Set Doppler steps for the grid search.
- void [set_doppler_center](#) (int doppler_center) override
Set Doppler center for the grid search.
- void [init](#) () override
Initializes acquisition algorithm.
- void [set_local_code](#) () override
Sets local Galileo E5a code for PCPS acquisition algorithm.
- signed int [mag](#) () override
Returns the maximum peak of grid search.
- void [reset](#) () override
Restart acquisition algorithm.
- void [set_state](#) (int state) override
If set to 1, ensures that acquisition starts at the first available sample.
- void [set_single_doppler_flag](#) (unsigned int single_doppler_flag)
This function is only used in the unit tests.
- void [stop_acquisition](#) () override
Stop running acquisition.
- void [set_resampler_latency](#) (uint32_t latency_samples __attribute__((unused))) override
Set resampler latency.

10.112.1 Detailed Description

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for Galileo E5a signals.

Definition at line 39 of file `galileo_e5a_pcps_acquisition_fpga.h`.

10.112.2 Constructor & Destructor Documentation

10.112.2.1 GalileoE5aPcpsAcquisitionFpga()

```
GalileoE5aPcpsAcquisitionFpga::GalileoE5aPcpsAcquisitionFpga (
    const ConfigurationInterface * configuration,
    const std::string & role,
    unsigned int in_streams,
    unsigned int out_streams )
```

Constructor.

10.112.2.2 ~GalileoE5aPcpsAcquisitionFpga()

```
GalileoE5aPcpsAcquisitionFpga::~GalileoE5aPcpsAcquisitionFpga ( ) [default]
```

Destructor.

10.112.3 Member Function Documentation

10.112.3.1 connect()

```
void GalileoE5aPcpsAcquisitionFpga::connect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Connect.

Implements [GNSSBlockInterface](#).

10.112.3.2 disconnect()

```
void GalileoE5aPcpsAcquisitionFpga::disconnect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Disconnect.

Implements [GNSSBlockInterface](#).

10.112.3.3 get_left_block()

```
gr::basic_block_sptr GalileoE5aPcpsAcquisitionFpga::get_left_block ( ) [override], [virtual]
```

Get left block.

Implements [GNSSBlockInterface](#).

10.112.3.4 get_right_block()

```
gr::basic_block_sptr GalileoE5aPcpsAcquisitionFpga::get_right_block ( ) [override], [virtual]
```

Get right block.

Implements [GNSSBlockInterface](#).

10.112.3.5 implementation()

```
std::string GalileoE5aPcpsAcquisitionFpga::implementation ( ) [inline], [override], [virtual]
```

Returns "Galileo_E5a_Pcps_Acquisition_Fpga".

Implements [GNSSBlockInterface](#).

Definition at line 67 of file `galileo_e5a_pcps_acquisition_fpga.h`.

10.112.3.6 init()

```
void GalileoE5aPcpsAcquisitionFpga::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.112.3.7 item_size()

```
size_t GalileoE5aPcpsAcquisitionFpga::item_size ( ) [inline], [override], [virtual]
```

Returns size of lv_16sc_t.

Implements [GNSSBlockInterface](#).

Definition at line 75 of file galileo_e5a_pcps_acquisition_fpga.h.

10.112.3.8 mag()

```
signed int GalileoE5aPcpsAcquisitionFpga::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.112.3.9 reset()

```
void GalileoE5aPcpsAcquisitionFpga::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.112.3.10 role()

```
std::string GalileoE5aPcpsAcquisitionFpga::role ( ) [inline], [override], [virtual]
```

Role.

Implements [GNSSBlockInterface](#).

Definition at line 59 of file galileo_e5a_pcps_acquisition_fpga.h.

10.112.3.11 set_channel()

```
void GalileoE5aPcpsAcquisitionFpga::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 110 of file galileo_e5a_pcps_acquisition_fpga.h.

10.112.3.12 `set_channel_fsm()`

```
void GalileoE5aPcpsAcquisitionFpga::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 119 of file `galileo_e5a_pcps_acquisition_fpga.h`.

10.112.3.13 `set_doppler_center()`

```
void GalileoE5aPcpsAcquisitionFpga::set_doppler_center (
    int doppler_center ) [override]
```

Set Doppler center for the grid search.

10.112.3.14 `set_doppler_max()`

```
void GalileoE5aPcpsAcquisitionFpga::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.112.3.15 `set_doppler_step()`

```
void GalileoE5aPcpsAcquisitionFpga::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.112.3.16 `set_gnss_synchro()`

```
void GalileoE5aPcpsAcquisitionFpga::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.112.3.17 set_local_code()

```
void GalileoE5aPcpsAcquisitionFpga::set_local_code ( ) [override], [virtual]
```

Sets local Galileo E5a code for PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.112.3.18 set_resampler_latency()

```
void GalileoE5aPcpsAcquisitionFpga::set_resampler_latency (
    uint32_t latency_samples __attribute__((unused)) ) [inline], [override]
```

Set resampler latency.

Definition at line 185 of file galileo_e5a_pcps_acquisition_fpga.h.

10.112.3.19 set_single_doppler_flag()

```
void GalileoE5aPcpsAcquisitionFpga::set_single_doppler_flag (
    unsigned int single_doppler_flag )
```

This function is only used in the unit tests.

10.112.3.20 set_state()

```
void GalileoE5aPcpsAcquisitionFpga::set_state (
    int state ) [override], [virtual]
```

If set to 1, ensures that acquisition starts at the first available sample.

Parameters

<i>state</i>	- int=1 forces start of acquisition
--------------	-------------------------------------

Implements [AcquisitionInterface](#).

10.112.3.21 set_threshold()

```
void GalileoE5aPcpsAcquisitionFpga::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.112.3.22 stop_acquisition()

```
void GalileoE5aPcpsAcquisitionFpga::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

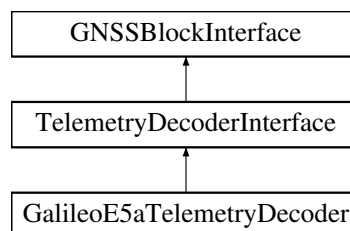
- [galileo_e5a_pcps_acquisition_fpga.h](#)

10.113 GalileoE5aTelemetryDecoder Class Reference

This class implements a NAV data decoder for Galileo INAV frames in E1B radio link.

```
#include <galileo_e5a_telemetry_decoder.h>
```

Inheritance diagram for GalileoE5aTelemetryDecoder:



Public Member Functions

- **GalileoE5aTelemetryDecoder** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_satellite** (const [Gnss_Satellite](#) &satellite) override
- std::string **role** () override
- std::string **implementation** () override
- *Returns "Galileo_E5a_Telemetry_Decoder".*
- void **set_channel** (int channel) override
- void **reset** () override
- size_t **item_size** () override

10.113.1 Detailed Description

This class implements a NAV data decoder for Galileo INAV frames in E1B radio link.

Definition at line 42 of file `galileo_e5a_telemetry_decoder.h`.

10.113.2 Member Function Documentation

10.113.2.1 `implementation()`

```
std::string GalileoE5aTelemetryDecoder::implementation ( ) [inline], [override], [virtual]
```

Returns "Galileo_E5a_Telemetry_Decoder".

Implements [GNSSBlockInterface](#).

Definition at line 68 of file `galileo_e5a_telemetry_decoder.h`.

The documentation for this class was generated from the following file:

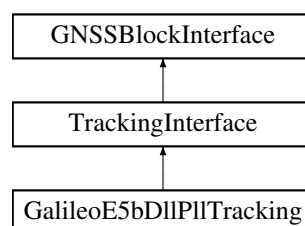
- [galileo_e5a_telemetry_decoder.h](#)

10.114 GalileoE5bDIIPIITracking Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <galileo_e5b_dll_pll_tracking.h>
```

Inheritance diagram for GalileoE5bDIIPIITracking:



Public Member Functions

- **GalileoE5bDlIPllTracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override

Returns "Galileo_E5b_DLL_PLL_Tracking".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override

Connect.
- void **disconnect** (gr::top_block_sptr top_block) override

Disconnect.
- gr::basic_block_sptr **get_left_block** () override

Get left block.
- gr::basic_block_sptr **get_right_block** () override

Get right block.
- void **set_channel** (unsigned int channel) override

Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override

Stop running tracking.

10.114.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 39 of file `galileo_e5b_dll_pll_tracking.h`.

10.114.2 Member Function Documentation

10.114.2.1 connect()

```
void GalileoE5bDllPllTracking::connect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Connect.

Implements [GNSSBlockInterface](#).

10.114.2.2 disconnect()

```
void GalileoE5bDllPllTracking::disconnect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Disconnect.

Implements [GNSSBlockInterface](#).

10.114.2.3 get_left_block()

```
gr::basic_block_sptr GalileoE5bDllPllTracking::get_left_block ( ) [override], [virtual]
```

Get left block.

Implements [GNSSBlockInterface](#).

10.114.2.4 get_right_block()

```
gr::basic_block_sptr GalileoE5bDllPllTracking::get_right_block ( ) [override], [virtual]
```

Get right block.

Implements [GNSSBlockInterface](#).

10.114.2.5 implementation()

```
std::string GalileoE5bDllPllTracking::implementation ( ) [inline], [override], [virtual]
```

Returns "Galileo_E5b_DLL_PLL_Tracking".

Implements [GNSSBlockInterface](#).

Definition at line 56 of file `galileo_e5b_dll_pll_tracking.h`.

10.114.2.6 set_channel()

```
void GalileoE5bDllPllTracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.114.2.7 `set_gnss_synchro()`

```
void GalileoE5bDllPllTracking::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.114.2.8 `stop_tracking()`

```
void GalileoE5bDllPllTracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

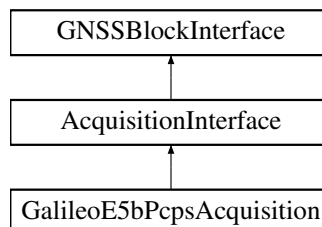
Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

- [galileo_e5b_dll_pll_tracking.h](#)

10.115 GalileoE5bPcpsAcquisition Class Reference

Inheritance diagram for GalileoE5bPcpsAcquisition:



Public Member Functions

- [GalileoE5bPcpsAcquisition](#) (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
Constructor.
- [~GalileoE5bPcpsAcquisition](#) ()=default
Destructor.
- std::string [role](#) () override
Role.
- std::string [implementation](#) () override
Returns "GALILEO_E5b_PCPS_Acquisition".
- size_t [item_size](#) () override
Returns size of lv_16sc_t.

- void [connect](#) (gr::top_block_sptr top_block) override
Connect.
- void [disconnect](#) (gr::top_block_sptr top_block) override
Disconnect.
- gr::basic_block_sptr [get_left_block](#) () override
Get left block.
- gr::basic_block_sptr [get_right_block](#) () override
Get right block.
- void [set_gnss_synchro](#) (Gnss_Synchro *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void [set_channel](#) (unsigned int channel) override
Set acquisition channel unique ID.
- void [set_channel_fsm](#) (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
Set channel fsm associated to this acquisition instance.
- void [set_threshold](#) (float threshold) override
Set statistics threshold of PCPS algorithm.
- void [set_doppler_max](#) (unsigned int doppler_max) override
Set maximum Doppler off grid search.
- void [set_doppler_step](#) (unsigned int doppler_step) override
Set Doppler steps for the grid search.
- void [set_doppler_center](#) (int doppler_center) override
Set Doppler center for the grid search.
- void [init](#) () override
Initializes acquisition algorithm.
- void [set_local_code](#) () override
Sets local Galileo E5b code for PCPS acquisition algorithm.
- signed int [mag](#) () override
Returns the maximum peak of grid search.
- void [reset](#) () override
Restart acquisition algorithm.
- void [set_state](#) (int state) override
If set to 1, ensures that acquisition starts at the first available sample.
- void [stop_acquisition](#) () override
Stop running acquisition.
- void [set_resampler_latency](#) (uint32_t latency_samples) override
Sets the resampler latency to account it in the acquisition code delay estimation.

10.115.1 Detailed Description

Definition at line 35 of file `galileo_e5b_pcps_acquisition.h`.

10.115.2 Constructor & Destructor Documentation

10.115.2.1 GalileoE5bPcpsAcquisition()

```
GalileoE5bPcpsAcquisition::GalileoE5bPcpsAcquisition (
    const ConfigurationInterface * configuration,
    const std::string & role,
    unsigned int in_streams,
    unsigned int out_streams )
```

Constructor.

10.115.2.2 ~GalileoE5bPcpsAcquisition()

```
GalileoE5bPcpsAcquisition::~GalileoE5bPcpsAcquisition ( ) [default]
```

Destructor.

10.115.3 Member Function Documentation

10.115.3.1 connect()

```
void GalileoE5bPcpsAcquisition::connect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Connect.

Implements [GNSSBlockInterface](#).

10.115.3.2 disconnect()

```
void GalileoE5bPcpsAcquisition::disconnect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Disconnect.

Implements [GNSSBlockInterface](#).

10.115.3.3 get_left_block()

```
gr::basic_block_sptr GalileoE5bPcpsAcquisition::get_left_block ( ) [override], [virtual]
```

Get left block.

Implements [GNSSBlockInterface](#).

10.115.3.4 get_right_block()

```
gr::basic_block_sptr GalileoE5bPcpsAcquisition::get_right_block ( ) [override], [virtual]
```

Get right block.

Implements [GNSSBlockInterface](#).

10.115.3.5 implementation()

```
std::string GalileoE5bPcpsAcquisition::implementation ( ) [inline], [override], [virtual]
```

Returns "GALILEO_E5b_PCPS_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 63 of file `galileo_e5b_pcps_acquisition.h`.

10.115.3.6 init()

```
void GalileoE5bPcpsAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.115.3.7 item_size()

```
size_t GalileoE5bPcpsAcquisition::item_size ( ) [inline], [override], [virtual]
```

Returns size of `lv_16sc_t`.

Implements [GNSSBlockInterface](#).

Definition at line 71 of file `galileo_e5b_pcps_acquisition.h`.

10.115.3.8 mag()

```
signed int GalileoE5bPcpsAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.115.3.9 reset()

```
void GalileoE5bPcpsAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.115.3.10 role()

```
std::string GalileoE5bPcpsAcquisition::role ( ) [inline], [override], [virtual]
```

Role.

Implements [GNSSBlockInterface](#).

Definition at line 54 of file `galileo_e5b_pcps_acquisition.h`.

10.115.3.11 set_channel()

```
void GalileoE5bPcpsAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 106 of file `galileo_e5b_pcps_acquisition.h`.

10.115.3.12 set_channel_fsm()

```
void GalileoE5bPcpsAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 115 of file `galileo_e5b_pcps_acquisition.h`.

10.115.3.13 set_doppler_center()

```
void GalileoE5bPcpsAcquisition::set_doppler_center (
    int doppler_center ) [override]
```

Set Doppler center for the grid search.

10.115.3.14 set_doppler_max()

```
void GalileoE5bPcpsAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.115.3.15 set_doppler_step()

```
void GalileoE5bPcpsAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.115.3.16 set_gnss_synchro()

```
void GalileoE5bPcpsAcquisition::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.115.3.17 set_local_code()

```
void GalileoE5bPcpsAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local Galileo E5b code for PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.115.3.18 `set_resampler_latency()`

```
void GalileoE5bPcpsAcquisition::set_resampler_latency (
    uint32_t latency_samples ) [override], [virtual]
```

Sets the resampler latency to account it in the acquisition code delay estimation.

Implements [AcquisitionInterface](#).

10.115.3.19 `set_state()`

```
void GalileoE5bPcpsAcquisition::set_state (
    int state ) [override], [virtual]
```

If set to 1, ensures that acquisition starts at the first available sample.

Parameters

<i>state</i>	- int=1 forces start of acquisition
--------------	-------------------------------------

Implements [AcquisitionInterface](#).

10.115.3.20 `set_threshold()`

```
void GalileoE5bPcpsAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.115.3.21 `stop_acquisition()`

```
void GalileoE5bPcpsAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

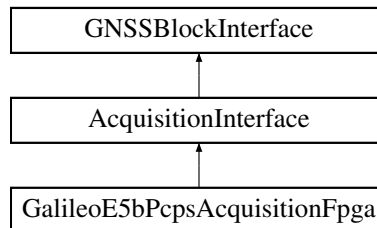
- [galileo_e5b_pcps_acquisition.h](#)

10.116 GalileoE5bPcpsAcquisitionFpga Class Reference

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for Galileo E5b signals.

```
#include <galileo_e5b_pcps_acquisition_fpga.h>
```

Inheritance diagram for GalileoE5bPcpsAcquisitionFpga:



Public Member Functions

- [GalileoE5bPcpsAcquisitionFpga](#) (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
Constructor.
- [~GalileoE5bPcpsAcquisitionFpga](#) ()=default
Destructor.
- std::string [role](#) () override
Role.
- std::string [implementation](#) () override
Returns "Galileo_E5b_Pcps_Acquisition_Fpga".
- size_t [item_size](#) () override
Returns size of lv_16sc_t.
- void [connect](#) (gr::top_block_sptr top_block) override
Connect.
- void [disconnect](#) (gr::top_block_sptr top_block) override
Disconnect.
- gr::basic_block_sptr [get_left_block](#) () override
Get left block.
- gr::basic_block_sptr [get_right_block](#) () override
Get right block.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common Gnss_Synchro object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void [set_channel](#) (unsigned int channel) override
Set acquisition channel unique ID.
- void [set_channel_fsm](#) (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
Set channel fsm associated to this acquisition instance.
- void [set_threshold](#) (float threshold) override
Set statistics threshold of PCPS algorithm.
- void [set_doppler_max](#) (unsigned int doppler_max) override
Set maximum Doppler off grid search.
- void [set_doppler_step](#) (unsigned int doppler_step) override
Set Doppler steps for the grid search.

- void [set_doppler_center](#) (int doppler_center) override
Set Doppler center for the grid search.
- void [init](#) () override
Initializes acquisition algorithm.
- void [set_local_code](#) () override
Sets local Galileo E5b code for PCPS acquisition algorithm.
- signed int [mag](#) () override
Returns the maximum peak of grid search.
- void [reset](#) () override
Restart acquisition algorithm.
- void [set_state](#) (int state) override
If set to 1, ensures that acquisition starts at the first available sample.
- void [set_single_doppler_flag](#) (unsigned int single_doppler_flag)
This function is only used in the unit tests.
- void [stop_acquisition](#) () override
Stop running acquisition.
- void [set_resampler_latency](#) (uint32_t latency_samples __attribute__((unused))) override
Set resampler latency.

10.116.1 Detailed Description

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for Galileo E5b signals.

Definition at line 39 of file `galileo_e5b_pcps_acquisition_fpga.h`.

10.116.2 Constructor & Destructor Documentation

10.116.2.1 GalileoE5bPcpsAcquisitionFpga()

```
GalileoE5bPcpsAcquisitionFpga::GalileoE5bPcpsAcquisitionFpga (
    const ConfigurationInterface * configuration,
    const std::string & role,
    unsigned int in_streams,
    unsigned int out_streams )
```

Constructor.

10.116.2.2 ~GalileoE5bPcpsAcquisitionFpga()

```
GalileoE5bPcpsAcquisitionFpga::~GalileoE5bPcpsAcquisitionFpga ( ) [default]
```

Destructor.

10.116.3 Member Function Documentation

10.116.3.1 connect()

```
void GalileoE5bPcpsAcquisitionFpga::connect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Connect.

Implements [GNSSBlockInterface](#).

10.116.3.2 disconnect()

```
void GalileoE5bPcpsAcquisitionFpga::disconnect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Disconnect.

Implements [GNSSBlockInterface](#).

10.116.3.3 get_left_block()

```
gr::basic_block_sptr GalileoE5bPcpsAcquisitionFpga::get_left_block ( ) [override], [virtual]
```

Get left block.

Implements [GNSSBlockInterface](#).

10.116.3.4 get_right_block()

```
gr::basic_block_sptr GalileoE5bPcpsAcquisitionFpga::get_right_block ( ) [override], [virtual]
```

Get right block.

Implements [GNSSBlockInterface](#).

10.116.3.5 implementation()

```
std::string GalileoE5bPcpsAcquisitionFpga::implementation ( ) [inline], [override], [virtual]
```

Returns "Galileo_E5b_Pcps_Acquisition_Fpga".

Implements [GNSSBlockInterface](#).

Definition at line 66 of file galileo_e5b_pcps_acquisition_fpga.h.

10.116.3.6 init()

```
void GalileoE5bPcpsAcquisitionFpga::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.116.3.7 item_size()

```
size_t GalileoE5bPcpsAcquisitionFpga::item_size ( ) [inline], [override], [virtual]
```

Returns size of lv_16sc_t.

Implements [GNSSBlockInterface](#).

Definition at line 74 of file galileo_e5b_pcps_acquisition_fpga.h.

10.116.3.8 mag()

```
signed int GalileoE5bPcpsAcquisitionFpga::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.116.3.9 reset()

```
void GalileoE5bPcpsAcquisitionFpga::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.116.3.10 `role()`

```
std::string GalileoE5bPcpsAcquisitionFpga::role ( ) [inline], [override], [virtual]
```

Role.

Implements [GNSSBlockInterface](#).

Definition at line 58 of file `galileo_e5b_pcps_acquisition_fpga.h`.

10.116.3.11 `set_channel()`

```
void GalileoE5bPcpsAcquisitionFpga::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 109 of file `galileo_e5b_pcps_acquisition_fpga.h`.

10.116.3.12 `set_channel_fsm()`

```
void GalileoE5bPcpsAcquisitionFpga::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 118 of file `galileo_e5b_pcps_acquisition_fpga.h`.

10.116.3.13 `set_doppler_center()`

```
void GalileoE5bPcpsAcquisitionFpga::set_doppler_center (
    int doppler_center ) [override]
```

Set Doppler center for the grid search.

10.116.3.14 `set_doppler_max()`

```
void GalileoE5bPcpsAcquisitionFpga::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.116.3.15 `set_doppler_step()`

```
void GalileoE5bPcpsAcquisitionFpga::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.116.3.16 `set_gnss_synchro()`

```
void GalileoE5bPcpsAcquisitionFpga::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.116.3.17 `set_local_code()`

```
void GalileoE5bPcpsAcquisitionFpga::set_local_code ( ) [override], [virtual]
```

Sets local Galileo E5b code for PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.116.3.18 `set_resampler_latency()`

```
void GalileoE5bPcpsAcquisitionFpga::set_resampler_latency (
    uint32_t latency_samples __attribute__((unused)) ) [inline], [override]
```

Set resampler latency.

Definition at line 184 of file `galileo_e5b_pcps_acquisition_fpga.h`.

10.116.3.19 set_single_doppler_flag()

```
void GalileoE5bPcpsAcquisitionFpga::set_single_doppler_flag (
    unsigned int single_doppler_flag )
```

This function is only used in the unit tests.

10.116.3.20 set_state()

```
void GalileoE5bPcpsAcquisitionFpga::set_state (
    int state ) [override], [virtual]
```

If set to 1, ensures that acquisition starts at the first available sample.

Parameters

<i>state</i>	- int=1 forces start of acquisition
--------------	-------------------------------------

Implements [AcquisitionInterface](#).

10.116.3.21 set_threshold()

```
void GalileoE5bPcpsAcquisitionFpga::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.116.3.22 stop_acquisition()

```
void GalileoE5bPcpsAcquisitionFpga::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

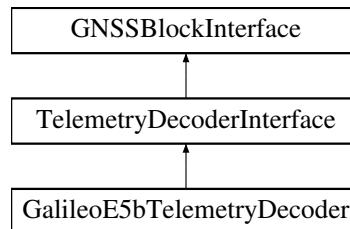
- [galileo_e5b_pcps_acquisition_fpga.h](#)

10.117 GalileoE5bTelemetryDecoder Class Reference

This class implements a NAV data decoder for Galileo INAV frames in E5b radio link.

```
#include <galileo_e5b_telemetry_decoder.h>
```

Inheritance diagram for GalileoE5bTelemetryDecoder:



Public Member Functions

- **GalileoE5bTelemetryDecoder** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string [implementation](#) () override
Returns "Galileo_E5b_Telemetry_Decoder".
- void [connect](#) (gr::top_block_sptr top_block) override
Connect.
- void [disconnect](#) (gr::top_block_sptr top_block) override
Disconnect.
- gr::basic_block_sptr [get_left_block](#) () override
Get left block.
- gr::basic_block_sptr [get_right_block](#) () override
Get right block.
- void [set_satellite](#) (const [Gnss_Satellite](#) &satellite) override
- std::string [role](#) () override
- void [set_channel](#) (int channel) override
- void [reset](#) () override
- size_t [item_size](#) () override

10.117.1 Detailed Description

This class implements a NAV data decoder for Galileo INAV frames in E5b radio link.

Definition at line 41 of file `galileo_e5b_telemetry_decoder.h`.

10.117.2 Member Function Documentation

10.117.2.1 connect()

```
void GalileoE5bTelemetryDecoder::connect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Connect.

Implements [GNSSBlockInterface](#).

10.117.2.2 disconnect()

```
void GalileoE5bTelemetryDecoder::disconnect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Disconnect.

Implements [GNSSBlockInterface](#).

10.117.2.3 get_left_block()

```
gr::basic_block_sptr GalileoE5bTelemetryDecoder::get_left_block ( ) [override], [virtual]
```

Get left block.

Implements [GNSSBlockInterface](#).

10.117.2.4 get_right_block()

```
gr::basic_block_sptr GalileoE5bTelemetryDecoder::get_right_block ( ) [override], [virtual]
```

Get right block.

Implements [GNSSBlockInterface](#).

10.117.2.5 implementation()

```
std::string GalileoE5bTelemetryDecoder::implementation ( ) [inline], [override], [virtual]
```

Returns "Galileo_E5b_Telemetry_Decoder".

Implements [GNSSBlockInterface](#).

Definition at line 55 of file `galileo_e5b_telemetry_decoder.h`.

The documentation for this class was generated from the following file:

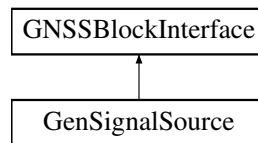
- [galileo_e5b_telemetry_decoder.h](#)

10.118 GenSignalSource Class Reference

This class wraps blocks that generates synthesized GNSS signal and filters the signal.

```
#include <gen_signal_source.h>
```

Inheritance diagram for GenSignalSource:



Public Member Functions

- [GenSignalSource](#) (std::shared_ptr< [GNSSBlockInterface](#) > signal_generator, std::shared_ptr< [GNSSBlockInterface](#) > filter, std::string role, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
Constructor.
- virtual [~GenSignalSource](#) ()=default
Virtual destructor.
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- std::string **role** () override
- std::string **implementation** () override
Returns "Signal Source".
- size_t **item_size** () override
- std::shared_ptr< [GNSSBlockInterface](#) > **signal_generator** () const

10.118.1 Detailed Description

This class wraps blocks that generates synthesized GNSS signal and filters the signal.

Definition at line 36 of file gen_signal_source.h.

10.118.2 Constructor & Destructor Documentation

10.118.2.1 GenSignalSource()

```

GenSignalSource::GenSignalSource (
    std::shared_ptr< GNSSBlockInterface > signal_generator,
    std::shared_ptr< GNSSBlockInterface > filter,
    std::string role,
    Concurrent\_Queue< pmt::pmt_t > * queue )

```

Constructor.

10.118.2.2 ~GenSignalSource()

```
virtual GenSignalSource::~~GenSignalSource ( ) [virtual], [default]
```

Virtual destructor.

10.118.3 Member Function Documentation

10.118.3.1 implementation()

```
std::string GenSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "Signal Source".

Implements [GNSSBlockInterface](#).

Definition at line 53 of file `gen_signal_source.h`.

The documentation for this class was generated from the following file:

- [gen_signal_source.h](#)

10.119 GeoJSON_Printer Class Reference

Prints PVT solutions in GeoJSON format file.

```
#include <geojson_printer.h>
```

Public Member Functions

- **GeoJSON_Printer** (const std::string &base_path=".")
- bool **set_headers** (const std::string &filename, bool time_tag_name=true)
- bool **print_position** (const [Pvt_Solution](#) *const position, bool print_average_values)
- bool **close_file** ()

10.119.1 Detailed Description

Prints PVT solutions in GeoJSON format file.

See <https://tools.ietf.org/html/rfc7946>

Definition at line 36 of file `geojson_printer.h`.

The documentation for this class was generated from the following file:

- [geojson_printer.h](#)

10.120 `geph_t` Struct Reference

Public Attributes

- int **sat**
- int **iode**
- int **frq**
- int **svh**
- int **sva**
- int **age**
- [gtime_t](#) **toe**
- [gtime_t](#) **tof**
- double **pos** [3]
- double **vel** [3]
- double **acc** [3]
- double **taun**
- double **gamn**
- double **dtaun**

10.120.1 Detailed Description

Definition at line 456 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.121 `Glonass_Gnav_Almanac` Class Reference

This class is a storage for the GLONASS SV ALMANAC data as described GLONASS ICD (Edition 5.1)

```
#include <glonass_gnav_almanac.h>
```

Public Member Functions

- [Glonass_Gnav_Almanac](#) ()=default
- `template<class Archive >`
void [serialize](#) (Archive &archive, const uint32_t version)

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the almanac data on disk file.

Public Attributes

- double [d_n_A](#) {}
Conventional number of satellite within GLONASS space segment [dimensionless].
- double [d_H_n_A](#) {}
Carrier frequency number of navigation RF signal transmitted by d_nA satellite as table 4.10 (0-31) [dimensionless].
- double [d_lambda_n_A](#) {}
Longitude of the first (within the d_NA day) ascending node of d_nA [radians].
- double [d_t_lambda_n_A](#) {}
Time of first ascending node passage [s].
- double [d_Delta_i_n_A](#) {}
Correction of the mean value of inclination of d_n_A satellite at instant t_lambda_n_A [radians].
- double [d_Delta_T_n_A](#) {}
Correction to the mean value of Draconian period of d_n_A satellite at instant t_lambda_n_A [s / orbital period].
- double [d_Delta_T_n_A_dot](#) {}
Rate of change of Draconian period of d_n_A satellite at instant t_lambda_n_A [s / orbital period²].
- double [d_epsilon_n_A](#) {}
Eccentricity of d_n_A satellite at instant t_lambda_n_A [dimensionless].
- double [d_omega_n_A](#) {}
Argument of perigee of d_n_A satellite at instant t_lambda_n_A [radians].
- double [d_M_n_A](#) {}
Type of satellite n_A [dimensionless].
- double [d_KP](#) {}
Notification on forthcoming leap second correction of UTC [dimensionless].
- double [d_tau_n_A](#) {}
Coarse value of d_n_A satellite time correction to GLONASS time at instant t_lambda_n_A[s].
- bool [d_C_n](#) {}
Generalized “unhealthy flag” of n_A satellite at instant of almanac upload [dimensionless].
- bool [d_I_n](#) {}
Health flag for nth satellite; In = 0 indicates the n-th satellite is healthy, In = 1 indicates malfunction of this nth satellite [dimensionless].
- int32_t [i_satellite_freq_channel](#) {}
SV Frequency [Channel](#) Number.
- uint32_t [i_satellite_PRN](#) {}
SV PRN Number, equivalent to slot number for compatibility with GPS.
- uint32_t [i_satellite_slot_number](#) {}
SV Slot Number.

10.121.1 Detailed Description

This class is a storage for the GLONASS SV ALMANAC data as described GLONASS ICD (Edition 5.1)

Note

Code added as part of GSoC 2017 program

See also

[GLONASS ICD](#)

Definition at line 34 of file glonass_gnav_almanac.h.

10.121.2 Constructor & Destructor Documentation

10.121.2.1 Glonass_Gnav_Almanac()

```
Glonass_Gnav_Almanac::Glonass_Gnav_Almanac ( ) [default]
```

Default constructor

10.121.3 Member Function Documentation

10.121.3.1 serialize()

```
template<class Archive >
void Glonass_Gnav_Almanac::serialize (
    Archive & archive,
    const uint32_t version ) [inline]
```

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the almanac data on disk file.

Definition at line 66 of file glonass_gnav_almanac.h.

References `d_C_n`, `d_Delta_i_n_A`, `d_Delta_T_n_A`, `d_Delta_T_n_A_dot`, `d_epsilon_n_A`, `d_H_n_A`, `d_KP`, `d_I_n`, `d_lambda_n_A`, `d_M_n_A`, `d_n_A`, `d_omega_n_A`, `d_t_lambda_n_A`, `d_tau_n_A`, `i_satellite_freq_channel`, `i_satellite_PRN`, and `i_satellite_slot_number`.

10.121.4 Member Data Documentation

10.121.4.1 d_C_n

```
bool Glonass_Gnav_Almanac::d_C_n {}
```

Generalized “unhealthy flag” of `n_A` satellite at instant of almanac upload [dimensionless].

Definition at line 54 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.2 d_Delta_i_n_A

```
double Glonass_Gnav_Almanac::d_Delta_i_n_A {}
```

Correction of the mean value of inclination of d_n_A satellite at instant t_lambda_n_A [radians].

Definition at line 46 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.3 d_Delta_T_n_A

```
double Glonass_Gnav_Almanac::d_Delta_T_n_A {}
```

Correction to the mean value of Draconian period of d_n_A satellite at instant t_lambda_n_A [s / orbital period].

Definition at line 47 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.4 d_Delta_T_n_A_dot

```
double Glonass_Gnav_Almanac::d_Delta_T_n_A_dot {}
```

Rate of change of Draconian period of d_n_A satellite at instant t_lambda_n_A [s / orbital period²].

Definition at line 48 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.5 d_epsilon_n_A

```
double Glonass_Gnav_Almanac::d_epsilon_n_A {}
```

Eccentricity of d_n_A satellite at instant t_lambda_n_A [dimensionless].

Definition at line 49 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.6 d_H_n_A

```
double Glonass_Gnav_Almanac::d_H_n_A {}
```

Carrier frequency number of navigation RF signal transmitted by d_nA satellite as table 4.10 (0-31) [dimensionless].

Definition at line 43 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.7 d_KP

```
double Glonass_Gnav_Almanac::d_KP {}
```

Notification on forthcoming leap second correction of UTC [dimensionless].

Definition at line 52 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.8 d_l_n

```
bool Glonass_Gnav_Almanac::d_l_n {}
```

Health flag for nth satellite; l_n = 0 indicates the n-th satellite is healthy, l_n = 1 indicates malfunction of this nth satellite [dimensionless].

Definition at line 55 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.9 d_lambda_n_A

```
double Glonass_Gnav_Almanac::d_lambda_n_A {}
```

Longitude of the first (within the d_NA day) ascending node of d_nA [radians].

Definition at line 44 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.10 d_M_n_A

```
double Glonass_Gnav_Almanac::d_M_n_A {}
```

Type of satellite n_A [dimensionless].

Definition at line 51 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.11 d_n_A

```
double Glonass_Gnav_Almanac::d_n_A {}
```

Conventional number of satellite within GLONASS space segment [dimensionless].

Definition at line 42 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.12 d_omega_n_A

```
double Glonass_Gnav_Almanac::d_omega_n_A {}
```

Argument of perigee of d_n_A satellite at instant t_lambdan_A [radians].

Definition at line 50 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.13 d_t_lambda_n_A

```
double Glonass_Gnav_Almanac::d_t_lambda_n_A {}
```

Time of first ascending node passage [s].

Definition at line 45 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.14 d_tau_n_A

```
double Glonass_Gnav_Almanac::d_tau_n_A {}
```

Coarse value of d_n_A satellite time correction to GLONASS time at instant t_lambdan_A[s].

Definition at line 53 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.15 i_satellite_freq_channel

```
int32_t Glonass_Gnav_Almanac::i_satellite_freq_channel {}
```

SV Frequency [Channel](#) Number.

Definition at line 58 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.16 i_satellite_PRN

```
uint32_t Glonass_Gnav_Almanac::i_satellite_PRN {}
```

SV PRN Number, equivalent to slot number for compatibility with GPS.

Definition at line 59 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

10.121.4.17 i_satellite_slot_number

```
uint32_t Glonass_Gnav_Almanac::i_satellite_slot_number {}
```

SV Slot Number.

Definition at line 60 of file glonass_gnav_almanac.h.

Referenced by `serialize()`.

The documentation for this class was generated from the following file:

- [glonass_gnav_almanac.h](#)

10.122 Glonass_Gnav_Ephemeris Class Reference

This class is a storage and orbital model functions for the GLONASS SV ephemeris data as described in GLONASS ICD (Edition 5.1)

```
#include <glonass_gnav_ephemeris.h>
```

Public Member Functions

- [Glonass_Gnav_Ephemeris](#) ()=default
- double [sv_clock_drift](#) (double transmitTime, double timeCorrUTC)
Sets (d_satClkDrift) and returns the clock drift in seconds according to the User Algorithm for SV Clock Correction (IS-GPS-200K, 20.3.3.3.3.1)
- boost::posix_time::ptime [compute_GLONASS_time](#) (double offset_time) const
Computes the GLONASS System Time and returns a boost::posix_time::ptime object \ param offset_time Is the start of day offset to compute the time.
- boost::posix_time::ptime [glot_to_utc](#) (const double offset_time, const double glot2utc_corr) const
Converts from GLONASST to UTC.
- void [glot_to_gpst](#) (double tod_offset, double glot2utc_corr, double glot2gpst_corr, int32_t *WN, double *T_{OW}) const
Converts from GLONASST to GPST.
- template<class Archive >
void [serialize](#) (Archive &archive, const uint32_t version)
Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

Public Attributes

- double [d_m](#) {}
String number within frame [dimensionless].
- double [d_t_k](#) {}
GLONASS Time (UTC(SU) + 3 h) referenced to the beginning of the frame within the current day [s].
- double [d_t_b](#) {}
Reference ephemeris relative time in GLONASS Time (UTC(SU) + 3 h). Index of a time interval within current day according to UTC(SU) + 03 hours 00 min. [s].
- double [d_M](#) {}
Type of satellite transmitting navigation signal [dimensionless].
- double [d_gamma_n](#) {}
Relative deviation of predicted carrier frequency value of n- satellite from nominal value at the instant tb [dimensionless].
- double [d_tau_n](#) {}
Correction to the nth satellite time (tn) relative to GLONASS time (te),.
- double [d_Xn](#) {}
Earth-fixed coordinate x of the satellite in PZ-90.02 coordinate system [km].
- double [d_Yn](#) {}
Earth-fixed coordinate y of the satellite in PZ-90.02 coordinate system [km].
- double [d_Zn](#) {}
Earth-fixed coordinate z of the satellite in PZ-90.02 coordinate system [km].
- double [d_VXn](#) {}
Earth-fixed velocity coordinate x of the satellite in PZ-90.02 coordinate system [km/s].
- double [d_VYn](#) {}

- Earth-fixed velocity coordinate y of the satellite in PZ-90.02 coordinate system [km/s].*

 - double [d_VZn](#) {}
- Earth-fixed velocity coordinate z of the satellite in PZ-90.02 coordinate system [km/s].*

 - double [d_AXn](#) {}
- Earth-fixed acceleration coordinate x of the satellite in PZ-90.02 coordinate system [km/s²].*

 - double [d_AYn](#) {}
- Earth-fixed acceleration coordinate y of the satellite in PZ-90.02 coordinate system [km/s²].*

 - double [d_AZn](#) {}
- Earth-fixed acceleration coordinate z of the satellite in PZ-90.02 coordinate system [km/s²].*

 - double [d_B_n](#) {}
- Health flag [dimensionless].*

 - double [d_P](#) {}
- Technological parameter of control segment, indication the satellite operation mode in respect of time parameters [dimensionless].*

 - double [d_N_T](#) {}
- Current date, calendar number of day within four-year interval starting from the 1-st of January in a leap year [days].*

 - double [d_F_T](#) {}
- Parameter that provides the predicted satellite user range accuracy at time tb [dimensionless].*

 - double [d_n](#) {}
- Index of the satellite transmitting given navigation signal. It corresponds to a slot number within GLONASS constellation.*

 - double [d_Delta_tau_n](#) {}
- Time difference between navigation RF signal transmitted in L2 sub- band and aviation RF signal transmitted in L1 sub-band by nth satellite. [dimensionless].*

 - double [d_E_n](#) {}
- Characterises "age" of a current information [days].*

 - double [d_P_1](#) {}
- Flag of the immediate data updating [minutes].*

 - bool [d_P_2](#) {}
- Flag of oddness ("1") or evenness ("0") of the value of (tb) [dimensionless].*

 - bool [d_P_3](#) {}
- Flag indicating a number of satellites for which almanac is transmitted within given frame: "1" corresponds to 5 satellites and "0" corresponds to 4 satellites [dimensionless].*

 - bool [d_P_4](#) {}
- Flag to show that ephemeris parameters are present. "1" indicates that updated ephemeris or frequency/time parameters have been uploaded by the control segment [dimensionless].*

 - bool [d_l3rd_n](#) {}
- Health flag for nth satellite; In = 0 indicates the n-th satellite is healthy, In = 1 indicates malfunction of this nth satellite [dimensionless].*

 - bool [d_l5th_n](#) {}
- Health flag for nth satellite; In = 0 indicates the n-th satellite is healthy, In = 1 indicates malfunction of this nth satellite [dimensionless].*

 - int32_t [i_satellite_freq_channel](#) {}
- SV Frequency [Channel](#) Number.*

 - uint32_t [i_satellite_PRN](#) {}
- SV PRN Number, equivalent to slot number for compatibility with GPS.*

 - uint32_t [i_satellite_slot_number](#) {}
- SV Slot Number.*

 - double [d_yr](#) = 1972.0
- Current year.*

 - double [d_satClkDrift](#) {}
- GLONASS clock error.*

- double [d_dtr](#) {}
relativistic clock correction term
- double [d_iodo](#) {}
Issue of data, ephemeris (Bit 0-6 of tb)
- double [d_tau_c](#) {}
GLONASST 2 UTC correction (todo) may be eliminated.
- double [d_TOW](#) {}
GLONASST IN GPST seconds of week.
- int32_t [d_WN](#) {}
GLONASST IN GPST week number of the start of frame.
- double [d_tod](#) {}
Time of Day since ephemeris where decoded.

10.122.1 Detailed Description

This class is a storage and orbital model functions for the GLONASS SV ephemeris data as described in GLONASS ICD (Edition 5.1)

Note

Code added as part of GSoC 2017 program

See also

[GLONASS ICD](#)

Definition at line 37 of file glonass_gnav_ephemeris.h.

10.122.2 Constructor & Destructor Documentation

10.122.2.1 Glonass_Gnav_Ephemeris()

```
Glonass_Gnav_Ephemeris::Glonass_Gnav_Ephemeris ( ) [default]
```

Default constructor

10.122.3 Member Function Documentation

10.122.3.1 compute_GLONASS_time()

```
boost::posix_time::ptime Glonass_Gnav_Ephemeris::compute_GLONASS_time (
    double offset_time ) const
```

Computes the GLONASS System Time and returns a boost::posix_time::ptime object \ param offset_time Is the start of day offset to compute the time.

10.122.3.2 glot_to_gpst()

```
void Glonass_Gnav_Ephemeris::glot_to_gpst (
    double tod_offset,
    double glot2utc_corr,
    double glot2gpst_corr,
    int32_t * WN,
    double * TOW ) const
```

Converts from GLONASST to GPST.

Converts from GLONASST to GPST in time of week (TOW) and week number (WN) format

Parameters

in	<i>tod_offset</i>	Is the start of day offset
in	<i>glot2utc_corr</i>	Correction from GLONASST to UTC
in	<i>glot2gpst_corr</i>	Correction from GLONASST to GPST
out	<i>WN</i>	Week Number, not in mod(1024) format
out	<i>TOW</i>	Time of Week in seconds of week

10.122.3.3 glot_to_utc()

```
boost::posix_time::ptime Glonass_Gnav_Ephemeris::glot_to_utc (
    const double offset_time,
    const double glot2utc_corr ) const
```

Converts from GLONASST to UTC.

The function simply adjust for the 6 hrs offset between GLONASST and UTC

Parameters

in	<i>offset_time</i>	Is the start of day offset
in	<i>glot2utc_corr</i>	Correction from GLONASST to UTC

Returns

UTC time as a boost::posix_time::ptime object

10.122.3.4 serialize()

```
template<class Archive >
void Glonass_Gnav_Ephemeris::serialize (
    Archive & archive,
    const uint32_t version ) [inline]
```

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

< SV PRN frequency channel number

< String number within frame [dimensionless]

< Time referenced to the beginning of the frame within the current day [hours, minutes, seconds]

< Index of a time interval within current day according to UTC(SU) + 03 hours 00 min. [minutes]

< Type of satellite transmitting navigation signal [dimensionless]

< Relative deviation of predicted carrier frequency value of n- satellite from nominal value at the instant tb [dimensionless]

< Correction to the nth satellite time (tn) relative to GLONASS time (te)

< Earth-fixed coordinate x of the satellite in PZ-90.02 coordinate system [km].

< Earth-fixed coordinate y of the satellite in PZ-90.02 coordinate system [km]

< Earth-fixed coordinate z of the satellite in PZ-90.02 coordinate system [km]

< Earth-fixed velocity coordinate x of the satellite in PZ-90.02 coordinate system [km/s]

< Earth-fixed velocity coordinate y of the satellite in PZ-90.02 coordinate system [km/s]

< Earth-fixed velocity coordinate z of the satellite in PZ-90.02 coordinate system [km/s]

< Earth-fixed acceleration coordinate x of the satellite in PZ-90.02 coordinate system [km/s²]

< Earth-fixed acceleration coordinate y of the satellite in PZ-90.02 coordinate system [km/s²]

< Earth-fixed acceleration coordinate z of the satellite in PZ-90.02 coordinate system [km/s²]

< Health flag [dimensionless]

< Technological parameter of control segment, indication the satellite operation mode in respect of time parameters [dimensionless]

< Current date, calendar number of day within four-year interval starting from the 1-st of January in a leap year [days]

< Parameter that provides the predicted satellite user range accuracy at time tb [dimensionless]

< Index of the satellite transmitting given navigation signal. It corresponds to a slot number within GLONASS constellation

< Time difference between navigation RF signal transmitted in L2 sub- band and aviation RF signal transmitted in L1 sub-band by nth satellite. [dimensionless]

< Characterises "age" of a current information [days]

< Flag of the immediate data updating.

< Flag of oddness ("1") or evenness ("0") of the value of (tb) [dimensionless]

< Flag indicating a number of satellites for which almanac is transmitted within given frame: "1" corresponds to 5 satellites and "0" corresponds to 4 satellites [dimensionless]

< Flag to show that ephemeris parameters are present. "1" indicates that updated ephemeris or frequency/time parameters have been uploaded by the control segment [dimensionless]

< Health flag for nth satellite; In = 0 indicates the n-th satellite is helthy, In = 1 indicates malfunction of this nth satellite [dimensionless]

< Health flag for nth satellite; In = 0 indicates the n-th satellite is helthy, In = 1 indicates malfunction of this nth satellite [dimensionless]

Definition at line 125 of file glonass_gnav_ephemeris.h.

References `d_AXn`, `d_AYn`, `d_AZn`, `d_B_n`, `d_Delta_tau_n`, `d_E_n`, `d_F_T`, `d_gamma_n`, `d_l3rd_n`, `d_l5th_n`, `d_m`, `d_M`, `d_n`, `d_N_T`, `d_P`, `d_P_1`, `d_P_2`, `d_P_3`, `d_P_4`, `d_t_b`, `d_t_k`, `d_tau_n`, `d_VXn`, `d_VYn`, `d_VZn`, `d_Xn`, `d_Yn`, `d_Zn`, `i_satellite_freq_channel`, `i_satellite_PRN`, and `i_satellite_slot_number`.

10.122.3.5 `sv_clock_drift()`

```
double Glonass_Gnav_Ephemeris::sv_clock_drift (
    double transmitTime,
    double timeCorrUTC )
```

Sets (`d_satClkDrift`) and returns the clock drift in seconds according to the User Algorithm for SV Clock Correction (IS-GPS-200K, 20.3.3.3.3.1)

10.122.4 Member Data Documentation

10.122.4.1 `d_AXn`

```
double Glonass_Gnav_Ephemeris::d_AXn {}
```

Earth-fixed acceleration coordinate x of the satellite in PZ-90.02 coordinate system [km/s²].

Definition at line 57 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.2 d_AYn

```
double Glonass_Gnav_Ephemeris::d_AYn {}
```

Earth-fixed acceleration coordinate y of the satellite in PZ-90.02 coordinate system [km/s²].

Definition at line 58 of file glonass_gnav_ephemeris.h.

Referenced by serialize().

10.122.4.3 d_AZn

```
double Glonass_Gnav_Ephemeris::d_AZn {}
```

Earth-fixed acceleration coordinate z of the satellite in PZ-90.02 coordinate system [km/s²].

Definition at line 59 of file glonass_gnav_ephemeris.h.

Referenced by serialize().

10.122.4.4 d_B_n

```
double Glonass_Gnav_Ephemeris::d_B_n {}
```

Health flag [dimensionless].

Definition at line 60 of file glonass_gnav_ephemeris.h.

Referenced by serialize().

10.122.4.5 d_Delta_tau_n

```
double Glonass_Gnav_Ephemeris::d_Delta_tau_n {}
```

Time difference between navigation RF signal transmitted in L2 sub- band and aviation RF signal transmitted in L1 sub-band by nth satellite. [dimensionless].

Definition at line 65 of file glonass_gnav_ephemeris.h.

Referenced by serialize().

10.122.4.6 d_dtr

```
double Glonass_Gnav_Ephemeris::d_dtr {}
```

relativistic clock correction term

Definition at line 81 of file glonass_gnav_ephemeris.h.

10.122.4.7 d_E_n

```
double Glonass_Gnav_Ephemeris::d_E_n {}
```

Characterises "age" of a current information [days].

Definition at line 66 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.8 d_F_T

```
double Glonass_Gnav_Ephemeris::d_F_T {}
```

Parameter that provides the predicted satellite user range accuracy at time `tb` [dimensionless].

Definition at line 63 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.9 d_gamma_n

```
double Glonass_Gnav_Ephemeris::d_gamma_n {}
```

Relative deviation of predicted carrier frequency value of `n`- satellite from nominal value at the instant `tb` [dimensionless].

Definition at line 49 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.10 d_iode

```
double Glonass_Gnav_Ephemeris::d_iode {}
```

Issue of data, ephemeris (Bit 0-6 of tb)

Definition at line 82 of file glonass_gnav_ephemeris.h.

10.122.4.11 d_l3rd_n

```
bool Glonass_Gnav_Ephemeris::d_l3rd_n {}
```

Health flag for nth satellite; ln = 0 indicates the n-th satellite is healthy, ln = 1 indicates malfunction of this nth satellite [dimensionless].

Definition at line 71 of file glonass_gnav_ephemeris.h.

Referenced by serialize().

10.122.4.12 d_l5th_n

```
bool Glonass_Gnav_Ephemeris::d_l5th_n {}
```

Health flag for nth satellite; ln = 0 indicates the n-th satellite is healthy, ln = 1 indicates malfunction of this nth satellite [dimensionless].

Definition at line 72 of file glonass_gnav_ephemeris.h.

Referenced by serialize().

10.122.4.13 d_m

```
double Glonass_Gnav_Ephemeris::d_m {}
```

String number within frame [dimensionless].

Definition at line 45 of file glonass_gnav_ephemeris.h.

Referenced by serialize().

10.122.4.14 d_M

```
double Glonass_Gnav_Ephemeris::d_M {}
```

Type of satellite transmitting navigation signal [dimensionless].

Definition at line 48 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.15 d_n

```
double Glonass_Gnav_Ephemeris::d_n {}
```

Index of the satellite transmitting given navigation signal. It corresponds to a slot number within GLONASS constellation.

Definition at line 64 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.16 d_N_T

```
double Glonass_Gnav_Ephemeris::d_N_T {}
```

Current date, calendar number of day within four-year interval starting from the 1-st of January in a leap year [days].

Definition at line 62 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.17 d_P

```
double Glonass_Gnav_Ephemeris::d_P {}
```

Technological parameter of control segment, indication the satellite operation mode in respect of time parameters [dimensionless].

Definition at line 61 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.18 d_P_1

```
double Glonass_Gnav_Ephemeris::d_P_1 {}
```

Flag of the immediate data updating [minutes].

Definition at line 67 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.19 d_P_2

```
bool Glonass_Gnav_Ephemeris::d_P_2 {}
```

Flag of oddness ("1") or evenness ("0") of the value of (tb) [dimensionless].

Definition at line 68 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.20 d_P_3

```
bool Glonass_Gnav_Ephemeris::d_P_3 {}
```

Flag indicating a number of satellites for which almanac is transmitted within given frame: "1" corresponds to 5 satellites and "0" corresponds to 4 satellites [dimensionless].

Definition at line 69 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.21 d_P_4

```
bool Glonass_Gnav_Ephemeris::d_P_4 {}
```

Flag to show that ephemeris parameters are present. "1" indicates that updated ephemeris or frequency/time parameters have been uploaded by the control segment [dimensionless].

Definition at line 70 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.22 d_satClkDrift

```
double Glonass_Gnav_Ephemeris::d_satClkDrift {}
```

GLONASS clock error.

Definition at line 80 of file glonass_gnav_ephemeris.h.

10.122.4.23 d_t_b

```
double Glonass_Gnav_Ephemeris::d_t_b {}
```

Reference ephemeris relative time in GLONASS Time (UTC(SU) + 3 h). Index of a time interval within current day according to UTC(SU) + 03 hours 00 min. [s].

Definition at line 47 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.24 d_t_k

```
double Glonass_Gnav_Ephemeris::d_t_k {}
```

GLONASS Time (UTC(SU) + 3 h) referenced to the beginning of the frame within the current day [s].

Definition at line 46 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.25 d_tau_c

```
double Glonass_Gnav_Ephemeris::d_tau_c {}
```

GLONASST 2 UTC correction (todo) may be eliminated.

Definition at line 83 of file glonass_gnav_ephemeris.h.

10.122.4.26 d_tau_n

```
double Glonass_Gnav_Ephemeris::d_tau_n {}
```

Correction to the nth satellite time (tn) relative to GLONASS time (te),.

Definition at line 50 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.27 d_tod

```
double Glonass_Gnav_Ephemeris::d_tod {}
```

Time of Day since ephemeris where decoded.

Definition at line 86 of file glonass_gnav_ephemeris.h.

10.122.4.28 d_TOW

```
double Glonass_Gnav_Ephemeris::d_TOW {}
```

GLONASST IN GPST seconds of week.

Definition at line 84 of file glonass_gnav_ephemeris.h.

10.122.4.29 d_VXn

```
double Glonass_Gnav_Ephemeris::d_VXn {}
```

Earth-fixed velocity coordinate x of the satellite in PZ-90.02 coordinate system [km/s].

Definition at line 54 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.30 d_VYn

```
double Glonass_Gnav_Ephemeris::d_VYn {}
```

Earth-fixed velocity coordinate y of the satellite in PZ-90.02 coordinate system [km/s].

Definition at line 55 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.31 d_VZn

```
double Glonass_Gnav_Ephemeris::d_VZn {}
```

Earth-fixed velocity coordinate z of the satellite in PZ-90.02 coordinate system [km/s].

Definition at line 56 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.32 d_WN

```
int32_t Glonass_Gnav_Ephemeris::d_WN {}
```

GLONASST IN GPST week number of the start of frame.

Definition at line 85 of file glonass_gnav_ephemeris.h.

10.122.4.33 d_Xn

```
double Glonass_Gnav_Ephemeris::d_Xn {}
```

Earth-fixed coordinate x of the satellite in PZ-90.02 coordinate system [km].

Definition at line 51 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.34 d_Yn

```
double Glonass_Gnav_Ephemeris::d_Yn {}
```

Earth-fixed coordinate y of the satellite in PZ-90.02 coordinate system [km].

Definition at line 52 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.35 d_yr

```
double Glonass_Gnav_Ephemeris::d_yr = 1972.0
```

Current year.

Definition at line 79 of file glonass_gnav_ephemeris.h.

10.122.4.36 d_Zn

```
double Glonass_Gnav_Ephemeris::d_Zn {}
```

Earth-fixed coordinate z of the satellite in PZ-90.02 coordinate system [km].

Definition at line 53 of file glonass_gnav_ephemeris.h.

Referenced by `serialize()`.

10.122.4.37 i_satellite_freq_channel

```
int32_t Glonass_Gnav_Ephemeris::i_satellite_freq_channel {}
```

SV Frequency [Channel](#) Number.

Definition at line 76 of file `glonass_gnav_ephemeris.h`.

Referenced by `serialize()`.

10.122.4.38 i_satellite_PRN

```
uint32_t Glonass_Gnav_Ephemeris::i_satellite_PRN {}
```

SV PRN Number, equivalent to slot number for compatibility with GPS.

Definition at line 77 of file `glonass_gnav_ephemeris.h`.

Referenced by `serialize()`.

10.122.4.39 i_satellite_slot_number

```
uint32_t Glonass_Gnav_Ephemeris::i_satellite_slot_number {}
```

SV Slot Number.

Definition at line 78 of file `glonass_gnav_ephemeris.h`.

Referenced by `serialize()`.

The documentation for this class was generated from the following file:

- [glonass_gnav_ephemeris.h](#)

10.123 Glonass_Gnav_Navigation_Message Class Reference

This class decodes a GLONASS GNAV Data message as described in GLONASS ICD (Edition 5.1)

```
#include <glonass_gnav_navigation_message.h>
```

Public Member Functions

- [Glonass_Gnav_Navigation_Message](#) ()
- bool [CRC_test](#) (std::bitset< [GLONASS_GNAV_STRING_BITS](#) > bits) const
Compute CRC for GLONASS GNAV strings.
- uint32_t [get_frame_number](#) (uint32_t satellite_slot_number)
Computes the frame number being decoded given the satellite slot number.
- [Glonass_Gnav_Ephemeris](#) [get_ephemeris](#) () const
Obtain a GLONASS GNAV SV Ephemeris class filled with current SV data.
- [Glonass_Gnav_Utc_Model](#) [get_utc_model](#) () const
Obtain a GLONASS GNAV UTC model parameters class filled with current SV data.
- [Glonass_Gnav_Almanac](#) [get_almanac](#) (uint32_t satellite_slot_number) const
Returns a [Glonass_Gnav_Almanac](#) object filled with the latest navigation data received.
- bool [have_new_ephemeris](#) ()
Returns true if a new [Glonass_Gnav_Ephemeris](#) object has arrived.
- bool [have_new_utc_model](#) ()
Returns true if new [Glonass_Gnav_Utc_Model](#) object has arrived.
- bool [have_new_almanac](#) ()
Returns true if new [Glonass_Gnav_Almanac](#) object has arrived.
- int32_t [string_decoder](#) (const std::string &frame_string)
Decodes the GLONASS GNAV string.
- bool [get_flag_CRC_test](#) () const
- void [set_rf_link](#) (int32_t rf_link)
- uint32_t [get_alm_satellite_slot_number](#) () const
- bool [get_flag_update_slot_number](#) () const
- void [set_flag_update_slot_number](#) (bool flag_slot)
- bool [get_flag_TOW_new](#) () const
- void [set_flag_TOW_new](#) (bool tow_new)
- bool [is_flag_TOW_set](#) () const
- void [set_flag_ephemeris_str_1](#) (bool ephemeris_str_1)
- void [set_flag_ephemeris_str_2](#) (bool ephemeris_str_2)
- void [set_flag_ephemeris_str_3](#) (bool ephemeris_str_3)
- void [set_flag_ephemeris_str_4](#) (bool ephemeris_str_4)

10.123.1 Detailed Description

This class decodes a GLONASS GNAV Data message as described in GLONASS ICD (Edition 5.1)

Note

Code added as part of GSoC 2017 program

See also

[GLONASS ICD](#)

Definition at line 43 of file [glonass_gnav_navigation_message.h](#).

10.123.2 Constructor & Destructor Documentation

10.123.2.1 Glonass_Gnav_Navigation_Message()

```
Glonass_Gnav_Navigation_Message::Glonass_Gnav_Navigation_Message ( )
```

Default constructor

10.123.3 Member Function Documentation

10.123.3.1 CRC_test()

```
bool Glonass_Gnav_Navigation_Message::CRC_test (
    std::bitset< GLONASS_GNAV_STRING_BITS > bits ) const
```

Compute CRC for GLONASS GNAV strings.

Parameters

<i>bits</i>	Bits of the string message where to compute CRC
-------------	---

10.123.3.2 get_almanac()

```
Glonass_Gnav_Almanac Glonass_Gnav_Navigation_Message::get_almanac (
    uint32_t satellite_slot_number ) const
```

Returns a [Glonass_Gnav_Almanac](#) object filled with the latest navigation data received.

Parameters

<i>satellite_slot_number</i>	Slot number identifier for the satellite
------------------------------	--

Returns

Returns the [Glonass_Gnav_Almanac](#) object for the input slot number

10.123.3.3 get_ephemeris()

```
Glonass_Gnav_Ephemeris Glonass_Gnav_Navigation_Message::get_ephemeris ( ) const [inline]
```

Obtain a GLONASS GNAV SV Ephemeris class filled with current SV data.

Definition at line 67 of file `glonass_gnav_navigation_message.h`.

10.123.3.4 get_frame_number()

```
uint32_t Glonass_Gnav_Navigation_Message::get_frame_number (
    uint32_t satellite_slot_number )
```

Computes the frame number being decoded given the satellite slot number.

Parameters

<i>satellite_slot_number</i>	[in] Satellite slot number identifier
------------------------------	---------------------------------------

Returns

Frame number being decoded, 0 if operation was not successful.

10.123.3.5 get_utc_model()

```
Glonass_Gnav_Utc_Model Glonass_Gnav_Navigation_Message::get_utc_model ( ) const [inline]
```

Obtain a GLONASS GNAV UTC model parameters class filled with current SV data.

Definition at line 75 of file glonass_gnav_navigation_message.h.

10.123.3.6 have_new_almanac()

```
bool Glonass_Gnav_Navigation_Message::have_new_almanac ( )
```

Returns true if new [Glonass_Gnav_Almanac](#) object has arrived.

10.123.3.7 have_new_ephemeris()

```
bool Glonass_Gnav_Navigation_Message::have_new_ephemeris ( )
```

Returns true if a new [Glonass_Gnav_Ephemeris](#) object has arrived.

10.123.3.8 have_new_utc_model()

```
bool Glonass_Gnav_Navigation_Message::have_new_utc_model ( )
```

Returns true if new [Glonass_Gnav_Utc_Model](#) object has arrived.

10.123.3.9 string_decoder()

```
int32_t Glonass_Gnav_Navigation_Message::string_decoder (
    const std::string & frame_string )
```

Decodes the GLONASS GNAV string.

Parameters

<code>frame_string</code>	[in] is the string message within the parsed frame
---------------------------	--

Returns

Returns the ID of the decoded string

The documentation for this class was generated from the following file:

- [glonass_gnav_navigation_message.h](#)

10.124 Glonass_Gnav_Utc_Model Class Reference

This class is a storage for the GLONASS GNAV UTC MODEL data as described in GLONASS ICD (Edition 5.1)

```
#include <glonass_gnav_utc_model.h>
```

Public Member Functions

- [Glonass_Gnav_Utc_Model](#) ()=default
- double [utc_time](#) (double glonass_time_corrected)
Computes the Coordinated Universal Time (UTC) and returns it in s
- template<class Archive >
void [serialize](#) (Archive &archive, const uint32_t version)
Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the almanac data on disk file.

Public Attributes

- bool **valid** {}
- double [d_tau_c](#) {}
GLONASS time scale correction to UTC(SU) time. [s].
- double [d_tau_gps](#) {}
Correction to GPS time to GLONASS time [day].
- double [d_N_4](#) {}
Four year interval number starting from 1996 [4 year interval].
- double [d_N_A](#) {}
Calendar day number within the four-year period beginning since the leap year for Almanac data [days].
- double [d_B1](#) {}
Coefficient to determine DeltaUT1 [s].
- double [d_B2](#) {}
Coefficient to determine DeltaUT1 [s/msd].

10.124.1 Detailed Description

This class is a storage for the GLONASS GNAV UTC MODEL data as described in GLONASS ICD (Edition 5.1)

Note

Code added as part of GSoC 2017 program

See also

[GLONASS ICD](#)

Definition at line 34 of file glonass_gnav_utc_model.h.

10.124.2 Constructor & Destructor Documentation

10.124.2.1 Glonass_Gnav_Utc_Model()

```
Glonass_Gnav_Utc_Model::Glonass_Gnav_Utc_Model ( ) [default]
```

Default constructor

10.124.3 Member Function Documentation

10.124.3.1 serialize()

```
template<class Archive >
void Glonass_Gnav_Utc_Model::serialize (
    Archive & archive,
    const uint32_t version ) [inline]
```

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the almanac data on disk file.

Definition at line 61 of file glonass_gnav_utc_model.h.

References `d_B1`, `d_B2`, `d_N_4`, `d_N_A`, `d_tau_c`, and `d_tau_gps`.

10.124.3.2 `utc_time()`

```
double Glonass_Gnav_Utc_Model::utc_time (
    double glonass_time_corrected )
```

Computes the Coordinated Universal Time (UTC) and returns it in [s](#)

10.124.4 Member Data Documentation

10.124.4.1 `d_B1`

```
double Glonass_Gnav_Utc_Model::d_B1 {}
```

Coefficient to determine DeltaUT1 [s].

Definition at line 48 of file `glonass_gnav_utc_model.h`.

Referenced by `serialize()`.

10.124.4.2 `d_B2`

```
double Glonass_Gnav_Utc_Model::d_B2 {}
```

Coefficient to determine DeltaUT1 [s/msd].

Definition at line 49 of file `glonass_gnav_utc_model.h`.

Referenced by `serialize()`.

10.124.4.3 `d_N_4`

```
double Glonass_Gnav_Utc_Model::d_N_4 {}
```

Four year interval number starting from 1996 [4 year interval].

Definition at line 46 of file `glonass_gnav_utc_model.h`.

Referenced by `serialize()`.

10.124.4.4 d_N_A

```
double Glonass_Gnav_Utc_Model::d_N_A {}
```

Calendar day number within the four-year period beginning since the leap year for Almanac data [days].

Definition at line 47 of file glonass_gnav_utc_model.h.

Referenced by `serialize()`.

10.124.4.5 d_tau_c

```
double Glonass_Gnav_Utc_Model::d_tau_c {}
```

GLONASS time scale correction to UTC(SU) time. [s].

Definition at line 44 of file glonass_gnav_utc_model.h.

Referenced by `serialize()`.

10.124.4.6 d_tau_gps

```
double Glonass_Gnav_Utc_Model::d_tau_gps {}
```

Correction to GPS time to GLONASS time [day].

Definition at line 45 of file glonass_gnav_utc_model.h.

Referenced by `serialize()`.

The documentation for this class was generated from the following file:

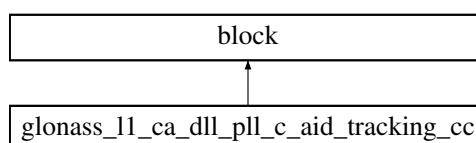
- [glonass_gnav_utc_model.h](#)

10.125 glonass_l1_ca_dll_pll_c_aid_tracking_cc Class Reference

This class implements a DLL + PLL tracking loop block.

```
#include <glonass_l1_ca_dll_pll_c_aid_tracking_cc.h>
```

Inheritance diagram for `glonass_l1_ca_dll_pll_c_aid_tracking_cc`:



Public Member Functions

- void **set_channel** (uint32_t channel)
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro)
- void **start_tracking** ()
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)

Friends

- glonass_l1_ca_dll_pll_c_aid_tracking_cc_sptr **glonass_l1_ca_dll_pll_c_aid_make_tracking_cc** (int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float pll_bw_hz, float dll_bw_hz, float pll_bw_narrow_hz, float dll_bw_narrow_hz, int32_t extend_correlation_ms, float early_late_space_chips)

10.125.1 Detailed Description

This class implements a DLL + PLL tracking loop block.

Definition at line 74 of file `glonass_l1_ca_dll_pll_c_aid_tracking_cc.h`.

The documentation for this class was generated from the following file:

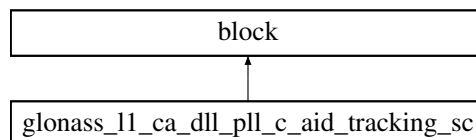
- [glonass_l1_ca_dll_pll_c_aid_tracking_cc.h](#)

10.126 glonass_l1_ca_dll_pll_c_aid_tracking_sc Class Reference

This class implements a DLL + PLL tracking loop block.

```
#include <glonass_l1_ca_dll_pll_c_aid_tracking_sc.h>
```

Inheritance diagram for `glonass_l1_ca_dll_pll_c_aid_tracking_sc`:



Public Member Functions

- void **set_channel** (uint32_t channel)
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro)
- void **start_tracking** ()
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)

Friends

- `glonass_l1_ca_dll_pll_c_aid_tracking_sc_sptr glonass_l1_ca_dll_pll_c_aid_make_tracking_sc` (int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float pll_bw_hz, float dll_bw_hz, float pll_bw_narrow_hz, float dll_bw_narrow_hz, int32_t extend_correlation_ms, float early_late_space_chips)

10.126.1 Detailed Description

This class implements a DLL + PLL tracking loop block.

Definition at line 73 of file `glonass_l1_ca_dll_pll_c_aid_tracking_sc.h`.

The documentation for this class was generated from the following file:

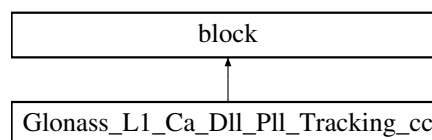
- [glonass_l1_ca_dll_pll_c_aid_tracking_sc.h](#)

10.127 Glonass_L1_Ca_Dll_Pll_Tracking_cc Class Reference

This class implements a DLL + PLL tracking loop block.

```
#include <glonass_l1_ca_dll_pll_tracking_cc.h>
```

Inheritance diagram for `Glonass_L1_Ca_Dll_Pll_Tracking_cc`:



Public Member Functions

- void **set_channel** (uint32_t channel)
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro)
- void **start_tracking** ()
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)

Friends

- `glonass_l1_ca_dll_pll_tracking_cc_sptr glonass_l1_ca_dll_pll_make_tracking_cc` (int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float pll_bw_hz, float dll_bw_hz, float early_late_space_chips)

10.127.1 Detailed Description

This class implements a DLL + PLL tracking loop block.

Definition at line 67 of file `glonass_l1_ca_dll_pll_tracking_cc.h`.

The documentation for this class was generated from the following file:

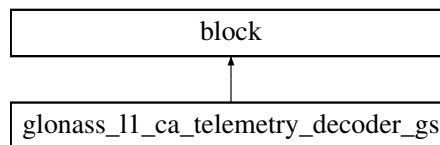
- [glonass_l1_ca_dll_pll_tracking_cc.h](#)

10.128 glonass_l1_ca_telemetry_decoder_gs Class Reference

This class implements a block that decodes the GNAV data defined in GLONASS ICD v5.1.

```
#include <glonass_l1_ca_telemetry_decoder_gs.h>
```

Inheritance diagram for `glonass_l1_ca_telemetry_decoder_gs`:



Public Member Functions

- [~glonass_l1_ca_telemetry_decoder_gs](#) ()
Class destructor.
- void [set_satellite](#) (const [Gnss_Satellite](#) &satellite)
Set satellite PRN.
- void [set_channel](#) (int32_t channel)
Set receiver's channel.
- void **reset** ()
- int [general_work](#) (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
This is where all signal processing takes place.

Friends

- `glonass_l1_ca_telemetry_decoder_gs_sptr` **glonass_l1_ca_make_telemetry_decoder_gs** (const [Gnss_Satellite](#) &satellite, bool dump)

10.128.1 Detailed Description

This class implements a block that decodes the GNAV data defined in GLONASS ICD v5.1.

Note

Code added as part of GSoC 2017 program

See also

[GLONASS ICD](#)

Definition at line 61 of file `glonass_l1_ca_telemetry_decoder_gs.h`.

10.128.2 Constructor & Destructor Documentation

10.128.2.1 `~glonass_l1_ca_telemetry_decoder_gs()`

```
glonass_l1_ca_telemetry_decoder_gs::~~glonass_l1_ca_telemetry_decoder_gs ( )
```

Class destructor.

10.128.3 Member Function Documentation

10.128.3.1 `general_work()`

```
int glonass_l1_ca_telemetry_decoder_gs::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

This is where all signal processing takes place.

10.128.3.2 `set_channel()`

```
void glonass_l1_ca_telemetry_decoder_gs::set_channel (
    int32_t channel )
```

Set receiver's channel.

10.128.3.3 set_satellite()

```
void glonass_l1_ca_telemetry_decoder_gs::set_satellite (
    const Gnss_Satellite & satellite )
```

Set satellite PRN.

The documentation for this class was generated from the following file:

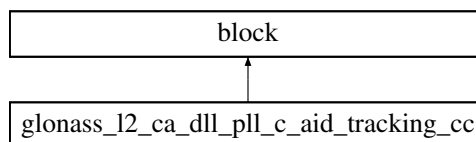
- [glonass_l1_ca_telemetry_decoder_gs.h](#)

10.129 glonass_l2_ca_dll_pll_c_aid_tracking_cc Class Reference

This class implements a DLL + PLL tracking loop block.

```
#include <glonass_l2_ca_dll_pll_c_aid_tracking_cc.h>
```

Inheritance diagram for glonass_l2_ca_dll_pll_c_aid_tracking_cc:



Public Member Functions

- void **set_channel** (uint32_t channel)
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro)
- void **start_tracking** ()
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)

Friends

- glonass_l2_ca_dll_pll_c_aid_tracking_cc_sptr **glonass_l2_ca_dll_pll_c_aid_make_tracking_cc** (int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float pll_bw_hz, float dll_bw_hz, float pll_bw_narrow_hz, float dll_bw_narrow_hz, int32_t extend_correlation_ms, float early_late_space_chips)

10.129.1 Detailed Description

This class implements a DLL + PLL tracking loop block.

Definition at line 70 of file [glonass_l2_ca_dll_pll_c_aid_tracking_cc.h](#).

The documentation for this class was generated from the following file:

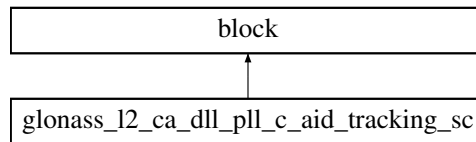
- [glonass_l2_ca_dll_pll_c_aid_tracking_cc.h](#)

10.130 glonass_l2_ca_dll_pll_c_aid_tracking_sc Class Reference

This class implements a DLL + PLL tracking loop block.

```
#include <glonass_l2_ca_dll_pll_c_aid_tracking_sc.h>
```

Inheritance diagram for glonass_l2_ca_dll_pll_c_aid_tracking_sc:



Public Member Functions

- void **set_channel** (uint32_t channel)
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro)
- void **start_tracking** ()
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)

Friends

- glonass_l2_ca_dll_pll_c_aid_tracking_sc_sptr **glonass_l2_ca_dll_pll_c_aid_make_tracking_sc** (int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float pll_bw_hz, float dll_bw_hz, float pll_bw_narrow_hz, float dll_bw_narrow_hz, int32_t extend_correlation_ms, float early_late_space_chips)

10.130.1 Detailed Description

This class implements a DLL + PLL tracking loop block.

Definition at line 71 of file glonass_l2_ca_dll_pll_c_aid_tracking_sc.h.

The documentation for this class was generated from the following file:

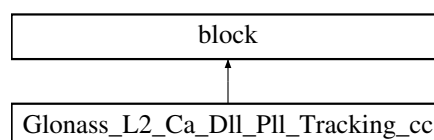
- [glonass_l2_ca_dll_pll_c_aid_tracking_sc.h](#)

10.131 Glonass_L2_Ca_Dll_Pll_Tracking_cc Class Reference

This class implements a DLL + PLL tracking loop block.

```
#include <glonass_l2_ca_dll_pll_tracking_cc.h>
```

Inheritance diagram for Glonass_L2_Ca_Dll_Pll_Tracking_cc:



Public Member Functions

- void **set_channel** (uint32_t channel)
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro)
- void **start_tracking** ()
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)

Friends

- [glonass_l2_ca_dll_pll_tracking_cc_sptr](#) **glonass_l2_ca_dll_pll_make_tracking_cc** (int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float pll_bw_hz, float dll_bw_hz, float early_late_space_chips)

10.131.1 Detailed Description

This class implements a DLL + PLL tracking loop block.

Definition at line 65 of file [glonass_l2_ca_dll_pll_tracking_cc.h](#).

The documentation for this class was generated from the following file:

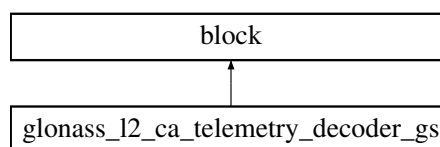
- [glonass_l2_ca_dll_pll_tracking_cc.h](#)

10.132 glonass_l2_ca_telemetry_decoder_gs Class Reference

This class implements a block that decodes the GNAV data defined in GLONASS ICD v5.1.

```
#include <glonass_l2_ca_telemetry_decoder_gs.h>
```

Inheritance diagram for [glonass_l2_ca_telemetry_decoder_gs](#):



Public Member Functions

- [~glonass_l2_ca_telemetry_decoder_gs](#) ()
Class destructor.
- void **set_satellite** (const [Gnss_Satellite](#) &satellite)
Set satellite PRN.
- void **set_channel** (int32_t channel)
Set receiver's channel.
- void **reset** ()
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
This is where all signal processing takes place.

Friends

- `glonass_l2_ca_telemetry_decoder_gs_sptr` **`glonass_l2_ca_make_telemetry_decoder_gs`** (const [Gnss_Satellite](#) &satellite, bool dump)

10.132.1 Detailed Description

This class implements a block that decodes the GNAV data defined in GLONASS ICD v5.1.

See also

[GLONASS ICD](#)

Definition at line 58 of file `glonass_l2_ca_telemetry_decoder_gs.h`.

10.132.2 Constructor & Destructor Documentation

10.132.2.1 `~glonass_l2_ca_telemetry_decoder_gs()`

```
glonass_l2_ca_telemetry_decoder_gs::~glonass_l2_ca_telemetry_decoder_gs ( )
```

Class destructor.

10.132.3 Member Function Documentation

10.132.3.1 `general_work()`

```
int glonass_l2_ca_telemetry_decoder_gs::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

This is where all signal processing takes place.

10.132.3.2 `set_channel()`

```
void glonass_l2_ca_telemetry_decoder_gs::set_channel (
    int32_t channel )
```

Set receiver's channel.

10.132.3.3 set_satellite()

```
void glonass_l2_ca_telemetry_decoder_gs::set_satellite (
    const Gnss_Satellite & satellite )
```

Set satellite PRN.

The documentation for this class was generated from the following file:

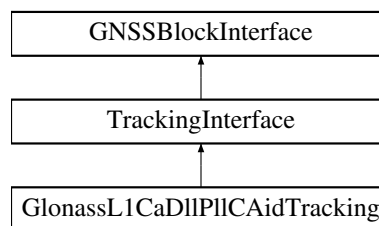
- [glonass_l2_ca_telemetry_decoder_gs.h](#)

10.133 GlonassL1CaDllPllCAidTracking Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <glonass_l1_ca_dll_pll_c_aid_tracking.h>
```

Inheritance diagram for GlonassL1CaDllPllCAidTracking:



Public Member Functions

- **GlonassL1CaDllPllCAidTracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "GLONASS_L1_CA_DLL_PLL_C_Aid_Tracking".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override
 - Set tracking channel unique ID.*
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.*
- void **start_tracking** () override
- void **stop_tracking** () override
 - Stop running tracking.*

10.133.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 42 of file `glonass_l1_ca_dll_pll_c_aid_tracking.h`.

10.133.2 Member Function Documentation

10.133.2.1 `implementation()`

```
std::string GlonassL1CaDllPllCAidTracking::implementation ( ) [inline], [override], [virtual]
```

Returns "GLONASS_L1_CA_DLL_PLL_C_Aid_Tracking".

Implements [GNSSBlockInterface](#).

Definition at line 59 of file `glonass_l1_ca_dll_pll_c_aid_tracking.h`.

10.133.2.2 `set_channel()`

```
void GlonassL1CaDllPllCAidTracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.133.2.3 `set_gnss_synchro()`

```
void GlonassL1CaDllPllCAidTracking::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.133.2.4 stop_tracking()

```
void GlonassL1CaDllPllCAidTracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

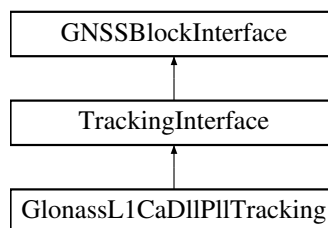
- [glonass_l1_ca_dll_pll_c_aid_tracking.h](#)

10.134 GlonassL1CaDllPllTracking Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <glonass_l1_ca_dll_pll_tracking.h>
```

Inheritance diagram for GlonassL1CaDllPllTracking:



Public Member Functions

- **GlonassL1CaDllPllTracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override

Returns "GLONASS_L1_CA_DLL_PLL_Tracking".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override

Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override

Stop running tracking.

10.134.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 40 of file `glonass_l1_ca_dll_pll_tracking.h`.

10.134.2 Member Function Documentation

10.134.2.1 `implementation()`

```
std::string GlonassL1CaDllPllTracking::implementation ( ) [inline], [override], [virtual]
```

Returns "GLONASS_L1_CA_DLL_PLL_Tracking".

Implements [GNSSBlockInterface](#).

Definition at line 57 of file `glonass_l1_ca_dll_pll_tracking.h`.

10.134.2.2 `set_channel()`

```
void GlonassL1CaDllPllTracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.134.2.3 `set_gnss_synchro()`

```
void GlonassL1CaDllPllTracking::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.134.2.4 stop_tracking()

```
void GlonassL1CaDllPllTracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

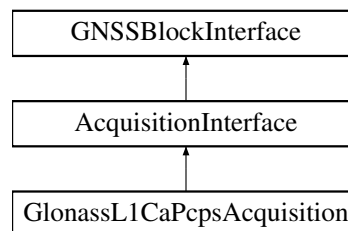
- [glonass_l1_ca_dll_pll_tracking.h](#)

10.135 GlonassL1CaPcpsAcquisition Class Reference

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

```
#include <glonass_l1_ca_pcps_acquisition.h>
```

Inheritance diagram for GlonassL1CaPcpsAcquisition:



Public Member Functions

- **GlonassL1CaPcpsAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "GLONASS_L1_CA_PCPS_Acquisition".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **set_channel** (unsigned int channel) override
Set acquisition channel unique ID.
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
Set channel fsm associated to this acquisition instance.
- void **set_threshold** (float threshold) override
Set statistics threshold of PCPS algorithm.
- void **set_doppler_max** (unsigned int doppler_max) override

- *Set maximum Doppler off grid search.*
- void [set_doppler_step](#) (unsigned int doppler_step) override
- *Set Doppler steps for the grid search.*
- void [init](#) () override
- *Initializes acquisition algorithm.*
- void [set_local_code](#) () override
- *Sets local code for GPS L1/CA PCPS acquisition algorithm.*
- signed int [mag](#) () override
- *Returns the maximum peak of grid search.*
- void [reset](#) () override
- *Restart acquisition algorithm.*
- void [set_state](#) (int state) override
- *If state = 1, it forces the block to start acquiring from the first sample.*
- void [stop_acquisition](#) () override
- *Stop running acquisition.*
- void [set_resampler_latency](#) (uint32_t latency_samples __attribute__((unused))) override

10.135.1 Detailed Description

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Definition at line 42 of file `glonass_l1_ca_pcps_acquisition.h`.

10.135.2 Member Function Documentation

10.135.2.1 implementation()

```
std::string GlonassL1CaPcpsAcquisition::implementation ( ) [inline], [override], [virtual]
```

Returns "GLONASS_L1_CA_PCPS_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 61 of file `glonass_l1_ca_pcps_acquisition.h`.

10.135.2.2 init()

```
void GlonassL1CaPcpsAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.135.2.3 mag()

```
signed int GlonassL1CaPcpsAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.135.2.4 reset()

```
void GlonassL1CaPcpsAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.135.2.5 set_channel()

```
void GlonassL1CaPcpsAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 86 of file glonass_l1_ca_pcps_acquisition.h.

10.135.2.6 set_channel_fsm()

```
void GlonassL1CaPcpsAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 95 of file glonass_l1_ca_pcps_acquisition.h.

10.135.2.7 set_doppler_max()

```
void GlonassL1CaPcpsAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.135.2.8 set_doppler_step()

```
void GlonassL1CaPcpsAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.135.2.9 set_gnss_synchro()

```
void GlonassL1CaPcpsAcquisition::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.135.2.10 set_local_code()

```
void GlonassL1CaPcpsAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local code for GPS L1/CA PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.135.2.11 set_state()

```
void GlonassL1CaPcpsAcquisition::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.135.2.12 `set_threshold()`

```
void GlonassL1CaPcpsAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.135.2.13 `stop_acquisition()`

```
void GlonassL1CaPcpsAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

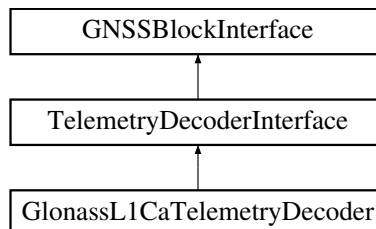
- [glonass_l1_ca_pcps_acquisition.h](#)

10.136 GlonassL1CaTelemetryDecoder Class Reference

This class implements a NAV data decoder for GLONASS L1 C/A.

```
#include <glonass_l1_ca_telemetry_decoder.h>
```

Inheritance diagram for GlonassL1CaTelemetryDecoder:



Public Member Functions

- **GlonassL1CaTelemetryDecoder** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_satellite** (const [Gnss_Satellite](#) &satellite) override
- void **set_channel** (int channel) override
- std::string **role** () override
- std::string **implementation** () override
- *Returns "GLONASS_L1_CA_Telemetry_Decoder".*
- void **reset** () override
- size_t **item_size** () override

10.136.1 Detailed Description

This class implements a NAV data decoder for GLONASS L1 C/A.

Definition at line 39 of file `glonass_l1_ca_telemetry_decoder.h`.

10.136.2 Member Function Documentation

10.136.2.1 `implementation()`

```
std::string GlonassL1CaTelemetryDecoder::implementation ( ) [inline], [override], [virtual]
```

Returns "GLONASS_L1_CA_Telemetry_Decoder".

Implements [GNSSBlockInterface](#).

Definition at line 64 of file `glonass_l1_ca_telemetry_decoder.h`.

The documentation for this class was generated from the following file:

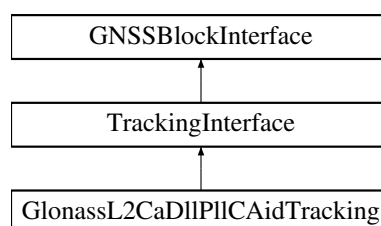
- [glonass_l1_ca_telemetry_decoder.h](#)

10.137 GlonassL2CaDllPllCAidTracking Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <glonass_l2_ca_dll_pll_c_aid_tracking.h>
```

Inheritance diagram for `GlonassL2CaDllPllCAidTracking`:



Public Member Functions

- **GlonassL2CaDllPllCAidTracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "GLONASS_L2_CA_DLL_PLL_C_Aid_Tracking".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override
Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override
Stop running tracking.

10.137.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 40 of file `glonass_l2_ca_dll_pll_c_aid_tracking.h`.

10.137.2 Member Function Documentation

10.137.2.1 implementation()

```
std::string GlonassL2CaDllPllCAidTracking::implementation ( ) [inline], [override], [virtual]
```

Returns "GLONASS_L2_CA_DLL_PLL_C_Aid_Tracking".

Implements [GNSSBlockInterface](#).

Definition at line 57 of file `glonass_l2_ca_dll_pll_c_aid_tracking.h`.

10.137.2.2 set_channel()

```
void GlonassL2CaDllPllCAidTracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.137.2.3 set_gnss_synchro()

```
void GlonassL2CaDllPllCAidTracking::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.137.2.4 stop_tracking()

```
void GlonassL2CaDllPllCAidTracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

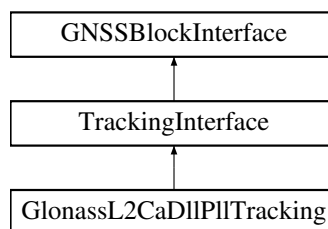
- [glonass_l2_ca_dll_pll_c_aid_tracking.h](#)

10.138 GlonassL2CaDllPllTracking Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <glonass_l2_ca_dll_pll_tracking.h>
```

Inheritance diagram for GlonassL2CaDllPllTracking:



Public Member Functions

- **GlonassL2CaDllPllTracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "GLONASS_L1_CA_DLL_PLL_Tracking".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override
 - Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override
 - Stop running tracking.

10.138.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 39 of file `glonass_l2_ca_dll_pll_tracking.h`.

10.138.2 Member Function Documentation

10.138.2.1 `implementation()`

```
std::string GlonassL2CaDllPllTracking::implementation ( ) [inline], [override], [virtual]
```

Returns "GLONASS_L1_CA_DLL_PLL_Tracking".

Implements [GNSSBlockInterface](#).

Definition at line 56 of file `glonass_l2_ca_dll_pll_tracking.h`.

10.138.2.2 `set_channel()`

```
void GlonassL2CaDllPllTracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.138.2.3 `set_gnss_synchro()`

```
void GlonassL2CaDllPllTracking::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.138.2.4 stop_tracking()

```
void GlonassL2CaDllPllTracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

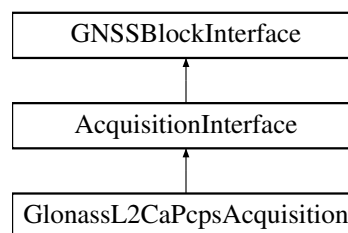
- [glonass_l2_ca_dll_pll_tracking.h](#)

10.139 GlonassL2CaPcpsAcquisition Class Reference

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GLONASS L2 C/A signals.

```
#include <glonass_l2_ca_pcps_acquisition.h>
```

Inheritance diagram for GlonassL2CaPcpsAcquisition:



Public Member Functions

- **GlonassL2CaPcpsAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override

Returns "GLONASS_L2_CA_PCPS_Acquisition".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **set_channel** (unsigned int channel) override

Set acquisition channel unique ID.
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override

Set channel fsm associated to this acquisition instance.
- void **set_threshold** (float threshold) override

Set statistics threshold of PCPS algorithm.
- void **set_doppler_max** (unsigned int doppler_max) override

- *Set maximum Doppler off grid search.*
• void [set_doppler_step](#) (unsigned int doppler_step) override
- *Set Doppler steps for the grid search.*
• void [init](#) () override
- *Initializes acquisition algorithm.*
• void [set_local_code](#) () override
- *Sets local code for GLONASS L2/CA PCPS acquisition algorithm.*
• signed int [mag](#) () override
- *Returns the maximum peak of grid search.*
• void [reset](#) () override
- *Restart acquisition algorithm.*
• void [set_state](#) (int state) override
- *If state = 1, it forces the block to start acquiring from the first sample.*
• void [stop_acquisition](#) () override
- *Stop running acquisition.*
• void [set_resampler_latency](#) (uint32_t latency_samples __attribute__((unused))) override

10.139.1 Detailed Description

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GLONASS L2 C/A signals.

Definition at line 41 of file `glonass_l2_ca_pcps_acquisition.h`.

10.139.2 Member Function Documentation

10.139.2.1 implementation()

```
std::string GlonassL2CaPcpsAcquisition::implementation ( ) [inline], [override], [virtual]
```

Returns "GLONASS_L2_CA_PCPS_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 60 of file `glonass_l2_ca_pcps_acquisition.h`.

10.139.2.2 init()

```
void GlonassL2CaPcpsAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.139.2.3 mag()

```
signed int GlonassL2CaPcpsAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.139.2.4 reset()

```
void GlonassL2CaPcpsAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.139.2.5 set_channel()

```
void GlonassL2CaPcpsAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 85 of file glonass_l2_ca_pcps_acquisition.h.

10.139.2.6 set_channel_fsm()

```
void GlonassL2CaPcpsAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 94 of file glonass_l2_ca_pcps_acquisition.h.

10.139.2.7 set_doppler_max()

```
void GlonassL2CaPcpsAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.139.2.8 set_doppler_step()

```
void GlonassL2CaPcpsAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.139.2.9 set_gnss_synchro()

```
void GlonassL2CaPcpsAcquisition::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.139.2.10 set_local_code()

```
void GlonassL2CaPcpsAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local code for GLONASS L2/CA PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.139.2.11 set_state()

```
void GlonassL2CaPcpsAcquisition::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.139.2.12 set_threshold()

```
void GlonassL2CaPcpsAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.139.2.13 stop_acquisition()

```
void GlonassL2CaPcpsAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

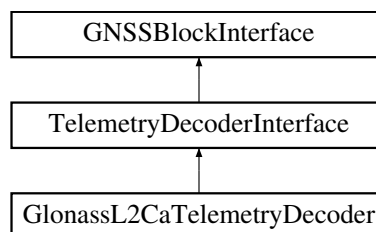
- [glonass_l2_ca_pcps_acquisition.h](#)

10.140 GlonassL2CaTelemetryDecoder Class Reference

This class implements a NAV data decoder for GLONASS L2 C/A.

```
#include <glonass_l2_ca_telemetry_decoder.h>
```

Inheritance diagram for GlonassL2CaTelemetryDecoder:

**Public Member Functions**

- **GlonassL2CaTelemetryDecoder** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_satellite** (const [Gnss_Satellite](#) &satellite) override
- void **set_channel** (int channel) override
- std::string **role** () override
- std::string **implementation** () override
- *Returns "GLONASS_L2_CA_Telemetry_Decoder".*
- void **reset** () override
- size_t **item_size** () override

10.140.1 Detailed Description

This class implements a NAV data decoder for GLONASS L2 C/A.

Definition at line 38 of file `glonass_l2_ca_telemetry_decoder.h`.

10.140.2 Member Function Documentation

10.140.2.1 implementation()

```
std::string GlonassL2CaTelemetryDecoder::implementation ( ) [inline], [override], [virtual]
```

Returns "GLONASS_L2_CA_Telemetry_Decoder".

Implements [GNSSBlockInterface](#).

Definition at line 63 of file `glonass_l2_ca_telemetry_decoder.h`.

The documentation for this class was generated from the following file:

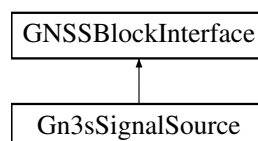
- [glonass_l2_ca_telemetry_decoder.h](#)

10.141 Gn3sSignalSource Class Reference

This class reads samples from a GN3S USB dongle, a RF front-end signal sampler.

```
#include <gn3s_signal_source.h>
```

Inheritance diagram for Gn3sSignalSource:



Public Member Functions

- **Gn3sSignalSource** (const [ConfigurationInterface](#) *configuration, std::string role, unsigned int in_stream, unsigned int out_stream, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "Gn3s_Signal_Source".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.141.1 Detailed Description

This class reads samples from a GN3S USB dongle, a RF front-end signal sampler.

Definition at line 39 of file gn3s_signal_source.h.

10.141.2 Member Function Documentation

10.141.2.1 implementation()

```
std::string Gn3sSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "Gn3s_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 56 of file gn3s_signal_source.h.

The documentation for this class was generated from the following file:

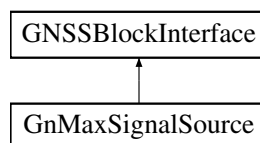
- [gn3s_signal_source.h](#)

10.142 GnMaxSignalSource Class Reference

This class reads samples from a gnMAX2769 USB dongle, a RF front-end signal sampler.

```
#include <gnmax_signal_source.h>
```

Inheritance diagram for GnMaxSignalSource:



Public Member Functions

- **GnMaxSignalSource** (const [ConfigurationInterface](#) *configuration, std::string role, unsigned int in_stream, unsigned int out_stream, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "GNMAX_Signal_Source".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.142.1 Detailed Description

This class reads samples from a gnMAX2769 USB dongle, a RF front-end signal sampler.

Definition at line 38 of file gnmax_signal_source.h.

10.142.2 Member Function Documentation

10.142.2.1 implementation()

```
std::string GnMaxSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "GNMAX_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 55 of file gnmax_signal_source.h.

The documentation for this class was generated from the following file:

- [gnmax_signal_source.h](#)

10.143 Gnss_circular_deque< T > Class Template Reference

Public Member Functions

- [Gnss_circular_deque](#) ()
Default constructor.
- [Gnss_circular_deque](#) (unsigned int max_size, unsigned int nchann)
nchann = number of channels; max_size = channel capacity
- unsigned int [size](#) (unsigned int ch)
Returns the number of available elements in a channel.
- T & [at](#) (unsigned int ch, unsigned int pos)
Returns a reference to an element with bound checking.
- const T & [get](#) (unsigned int ch, unsigned int pos) const
Returns a const reference to an element without bound checking.
- T & [front](#) (unsigned int ch)
Returns a reference to the first element in the deque.
- T & [back](#) (unsigned int ch)
Returns a reference to the last element in the deque.
- void [push_back](#) (unsigned int ch, const T &new_data)
Inserts an element at the end of the deque.
- void [pop_front](#) (unsigned int ch)
Removes the first element of the deque.
- void [clear](#) (unsigned int ch)
Removes all the elements of the deque (Sets size to 0). Capacity is not modified.
- void [reset](#) (unsigned int max_size, unsigned int nchann)
Removes all the elements in all the channels. Re-sets the number of channels and their capacity.
- void [reset](#) ()
Removes all the channels (Sets nchann to 0)

10.143.1 Detailed Description

```
template<class T>
class Gnss_circular_deque< T >
```

Definition at line 28 of file gnss_circular_deque.h.

10.143.2 Constructor & Destructor Documentation

10.143.2.1 Gnss_circular_deque() [1/2]

```
template<class T >
Gnss_circular_deque< T >::Gnss_circular_deque ( )
```

Default constructor.

Definition at line 50 of file gnss_circular_deque.h.

10.143.2.2 Gnss_circular_deque() [2/2]

```
template<class T >
Gnss_circular_deque< T >::Gnss_circular_deque (
    unsigned int max_size,
    unsigned int nchann )
```

nchann = number of channels; *max_size* = channel capacity

Definition at line 57 of file gnss_circular_deque.h.

10.143.3 Member Function Documentation

10.143.3.1 at()

```
template<class T >
T & Gnss_circular_deque< T >::at (
    unsigned int ch,
    unsigned int pos )
```

Returns a reference to an element with bound checking.

Definition at line 85 of file gnss_circular_deque.h.

10.143.3.2 back()

```
template<class T >
T & Gnss_circular_deque< T >::back (
    unsigned int ch )
```

Returns a reference to the last element in the deque.

Definition at line 71 of file gnss_circular_deque.h.

10.143.3.3 clear()

```
template<class T >
void Gnss_circular_deque< T >::clear (
    unsigned int ch )
```

Removes all the elements of the deque (Sets size to 0). Capacity is not modified.

Definition at line 99 of file gnss_circular_deque.h.

10.143.3.4 front()

```
template<class T >
T & Gnss_circular_deque< T >::front (
    unsigned int ch )
```

Returns a reference to the first element in the deque.

Definition at line 78 of file gnss_circular_deque.h.

10.143.3.5 get()

```
template<class T >
const T & Gnss_circular_deque< T >::get (
    unsigned int ch,
    unsigned int pos ) const
```

Returns a const reference to an element without bound checking.

Definition at line 92 of file gnss_circular_deque.h.

10.143.3.6 pop_front()

```
template<class T >
void Gnss_circular_deque< T >::pop_front (
    unsigned int ch )
```

Removes the first element of the deque.

Definition at line 127 of file gnss_circular_deque.h.

10.143.3.7 push_back()

```
template<class T>
void Gnss_circular_deque< T >::push_back (
    unsigned int ch,
    const T & new_data )
```

Inserts an element at the end of the deque.

Definition at line 134 of file gnss_circular_deque.h.

10.143.3.8 reset() [1/2]

```
template<class T >
void Gnss_circular_deque< T >::reset (
    unsigned int max_size,
    unsigned int nchann )
```

Removes all the elements in all the channels. Re-sets the number of channels and their capacity.

Definition at line 106 of file gnss_circular_deque.h.

10.143.3.9 reset() [2/2]

```
template<class T >
void Gnss_circular_deque< T >::reset ( )
```

Removes all the channels (Sets nchann to 0)

Definition at line 120 of file gnss_circular_deque.h.

10.143.3.10 size()

```
template<class T >
unsigned int Gnss_circular_deque< T >::size (
    unsigned int ch )
```

Returns the number of available elements in a channel.

Definition at line 64 of file gnss_circular_deque.h.

The documentation for this class was generated from the following file:

- [gnss_circular_deque.h](#)

10.144 Gnss_Satellite Class Reference

This class represents a GNSS satellite.

```
#include <gnss_satellite.h>
```

Public Member Functions

- [Gnss_Satellite](#) ()=default
Default Constructor.
- [Gnss_Satellite](#) (const std::string &system_, uint32_t PRN_)
Concrete GNSS satellite Constructor.
- [~Gnss_Satellite](#) ()=default
Default Destructor.
- [Gnss_Satellite](#) (const [Gnss_Satellite](#) &other) noexcept
Copy constructor.
- [Gnss_Satellite](#) & operator= (const [Gnss_Satellite](#) &)
Copy assignment operator.
- [Gnss_Satellite](#) ([Gnss_Satellite](#) &&other) noexcept
Move constructor.
- [Gnss_Satellite](#) & operator= ([Gnss_Satellite](#) &&other) noexcept
Move assignment operator.
- void [update_PRN](#) (uint32_t PRN)
Updates the PRN Number when information is decoded, only applies to GLONASS GNAV messages.
- uint32_t [get_PRN](#) () const
Gets satellite's PRN.
- int32_t [get_rf_link](#) () const
Gets the satellite's rf link.
- std::string [get_system](#) () const
Gets the satellite system {"GPS", "GLONASS", "SBAS", "Galileo", "Beidou"}.
- std::string [get_system_short](#) () const
Gets the satellite system {"G", "R", "SBAS", "E", "C"}.
- std::string [get_block](#) () const
Gets the satellite block. If GPS, returns {"IIA", "IIR", "IIR-M", "IIF"}.
- std::string [what_block](#) (const std::string &system_, uint32_t PRN_)
Gets the block of a given satellite.

Friends

- bool `operator==` (const [Gnss_Satellite](#) &, const [Gnss_Satellite](#) &)
operator== for comparison
- std::ostream & `operator<<` (std::ostream &, const [Gnss_Satellite](#) &)
operator<< for pretty printing

10.144.1 Detailed Description

This class represents a GNSS satellite.

It contains information about the space vehicles currently operational of GPS, Glonass, SBAS and Galileo constellations.

Definition at line 37 of file `gnss_satellite.h`.

10.144.2 Constructor & Destructor Documentation

10.144.2.1 `Gnss_Satellite()` [1/4]

```
Gnss_Satellite::Gnss_Satellite ( ) [default]
```

Default Constructor.

10.144.2.2 `Gnss_Satellite()` [2/4]

```
Gnss_Satellite::Gnss_Satellite (
    const std::string & system_,
    uint32_t PRN_ )
```

Concrete GNSS satellite Constructor.

10.144.2.3 `~Gnss_Satellite()`

```
Gnss_Satellite::~Gnss_Satellite ( ) [default]
```

Default Destructor.

10.144.2.4 Gnss_Satellite() [3/4]

```
Gnss_Satellite::Gnss_Satellite (
    const Gnss_Satellite & other ) [noexcept]
```

Copy constructor.

10.144.2.5 Gnss_Satellite() [4/4]

```
Gnss_Satellite::Gnss_Satellite (
    Gnss_Satellite && other ) [noexcept]
```

Move constructor.

10.144.3 Member Function Documentation

10.144.3.1 get_block()

```
std::string Gnss_Satellite::get_block ( ) const
```

Gets the satellite block. If GPS, returns {"IIA", "IIR", "IIR-M", "IIF"}.

10.144.3.2 get_PRN()

```
uint32_t Gnss_Satellite::get_PRN ( ) const
```

Gets satellite's PRN.

10.144.3.3 get_rf_link()

```
int32_t Gnss_Satellite::get_rf_link ( ) const
```

Gets the satellite's rf link.

10.144.3.4 get_system()

```
std::string Gnss_Satellite::get_system ( ) const
```

Gets the satellite system {"GPS", "GLONASS", "SBAS", "Galileo", "Beidou"}.

10.144.3.5 get_system_short()

```
std::string Gnss_Satellite::get_system_short ( ) const
```

Gets the satellite system {"G", "R", "SBAS", "E", "C"}.

10.144.3.6 operator=() [1/2]

```
Gnss_Satellite& Gnss_Satellite::operator= (
    const Gnss_Satellite & )
```

Copy assignment operator.

10.144.3.7 operator=() [2/2]

```
Gnss_Satellite& Gnss_Satellite::operator= (
    Gnss_Satellite && other ) [noexcept]
```

Move assignment operator.

10.144.3.8 update_PRN()

```
void Gnss_Satellite::update_PRN (
    uint32_t PRN )
```

Updates the PRN Number when information is decoded, only applies to GLONASS GNAV messages.

10.144.3.9 what_block()

```
std::string Gnss_Satellite::what_block (
    const std::string & system_,
    uint32_t PRN_ )
```

Gets the block of a given satellite.

10.144.4 Friends And Related Function Documentation

10.144.4.1 operator<<

```
std::ostream& operator<< (
    std::ostream & ,
    const Gnss_Satellite & ) [friend]
```

operator<< for pretty printing

10.144.4.2 operator==

```
bool operator== (
    const Gnss_Satellite & ,
    const Gnss_Satellite & ) [friend]
```

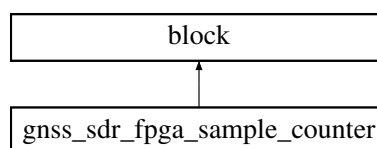
operator== for comparison

The documentation for this class was generated from the following file:

- [gnss_satellite.h](#)

10.145 gnss_sdr_fpga_sample_counter Class Reference

Inheritance diagram for gnss_sdr_fpga_sample_counter:



Public Member Functions

- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- gnss_sdr_fpga_sample_counter_sptr **gnss_sdr_make_fpga_sample_counter** (double _fs, int32_t _↔ interval_ms)

10.145.1 Detailed Description

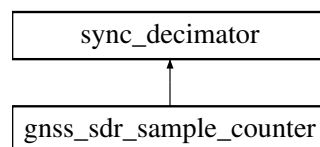
Definition at line 45 of file `gnss_sdr_fpga_sample_counter.h`.

The documentation for this class was generated from the following file:

- [gnss_sdr_fpga_sample_counter.h](#)

10.146 gnss_sdr_sample_counter Class Reference

Inheritance diagram for `gnss_sdr_sample_counter`:



Public Member Functions

- `int work` (`int` noutput_items, `gr_vector_const_void_star` &input_items, `gr_vector_void_star` &output_items)

Friends

- `gnss_sdr_sample_counter_sptr gnss_sdr_make_sample_counter` (`double` _fs, `int32_t` _interval_ms, `size_t` _size)

10.146.1 Detailed Description

Definition at line 47 of file `gnss_sdr_sample_counter.h`.

The documentation for this class was generated from the following file:

- [gnss_sdr_sample_counter.h](#)

10.147 Gnss_Sdr_Supl_Client Class Reference

class that implements a C++ interface to external Secure User Location Protocol (SUPL) client library..

```
#include <gnss_sdr_supl_client.h>
```

Public Member Functions

- int **get_assistance** (int i_mcc, int i_mns, int i_lac, int i_ci)
- void **read_supl_data** ()
- bool **load_ephemeris_xml** (const std::string &file_name)
Read GPS NAV ephemeris map from XML file.
- bool **save_ephemeris_map_xml** (const std::string &file_name, std::map< int, [Gps_Ephemeris](#) > eph_map)
Save ephemeris map to XML file.
- bool **load_cnav_ephemeris_xml** (const std::string &file_name)
Read GPS CNAV ephemeris map from XML file.
- bool **save_cnav_ephemeris_map_xml** (const std::string file_name, std::map< int, [Gps_CNAV_Ephemeris](#) > eph_map)
Save GPS CNAV ephemeris map to XML file.
- bool **load_gal_ephemeris_xml** (const std::string &file_name)
Read Galileo ephemeris map from XML file.
- bool **save_gal_ephemeris_map_xml** (const std::string file_name, std::map< int, [Galileo_Ephemeris](#) > eph_map)
Save Galileo ephemeris map to XML file.
- bool **load_gnav_ephemeris_xml** (const std::string &file_name)
Read GLONASS GNAV ephemeris map from XML file.
- bool **save_gnav_ephemeris_map_xml** (const std::string file_name, std::map< int, [Glonass_Gnav_Ephemeris](#) > eph_map)
Save GLONASS GNAV ephemeris map to XML file.
- bool **load_utc_xml** (const std::string &file_name)
Read GPS utc model from XML file.
- bool **save_utc_xml** (const std::string &file_name, [Gps_Utc_Model](#) &utc)
Save UTC model map to XML file.
- bool **load_cnav_utc_xml** (const std::string &file_name)
Read CNAV GPS utc model from XML file.
- bool **save_cnav_utc_xml** (const std::string &file_name, [Gps_CNAV_Utc_Model](#) &utc)
Save CNAV UTC model map to XML file.
- bool **load_gal_utc_xml** (const std::string &file_name)
Read Galileo utc model from XML file.
- bool **save_gal_utc_xml** (const std::string &file_name, [Galileo_Utc_Model](#) &utc)
Save Galileo UTC model map to XML file.
- bool **load_gal_almanac_xml** (const std::string &file_name)
Read Galileo almanac map from XML file.
- bool **save_gal_almanac_xml** (const std::string &file_name, std::map< int, [Galileo_Almanac](#) > galileo_almanac_map_to_save)
Save Galileo almanac map to XML file.
- bool **load_gps_almanac_xml** (const std::string &file_name)
Read GPS almanac map from XML file.
- bool **save_gps_almanac_xml** (const std::string &file_name, std::map< int, [Gps_Almanac](#) > gps_almanac_map_to_save)
Save GPS almanac map to XML file.
- bool **load_iono_xml** (const std::string &file_name)
Read iono from XML file.
- bool **save_iono_xml** (const std::string &file_name, [Gps_Iono](#) &iono)
Save iono map to XML file.
- bool **load_gal_iono_xml** (const std::string &file_name)
Read Galileo iono from XML file.
- bool **save_gal_iono_xml** (const std::string &file_name, [Galileo_Iono](#) &iono)

- *Save Galileo iono map to XML file.*
- bool [load_glo_utc_xml](#) (const std::string &file_name)
- *Read Glonass utc model from XML file.*
- bool [save_glo_utc_xml](#) (const std::string &file_name, [Glonass_Gnav_Utc_Model](#) &utc)
- *Save Glonass UTC model map to XML file.*
- bool [load_ref_time_xml](#) (const std::string &file_name)
- *Read ref time from XML file.*
- bool [save_ref_time_xml](#) (const std::string &file_name, [Agnss_Ref_Time](#) &ref_time_map)
- *Save ref time map to XML file.*
- bool [load_ref_location_xml](#) (const std::string &file_name)
- *Read ref location from XML file.*
- bool [save_ref_location_xml](#) (const std::string &file_name, [Agnss_Ref_Location](#) &ref_location)
- *Save ref location map to XML file.*
- void [print_assistance](#) ()

Public Attributes

- std::string **server_name**
- int **server_port**
- int **request**
- std::map< int, [Gps_Ephemeris](#) > **gps_ephemeris_map**
- std::map< int, [Galileo_Ephemeris](#) > **gal_ephemeris_map**
- std::map< int, [Gps_CNAV_Ephemeris](#) > **gps_cnav_ephemeris_map**
- std::map< int, [Glonass_Gnav_Ephemeris](#) > **glonass_gnav_ephemeris_map**
- std::map< int, [Gps_Almanac](#) > **gps_almanac_map**
- std::map< int, [Galileo_Almanac](#) > **gal_almanac_map**
- [Gps_Iono](#) **gps_iono**
- [Galileo_Iono](#) **gal_iono**
- [Agnss_Ref_Time](#) **gps_time**
- [Gps_Utc_Model](#) **gps_utc**
- [Galileo_Utc_Model](#) **gal_utc**
- [Gps_CNAV_Utc_Model](#) **gps_cnav_utc**
- [Glonass_Gnav_Utc_Model](#) **glo_gnav_utc**
- [Agnss_Ref_Location](#) **gps_ref_loc**
- std::map< int, [Gps_Acq_Assist](#) > **gps_acq_map**

10.147.1 Detailed Description

class that implements a C++ interface to external Secure User Location Protocol (SUPL) client library..

Definition at line 52 of file gnss_sdr_supl_client.h.

10.147.2 Member Function Documentation

10.147.2.1 load_cnav_ephemeris_xml()

```
bool Gnss_Sdr_Supl_Client::load_cnav_ephemeris_xml (
    const std::string & file_name )
```

Read GPS CNAV ephemeris map from XML file.

10.147.2.2 load_cnav_utc_xml()

```
bool Gnss_Sdr_Supl_Client::load_cnav_utc_xml (
    const std::string & file_name )
```

Read CNAV GPS utc model from XML file.

10.147.2.3 load_ephemeris_xml()

```
bool Gnss_Sdr_Supl_Client::load_ephemeris_xml (
    const std::string & file_name )
```

Read GPS NAV ephemeris map from XML file.

10.147.2.4 load_gal_almanac_xml()

```
bool Gnss_Sdr_Supl_Client::load_gal_almanac_xml (
    const std::string & file_name )
```

Read Galileo almanac map from XML file.

10.147.2.5 load_gal_ephemeris_xml()

```
bool Gnss_Sdr_Supl_Client::load_gal_ephemeris_xml (
    const std::string & file_name )
```

Read Galileo ephemeris map from XML file.

10.147.2.6 load_gal_iono_xml()

```
bool Gnss_Sdr_Supl_Client::load_gal_iono_xml (
    const std::string & file_name )
```

Read Galileo iono from XML file.

10.147.2.7 load_gal_utc_xml()

```
bool Gnss_Sdr_Supl_Client::load_gal_utc_xml (
    const std::string & file_name )
```

Read Galileo utc model from XML file.

10.147.2.8 load_glo_utc_xml()

```
bool Gnss_Sdr_Supl_Client::load_glo_utc_xml (
    const std::string & file_name )
```

Read Glonass utc model from XML file.

10.147.2.9 load_gnav_ephemeris_xml()

```
bool Gnss_Sdr_Supl_Client::load_gnav_ephemeris_xml (
    const std::string & file_name )
```

Read GLONASS GNAV ephemeris map from XML file.

10.147.2.10 load_gps_almanac_xml()

```
bool Gnss_Sdr_Supl_Client::load_gps_almanac_xml (
    const std::string & file_name )
```

Read GPS almanac map from XML file.

10.147.2.11 load_iono_xml()

```
bool Gnss_Sdr_Supl_Client::load_iono_xml (
    const std::string & file_name )
```

Read iono from XML file.

10.147.2.12 load_ref_location_xml()

```
bool Gnss_Sdr_Supl_Client::load_ref_location_xml (
    const std::string & file_name )
```

Read ref location from XML file.

10.147.2.13 load_ref_time_xml()

```
bool Gnss_Sdr_Supl_Client::load_ref_time_xml (
    const std::string & file_name )
```

Read ref time from XML file.

10.147.2.14 load_utc_xml()

```
bool Gnss_Sdr_Supl_Client::load_utc_xml (
    const std::string & file_name )
```

Read GPS utc model from XML file.

10.147.2.15 save_cnav_ephemeris_map_xml()

```
bool Gnss_Sdr_Supl_Client::save_cnav_ephemeris_map_xml (
    const std::string file_name,
    std::map< int, Gps_CNAV_Ephemeris > eph_map )
```

Save GPS CNAV ephemeris map to XML file.

10.147.2.16 save_cnav_utc_xml()

```
bool Gnss_Sdr_Supl_Client::save_cnav_utc_xml (
    const std::string & file_name,
    Gps_CNAV_Utc_Model & utc )
```

Save CNAV UTC model map to XML file.

10.147.2.17 save_ephemeris_map_xml()

```
bool Gnss_Sdr_Supl_Client::save_ephemeris_map_xml (
    const std::string & file_name,
    std::map< int, Gps_Ephemeris > eph_map )
```

Save ephemeris map to XML file.

10.147.2.18 save_gal_almanac_xml()

```
bool Gnss_Sdr_Supl_Client::save_gal_almanac_xml (
    const std::string & file_name,
    std::map< int, Galileo_Almanac > galileo_almanac_map_to_save )
```

Save Galileo almanac map to XML file.

10.147.2.19 save_gal_ephemeris_map_xml()

```
bool Gnss_Sdr_Supl_Client::save_gal_ephemeris_map_xml (
    const std::string file_name,
    std::map< int, Galileo_Ephemeris > eph_map )
```

Save Galileo ephemeris map to XML file.

10.147.2.20 save_gal_iono_xml()

```
bool Gnss_Sdr_Supl_Client::save_gal_iono_xml (
    const std::string & file_name,
    Galileo_Iono & iono )
```

Save Galileo iono map to XML file.

10.147.2.21 save_gal_utc_xml()

```
bool Gnss_Sdr_Supl_Client::save_gal_utc_xml (
    const std::string & file_name,
    Galileo_Utc_Model & utc )
```

Save Galileo UTC model map to XML file.

10.147.2.22 save_glo_utc_xml()

```
bool Gnss_Sdr_Supl_Client::save_glo_utc_xml (
    const std::string & file_name,
    Glonass_Gnav_Utc_Model & utc )
```

Save Glonass UTC model map to XML file.

10.147.2.23 save_gnav_ephemeris_map_xml()

```
bool Gnss_Sdr_Supl_Client::save_gnav_ephemeris_map_xml (
    const std::string file_name,
    std::map< int, Glonass\_Gnav\_Ephemeris > eph_map )
```

Save GLONASS GNAV ephemeris map to XML file.

10.147.2.24 save_gps_almanac_xml()

```
bool Gnss_Sdr_Supl_Client::save_gps_almanac_xml (
    const std::string & file_name,
    std::map< int, Gps\_Almanac > gps_almanac_map_to_save )
```

Save GPS almanac map to XML file.

10.147.2.25 save_iono_xml()

```
bool Gnss_Sdr_Supl_Client::save_iono_xml (
    const std::string & file_name,
    Gps\_Iono & iono )
```

Save iono map to XML file.

10.147.2.26 save_ref_location_xml()

```
bool Gnss_Sdr_Supl_Client::save_ref_location_xml (
    const std::string & file_name,
    Agnss\_Ref\_Location & ref_location )
```

Save ref location map to XML file.

10.147.2.27 save_ref_time_xml()

```
bool Gnss_Sdr_Supl_Client::save_ref_time_xml (
    const std::string & file_name,
    Agnss\_Ref\_Time & ref_time_map )
```

Save ref time map to XML file.

10.147.2.28 save_utc_xml()

```
bool Gnss_Sdr_Supl_Client::save_utc_xml (
    const std::string & file_name,
    Gps_Utc_Model & utc )
```

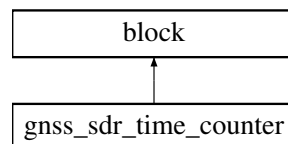
Save UTC model map to XML file.

The documentation for this class was generated from the following file:

- [gnss_sdr_supl_client.h](#)

10.148 gnss_sdr_time_counter Class Reference

Inheritance diagram for gnss_sdr_time_counter:



Public Member Functions

- int **general_work** (int noutput_items __attribute__((unused)), gr_vector_int &ninput_items __attribute__((unused)), gr_vector_const_void_star &input_items __attribute__((unused)), gr_vector_void_star &output_items)

Friends

- gnss_sdr_time_counter_sptr **gnss_sdr_make_time_counter** ()

10.148.1 Detailed Description

Definition at line 35 of file gnss_sdr_time_counter.h.

The documentation for this class was generated from the following file:

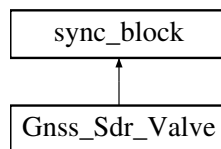
- [gnss_sdr_time_counter.h](#)

10.149 Gnss_Sdr_Valve Class Reference

Implementation of a GNU Radio block that sends a STOP message to the control queue right after a specific number of samples have passed through it.

```
#include <gnss_sdr_valve.h>
```

Inheritance diagram for Gnss_Sdr_Valve:



Public Member Functions

- void **open_valve** ()
- int **work** (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- boost::shared_ptr< [Gnss_Sdr_Valve](#) > **gnss_sdr_make_valve** (size_t sizeof_stream_item, uint64_t nitems, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- boost::shared_ptr< [Gnss_Sdr_Valve](#) > **gnss_sdr_make_valve** (size_t sizeof_stream_item, uint64_t nitems, [Concurrent_Queue](#)< pmt::pmt_t > *queue, bool stop_flowgraph)

10.149.1 Detailed Description

Implementation of a GNU Radio block that sends a STOP message to the control queue right after a specific number of samples have passed through it.

Definition at line 68 of file gnss_sdr_valve.h.

The documentation for this class was generated from the following file:

- [gnss_sdr_valve.h](#)

10.150 Gnss_Signal Class Reference

This class represents a GNSS signal.

```
#include <gnss_signal.h>
```

Public Member Functions

- **Gnss_Signal** (const std::string &signal_)
- **Gnss_Signal** (const [Gnss_Satellite](#) &satellite_, const std::string &signal_)
- std::string [get_signal_str](#) () const
Get the satellite signal {"1C" for GPS L1 C/A, "2S" for GPS L2C (M), "L5" for GPS L5, "1G" for GLONASS L1 C/A, "1B" for Galileo E1B, "5X" for Galileo E5a.
- [Gnss_Satellite](#) [get_satellite](#) () const
Get the [Gnss_Satellite](#) associated to the signal.

Friends

- bool [operator==](#) (const [Gnss_Signal](#) &, const [Gnss_Signal](#) &)
operator== for comparison
- std::ostream & [operator<<](#) (std::ostream &, const [Gnss_Signal](#) &)
operator<< for pretty printing

10.150.1 Detailed Description

This class represents a GNSS signal.

It contains information about the space vehicle and the specific signal.

Definition at line 33 of file gnss_signal.h.

10.150.2 Member Function Documentation

10.150.2.1 [get_satellite\(\)](#)

```
Gnss\_Satellite Gnss_Signal::get_satellite ( ) const
```

Get the [Gnss_Satellite](#) associated to the signal.

10.150.2.2 [get_signal_str\(\)](#)

```
std::string Gnss_Signal::get_signal_str ( ) const
```

Get the satellite signal {"1C" for GPS L1 C/A, "2S" for GPS L2C (M), "L5" for GPS L5, "1G" for GLONASS L1 C/A, "1B" for Galileo E1B, "5X" for Galileo E5a.

10.150.3 Friends And Related Function Documentation

10.150.3.1 operator<<

```
std::ostream& operator<< (
    std::ostream & ,
    const Gnss_Signal & ) [friend]
```

operator<< for pretty printing

10.150.3.2 operator==

```
bool operator== (
    const Gnss_Signal & ,
    const Gnss_Signal & ) [friend]
```

operator== for comparison

The documentation for this class was generated from the following file:

- [gnss_signal.h](#)

10.151 Gnss_Synchro Class Reference

This is the class that contains the information that is shared by the processing blocks.

```
#include <gnss_synchro.h>
```

Public Member Functions

- [Gnss_Synchro](#) ()=default
Default constructor.
- [~Gnss_Synchro](#) ()=default
Default destructor.
- [Gnss_Synchro](#) (const [Gnss_Synchro](#) &other) noexcept
Copy constructor.
- [Gnss_Synchro](#) & operator= (const [Gnss_Synchro](#) &rhs) noexcept
Copy assignment operator.
- [Gnss_Synchro](#) ([Gnss_Synchro](#) &&other) noexcept
Move constructor.
- [Gnss_Synchro](#) & operator= ([Gnss_Synchro](#) &&other) noexcept
Move assignment operator.
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
This member function serializes and restores [Gnss_Synchro](#) objects from a byte stream.

Public Attributes

- char [System](#) {}
Set by Channel::set_signal(Gnss_Signal gnss_signal)
- char [Signal](#) [3] {}
Set by Channel::set_signal(Gnss_Signal gnss_signal)
- uint32_t [PRN](#) {}
Set by Channel::set_signal(Gnss_Signal gnss_signal)
- int32_t [Channel_ID](#) {}
Set by Channel constructor.
- double [Acq_delay_samples](#) {}
Set by Acquisition processing block.
- double [Acq_doppler_hz](#) {}
Set by Acquisition processing block.
- uint64_t [Acq_samplestamp_samples](#) {}
Set by Acquisition processing block.
- uint32_t [Acq_doppler_step](#) {}
Set by Acquisition processing block.
- int64_t [fs](#) {}
Set by Tracking processing block.
- double [Prompt_I](#) {}
Set by Tracking processing block.
- double [Prompt_Q](#) {}
Set by Tracking processing block.
- double [CN0_dB_hz](#) {}
Set by Tracking processing block.
- double [Carrier_Doppler_hz](#) {}
Set by Tracking processing block.
- double [Carrier_phase_rads](#) {}
Set by Tracking processing block.
- double [Code_phase_samples](#) {}
Set by Tracking processing block.
- uint64_t [Tracking_sample_counter](#) {}
Set by Tracking processing block.
- int32_t [correlation_length_ms](#) {}
Set by Tracking processing block.
- uint32_t [TOW_at_current_symbol_ms](#) {}
Set by Telemetry Decoder processing block.
- double [Pseudorange_m](#) {}
Set by Observables processing block.
- double [RX_time](#) {}
Set by Observables processing block.
- double [interp_TOW_ms](#) {}
Set by Observables processing block.
- bool [Flag_valid_acquisition](#) {}
Set by Acquisition processing block.
- bool [Flag_valid_symbol_output](#) {}
Set by Tracking processing block.
- bool [Flag_valid_word](#) {}
Set by Telemetry Decoder processing block.
- bool [Flag_valid_pseudorange](#) {}
Set by Observables processing block.

10.151.1 Detailed Description

This is the class that contains the information that is shared by the processing blocks.

Definition at line 33 of file gnss_synchro.h.

10.151.2 Constructor & Destructor Documentation

10.151.2.1 Gnss_Synchro() [1/3]

```
Gnss_Synchro::Gnss_Synchro ( ) [default]
```

Default constructor.

10.151.2.2 ~Gnss_Synchro()

```
Gnss_Synchro::~~Gnss_Synchro ( ) [default]
```

Default destructor.

10.151.2.3 Gnss_Synchro() [2/3]

```
Gnss_Synchro::Gnss_Synchro (
    const Gnss_Synchro & other ) [inline], [noexcept]
```

Copy constructor.

Definition at line 78 of file gnss_synchro.h.

10.151.2.4 Gnss_Synchro() [3/3]

```
Gnss_Synchro::Gnss_Synchro (
    Gnss_Synchro && other ) [inline], [noexcept]
```

Move constructor.

Definition at line 121 of file gnss_synchro.h.

10.151.3 Member Function Documentation

10.151.3.1 `operator=()` [1/2]

```
Gnss_Synchro& Gnss_Synchro::operator= (
    const Gnss_Synchro & rhs ) [inline], [noexcept]
```

Copy assignment operator.

Definition at line 84 of file `gnss_synchro.h`.

References `Acq_delay_samples`, `Acq_doppler_hz`, `Acq_doppler_step`, `Acq_samplestamp_samples`, `Carrier_Doppler_hz`, `Carrier_phase_rads`, `Channel_ID`, `CN0_dB_hz`, `Code_phase_samples`, `correlation_length_ms`, `Flag_valid_acquisition`, `Flag_valid_pseudorange`, `Flag_valid_symbol_output`, `Flag_valid_word`, `fs`, `interp_TOW_ms`, `PRN`, `Prompt_I`, `Prompt_Q`, `Pseudorange_m`, `RX_time`, `Signal`, `System`, `TOW_at_current_symbol_ms`, and `Tracking_sample_counter`.

10.151.3.2 `operator=()` [2/2]

```
Gnss_Synchro& Gnss_Synchro::operator= (
    Gnss_Synchro && other ) [inline], [noexcept]
```

Move assignment operator.

Definition at line 127 of file `gnss_synchro.h`.

References `Acq_delay_samples`, `Acq_doppler_hz`, `Acq_doppler_step`, `Acq_samplestamp_samples`, `Carrier_Doppler_hz`, `Carrier_phase_rads`, `Channel_ID`, `CN0_dB_hz`, `Code_phase_samples`, `correlation_length_ms`, `Flag_valid_acquisition`, `Flag_valid_pseudorange`, `Flag_valid_symbol_output`, `Flag_valid_word`, `fs`, `interp_TOW_ms`, `PRN`, `Prompt_I`, `Prompt_Q`, `Pseudorange_m`, `RX_time`, `Signal`, `System`, `TOW_at_current_symbol_ms`, and `Tracking_sample_counter`.

10.151.3.3 `serialize()`

```
template<class Archive >
void Gnss_Synchro::serialize (
    Archive & ar,
    const unsigned int version ) [inline]
```

This member function serializes and restores `Gnss_Synchro` objects from a byte stream.

Definition at line 168 of file `gnss_synchro.h`.

References `Acq_delay_samples`, `Acq_doppler_hz`, `Acq_doppler_step`, `Acq_samplestamp_samples`, `Carrier_Doppler_hz`, `Carrier_phase_rads`, `Channel_ID`, `CN0_dB_hz`, `Code_phase_samples`, `correlation_length_ms`, `Flag_valid_acquisition`, `Flag_valid_pseudorange`, `Flag_valid_symbol_output`, `Flag_valid_word`, `fs`, `interp_TOW_ms`, `PRN`, `Prompt_I`, `Prompt_Q`, `Pseudorange_m`, `RX_time`, `Signal`, `System`, `TOW_at_current_symbol_ms`, and `Tracking_sample_counter`.

10.151.4 Member Data Documentation

10.151.4.1 Acq_delay_samples

```
double Gnss_Synchro::Acq_delay_samples {}
```

Set by Acquisition processing block.

Definition at line 47 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.2 Acq_doppler_hz

```
double Gnss_Synchro::Acq_doppler_hz {}
```

Set by Acquisition processing block.

Definition at line 48 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.3 Acq_doppler_step

```
uint32_t Gnss_Synchro::Acq_doppler_step {}
```

Set by Acquisition processing block.

Definition at line 50 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.4 Acq_samplestamp_samples

```
uint64_t Gnss_Synchro::Acq_samplestamp_samples {}
```

Set by Acquisition processing block.

Definition at line 49 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.5 Carrier_Doppler_hz

```
double Gnss_Synchro::Carrier_Doppler_hz {}
```

Set by Tracking processing block.

Definition at line 57 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.6 Carrier_phase_rads

```
double Gnss_Synchro::Carrier_phase_rads {}
```

Set by Tracking processing block.

Definition at line 58 of file gnss_synchro.h.

Referenced by operator=(), and serialize().

10.151.4.7 Channel_ID

```
int32_t Gnss_Synchro::Channel_ID {}
```

Set by [Channel](#) constructor.

Definition at line 44 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.8 CN0_dB_hz

```
double Gnss_Synchro::CN0_dB_hz {}
```

Set by Tracking processing block.

Definition at line 56 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.9 Code_phase_samples

```
double Gnss_Synchro::Code_phase_samples {}
```

Set by Tracking processing block.

Definition at line 59 of file gnss_synchro.h.

Referenced by operator=(), and serialize().

10.151.4.10 correlation_length_ms

```
int32_t Gnss_Synchro::correlation_length_ms {}
```

Set by Tracking processing block.

Definition at line 61 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.11 Flag_valid_acquisition

```
bool Gnss_Synchro::Flag_valid_acquisition {}
```

Set by Acquisition processing block.

Definition at line 72 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.12 Flag_valid_pseudorange

```
bool Gnss_Synchro::Flag_valid_pseudorange {}
```

Set by Observables processing block.

Definition at line 75 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.13 Flag_valid_symbol_output

```
bool Gnss_Synchro::Flag_valid_symbol_output {}
```

Set by Tracking processing block.

Definition at line 73 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.14 Flag_valid_word

```
bool Gnss_Synchro::Flag_valid_word {}
```

Set by Telemetry Decoder processing block.

Definition at line 74 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.15 fs

```
int64_t Gnss_Synchro::fs {}
```

Set by Tracking processing block.

Definition at line 53 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.16 interp_TOW_ms

```
double Gnss_Synchro::interp_TOW_ms {}
```

Set by Observables processing block.

Definition at line 69 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.17 PRN

```
uint32_t Gnss_Synchro::PRN {}
```

Set by Channel::set_signal(Gnss_Signal gnss_signal)

Definition at line 43 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.18 Prompt_I

```
double Gnss_Synchro::Prompt_I {}
```

Set by Tracking processing block.

Definition at line 54 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.19 Prompt_Q

```
double Gnss_Synchro::Prompt_Q {}
```

Set by Tracking processing block.

Definition at line 55 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.20 Pseudorange_m

```
double Gnss_Synchro::Pseudorange_m {}
```

Set by Observables processing block.

Definition at line 67 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.21 RX_time

```
double Gnss_Synchro::RX_time {}
```

Set by Observables processing block.

Definition at line 68 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.22 Signal

```
char Gnss_Synchro::Signal[3] {}
```

Set by Channel::set_signal(Gnss_Signal gnss_signal)

Definition at line 42 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.23 System

```
char Gnss_Synchro::System {}
```

Set by Channel::set_signal(Gnss_Signal gnss_signal)

Definition at line 41 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.24 TOW_at_current_symbol_ms

```
uint32_t Gnss_Synchro::TOW_at_current_symbol_ms {}
```

Set by Telemetry Decoder processing block.

Definition at line 64 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

10.151.4.25 Tracking_sample_counter

```
uint64_t Gnss_Synchro::Tracking_sample_counter {}
```

Set by Tracking processing block.

Definition at line 60 of file gnss_synchro.h.

Referenced by operator=(), Serdes_Gnss_Synchro::readProtobuffer(), and serialize().

The documentation for this class was generated from the following file:

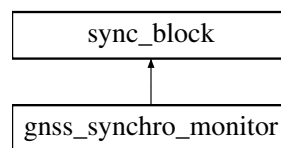
- [gnss_synchro.h](#)

10.152 gnss_synchro_monitor Class Reference

This class implements a monitoring block which allows sending a data stream with the receiver internal parameters ([Gnss_Synchro](#) objects) to local or remote clients over UDP.

```
#include <gnss_synchro_monitor.h>
```

Inheritance diagram for gnss_synchro_monitor:



Public Member Functions

- [~gnss_synchro_monitor](#) ()=default
Default destructor.
- int **work** (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- gnss_synchro_monitor_sptr **gnss_synchro_make_monitor** (int n_channels, int decimation_factor, int udp_port, const std::vector< std::string > &udp_addresses, bool enable_protobuf)

10.152.1 Detailed Description

This class implements a monitoring block which allows sending a data stream with the receiver internal parameters ([Gnss_Synchro](#) objects) to local or remote clients over UDP.

Definition at line 57 of file gnss_synchro_monitor.h.

10.152.2 Constructor & Destructor Documentation

10.152.2.1 ~gnss_synchro_monitor()

```
gnss_synchro_monitor::~gnss_synchro_monitor ( ) [default]
```

Default destructor.

The documentation for this class was generated from the following file:

- [gnss_synchro_monitor.h](#)

10.153 Gnss_Synchro_Udp_Sink Class Reference

This class sends serialized [Gnss_Synchro](#) objects over UDP to one or multiple endpoints.

```
#include <gnss_synchro_udp_sink.h>
```

Public Member Functions

- **Gnss_Synchro_Udp_Sink** (const std::vector< std::string > &addresses, const uint16_t &port, bool enable_protobuf)
- bool **write_gnss_synchro** (const std::vector< [Gnss_Synchro](#) > &stocks)

10.153.1 Detailed Description

This class sends serialized [Gnss_Synchro](#) objects over UDP to one or multiple endpoints.

Definition at line 42 of file gnss_synchro_udp_sink.h.

The documentation for this class was generated from the following file:

- [gnss_synchro_udp_sink.h](#)

10.154 GNSSBlockFactory Class Reference

Class that produces all kinds of GNSS blocks.

```
#include <gnss_block_factory.h>
```

Public Member Functions

- `std::unique_ptr< GNSSBlockInterface > GetSignalSource` (const `ConfigurationInterface` *configuration, `Concurrent_Queue`< pmt::pmt_t > *queue, int ID=-1)
- `std::unique_ptr< GNSSBlockInterface > GetSignalConditioner` (const `ConfigurationInterface` *configuration, int ID=-1)
- `std::unique_ptr< std::vector< std::unique_ptr< GNSSBlockInterface > > > GetChannels` (const `ConfigurationInterface` *configuration, `Concurrent_Queue`< pmt::pmt_t > *queue)
- `std::unique_ptr< GNSSBlockInterface > GetObservables` (const `ConfigurationInterface` *configuration)
- `std::unique_ptr< GNSSBlockInterface > GetPVT` (const `ConfigurationInterface` *configuration)
- `std::unique_ptr< GNSSBlockInterface > GetBlock` (const `ConfigurationInterface` *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams, `Concurrent_Queue`< pmt::pmt_t > *queue=nullptr)

Returns the block with the required role implementation and its configuration parameters.

10.154.1 Detailed Description

Class that produces all kinds of GNSS blocks.

Definition at line 46 of file `gnss_block_factory.h`.

10.154.2 Member Function Documentation

10.154.2.1 GetBlock()

```
std::unique_ptr<GNSSBlockInterface> GNSSBlockFactory::GetBlock (
    const ConfigurationInterface * configuration,
    const std::string & role,
    unsigned int in_streams,
    unsigned int out_streams,
    Concurrent_Queue< pmt::pmt_t > * queue = nullptr )
```

Returns the block with the required role implementation and its configuration parameters.

The documentation for this class was generated from the following file:

- [gnss_block_factory.h](#)

10.155.1 Detailed Description

This abstract class represents an interface to GNSS blocks.

Abstract class for GNSS block interfaces. Since all its methods are virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

Definition at line 42 of file `gnss_block_interface.h`.

The documentation for this class was generated from the following file:

- [gnss_block_interface.h](#)

10.156 GNSSFlowgraph Class Reference

This class represents a GNSS flow graph.

```
#include <gnss_flowgraph.h>
```

Public Member Functions

- [GNSSFlowgraph](#) (std::shared_ptr< [ConfigurationInterface](#) > configuration, std::shared_ptr< [Concurrent_Queue](#)< pmt::pmt_t >> queue)
Constructor that initializes the receiver flow graph.
- [~GNSSFlowgraph](#) ()
Destructor.
- void [start](#) ()
Start the flow graph.
- void [stop](#) ()
Stop the flow graph.
- void [connect](#) ()
Connects the defined blocks in the flow graph.
- void [disconnect](#) ()
Disconnect the blocks in the flow graph.
- void [wait](#) ()
Wait for a flowgraph to complete.
- void [acquisition_manager](#) (unsigned int who)
Manage satellite acquisition.
- void [apply_action](#) (unsigned int who, unsigned int what)
Applies an action to the flow graph.
- void [set_configuration](#) (const std::shared_ptr< [ConfigurationInterface](#) > &configuration)
Set flow graph configuration.
- bool [connected](#) () const
- bool [running](#) () const
- bool [send_telemetry_msg](#) (const pmt::pmt_t &msg)
Sends a GNU Radio asynchronous message from telemetry to PVT.
- std::shared_ptr< [PvtInterface](#) > [get_pvt](#) ()
Returns a smart pointer to the PVT object.
- void [prioritize_satellites](#) (const std::vector< std::pair< int, [Gnss_Satellite](#) >> &visible_satellites)
Priorize visible satellites in the specified vector.

10.156.1 Detailed Description

This class represents a GNSS flow graph.

It contains a signal source, a signal conditioner, a set of channels, a PVT and an output filter.

Definition at line 59 of file gnss_flowgraph.h.

10.156.2 Constructor & Destructor Documentation

10.156.2.1 GNSSFlowgraph()

```
GNSSFlowgraph::GNSSFlowgraph (
    std::shared_ptr< ConfigurationInterface > configuration,
    std::shared_ptr< Concurrent_Queue< pmt::pmt_t >> queue )
```

Constructor that initializes the receiver flow graph.

10.156.2.2 ~GNSSFlowgraph()

```
GNSSFlowgraph::~GNSSFlowgraph ( )
```

Destructor.

10.156.3 Member Function Documentation

10.156.3.1 acquisition_manager()

```
void GNSSFlowgraph::acquisition_manager (
    unsigned int who )
```

Manage satellite acquisition.

Parameters

in	who	Channel ID
----	-----	------------

10.156.3.2 apply_action()

```
void GNSSFlowgraph::apply_action (
    unsigned int who,
    unsigned int what )
```

Applies an action to the flow graph.

Parameters

in	<i>who</i>	Who generated the action
in	<i>what</i>	What is the action. 0: acquisition failed; 1: acquisition success; 2: tracking lost

10.156.3.3 connect()

```
void GNSSFlowgraph::connect ( )
```

Connects the defined blocks in the flow graph.

Signal Source > Signal conditioner > Channels >> Observables >> PVT > Output filter

10.156.3.4 disconnect()

```
void GNSSFlowgraph::disconnect ( )
```

Disconnect the blocks in the flow graph.

10.156.3.5 get_pvt()

```
std::shared_ptr<PvtInterface> GNSSFlowgraph::get_pvt ( ) [inline]
```

Returns a smart pointer to the PVT object.

Definition at line 143 of file gnss_flowgraph.h.

10.156.3.6 prioritize_satellites()

```
void GNSSFlowgraph::prioritize_satellites (
    const std::vector< std::pair< int, Gnss_Satellite >> & visible_satellites )
```

Priorize visible satellites in the specified vector.

10.156.3.7 send_telemetry_msg()

```
bool GNSSFlowgraph::send_telemetry_msg (
    const pmt::pmt_t & msg )
```

Sends a GNU Radio asynchronous message from telemetry to PVT.

It is used to assist the receiver with external ephemeris data

10.156.3.8 set_configuration()

```
void GNSSFlowgraph::set_configuration (
    const std::shared_ptr< ConfigurationInterface > & configuration )
```

Set flow graph configuration.

10.156.3.9 start()

```
void GNSSFlowgraph::start ( )
```

Start the flow graph.

10.156.3.10 stop()

```
void GNSSFlowgraph::stop ( )
```

Stop the flow graph.

10.156.3.11 wait()

```
void GNSSFlowgraph::wait ( )
```

Wait for a flowgraph to complete.

Flowgraphs complete when either (1) all blocks indicate that they are done, or (2) after [stop\(\)](#) has been called to request shutdown.

The documentation for this class was generated from the following file:

- [gnss_flowgraph.h](#)

10.157 Gnuplot Class Reference

Public Member Functions

- [Gnuplot](#) (const std::string &style="points")
 - set a style during construction*
- **Gnuplot** (const std::vector< double > &x, const std::string &title="", const std::string &style="points", const std::string &labelx="x", const std::string &labely="y")
- **Gnuplot** (const std::vector< double > &x, const std::vector< double > &y, const std::string &title="", const std::string &style="points", const std::string &labelx="x", const std::string &labely="y")
- **Gnuplot** (const std::vector< double > &x, const std::vector< double > &y, const std::vector< double > &z, const std::string &title="", const std::string &style="points", const std::string &labelx="x", const std::string &labely="y", const std::string &labelz="z")
- [Gnuplot](#) & **cmd** (const std::string &cmdstr)
- [Gnuplot](#) & **operator<<** (const std::string &cmdstr)
 - Sends a command to an active gnuplot session, identical to cmd()*
- [Gnuplot](#) & **showonscreen** ()
- [Gnuplot](#) & **disablescreen** ()
- [Gnuplot](#) & **savetops** (const std::string &filename="gnuplot_output")
- [Gnuplot](#) & **savetopdf** (const std::string &filename="gnuplot_output", unsigned int font_size=12)
- [Gnuplot](#) & **set_style** (const std::string &stylestr="points")
- [Gnuplot](#) & **set_smooth** (const std::string &stylestr="csplines")
- [Gnuplot](#) & **unset_smooth** ()
- [Gnuplot](#) & **set_pointsize** (const double pointsize=1.0)
- [Gnuplot](#) & **set_grid** ()
- [Gnuplot](#) & **unset_grid** ()
- [Gnuplot](#) & **set_multiplot** (int rows, int cols)
- [Gnuplot](#) & **unset_multiplot** ()
- [Gnuplot](#) & **set_samples** (const int samples=100)
- [Gnuplot](#) & **set_isosamples** (const int isolines=10)
- [Gnuplot](#) & **set_hidden3d** ()
- [Gnuplot](#) & **unset_hidden3d** ()
- [Gnuplot](#) & **set_contour** (const std::string &position="base")
- [Gnuplot](#) & **unset_contour** ()
- [Gnuplot](#) & **set_surface** ()
- [Gnuplot](#) & **unset_surface** ()
- [Gnuplot](#) & **set_legend** (const std::string &position="default")
- [Gnuplot](#) & **unset_legend** ()
- [Gnuplot](#) & **set_title** (const std::string &title="")
- [Gnuplot](#) & **unset_title** ()
 - Clears the title of a gnuplot session.*
- [Gnuplot](#) & **set_ylabel** (const std::string &label="x")
- [Gnuplot](#) & **set_xlabel** (const std::string &label="y")
- [Gnuplot](#) & **set_zlabel** (const std::string &label="z")
- [Gnuplot](#) & **set_xrange** (const double iFrom, const double iTo)
- [Gnuplot](#) & **set_yrange** (const double iFrom, const double iTo)
- [Gnuplot](#) & **set_zrange** (const double iFrom, const double iTo)
- [Gnuplot](#) & **set_xautoscale** ()
- [Gnuplot](#) & **set_yautoscale** ()
- [Gnuplot](#) & **set_zautoscale** ()
- [Gnuplot](#) & **set_xlogscale** (const double base=10)
- [Gnuplot](#) & **set_ylogscale** (const double base=10)
- [Gnuplot](#) & **set_zlogscale** (const double base=10)
- [Gnuplot](#) & **unset_xlogscale** ()

- [Gnuplot](#) & [unset_ylogscale](#) ()
- [Gnuplot](#) & [unset_zlogscale](#) ()
- [Gnuplot](#) & [set_cbrange](#) (const double iFrom, const double iTo)
- [Gnuplot](#) & [plotfile_x](#) (const std::string &filename, const unsigned int column=1, const std::string &title="")
- [template<typename X >](#)
[Gnuplot](#) & [plot_x](#) (const X &x, const std::string &title="")
- [Gnuplot](#) & [plotfile_xy](#) (const std::string &filename, const unsigned int column_x=1, const unsigned int column_y=2, const std::string &title="", const unsigned int decimate=1)
- [template<typename X , typename Y >](#)
[Gnuplot](#) & [plot_xy](#) (const X &x, const Y &y, const std::string &title="", const unsigned int decimate=1)
- [Gnuplot](#) & [plotfile_xy_err](#) (const std::string &filename, const unsigned int column_x=1, const unsigned int column_y=2, const unsigned int column_dy=3, const std::string &title="")
- [template<typename X , typename Y , typename E >](#)
[Gnuplot](#) & [plot_xy_err](#) (const X &x, const Y &y, const E &dy, const std::string &title="")
- [template<typename X , typename Y , typename E >](#)
[Gnuplot](#) & [plot_grid3d](#) (const X &x, const Y &y, const E &mag, const std::string &title="")
- [Gnuplot](#) & [plotfile_xyz](#) (const std::string &filename, const unsigned int column_x=1, const unsigned int column_y=2, const unsigned int column_z=3, const std::string &title="")
- [template<typename X , typename Y , typename Z >](#)
[Gnuplot](#) & [plot_xyz](#) (const X &x, const Y &y, const Z &z, const std::string &title="")
- [Gnuplot](#) & [plot_slope](#) (const double a, const double b, const std::string &title="")
- [Gnuplot](#) & [plot_equation](#) (const std::string &equation, const std::string &title="")
- [Gnuplot](#) & [plot_equation3d](#) (const std::string &equation, const std::string &title="")
- [Gnuplot](#) & [plot_image](#) (const unsigned char *ucPicBuf, const unsigned int iWidth, const unsigned int iHeight, const std::string &title="")
- [Gnuplot](#) & [plot_circle](#) (double east, double north, double radius, const std::string &label="")
- [Gnuplot](#) & [replot](#) (void)
replot repeats the last plot or splot command.
- [Gnuplot](#) & [reset_plot](#) ()
- [Gnuplot](#) & [reset_all](#) ()
- void [remove_tmpfiles](#) ()
- bool [is_valid](#) ()

Static Public Member Functions

- static bool [set_GNUPlotPath](#) (const std::string &path)
- static void [set_terminal_std](#) (const std::string &type)

10.157.1 Detailed Description

Definition at line 75 of file `gnuplot_i.h`.

10.157.2 Constructor & Destructor Documentation

10.157.2.1 Gnuplot()

```
Gnuplot::Gnuplot (
    const std::string & style = "points" ) [inline], [explicit]
```

set a style during construction

Definition at line 695 of file `gnuplot_i.h`.

10.157.3 Member Function Documentation

10.157.3.1 `operator<<()`

```
Gnuplot& Gnuplot::operator<< (
    const std::string & cmdstr ) [inline]
```

Sends a command to an active gnuplot session, identical to `cmd()`

Definition at line 225 of file `gnuplot_i.h`.

10.157.3.2 `replot()`

```
Gnuplot& Gnuplot::replot (
    void ) [inline]
```

`replot` repeats the last plot or `splot` command.

Definition at line 639 of file `gnuplot_i.h`.

10.157.3.3 `unset_title()`

```
Gnuplot& Gnuplot::unset_title ( ) [inline]
```

Clears the title of a gnuplot session.

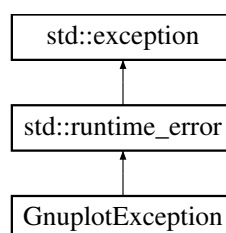
Definition at line 434 of file `gnuplot_i.h`.

The documentation for this class was generated from the following file:

- [gnuplot_i.h](#)

10.158 GnuplotException Class Reference

Inheritance diagram for `GnuplotException`:



Public Member Functions

- **GnuplotException** (const std::string &msg)

10.158.1 Detailed Description

Definition at line 68 of file `gnuplot_i.h`.

The documentation for this class was generated from the following file:

- [gnuplot_i.h](#)

10.159 Gps_Acq_Assist Class Reference

This class is a storage for the GPS GSM RRLP acquisition assistance data as described in Digital cellular telecommunications system (Phase 2+); Location Services (LCS); Mobile Station (MS) - Serving Mobile Location Centre (SMLC) Radio Resource LCS Protocol (RRLP) (3GPP TS 44.031 version 5.12.0 Release 5)

```
#include <gps_acq_assist.h>
```

Public Member Functions

- [Gps_Acq_Assist](#) ()=default

Public Attributes

- uint32_t [i_satellite_PRN](#) {}
SV PRN NUMBER.
- double [d_TOW](#) {}
Time Of Week assigned to the acquisition data.
- double [d_Doppler0](#) {}
Doppler (0 order term) [Hz].
- double [d_Doppler1](#) {}
Doppler (1 order term) [Hz].
- double [dopplerUncertainty](#) {}
Doppler Uncertainty [Hz].
- double [Code_Phase](#) {}
Code phase [chips].
- double [Code_Phase_int](#) {}
Integer Code Phase [1 C/A code period].
- double [GPS_Bit_Number](#) {}
GPS Bit Number.
- double [Code_Phase_window](#) {}
Code Phase search window [chips].
- double [Azimuth](#) {}
Satellite Azimuth [deg].
- double [Elevation](#) {}
Satellite Elevation [deg].

10.159.1 Detailed Description

This class is a storage for the GPS GSM RRLP acquisition assistance data as described in Digital cellular telecommunications system (Phase 2+); Location Services (LCS); Mobile Station (MS) - Serving Mobile Location Centre (SM-LC) Radio Resource LCS Protocol (RRLP) (3GPP TS 44.031 version 5.12.0 Release 5)

Definition at line 34 of file gps_acq_assist.h.

10.159.2 Constructor & Destructor Documentation

10.159.2.1 Gps_Acq_Assist()

```
Gps_Acq_Assist::Gps_Acq_Assist ( ) [default]
```

Default constructor

10.159.3 Member Data Documentation

10.159.3.1 Azimuth

```
double Gps_Acq_Assist::Azimuth {}
```

Satellite Azimuth [deg].

Definition at line 51 of file gps_acq_assist.h.

10.159.3.2 Code_Phase

```
double Gps_Acq_Assist::Code_Phase {}
```

Code phase [chips].

Definition at line 47 of file gps_acq_assist.h.

10.159.3.3 Code_Phase_int

```
double Gps_Acq_Assist::Code_Phase_int {}
```

Integer Code Phase [1 C/A code period].

Definition at line 48 of file gps_acq_assist.h.

10.159.3.4 Code_Phase_window

```
double Gps_Acq_Assist::Code_Phase_window {}
```

Code Phase search window [chips].

Definition at line 50 of file `gps_acq_assist.h`.

10.159.3.5 d_Doppler0

```
double Gps_Acq_Assist::d_Doppler0 {}
```

Doppler (0 order term) [Hz].

Definition at line 44 of file `gps_acq_assist.h`.

10.159.3.6 d_Doppler1

```
double Gps_Acq_Assist::d_Doppler1 {}
```

Doppler (1 order term) [Hz].

Definition at line 45 of file `gps_acq_assist.h`.

10.159.3.7 d_TOW

```
double Gps_Acq_Assist::d_TOW {}
```

Time Of Week assigned to the acquisition data.

Definition at line 43 of file `gps_acq_assist.h`.

10.159.3.8 dopplerUncertainty

```
double Gps_Acq_Assist::dopplerUncertainty {}
```

Doppler Uncertainty [Hz].

Definition at line 46 of file `gps_acq_assist.h`.

10.159.3.9 Elevation

```
double Gps_Acq_Assist::Elevation {}
```

Satellite Elevation [deg].

Definition at line 52 of file `gps_acq_assist.h`.

10.159.3.10 GPS_Bit_Number

```
double Gps_Acq_Assist::GPS_Bit_Number {}
```

GPS Bit Number.

Definition at line 49 of file `gps_acq_assist.h`.

10.159.3.11 i_satellite_PRN

```
uint32_t Gps_Acq_Assist::i_satellite_PRN {}
```

SV PRN NUMBER.

Definition at line 42 of file `gps_acq_assist.h`.

The documentation for this class was generated from the following file:

- [gps_acq_assist.h](#)

10.160 Gps_Almanac Class Reference

This class is a storage for the GPS SV ALMANAC data as described in IS-GPS-200K.

```
#include <gps_almanac.h>
```

Public Member Functions

- [Gps_Almanac](#) ()=default
- `template<class Archive >`
void **serialize** (Archive &ar, const unsigned int version)

Public Attributes

- uint32_t [i_satellite_PRN](#) {}
SV PRN NUMBER.
- double [d_Delta_i](#) {}
Inclination Angle at Reference Time (relative to $i_0 = 0.30$ semi-circles)
- int32_t [i_Toa](#) {}
Almanac data reference time of week (Ref. 20.3.3.4.3 IS-GPS-200K) [s].
- int32_t [i_WNa](#) {}
Almanac week number.
- double [d_M_0](#) {}
Mean Anomaly at Reference Time [semi-circles].
- double [d_e_eccentricity](#) {}
Eccentricity [dimensionless].
- double [d_sqrt_A](#) {}
Square Root of the Semi-Major Axis [sqrt(m)].
- double [d_OMEGA0](#) {}
Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-circles].
- double [d_OMEGA](#) {}
Argument of Perigee [semi-circles].
- double [d_OMEGA_DOT](#) {}
Rate of Right Ascension [semi-circles/s].
- int32_t [i_SV_health](#) {}
SV Health.
- int32_t [i_AS_status](#) {}
Anti-Spoofing Flags and SV Configuration.
- double [d_A_f0](#) {}
Coefficient 0 of code phase offset model [s].
- double [d_A_f1](#) {}
Coefficient 1 of code phase offset model [s/s].

10.160.1 Detailed Description

This class is a storage for the GPS SV ALMANAC data as described in IS-GPS-200K.

See <https://www.gps.gov/technical/icwg/IS-GPS-200K.pdf> Appendix II

Definition at line 32 of file `gps_almanac.h`.

10.160.2 Constructor & Destructor Documentation

10.160.2.1 Gps_Almanac()

```
Gps_Almanac::Gps_Almanac ( ) [default]
```

Default constructor

10.160.3 Member Data Documentation

10.160.3.1 d_A_f0

```
double Gps_Almanac::d_A_f0 {}
```

Coefficient 0 of code phase offset model [s].

Definition at line 52 of file gps_almanac.h.

10.160.3.2 d_A_f1

```
double Gps_Almanac::d_A_f1 {}
```

Coefficient 1 of code phase offset model [s/s].

Definition at line 53 of file gps_almanac.h.

10.160.3.3 d_Delta_i

```
double Gps_Almanac::d_Delta_i {}
```

Inclination Angle at Reference Time (relative to $i_0 = 0.30$ semi-circles)

Definition at line 41 of file gps_almanac.h.

10.160.3.4 d_e_eccentricity

```
double Gps_Almanac::d_e_eccentricity {}
```

Eccentricity [dimensionless].

Definition at line 45 of file gps_almanac.h.

10.160.3.5 d_M_0

```
double Gps_Almanac::d_M_0 {}
```

Mean Anomaly at Reference Time [semi-circles].

Definition at line 44 of file gps_almanac.h.

10.160.3.6 d_OMEGA

```
double Gps_Almanac::d_OMEGA {}
```

Argument of Perigee [semi-cicles].

Definition at line 48 of file gps_almanac.h.

10.160.3.7 d_OMEGA0

```
double Gps_Almanac::d_OMEGA0 {}
```

Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-circles].

Definition at line 47 of file gps_almanac.h.

10.160.3.8 d_OMEGA_DOT

```
double Gps_Almanac::d_OMEGA_DOT {}
```

Rate of Right Ascension [semi-circles/s].

Definition at line 49 of file gps_almanac.h.

10.160.3.9 d_sqrt_A

```
double Gps_Almanac::d_sqrt_A {}
```

Square Root of the Semi-Major Axis [sqrt(m)].

Definition at line 46 of file gps_almanac.h.

10.160.3.10 i_AS_status

```
int32_t Gps_Almanac::i_AS_status {}
```

Anti-Spoofing Flags and SV Configuration.

Definition at line 51 of file gps_almanac.h.

10.160.3.11 i_satellite_PRN

```
uint32_t Gps_Almanac::i_satellite_PRN {}
```

SV PRN NUMBER.

Definition at line 40 of file `gps_almanac.h`.

10.160.3.12 i_SV_health

```
int32_t Gps_Almanac::i_SV_health {}
```

SV Health.

Definition at line 50 of file `gps_almanac.h`.

10.160.3.13 i_Toa

```
int32_t Gps_Almanac::i_Toa {}
```

Almanac data reference time of week (Ref. 20.3.3.4.3 IS-GPS-200K) [s].

Definition at line 42 of file `gps_almanac.h`.

10.160.3.14 i_WNa

```
int32_t Gps_Almanac::i_WNa {}
```

Almanac week number.

Definition at line 43 of file `gps_almanac.h`.

The documentation for this class was generated from the following file:

- [gps_almanac.h](#)

10.161 Gps_CNAV_Ephemeris Class Reference

This class is a storage and orbital model functions for the GPS SV ephemeris data as described in IS-GPS-200K.

```
#include <gps_cnav_ephemeris.h>
```

Public Member Functions

- [Gps_CNAV_Ephemeris](#) ()=default
- double [satellitePosition](#) (double transmitTime)
Compute the ECEF SV coordinates and ECEF velocity Implementation of Table 20-IV (IS-GPS-200K)
- double [sv_clock_drift](#) (double transmitTime)
Sets (d_satClkDrift)and returns the clock drift in seconds according to the User Algorithm for SV Clock Correction (IS-GPS-200K, 20.3.3.3.3.1)
- double [sv_clock_relativistic_term](#) (double transmitTime)
Sets (d_dtr) and returns the clock relativistic correction term in seconds according to the User Algorithm for SV Clock Correction (IS-GPS-200K, 20.3.3.3.3.1)
- template<class Archive >
void [serialize](#) (Archive &archive, const uint32_t version)
Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

Public Attributes

- uint32_t [i_satellite_PRN](#) {}
- int32_t [i_GPS_week](#) {}
GPS week number, aka WN [week].
- int32_t [i_URA](#) {}
ED Accuracy Index.
- int32_t [i_signal_health](#) {}
Signal health (L1/L2/L5)
- int32_t [d_Top](#) {}
Data predict time of week.
- double [d_DELTA_A](#) {}
Semi-major axis difference at reference time.
- double [d_A_DOT](#) {}
Change rate in semi-major axis.
- double [d_Delta_n](#) {}
Mean Motion Difference From Computed Value [semi-circles/s].
- double [d_DELTA_DOT_N](#) {}
Rate of mean motion difference from computed value.
- double [d_M_0](#) {}
Mean Anomaly at Reference Time [semi-circles].
- double [d_e_eccentricity](#) {}
Eccentricity.
- double [d_OMEGA](#) {}
Argument of Perigee [semi-circles].
- double [d_OMEGA0](#) {}
Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-circles].
- int32_t [d_Toe1](#) {}
Ephemeris data reference time of week (Ref. 20.3.3.4.3 IS-GPS-200K) [s].
- int32_t [d_Toe2](#) {}
Ephemeris data reference time of week (Ref. 20.3.3.4.3 IS-GPS-200K) [s].
- double [d_DELTA_OMEGA_DOT](#) {}
Rate of Right Ascension difference [semi-circles/s].
- double [d_i_0](#) {}
Inclination Angle at Reference Time [semi-circles].

- double [d_IDOT](#) {}
Rate of Inclination Angle [semi-circles/s].
- double [d_Cis](#) {}
Amplitude of the Sine Harmonic Correction Term to the Angle of Inclination [rad].
- double [d_Cic](#) {}
Amplitude of the Cosine Harmonic Correction Term to the Angle of Inclination [rad].
- double [d_Crs](#) {}
Amplitude of the Sine Harmonic Correction Term to the Orbit Radius [m].
- double [d_Crc](#) {}
Amplitude of the Cosine Harmonic Correction Term to the Orbit Radius [m].
- double [d_Cus](#) {}
Amplitude of the Sine Harmonic Correction Term to the Argument of Latitude [rad].
- double [d_Cuc](#) {}
Amplitude of the Cosine Harmonic Correction Term to the Argument of Latitude [rad].
- int32_t [d_Toc](#) {}
clock data reference time (Ref. 20.3.3.3.3.1 IS-GPS-200K) [s]
- double [d_A_f0](#) {}
Coefficient 0 of code phase offset model [s].
- double [d_A_f1](#) {}
Coefficient 1 of code phase offset model [s/s].
- double [d_A_f2](#) {}
Coefficient 2 of code phase offset model [s/s²].
- double [d_URA0](#) {}
NED Accuracy Index.
- double [d_URA1](#) {}
NED Accuracy Change Index.
- double [d_URA2](#) {}
NED Accuracy Change Rate Index.
- double [d_TGD](#) {}
Estimated Group Delay Differential: L1-L2 correction term only for the benefit of "L1 P(Y)" or "L2 P(Y)" s users [s].
- double [d_ISCL1](#) {}
- double [d_ISCL2](#) {}
- double [d_ISCL5I](#) {}
- double [d_ISCL5Q](#) {}
- int32_t [d_TOW](#) {}
Time of GPS Week of the ephemeris set (taken from subframes TOW) [s].
- double [d_satClkDrift](#) {}
GPS clock error.
- double [d_dtr](#) {}
relativistic clock correction term
- double [d_satpos_X](#) {}
Earth-fixed coordinate x of the satellite [m]. Intersection of the IERS Reference Meridian (IRM) and the plane passing through the origin and normal to the Z-axis.
- double [d_satpos_Y](#) {}
Earth-fixed coordinate y of the satellite [m]. Completes a right-handed, Earth-Centered, Earth-Fixed orthogonal coordinate system.
- double [d_satpos_Z](#) {}
Earth-fixed coordinate z of the satellite [m]. The direction of the IERS (International Earth Rotation and Reference Systems Service) Reference Pole (IRP).
- double [d_satvel_X](#) {}
Earth-fixed velocity coordinate x of the satellite [m].
- double [d_satvel_Y](#) {}

- *Earth-fixed velocity coordinate y of the satellite [m].*
 • double [d_satvel_Z](#) {}
- *Earth-fixed velocity coordinate z of the satellite [m].*
 • bool [b_integrity_status_flag](#) {}
- *If true, enhanced level of integrity assurance.*
 • bool [b_l2c_phasing_flag](#) {}
- bool [b_alert_flag](#) {}
- *If true, indicates that the SV URA may be worse than indicated in d_SV_accuracy, use that SV at our own risk.*
 • bool [b_antispoofing_flag](#) {}
- *If true, the AntiSpoofing mode is ON in that SV.*

10.161.1 Detailed Description

This class is a storage and orbital model functions for the GPS SV ephemeris data as described in IS-GPS-200K.

See <https://www.gps.gov/technical/icwg/IS-GPS-200K.pdf> Appendix III

Definition at line 33 of file `gps_cnav_ephemeris.h`.

10.161.2 Constructor & Destructor Documentation

10.161.2.1 Gps_CNAV_Ephemeris()

```
Gps_CNAV_Ephemeris::Gps_CNAV_Ephemeris ( ) [default]
```

Default constructor

10.161.3 Member Function Documentation

10.161.3.1 satellitePosition()

```
double Gps_CNAV_Ephemeris::satellitePosition (
    double transmitTime )
```

Compute the ECEF SV coordinates and ECEF velocity Implementation of Table 20-IV (IS-GPS-200K)

10.161.3.2 `serialize()`

```
template<class Archive >
void Gps_CNAV_Ephemeris::serialize (
    Archive & archive,
    const uint32_t version ) [inline]
```

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

- < Time of GPS Week of the ephemeris set (taken from subframes TOW) [s]
- < Amplitude of the Sine Harmonic Correction Term to the Orbit Radius [m]
- < Mean Anomaly at Reference Time [semi-circles]
- < Amplitude of the Cosine Harmonic Correction Term to the Argument of Latitude [rad]
- < Eccentricity [dimensionless]
- < Amplitude of the Sine Harmonic Correction Term to the Argument of Latitude [rad]
- < Ephemeris data reference time of week (Ref. 20.3.3.4.3 IS-GPS-200K) [s]
- < Ephemeris data reference time of week (Ref. 20.3.3.4.3 IS-GPS-200K) [s]
- < clock data reference time (Ref. 20.3.3.3.1 IS-GPS-200K) [s]
- < Amplitude of the Cosine Harmonic Correction Term to the Angle of Inclination [rad]
- < Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-circles]
- < Amplitude of the Sine Harmonic Correction Term to the Angle of Inclination [rad]
- < Inclination Angle at Reference Time [semi-circles]
- < Amplitude of the Cosine Harmonic Correction Term to the Orbit Radius [m]
- < Argument of Perigee [semi-circles]
- < Rate of Inclination Angle [semi-circles/s]
- < GPS week number, aka WN [week]
- < Estimated Group Delay Differential: L1-L2 correction term only for the benefit of "L1 P(Y)" or "L2 P(Y)" s users [s]
- < Estimated Group Delay Differential: L1P(Y)-L1C/A correction term only for the benefit of "L1 P(Y)" or "L2 P(Y)" s users [s]
- < Estimated Group Delay Differential: L1P(Y)-L2C correction term only for the benefit of "L1 P(Y)" or "L2 P(Y)" s users [s]
- < Estimated Group Delay Differential: L1P(Y)-L5i correction term only for the benefit of "L1 P(Y)" or "L2 P(Y)" s users [s]
- < Estimated Group Delay Differential: L1P(Y)-L5q correction term only for the benefit of "L1 P(Y)" or "L2 P(Y)" s users [s]
- < Semi-major axis difference at reference time [m]

- < Change rate in semi-major axis [m/s]
- < Rate of Right Ascension difference [semi-circles/s]
- < Coefficient 0 of code phase offset model [s]
- < Coefficient 1 of code phase offset model [s/s]
- < Coefficient 2 of code phase offset model [s/s²]
- < If true, indicates that the SV URA may be worse than indicated in `d_SV_accuracy`, use that SV at our own risk.
- < If true, the AntiSpoofing mode is ON in that SV

Definition at line 139 of file `gps_cnav_ephemeris.h`.

References `b_alert_flag`, `b_antispoofing_flag`, `b_integrity_status_flag`, `d_A_DOT`, `d_A_f0`, `d_A_f1`, `d_A_f2`, `d_Cic`, `d_Cis`, `d_Crc`, `d_Crs`, `d_Cuc`, `d_Cus`, `d_DELTA_A`, `d_DELTA_OMEGA_DOT`, `d_e_eccentricity`, `d_i_0`, `d_IDOT`, `d_M_0`, `d_OMEGA`, `d_OMEGA0`, `d_TGD`, `d_Toc`, `d_Toe1`, `d_Toe2`, `d_TOW`, and `i_GPS_week`.

10.161.3.3 `sv_clock_drift()`

```
double Gps_CNAV_Ephemeris::sv_clock_drift (
    double transmitTime )
```

Sets (*`d_satClkDrift`*) and returns the clock drift in seconds according to the User Algorithm for SV Clock Correction (IS-GPS-200K, 20.3.3.3.3.1)

10.161.3.4 `sv_clock_relativistic_term()`

```
double Gps_CNAV_Ephemeris::sv_clock_relativistic_term (
    double transmitTime )
```

Sets (*`d_dtr`*) and returns the clock relativistic correction term in seconds according to the User Algorithm for SV Clock Correction (IS-GPS-200K, 20.3.3.3.3.1)

10.161.4 Member Data Documentation

10.161.4.1 `b_alert_flag`

```
bool Gps_CNAV_Ephemeris::b_alert_flag {}
```

If true, indicates that the SV URA may be worse than indicated in `d_SV_accuracy`, use that SV at our own risk.

Definition at line 131 of file `gps_cnav_ephemeris.h`.

Referenced by `serialize()`.

10.161.4.2 b_antispoofing_flag

```
bool Gps_CNAV_Ephemeris::b_antispoofing_flag {}
```

If true, the AntiSpoofing mode is ON in that SV.

Definition at line 132 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.3 b_integrity_status_flag

```
bool Gps_CNAV_Ephemeris::b_integrity_status_flag {}
```

If true, enhanced level of integrity assurance.

If false, indicates that the conveying signal is provided with the legacy level of integrity assurance. That is, the probability that the instantaneous URE of the conveying signal exceeds 4.42 times the upper bound value of the current broadcast URA index, for more than 5.2 seconds, without an accompanying alert, is less than 1E-5 per hour. If true, indicates that the conveying signal is provided with an enhanced level of integrity assurance. That is, the probability that the instantaneous URE of the conveying signal exceeds 5.73 times the upper bound value of the current broadcast URA index, for more than 5.2 seconds, without an accompanying alert, is less than 1E-8 per hour.

Definition at line 129 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.4 d_A_DOT

```
double Gps_CNAV_Ephemeris::d_A_DOT {}
```

Change rate in semi-major axis.

Definition at line 67 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.5 d_A_f0

```
double Gps_CNAV_Ephemeris::d_A_f0 {}
```

Coefficient 0 of code phase offset model [s].

Definition at line 88 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.6 d_A_f1

```
double Gps_CNAV_Ephemeris::d_A_f1 {}
```

Coefficient 1 of code phase offset model [s/s].

Definition at line 89 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.7 d_A_f2

```
double Gps_CNAV_Ephemeris::d_A_f2 {}
```

Coefficient 2 of code phase offset model [s/s²].

Definition at line 90 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.8 d_Cic

```
double Gps_CNAV_Ephemeris::d_Cic {}
```

Amplitude of the Cosine Harmonic Correction Term to the Angle of Inclination [rad].

Definition at line 80 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.9 d_Cis

```
double Gps_CNAV_Ephemeris::d_Cis {}
```

Amplitude of the Sine Harmonic Correction Term to the Angle of Inclination [rad].

Definition at line 79 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.10 d_Crc

```
double Gps_CNAV_Ephemeris::d_Crc {}
```

Amplitude of the Cosine Harmonic Correction Term to the Orbit Radius [m].

Definition at line 82 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.11 d_Crs

```
double Gps_CNAV_Ephemeris::d_Crs {}
```

Amplitude of the Sine Harmonic Correction Term to the Orbit Radius [m].

Definition at line 81 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.12 d_Cuc

```
double Gps_CNAV_Ephemeris::d_Cuc {}
```

Amplitude of the Cosine Harmonic Correction Term to the Argument of Latitude [rad].

Definition at line 84 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.13 d_Cus

```
double Gps_CNAV_Ephemeris::d_Cus {}
```

Amplitude of the Sine Harmonic Correction Term to the Argument of Latitude [rad].

Definition at line 83 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.14 d_DELTA_A

```
double Gps_CNAV_Ephemeris::d_DELTA_A {}
```

Semi-major axis difference at reference time.

Definition at line 66 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.15 d_DELTA_DOT_N

```
double Gps_CNAV_Ephemeris::d_DELTA_DOT_N {}
```

Rate of mean motion difference from computed value.

Definition at line 69 of file gps_cnav_ephemeris.h.

10.161.4.16 d_Delta_n

```
double Gps_CNAV_Ephemeris::d_Delta_n {}
```

Mean Motion Difference From Computed Value [semi-circles/s].

Definition at line 68 of file gps_cnav_ephemeris.h.

10.161.4.17 d_DELTA_OMEGA_DOT

```
double Gps_CNAV_Ephemeris::d_DELTA_OMEGA_DOT {}
```

Rate of Right Ascension difference [semi-circles/s].

Definition at line 76 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.18 d_dtr

```
double Gps_CNAV_Ephemeris::d_dtr {}
```

relativistic clock correction term

Definition at line 107 of file gps_cnav_ephemeris.h.

10.161.4.19 d_e_eccentricity

```
double Gps_CNAV_Ephemeris::d_e_eccentricity {}
```

Eccentricity.

Definition at line 71 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.20 d_i_0

```
double Gps_CNAV_Ephemeris::d_i_0 {}
```

Inclination Angle at Reference Time [semi-circles].

Definition at line 77 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.21 d_IDOT

```
double Gps_CNAV_Ephemeris::d_IDOT {}
```

Rate of Inclination Angle [semi-circles/s].

Definition at line 78 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.22 d_M_0

```
double Gps_CNAV_Ephemeris::d_M_0 {}
```

Mean Anomaly at Reference Time [semi-circles].

Definition at line 70 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.23 d_OMEGA

```
double Gps_CNAV_Ephemeris::d_OMEGA {}
```

Argument of Perigee [semi-cycles].

Definition at line 72 of file `gps_cnav_ephemeris.h`.

Referenced by `serialize()`.

10.161.4.24 d_OMEGA0

```
double Gps_CNAV_Ephemeris::d_OMEGA0 {}
```

Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-cycles].

Definition at line 73 of file `gps_cnav_ephemeris.h`.

Referenced by `serialize()`.

10.161.4.25 d_satClkDrift

```
double Gps_CNAV_Ephemeris::d_satClkDrift {}
```

GPS clock error.

Definition at line 106 of file `gps_cnav_ephemeris.h`.

10.161.4.26 d_satpos_X

```
double Gps_CNAV_Ephemeris::d_satpos_X {}
```

Earth-fixed coordinate x of the satellite [m]. Intersection of the IERS Reference Meridian (IRM) and the plane passing through the origin and normal to the Z-axis.

Definition at line 110 of file `gps_cnav_ephemeris.h`.

10.161.4.27 d_satpos_Y

```
double Gps_CNAV_Ephemeris::d_satpos_Y {}
```

Earth-fixed coordinate y of the satellite [m]. Completes a right-handed, Earth-Centered, Earth-Fixed orthogonal coordinate system.

Definition at line 111 of file `gps_cnav_ephemeris.h`.

10.161.4.28 d_satpos_Z

```
double Gps_CNAV_Ephemeris::d_satpos_Z {}
```

Earth-fixed coordinate z of the satellite [m]. The direction of the IERS (International Earth Rotation and Reference Systems Service) Reference Pole (IRP).

Definition at line 112 of file gps_cnav_ephemeris.h.

10.161.4.29 d_satvel_X

```
double Gps_CNAV_Ephemeris::d_satvel_X {}
```

Earth-fixed velocity coordinate x of the satellite [m].

Definition at line 115 of file gps_cnav_ephemeris.h.

10.161.4.30 d_satvel_Y

```
double Gps_CNAV_Ephemeris::d_satvel_Y {}
```

Earth-fixed velocity coordinate y of the satellite [m].

Definition at line 116 of file gps_cnav_ephemeris.h.

10.161.4.31 d_satvel_Z

```
double Gps_CNAV_Ephemeris::d_satvel_Z {}
```

Earth-fixed velocity coordinate z of the satellite [m].

Definition at line 117 of file gps_cnav_ephemeris.h.

10.161.4.32 d_TGD

```
double Gps_CNAV_Ephemeris::d_TGD {}
```

Estimated Group Delay Differential: L1-L2 correction term only for the benefit of "L1 P(Y)" or "L2 P(Y)" s users [s].

Definition at line 97 of file gps_cnav_ephemeris.h.

Referenced by `serialize()`.

10.161.4.33 d_Toc

```
int32_t Gps_CNAV_Ephemeris::d_Toc {}
```

clock data reference time (Ref. 20.3.3.3.1 IS-GPS-200K) [s]

Definition at line 87 of file `gps_cnav_ephemeris.h`.

Referenced by `serialize()`.

10.161.4.34 d_Toe1

```
int32_t Gps_CNAV_Ephemeris::d_Toe1 {}
```

Ephemeris data reference time of week (Ref. 20.3.3.4.3 IS-GPS-200K) [s].

Definition at line 74 of file `gps_cnav_ephemeris.h`.

Referenced by `serialize()`.

10.161.4.35 d_Toe2

```
int32_t Gps_CNAV_Ephemeris::d_Toe2 {}
```

Ephemeris data reference time of week (Ref. 20.3.3.4.3 IS-GPS-200K) [s].

Definition at line 75 of file `gps_cnav_ephemeris.h`.

Referenced by `serialize()`.

10.161.4.36 d_Top

```
int32_t Gps_CNAV_Ephemeris::d_Top {}
```

Data predict time of week.

Definition at line 65 of file `gps_cnav_ephemeris.h`.

10.161.4.37 d_TOW

```
int32_t Gps_CNAV_Ephemeris::d_TOW {}
```

Time of GPS Week of the ephemeris set (taken from subframes TOW) [s].

Definition at line 103 of file `gps_cnav_ephemeris.h`.

Referenced by `serialize()`.

10.161.4.38 d_URA0

```
double Gps_CNAV_Ephemeris::d_URA0 {}
```

NED Accuracy Index.

Definition at line 92 of file `gps_cnav_ephemeris.h`.

10.161.4.39 d_URA1

```
double Gps_CNAV_Ephemeris::d_URA1 {}
```

NED Accuracy Change Index.

Definition at line 93 of file `gps_cnav_ephemeris.h`.

10.161.4.40 d_URA2

```
double Gps_CNAV_Ephemeris::d_URA2 {}
```

NED Accuracy Change Rate Index.

Definition at line 94 of file `gps_cnav_ephemeris.h`.

10.161.4.41 i_GPS_week

```
int32_t Gps_CNAV_Ephemeris::i_GPS_week {}
```

GPS week number, aka WN [week].

Definition at line 62 of file `gps_cnav_ephemeris.h`.

Referenced by `serialize()`.

10.161.4.42 i_signal_health

```
int32_t Gps_CNAV_Ephemeris::i_signal_health {}
```

Signal health (L1/L2/L5)

Definition at line 64 of file `gps_cnav_ephemeris.h`.

10.161.4.43 i_URA

```
int32_t Gps_CNAV_Ephemeris::i_URA {}
```

ED Accuracy Index.

Definition at line 63 of file `gps_cnav_ephemeris.h`.

The documentation for this class was generated from the following file:

- [gps_cnav_ephemeris.h](#)

10.162 Gps_CNAV_Iono Class Reference

This class is a storage for the GPS IONOSPHERIC data as described in IS-GPS-200K.

```
#include <gps_cnav_iono.h>
```

Public Member Functions

- [Gps_CNAV_Iono](#) ()=default
Default constructor.
- `template<class Archive >`
`void serialize (Archive &archive, const unsigned int version) const`
Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

Public Attributes

- double [d_alpha0](#) {}
Coefficient 0 of a cubic equation representing the amplitude of the vertical delay [s].
- double [d_alpha1](#) {}
Coefficient 1 of a cubic equation representing the amplitude of the vertical delay [s/semi-circle].
- double [d_alpha2](#) {}
Coefficient 2 of a cubic equation representing the amplitude of the vertical delay [s(semi-circle)²].
- double [d_alpha3](#) {}
Coefficient 3 of a cubic equation representing the amplitude of the vertical delay [s(semi-circle)³].
- double [d_beta0](#) {}
Coefficient 0 of a cubic equation representing the period of the model [s].
- double [d_beta1](#) {}
Coefficient 1 of a cubic equation representing the period of the model [s/semi-circle].
- double [d_beta2](#) {}
Coefficient 2 of a cubic equation representing the period of the model [s(semi-circle)²].
- double [d_beta3](#) {}
Coefficient 3 of a cubic equation representing the period of the model [s(semi-circle)³].
- bool [valid](#) {}
Valid flag.

10.162.1 Detailed Description

This class is a storage for the GPS IONOSPHERIC data as described in IS-GPS-200K.

See <https://www.gps.gov/technical/icwg/IS-GPS-200K.pdf> Appendix III

Definition at line 33 of file gps_cnav_iono.h.

10.162.2 Constructor & Destructor Documentation

10.162.2.1 Gps_CNAV_Iono()

```
Gps_CNAV_Iono::Gps_CNAV_Iono ( ) [default]
```

Default constructor.

10.162.3 Member Function Documentation

10.162.3.1 serialize()

```
template<class Archive >
void Gps_CNAV_Iono::serialize (
    Archive & archive,
    const unsigned int version ) const [inline]
```

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

Definition at line 54 of file gps_cnav_iono.h.

References `d_alpha0`, `d_alpha1`, `d_alpha2`, `d_alpha3`, `d_beta0`, `d_beta1`, `d_beta2`, and `d_beta3`.

10.162.4 Member Data Documentation

10.162.4.1 d_alpha0

```
double Gps_CNAV_Iono::d_alpha0 {}
```

Coefficient 0 of a cubic equation representing the amplitude of the vertical delay [s].

Definition at line 39 of file gps_cnav_iono.h.

Referenced by `serialize()`.

10.162.4.2 d_alpha1

```
double Gps_CNAV_Iono::d_alpha1 {}
```

Coefficient 1 of a cubic equation representing the amplitude of the vertical delay [s/semi-circle].

Definition at line 40 of file gps_cnav_iono.h.

Referenced by `serialize()`.

10.162.4.3 d_alpha2

```
double Gps_CNAV_Iono::d_alpha2 {}
```

Coefficient 2 of a cubic equation representing the amplitude of the vertical delay [s(semi-circle)²].

Definition at line 41 of file gps_cnav_iono.h.

Referenced by `serialize()`.

10.162.4.4 d_alpha3

```
double Gps_CNAV_Iono::d_alpha3 {}
```

Coefficient 3 of a cubic equation representing the amplitude of the vertical delay [s(semi-circle)³].

Definition at line 42 of file gps_cnav_iono.h.

Referenced by `serialize()`.

10.162.4.5 d_beta0

```
double Gps_CNAV_Iono::d_beta0 {}
```

Coefficient 0 of a cubic equation representing the period of the model [s].

Definition at line 43 of file gps_cnav_iono.h.

Referenced by `serialize()`.

10.162.4.6 d_beta1

```
double Gps_CNAV_Iono::d_beta1 {}
```

Coefficient 1 of a cubic equation representing the period of the model [s/semi-circle].

Definition at line 44 of file `gps_cnav_iono.h`.

Referenced by `serialize()`.

10.162.4.7 d_beta2

```
double Gps_CNAV_Iono::d_beta2 {}
```

Coefficient 2 of a cubic equation representing the period of the model [s(semi-circle)²].

Definition at line 45 of file `gps_cnav_iono.h`.

Referenced by `serialize()`.

10.162.4.8 d_beta3

```
double Gps_CNAV_Iono::d_beta3 {}
```

Coefficient 3 of a cubic equation representing the period of the model [s(semi-circle)³].

Definition at line 46 of file `gps_cnav_iono.h`.

Referenced by `serialize()`.

10.162.4.9 valid

```
bool Gps_CNAV_Iono::valid {}
```

Valid flag.

Definition at line 47 of file `gps_cnav_iono.h`.

The documentation for this class was generated from the following file:

- [gps_cnav_iono.h](#)

10.163 Gps_CNAV_Navigation_Message Class Reference

This class decodes a GPS CNAV Data message as described in IS-GPS-200K.

```
#include <gps_cnav_navigation_message.h>
```

Public Member Functions

- [Gps_CNAV_Navigation_Message](#) ()
- void **decode_page** (std::bitset< GPS_CNAV_DATA_PAGE_BITS > data_bits)
- [Gps_CNAV_Ephemeris get_ephemeris](#) () const
Obtain a GPS SV Ephemeris class filled with current SV data.
- bool [have_new_iono](#) ()
Check if we have a new iono record stored in the GPS ephemeris class.
- [Gps_CNAV_Iono get_iono](#) () const
Obtain a GPS ionospheric correction parameters class filled with current SV data.
- [Gps_CNAV_Utc_Model get_utc_model](#) ()
Obtain a GPS UTC model parameters class filled with current SV data.
- bool [have_new_utc_model](#) ()
- bool [have_new_ephemeris](#) ()
Check if we have a new ephemeris stored in the GPS ephemeris class.

10.163.1 Detailed Description

This class decodes a GPS CNAV Data message as described in IS-GPS-200K.

See <https://www.gps.gov/technical/icwg/IS-GPS-200K.pdf> Appendix III

Definition at line 42 of file `gps_cnav_navigation_message.h`.

10.163.2 Constructor & Destructor Documentation

10.163.2.1 Gps_CNAV_Navigation_Message()

```
Gps_CNAV_Navigation_Message::Gps_CNAV_Navigation_Message ( )
```

Default constructor

10.163.3 Member Function Documentation

10.163.3.1 get_ephemeris()

`Gps_CNAV_Ephemeris` `Gps_CNAV_Navigation_Message::get_ephemeris () const`

Obtain a GPS SV Ephemeris class filled with current SV data.

10.163.3.2 get_iono()

`Gps_CNAV_Iono` `Gps_CNAV_Navigation_Message::get_iono () const`

Obtain a GPS ionospheric correction parameters class filled with current SV data.

10.163.3.3 get_utc_model()

`Gps_CNAV_Utc_Model` `Gps_CNAV_Navigation_Message::get_utc_model ()`

Obtain a GPS UTC model parameters class filled with current SV data.

10.163.3.4 have_new_ephemeris()

`bool` `Gps_CNAV_Navigation_Message::have_new_ephemeris ()`

Check if we have a new ephemeris stored in the GPS ephemeris class.

10.163.3.5 have_new_iono()

`bool` `Gps_CNAV_Navigation_Message::have_new_iono ()`

Check if we have a new iono record stored in the GPS ephemeris class.

10.163.3.6 have_new_utc_model()

`bool` `Gps_CNAV_Navigation_Message::have_new_utc_model ()`

if we have a new GPS UTC model record stored in the GPS ephemeris class

The documentation for this class was generated from the following file:

- [gps_cnav_navigation_message.h](#)

10.164 Gps_CNAV_Utc_Model Class Reference

This class is a storage for the GPS UTC MODEL data as described in in IS-GPS-200K.

```
#include <gps_cnav_utc_model.h>
```

Public Member Functions

- [Gps_CNAV_Utc_Model](#) ()=default
- `template<class Archive >`
void **serialize** (Archive &archive, const uint32_t version)

Public Attributes

- double [d_A2](#) {}
2nd order term of a model that relates GPS and UTC time (ref. 20.3.3.5.2.4 IS-GPS-200K) [s/s]
- double [d_A1](#) {}
1st order term of a model that relates GPS and UTC time (ref. 20.3.3.5.2.4 IS-GPS-200K) [s/s]
- double [d_A0](#) {}
Constant of a model that relates GPS and UTC time (ref. 20.3.3.5.2.4 IS-GPS-200K) [s].
- int32_t [d_t_OT](#) {}
Reference time for UTC data (reference 20.3.4.5 and 20.3.3.5.2.4 IS-GPS-200K) [s].
- int32_t [i_WN_T](#) {}
UTC reference week number [weeks].
- int32_t [d_DeltaT_LS](#) {}
delta time due to leap seconds [s]. Number of leap seconds since 6-Jan-1980 as transmitted by the GPS almanac.
- int32_t [i_WN_LSF](#) {}
Week number at the end of which the leap second becomes effective [weeks].
- int32_t [i_DN](#) {}
Day number (DN) at the end of which the leap second becomes effective [days].
- int32_t [d_DeltaT_LSF](#) {}
Scheduled future or recent past (relative to NAV message upload) value of the delta time due to leap seconds [s].
- bool **valid** {}

10.164.1 Detailed Description

This class is a storage for the GPS UTC MODEL data as described in in IS-GPS-200K.

See <https://www.gps.gov/technical/icwg/IS-GPS-200K.pdf> Appendix III

Definition at line 32 of file `gps_cnav_utc_model.h`.

10.164.2 Constructor & Destructor Documentation

10.164.2.1 Gps_CNAV_Utc_Model()

```
Gps_CNAV_Utc_Model::Gps_CNAV_Utc_Model ( ) [default]
```

Default constructor

10.164.3 Member Data Documentation

10.164.3.1 d_A0

```
double Gps_CNAV_Utc_Model::d_A0 {}
```

Constant of a model that relates GPS and UTC time (ref. 20.3.3.5.2.4 IS-GPS-200K) [s].

Definition at line 43 of file gps_cnav_utc_model.h.

10.164.3.2 d_A1

```
double Gps_CNAV_Utc_Model::d_A1 {}
```

1st order term of a model that relates GPS and UTC time (ref. 20.3.3.5.2.4 IS-GPS-200K) [s/s]

Definition at line 42 of file gps_cnav_utc_model.h.

10.164.3.3 d_A2

```
double Gps_CNAV_Utc_Model::d_A2 {}
```

2nd order term of a model that relates GPS and UTC time (ref. 20.3.3.5.2.4 IS-GPS-200K) [s/s]

Definition at line 41 of file gps_cnav_utc_model.h.

10.164.3.4 d_DeltaT_LS

```
int32_t Gps_CNAV_Utc_Model::d_DeltaT_LS {}
```

delta time due to leap seconds [s]. Number of leap seconds since 6-Jan-1980 as transmitted by the GPS almanac.

Definition at line 46 of file gps_cnav_utc_model.h.

10.164.3.5 d_DeltaT_LSF

```
int32_t Gps_CNAV_Utc_Model::d_DeltaT_LSF {}
```

Scheduled future or recent past (relative to NAV message upload) value of the delta time due to leap seconds [s].

Definition at line 49 of file `gps_cnav_utc_model.h`.

10.164.3.6 d_t_OT

```
int32_t Gps_CNAV_Utc_Model::d_t_OT {}
```

Reference time for UTC data (reference 20.3.4.5 and 20.3.3.5.2.4 IS-GPS-200K) [s].

Definition at line 44 of file `gps_cnav_utc_model.h`.

10.164.3.7 i_DN

```
int32_t Gps_CNAV_Utc_Model::i_DN {}
```

Day number (DN) at the end of which the leap second becomes effective [days].

Definition at line 48 of file `gps_cnav_utc_model.h`.

10.164.3.8 i_WN_LSF

```
int32_t Gps_CNAV_Utc_Model::i_WN_LSF {}
```

Week number at the end of which the leap second becomes effective [weeks].

Definition at line 47 of file `gps_cnav_utc_model.h`.

10.164.3.9 i_WN_T

```
int32_t Gps_CNAV_Utc_Model::i_WN_T {}
```

UTC reference week number [weeks].

Definition at line 45 of file `gps_cnav_utc_model.h`.

The documentation for this class was generated from the following file:

- [gps_cnav_utc_model.h](#)

10.165 Gps_Ephemeris Class Reference

This class is a storage and orbital model functions for the GPS SV ephemeris data as described in IS-GPS-200K.

```
#include <gps_ephemeris.h>
```

Public Member Functions

- [Gps_Ephemeris](#) ()
- double [satellitePosition](#) (double transmitTime)
Compute the ECEF SV coordinates and ECEF velocity Implementation of Table 20-IV (IS-GPS-200K) and compute the clock bias term including relativistic effect (return value)
- double [sv_clock_drift](#) (double transmitTime)
Sets (d_satClkDrift) and returns the clock drift in seconds according to the User Algorithm for SV Clock Correction (IS-GPS-200K, 20.3.3.3.3.1)
- double [sv_clock_relativistic_term](#) (double transmitTime)
Sets (d_dtr) and returns the clock relativistic correction term in seconds according to the User Algorithm for SV Clock Correction (IS-GPS-200K, 20.3.3.3.3.1)
- template<class Archive >
void [serialize](#) (Archive &archive, const uint32_t version)
Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

Public Attributes

- uint32_t [i_satellite_PRN](#) {}
- int32_t [d_TOW](#) {}
Time of GPS Week of the ephemeris set (taken from subframes TOW) [s].
- double [d_Crs](#) {}
Amplitude of the Sine Harmonic Correction Term to the Orbit Radius [m].
- double [d_Delta_n](#) {}
Mean Motion Difference From Computed Value [semi-circles/s].
- double [d_M_0](#) {}
Mean Anomaly at Reference Time [semi-circles].
- double [d_Cuc](#) {}
Amplitude of the Cosine Harmonic Correction Term to the Argument of Latitude [rad].
- double [d_e_eccentricity](#) {}
Eccentricity [dimensionless].
- double [d_Cus](#) {}
Amplitude of the Sine Harmonic Correction Term to the Argument of Latitude [rad].
- double [d_sqrt_A](#) {}
Square Root of the Semi-Major Axis [sqrt(m)].
- int32_t [d_Toe](#) {}
Ephemeris data reference time of week (Ref. 20.3.3.4.3 IS-GPS-200K) [s].
- int32_t [d_Toc](#) {}
clock data reference time (Ref. 20.3.3.3.3.1 IS-GPS-200K) [s]
- double [d_Cic](#) {}
Amplitude of the Cosine Harmonic Correction Term to the Angle of Inclination [rad].
- double [d_OMEGA0](#) {}
Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-circles].

- double [d_Cis](#) {}
Amplitude of the Sine Harmonic Correction Term to the Angle of Inclination [rad].
- double [d_i_0](#) {}
Inclination Angle at Reference Time [semi-circles].
- double [d_Crc](#) {}
Amplitude of the Cosine Harmonic Correction Term to the Orbit Radius [m].
- double [d_OMEGA](#) {}
Argument of Perigee [semi-circles].
- double [d_OMEGA_DOT](#) {}
Rate of Right Ascension [semi-circles/s].
- double [d_IDOT](#) {}
Rate of Inclination Angle [semi-circles/s].
- int32_t [i_code_on_L2](#) {}
If 1, P code ON in L2; if 2, C/A code ON in L2;.
- int32_t [i_GPS_week](#) {}
GPS week number, aka WN [week].
- bool [b_L2_P_data_flag](#) {}
When true, indicates that the NAV data stream was commanded OFF on the P-code of the L2 channel.
- int32_t [i_SV_accuracy](#) {}
User Range Accuracy (URA) index of the SV (reference paragraph 6.2.1) for the standard positioning service user (Ref 20.3.3.3.1.3 IS-GPS-200K)
- int32_t [i_SV_health](#) {}
- double [d_TGD](#) {}
Estimated Group Delay Differential: L1-L2 correction term only for the benefit of "L1 P(Y)" or "L2 P(Y)" s users [s].
- int32_t [d_IODC](#) {}
Issue of Data, Clock.
- int32_t [d_IODE_SF2](#) {}
Issue of Data, Ephemeris (IODE), subframe 2.
- int32_t [d_IODE_SF3](#) {}
Issue of Data, Ephemeris(IODE), subframe 3.
- int32_t [i_AODO](#) {}
Age of Data Offset (AODO) term for the navigation message correction table (NMCT) contained in subframe 4 (reference paragraph 20.3.3.5.1.9) [s].
- bool [b_fit_interval_flag](#) {}
indicates the curve-fit interval used by the CS (Block II/IIA/IIR/IIR-M/IIF) and SS (Block IIIA) in determining the ephemeris parameters, as follows: 0 = 4 hours, 1 = greater than 4 hours.
- double [d_spare1](#) {}
- double [d_spare2](#) {}
- double [d_A_f0](#) {}
Coefficient 0 of code phase offset model [s].
- double [d_A_f1](#) {}
Coefficient 1 of code phase offset model [s/s].
- double [d_A_f2](#) {}
Coefficient 2 of code phase offset model [s/s²].
- bool [b_integrity_status_flag](#) {}
If true, enhanced level of integrity assurance.
- bool [b_alert_flag](#) {}
If true, indicates that the SV URA may be worse than indicated in [d_SV_accuracy](#), use that SV at our own risk.
- bool [b_antispoofing_flag](#) {}
If true, the AntiSpoofing mode is ON in that SV.
- double [d_satClkDrift](#) {}

- GPS clock error.*

 - double `d_dtr` {}
relativistic clock correction term
- double `d_satpos_X` {}
Earth-fixed coordinate x of the satellite [m]. Intersection of the IERS Reference Meridian (IRM) and the plane passing through the origin and normal to the Z-axis.
- double `d_satpos_Y` {}
Earth-fixed coordinate y of the satellite [m]. Completes a right-handed, Earth-Centered, Earth-Fixed orthogonal coordinate system.
- double `d_satpos_Z` {}
Earth-fixed coordinate z of the satellite [m]. The direction of the IERS (International Earth Rotation and Reference Systems Service) Reference Pole (IRP).
- double `d_satvel_X` {}
Earth-fixed velocity coordinate x of the satellite [m].
- double `d_satvel_Y` {}
Earth-fixed velocity coordinate y of the satellite [m].
- double `d_satvel_Z` {}
Earth-fixed velocity coordinate z of the satellite [m].
- `std::map< int, std::string >` `satelliteBlock`
Map that stores to which block the PRN belongs <https://www.navcen.uscg.gov/?Do=constellation&Status>.

10.165.1 Detailed Description

This class is a storage and orbital model functions for the GPS SV ephemeris data as described in IS-GPS-200K.

See <https://www.gps.gov/technical/icwg/IS-GPS-200K.pdf> Appendix II

Definition at line 36 of file `gps_ephemeris.h`.

10.165.2 Constructor & Destructor Documentation

10.165.2.1 Gps_Ephemeris()

```
Gps_Ephemeris::Gps_Ephemeris ( )
```

Default constructor

10.165.3 Member Function Documentation

10.165.3.1 satellitePosition()

```
double Gps_Ephemeris::satellitePosition (
    double transmitTime )
```

Compute the ECEF SV coordinates and ECEF velocity Implementation of Table 20-IV (IS-GPS-200K) and compute the clock bias term including relativistic effect (return value)

10.165.3.2 serialize()

```
template<class Archive >
void Gps_Ephemeris::serialize (
    Archive & archive,
    const uint32_t version ) [inline]
```

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

< Time of GPS Week of the ephemeris set (taken from subframes TOW) [s]

< Amplitude of the Sine Harmonic Correction Term to the Orbit Radius [m]

< Mean Motion Difference From Computed Value [semi-circles/s]

< Mean Anomaly at Reference Time [semi-circles]

< Amplitude of the Cosine Harmonic Correction Term to the Argument of Latitude [rad]

< Eccentricity [dimensionless]

< Amplitude of the Sine Harmonic Correction Term to the Argument of Latitude [rad]

< Square Root of the Semi-Major Axis [sqrt(m)]

< Ephemeris data reference time of week (Ref. 20.3.3.4.3 IS-GPS-200K) [s]

< clock data reference time (Ref. 20.3.3.3.3.1 IS-GPS-200K) [s]

< Amplitude of the Cosine Harmonic Correction Term to the Angle of Inclination [rad]

< Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-circles]

< Amplitude of the Sine Harmonic Correction Term to the Angle of Inclination [rad]

< Inclination Angle at Reference Time [semi-circles]

< Amplitude of the Cosine Harmonic Correction Term to the Orbit Radius [m]

< Argument of Perigee [semi-circles]

< Rate of Right Ascension [semi-circles/s]

< Rate of Inclination Angle [semi-circles/s]

< If 1, P code ON in L2; if 2, C/A code ON in L2;

- < GPS week number, aka WN [week]
- < When true, indicates that the NAV data stream was commanded OFF on the P-code of the L2 channel
- < User Range Accuracy (URA) index of the SV (reference paragraph 6.2.1) for the standard positioning service user (Ref 20.3.3.3.1.3 IS-GPS-200K)
- < Estimated Group Delay Differential: L1-L2 correction term only for the benefit of "L1 P(Y)" or "L2 P(Y)" s users [s]
- < Issue of Data, Clock
- < Age of Data Offset (AODO) term for the navigation message correction table (NMCT) contained in subframe 4 (reference paragraph 20.3.3.5.1.9) [s]
- < Indicates the curve-fit interval used by the CS (Block II/IIA/IIR/IIR-M/IIF) and SS (Block IIIA) in determining the ephemeris parameters, as follows: 0 = 4 hours, 1 = greater than 4 hours.
- < Coefficient 0 of code phase offset model [s]
- < Coefficient 1 of code phase offset model [s/s]
- < Coefficient 2 of code phase offset model [s/s²]
- < If true, indicates that the SV URA may be worse than indicated in d_SV_accuracy, use that SV at our own risk.
- < If true, the AntiSpoofing mode is ON in that SV

Definition at line 138 of file gps_ephemeris.h.

References `b_alert_flag`, `b_antispoofing_flag`, `b_fit_interval_flag`, `b_integrity_status_flag`, `b_L2_P_data_flag`, `d_A_f0`, `d_A_f1`, `d_A_f2`, `d_Cic`, `d_Cis`, `d_Crc`, `d_Crs`, `d_Cuc`, `d_Cus`, `d_Delta_n`, `d_e_eccentricity`, `d_i_0`, `d_IDOT`, `d_IODC`, `d_IODE_SF2`, `d_IODE_SF3`, `d_M_0`, `d_OMEGA`, `d_OMEGA0`, `d_OMEGA_DOT`, `d_sqrt_A`, `d_TGD`, `d_Toc`, `d_Toe`, `d_TOW`, `i_AODO`, `i_code_on_L2`, `i_GPS_week`, and `i_SV_accuracy`.

10.165.3.3 `sv_clock_drift()`

```
double Gps_Ephemeris::sv_clock_drift (
    double transmitTime )
```

Sets (`d_satClkDrift`) and returns the clock drift in seconds according to the User Algorithm for SV Clock Correction (IS-GPS-200K, 20.3.3.3.3.1)

10.165.3.4 `sv_clock_relativistic_term()`

```
double Gps_Ephemeris::sv_clock_relativistic_term (
    double transmitTime )
```

Sets (`d_dtr`) and returns the clock relativistic correction term in seconds according to the User Algorithm for SV Clock Correction (IS-GPS-200K, 20.3.3.3.3.1)

10.165.4 Member Data Documentation

10.165.4.1 `b_alert_flag`

```
bool Gps_Ephemeris::b_alert_flag {}
```

If true, indicates that the SV URA may be worse than indicated in `d_SV_accuracy`, use that SV at our own risk.

Definition at line 114 of file `gps_ephemeris.h`.

Referenced by `serialize()`.

10.165.4.2 `b_antispoofing_flag`

```
bool Gps_Ephemeris::b_antispoofing_flag {}
```

If true, the AntiSpoofing mode is ON in that SV.

Definition at line 115 of file `gps_ephemeris.h`.

Referenced by `serialize()`.

10.165.4.3 `b_fit_interval_flag`

```
bool Gps_Ephemeris::b_fit_interval_flag {}
```

indicates the curve-fit interval used by the CS (Block II/IIA/IIR/IIR-M/IIF) and SS (Block IIIA) in determining the ephemeris parameters, as follows: 0 = 4 hours, 1 = greater than 4 hours.

Definition at line 93 of file `gps_ephemeris.h`.

Referenced by `serialize()`.

10.165.4.4 `b_integrity_status_flag`

```
bool Gps_Ephemeris::b_integrity_status_flag {}
```

If true, enhanced level of integrity assurance.

If false, indicates that the conveying signal is provided with the legacy level of integrity assurance. That is, the probability that the instantaneous URE of the conveying signal exceeds 4.42 times the upper bound value of the current broadcast URA index, for more than 5.2 seconds, without an accompanying alert, is less than 1E-5 per hour. If true, indicates that the conveying signal is provided with an enhanced level of integrity assurance. That is, the probability that the instantaneous URE of the conveying signal exceeds 5.73 times the upper bound value of the current broadcast URA index, for more than 5.2 seconds, without an accompanying alert, is less than 1E-8 per hour.

Definition at line 113 of file `gps_ephemeris.h`.

Referenced by `serialize()`.

10.165.4.5 b_L2_P_data_flag

```
bool Gps_Ephemeris::b_L2_P_data_flag {}
```

When true, indicates that the NAV data stream was commanded OFF on the P-code of the L2 channel.

Definition at line 84 of file `gps_ephemeris.h`.

Referenced by `serialize()`.

10.165.4.6 d_A_f0

```
double Gps_Ephemeris::d_A_f0 {}
```

Coefficient 0 of code phase offset model [s].

Definition at line 97 of file `gps_ephemeris.h`.

Referenced by `serialize()`.

10.165.4.7 d_A_f1

```
double Gps_Ephemeris::d_A_f1 {}
```

Coefficient 1 of code phase offset model [s/s].

Definition at line 98 of file `gps_ephemeris.h`.

Referenced by `serialize()`.

10.165.4.8 d_A_f2

```
double Gps_Ephemeris::d_A_f2 {}
```

Coefficient 2 of code phase offset model [s/s²].

Definition at line 99 of file `gps_ephemeris.h`.

Referenced by `serialize()`.

10.165.4.9 d_Cic

```
double Gps_Ephemeris::d_Cic {}
```

Amplitude of the Cosine Harmonic Correction Term to the Angle of Inclination [rad].

Definition at line 74 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.10 d_Cis

```
double Gps_Ephemeris::d_Cis {}
```

Amplitude of the Sine Harmonic Correction Term to the Angle of Inclination [rad].

Definition at line 76 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.11 d_Crc

```
double Gps_Ephemeris::d_Crc {}
```

Amplitude of the Cosine Harmonic Correction Term to the Orbit Radius [m].

Definition at line 78 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.12 d_Crs

```
double Gps_Ephemeris::d_Crs {}
```

Amplitude of the Sine Harmonic Correction Term to the Orbit Radius [m].

Definition at line 65 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.13 d_Cuc

```
double Gps_Ephemeris::d_Cuc {}
```

Amplitude of the Cosine Harmonic Correction Term to the Argument of Latitude [rad].

Definition at line 68 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.14 d_Cus

```
double Gps_Ephemeris::d_Cus {}
```

Amplitude of the Sine Harmonic Correction Term to the Argument of Latitude [rad].

Definition at line 70 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.15 d_Delta_n

```
double Gps_Ephemeris::d_Delta_n {}
```

Mean Motion Difference From Computed Value [semi-circles/s].

Definition at line 66 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.16 d_dtr

```
double Gps_Ephemeris::d_dtr {}
```

relativistic clock correction term

Definition at line 119 of file gps_ephemeris.h.

10.165.4.17 d_e_eccentricity

```
double Gps_Ephemeris::d_e_eccentricity {}
```

Eccentricity [dimensionless].

Definition at line 69 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.18 d_i_0

```
double Gps_Ephemeris::d_i_0 {}
```

Inclination Angle at Reference Time [semi-circles].

Definition at line 77 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.19 d_IDOT

```
double Gps_Ephemeris::d_IDOT {}
```

Rate of Inclination Angle [semi-circles/s].

Definition at line 81 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.20 d_IODC

```
int32_t Gps_Ephemeris::d_IODC {}
```

Issue of Data, Clock.

Definition at line 88 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.21 d_IODE_SF2

```
int32_t Gps_Ephemeris::d_IODE_SF2 {}
```

Issue of Data, Ephemeris (IODF), subframe 2.

Definition at line 89 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.22 d_IODE_SF3

```
int32_t Gps_Ephemeris::d_IODE_SF3 {}
```

Issue of Data, Ephemeris (IODF), subframe 3.

Definition at line 90 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.23 d_M_0

```
double Gps_Ephemeris::d_M_0 {}
```

Mean Anomaly at Reference Time [semi-circles].

Definition at line 67 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.24 d_OMEGA

```
double Gps_Ephemeris::d_OMEGA {}
```

Argument of Perigee [semi-circles].

Definition at line 79 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.25 d_OMEGA0

```
double Gps_Ephemeris::d_OMEGA0 {}
```

Longitude of Ascending Node of Orbit Plane at Weekly Epoch [semi-circles].

Definition at line 75 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.26 d_OMEGA_DOT

```
double Gps_Ephemeris::d_OMEGA_DOT {}
```

Rate of Right Ascension [semi-circles/s].

Definition at line 80 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.27 d_satClkDrift

```
double Gps_Ephemeris::d_satClkDrift {}
```

GPS clock error.

Definition at line 118 of file gps_ephemeris.h.

10.165.4.28 d_satpos_X

```
double Gps_Ephemeris::d_satpos_X {}
```

Earth-fixed coordinate x of the satellite [m]. Intersection of the IERS Reference Meridian (IRM) and the plane passing through the origin and normal to the Z-axis.

Definition at line 122 of file gps_ephemeris.h.

10.165.4.29 d_satpos_Y

```
double Gps_Ephemeris::d_satpos_Y {}
```

Earth-fixed coordinate y of the satellite [m]. Completes a right-handed, Earth-Centered, Earth-Fixed orthogonal coordinate system.

Definition at line 123 of file gps_ephemeris.h.

10.165.4.30 d_satpos_Z

```
double Gps_Ephemeris::d_satpos_Z {}
```

Earth-fixed coordinate z of the satellite [m]. The direction of the IERS (International Earth Rotation and Reference Systems Service) Reference Pole (IRP).

Definition at line 124 of file gps_ephemeris.h.

10.165.4.31 d_satvel_X

```
double Gps_Ephemeris::d_satvel_X {}
```

Earth-fixed velocity coordinate x of the satellite [m].

Definition at line 127 of file gps_ephemeris.h.

10.165.4.32 d_satvel_Y

```
double Gps_Ephemeris::d_satvel_Y {}
```

Earth-fixed velocity coordinate y of the satellite [m].

Definition at line 128 of file gps_ephemeris.h.

10.165.4.33 d_satvel_Z

```
double Gps_Ephemeris::d_satvel_Z {}
```

Earth-fixed velocity coordinate z of the satellite [m].

Definition at line 129 of file gps_ephemeris.h.

10.165.4.34 d_sqrt_A

```
double Gps_Ephemeris::d_sqrt_A {}
```

Square Root of the Semi-Major Axis [sqrt(m)].

Definition at line 71 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.35 d_TGD

```
double Gps_Ephemeris::d_TGD {}
```

Estimated Group Delay Differential: L1-L2 correction term only for the benefit of "L1 P(Y)" or "L2 P(Y)" s users [s].

Definition at line 87 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.36 d_Toc

```
int32_t Gps_Ephemeris::d_Toc {}
```

clock data reference time (Ref. 20.3.3.3.1 IS-GPS-200K) [s]

Definition at line 73 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.37 d_Toe

```
int32_t Gps_Ephemeris::d_Toe {}
```

Ephemeris data reference time of week (Ref. 20.3.3.4.3 IS-GPS-200K) [s].

Definition at line 72 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.38 d_TOW

```
int32_t Gps_Ephemeris::d_TOW {}
```

Time of GPS Week of the ephemeris set (taken from subframes TOW) [s].

Definition at line 64 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.39 i_AODO

```
int32_t Gps_Ephemeris::i_AODO {}
```

Age of Data Offset (AODO) term for the navigation message correction table (NMCT) contained in subframe 4 (reference paragraph 20.3.3.5.1.9) [s].

Definition at line 91 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.40 i_code_on_L2

```
int32_t Gps_Ephemeris::i_code_on_L2 {}
```

If 1, P code ON in L2; if 2, C/A code ON in L2;.

Definition at line 82 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.41 i_GPS_week

```
int32_t Gps_Ephemeris::i_GPS_week {}
```

GPS week number, aka WN [week].

Definition at line 83 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.42 i_SV_accuracy

```
int32_t Gps_Ephemeris::i_SV_accuracy {}
```

User Range Accuracy (URA) index of the SV (reference paragraph 6.2.1) for the standard positioning service user (Ref 20.3.3.3.1.3 IS-GPS-200K)

Definition at line 85 of file gps_ephemeris.h.

Referenced by `serialize()`.

10.165.4.43 `satelliteBlock`

```
std::map<int, std::string> Gps_Ephemeris::satelliteBlock
```

Map that stores to which block the PRN belongs <https://www.navcen.uscg.gov/?Do=constellation&Status>.

Definition at line 131 of file `gps_ephemeris.h`.

The documentation for this class was generated from the following file:

- [gps_ephemeris.h](#)

10.166 `Gps_Iono` Class Reference

This class is a storage for the GPS IONOSPHERIC data as described in IS-GPS-200K.

```
#include <gps_iono.h>
```

Public Member Functions

- `Gps_Iono()` = default
Default constructor.
- `template<class Archive > void serialize (Archive &archive, const unsigned int version)`
Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

Public Attributes

- `bool valid {}`
Valid flag.
- `double d_alpha0 {}`
Coefficient 0 of a cubic equation representing the amplitude of the vertical delay [s].
- `double d_alpha1 {}`
Coefficient 1 of a cubic equation representing the amplitude of the vertical delay [s/semi-circle].
- `double d_alpha2 {}`
Coefficient 2 of a cubic equation representing the amplitude of the vertical delay [s(semi-circle)^2].
- `double d_alpha3 {}`
Coefficient 3 of a cubic equation representing the amplitude of the vertical delay [s(semi-circle)^3].
- `double d_beta0 {}`
Coefficient 0 of a cubic equation representing the period of the model [s].
- `double d_beta1 {}`
Coefficient 1 of a cubic equation representing the period of the model [s/semi-circle].
- `double d_beta2 {}`
Coefficient 2 of a cubic equation representing the period of the model [s(semi-circle)^2].
- `double d_beta3 {}`
Coefficient 3 of a cubic equation representing the period of the model [s(semi-circle)^3].

10.166.1 Detailed Description

This class is a storage for the GPS IONOSPHERIC data as described in IS-GPS-200K.

See <https://www.gps.gov/technical/icwg/IS-GPS-200K.pdf> Appendix II

Definition at line 33 of file `gps_iono.h`.

10.166.2 Constructor & Destructor Documentation

10.166.2.1 Gps_Iono()

```
Gps_Iono::Gps_Iono ( ) [default]
```

Default constructor.

10.166.3 Member Function Documentation

10.166.3.1 serialize()

```
template<class Archive >
void Gps_Iono::serialize (
    Archive & archive,
    const unsigned int version ) [inline]
```

Serialize is a boost standard method to be called by the boost XML serialization. Here is used to save the ephemeris data on disk file.

Definition at line 54 of file `gps_iono.h`.

References `d_alpha0`, `d_alpha1`, `d_alpha2`, `d_alpha3`, `d_beta0`, `d_beta1`, `d_beta2`, and `d_beta3`.

10.166.4 Member Data Documentation

10.166.4.1 d_alpha0

```
double Gps_Iono::d_alpha0 {}
```

Coefficient 0 of a cubic equation representing the amplitude of the vertical delay [s].

Definition at line 38 of file `gps_iono.h`.

Referenced by `serialize()`.

10.166.4.2 d_alpha1

```
double Gps_Iono::d_alpha1 {}
```

Coefficient 1 of a cubic equation representing the amplitude of the vertical delay [s/semi-circle].

Definition at line 39 of file gps_iono.h.

Referenced by `serialize()`.

10.166.4.3 d_alpha2

```
double Gps_Iono::d_alpha2 {}
```

Coefficient 2 of a cubic equation representing the amplitude of the vertical delay [s(semi-circle)²].

Definition at line 40 of file gps_iono.h.

Referenced by `serialize()`.

10.166.4.4 d_alpha3

```
double Gps_Iono::d_alpha3 {}
```

Coefficient 3 of a cubic equation representing the amplitude of the vertical delay [s(semi-circle)³].

Definition at line 41 of file gps_iono.h.

Referenced by `serialize()`.

10.166.4.5 d_beta0

```
double Gps_Iono::d_beta0 {}
```

Coefficient 0 of a cubic equation representing the period of the model [s].

Definition at line 42 of file gps_iono.h.

Referenced by `serialize()`.

10.166.4.6 d_beta1

```
double Gps_Iono::d_beta1 {}
```

Coefficient 1 of a cubic equation representing the period of the model [s/semi-circle].

Definition at line 43 of file gps_iono.h.

Referenced by `serialize()`.

10.166.4.7 d_beta2

```
double Gps_Iono::d_beta2 {}
```

Coefficient 2 of a cubic equation representing the period of the model [s(semi-circle)²].

Definition at line 44 of file gps_iono.h.

Referenced by `serialize()`.

10.166.4.8 d_beta3

```
double Gps_Iono::d_beta3 {}
```

Coefficient 3 of a cubic equation representing the period of the model [s(semi-circle)³].

Definition at line 45 of file gps_iono.h.

Referenced by `serialize()`.

10.166.4.9 valid

```
bool Gps_Iono::valid {}
```

Valid flag.

Definition at line 36 of file gps_iono.h.

The documentation for this class was generated from the following file:

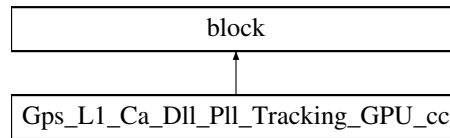
- [gps_iono.h](#)

10.167 Gps_L1_Ca_Dll_Pll_Tracking_GPU_cc Class Reference

This class implements a DLL + PLL tracking loop block.

```
#include <gps_l1_ca_dll_pll_tracking_gpu_cc.h>
```

Inheritance diagram for Gps_L1_Ca_Dll_Pll_Tracking_GPU_cc:



Public Member Functions

- void **set_channel** (uint32_t channel)
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro)
- void **start_tracking** ()
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)

Friends

- `gps_l1_ca_dll_pll_tracking_gpu_cc_sptr` **gps_l1_ca_dll_pll_make_tracking_gpu_cc** (int64_t fs, in, uint32_t vector_length, bool dump, std::string dump_filename, float pll_bw_hz, float dll_bw_hz, float early_late_space_chips)

10.167.1 Detailed Description

This class implements a DLL + PLL tracking loop block.

Definition at line 67 of file `gps_l1_ca_dll_pll_tracking_gpu_cc.h`.

The documentation for this class was generated from the following file:

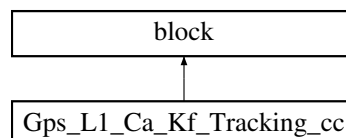
- [gps_l1_ca_dll_pll_tracking_gpu_cc.h](#)

10.168 Gps_L1_Ca_Kf_Tracking_cc Class Reference

This class implements a DLL + PLL tracking loop block.

```
#include <gps_l1_ca_kf_tracking_cc.h>
```

Inheritance diagram for Gps_L1_Ca_Kf_Tracking_cc:



Public Member Functions

- void **set_channel** (uint32_t channel)
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro)
- void **start_tracking** ()
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)

Friends

- [gps_l1_ca_kf_tracking_cc_sptr](#) **gps_l1_ca_kf_make_tracking_cc** (uint32_t order, int64_t if_freq, int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float dll_bw_hz, float early_late←_space_chips, bool bce_run, uint32_t bce_ptrans, uint32_t bce_strans, int32_t bce_nu, int32_t bce_kappa)

10.168.1 Detailed Description

This class implements a DLL + PLL tracking loop block.

Definition at line 79 of file `gps_l1_ca_kf_tracking_cc.h`.

The documentation for this class was generated from the following file:

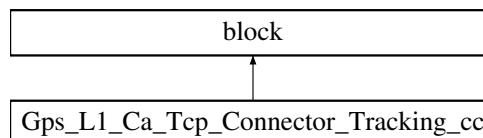
- [gps_l1_ca_kf_tracking_cc.h](#)

10.169 Gps_L1_Ca_Tcp_Connector_Tracking_cc Class Reference

This class implements a DLL + PLL tracking loop block.

```
#include <gps_l1_ca_tcp_connector_tracking_cc.h>
```

Inheritance diagram for `Gps_L1_Ca_Tcp_Connector_Tracking_cc`:



Public Member Functions

- void **set_channel** (uint32_t channel)
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro)
- void **start_tracking** ()
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)

Friends

- `gps_l1_ca_tcp_connector_tracking_cc_sptr` **`gps_l1_ca_tcp_connector_make_tracking_cc`** (`int64_t fs_in`, `uint32_t vector_length`, `bool dump`, `const std::string &dump_filename`, `float early_late_space_chips`, `size_t port_ch0`)

10.169.1 Detailed Description

This class implements a DLL + PLL tracking loop block.

Definition at line 64 of file `gps_l1_ca_tcp_connector_tracking_cc.h`.

The documentation for this class was generated from the following file:

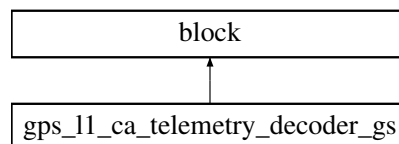
- [gps_l1_ca_tcp_connector_tracking_cc.h](#)

10.170 `gps_l1_ca_telemetry_decoder_gs` Class Reference

This class implements a block that decodes the NAV data defined in IS-GPS-200K.

```
#include <gps_l1_ca_telemetry_decoder_gs.h>
```

Inheritance diagram for `gps_l1_ca_telemetry_decoder_gs`:



Public Member Functions

- void **`set_satellite`** (`const Gnss_Satellite &satellite`)
Set satellite PRN.
- void **`set_channel`** (`int channel`)
Set receiver's channel.
- void **`reset`** ()
- int **`general_work`** (`int noutput_items`, `gr_vector_int &ninput_items`, `gr_vector_const_void_star &input_items`, `gr_vector_void_star &output_items`)
This is where all signal processing takes place.

Friends

- `gps_l1_ca_telemetry_decoder_gs_sptr` **`gps_l1_ca_make_telemetry_decoder_gs`** (`const Gnss_Satellite &satellite`, `bool dump`)

10.170.1 Detailed Description

This class implements a block that decodes the NAV data defined in IS-GPS-200K.

Definition at line 55 of file `gps_l1_ca_telemetry_decoder_gs.h`.

10.170.2 Member Function Documentation

10.170.2.1 `general_work()`

```
int gps_l1_ca_telemetry_decoder_gs::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

This is where all signal processing takes place.

10.170.2.2 `set_channel()`

```
void gps_l1_ca_telemetry_decoder_gs::set_channel (
    int channel )
```

Set receiver's channel.

10.170.2.3 `set_satellite()`

```
void gps_l1_ca_telemetry_decoder_gs::set_satellite (
    const Gnss_Satellite & satellite )
```

Set satellite PRN.

The documentation for this class was generated from the following file:

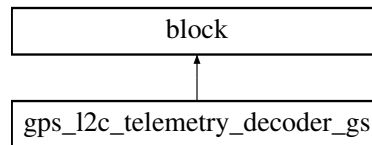
- [gps_l1_ca_telemetry_decoder_gs.h](#)

10.171 `gps_l2c_telemetry_decoder_gs` Class Reference

This class implements a block that decodes CNAV data defined in IS-GPS-200K.

```
#include <gps_l2c_telemetry_decoder_gs.h>
```

Inheritance diagram for `gps_l2c_telemetry_decoder_gs`:



Public Member Functions

- void `set_satellite` (const `Gnss_Satellite` &satellite)
Set satellite PRN.
- void `set_channel` (int32_t channel)
Set receiver's channel.
- void `reset` ()
- int `general_work` (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
This is where all signal processing takes place.

Friends

- `gps_l2c_telemetry_decoder_gs_sptr` `gps_l2c_make_telemetry_decoder_gs` (const `Gnss_Satellite` &satellite, bool dump)

10.171.1 Detailed Description

This class implements a block that decodes CNAV data defined in IS-GPS-200K.

Definition at line 57 of file `gps_l2c_telemetry_decoder_gs.h`.

10.171.2 Member Function Documentation

10.171.2.1 `general_work()`

```
int gps_l2c_telemetry_decoder_gs::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

This is where all signal processing takes place.

10.171.2.2 set_channel()

```
void gps_l2c_telemetry_decoder_gs::set_channel (
    int32_t channel )
```

Set receiver's channel.

10.171.2.3 set_satellite()

```
void gps_l2c_telemetry_decoder_gs::set_satellite (
    const Gnss_Satellite & satellite )
```

Set satellite PRN.

The documentation for this class was generated from the following file:

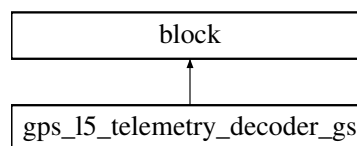
- [gps_l2c_telemetry_decoder_gs.h](#)

10.172 gps_l5_telemetry_decoder_gs Class Reference

This class implements a GPS L5 Telemetry decoder.

```
#include <gps_l5_telemetry_decoder_gs.h>
```

Inheritance diagram for gps_l5_telemetry_decoder_gs:



Public Member Functions

- void [set_satellite](#) (const [Gnss_Satellite](#) &satellite)
Set satellite PRN.
- void [set_channel](#) (int32_t channel)
Set receiver's channel.
- void **reset** ()
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- `gps_l5_telemetry_decoder_gs_sptr` **gps_l5_make_telemetry_decoder_gs** (const [Gnss_Satellite](#) &satellite, bool dump)

10.172.1 Detailed Description

This class implements a GPS L5 Telemetry decoder.

Definition at line 60 of file `gps_l5_telemetry_decoder_gs.h`.

10.172.2 Member Function Documentation

10.172.2.1 `set_channel()`

```
void gps_l5_telemetry_decoder_gs::set_channel (
    int32_t channel )
```

Set receiver's channel.

10.172.2.2 `set_satellite()`

```
void gps_l5_telemetry_decoder_gs::set_satellite (
    const Gnss_Satellite & satellite )
```

Set satellite PRN.

The documentation for this class was generated from the following file:

- [gps_l5_telemetry_decoder_gs.h](#)

10.173 Gps_Navigation_Message Class Reference

This class decodes a GPS NAV Data message as described in IS-GPS-200K.

```
#include <gps_navigation_message.h>
```

Public Member Functions

- [Gps_Navigation_Message](#) ()
- [Gps_Ephemeris](#) [get_ephemeris](#) () const
Obtain a GPS SV Ephemeris class filled with current SV data.
- [Gps_Iono](#) [get_iono](#) ()
Obtain a GPS ionospheric correction parameters class filled with current SV data.
- [Gps_Utc_Model](#) [get_utc_model](#) ()
Obtain a GPS UTC model parameters class filled with current SV data.
- `int32_t` [subframe_decoder](#) (char *subframe)
Decodes the GPS NAV message.
- `double` [utc_time](#) (const `double` gpstime_corrected) const
Computes the Coordinated Universal Time (UTC) and returns it in [s](#)
- `int32_t` [get_TOW](#) () const
Gets Time of Week, in seconds.
- `int32_t` [get_GPS_week](#) () const
Sets Time of Week, in seconds.
- `void` [set_satellite_PRN](#) (`uint32_t` prn)
Sets satellite PRN number.
- `uint32_t` [get_satellite_PRN](#) () const
Gets satellite PRN number.
- `void` [set_channel](#) (`int32_t` channel_id)
Sets channel ID.
- `bool` [get_flag_iono_valid](#) () const
Gets flag_iono_valid.
- `bool` [get_flag_utc_model_valid](#) () const
Gets flag_utc_model_valid.
- `bool` [satellite_validation](#) ()

10.173.1 Detailed Description

This class decodes a GPS NAV Data message as described in IS-GPS-200K.

See <https://www.gps.gov/technical/icwg/IS-GPS-200K.pdf> Appendix II

Definition at line 43 of file `gps_navigation_message.h`.

10.173.2 Constructor & Destructor Documentation

10.173.2.1 Gps_Navigation_Message()

```
Gps_Navigation_Message::Gps_Navigation_Message ( )
```

Default constructor

10.173.3 Member Function Documentation

10.173.3.1 `get_ephemeris()`

`Gps_Ephemeris` `Gps_Navigation_Message::get_ephemeris () const`

Obtain a GPS SV Ephemeris class filled with current SV data.

10.173.3.2 `get_flag_iono_valid()`

`bool` `Gps_Navigation_Message::get_flag_iono_valid () const` `[inline]`

Gets `flag_iono_valid`.

Definition at line 120 of file `gps_navigation_message.h`.

10.173.3.3 `get_flag_utc_model_valid()`

`bool` `Gps_Navigation_Message::get_flag_utc_model_valid () const` `[inline]`

Gets `flag_utc_model_valid`.

Definition at line 128 of file `gps_navigation_message.h`.

10.173.3.4 `get_GPS_week()`

`int32_t` `Gps_Navigation_Message::get_GPS_week () const` `[inline]`

Sets Time of Week, in seconds.

Definition at line 88 of file `gps_navigation_message.h`.

10.173.3.5 `get_iono()`

`Gps_Iono` `Gps_Navigation_Message::get_iono ()`

Obtain a GPS ionospheric correction parameters class filled with current SV data.

10.173.3.6 get_satellite_PRN()

```
uint32_t Gps_Navigation_Message::get_satellite_PRN ( ) const [inline]
```

Gets satellite PRN number.

Definition at line 104 of file `gps_navigation_message.h`.

10.173.3.7 get_TOW()

```
int32_t Gps_Navigation_Message::get_TOW ( ) const [inline]
```

Gets Time of Week, in seconds.

Definition at line 80 of file `gps_navigation_message.h`.

10.173.3.8 get_utc_model()

```
Gps_Utc_Model Gps_Navigation_Message::get_utc_model ( )
```

Obtain a GPS UTC model parameters class filled with current SV data.

10.173.3.9 set_channel()

```
void Gps_Navigation_Message::set_channel (
    int32_t channel_id ) [inline]
```

Sets channel ID.

Definition at line 112 of file `gps_navigation_message.h`.

10.173.3.10 set_satellite_PRN()

```
void Gps_Navigation_Message::set_satellite_PRN (
    uint32_t prn ) [inline]
```

Sets satellite PRN number.

Definition at line 96 of file `gps_navigation_message.h`.

10.173.3.11 subframe_decoder()

```
int32_t Gps_Navigation_Message::subframe_decoder (
    char * subframe )
```

Decodes the GPS NAV message.

10.173.3.12 utc_time()

```
double Gps_Navigation_Message::utc_time (
    const double gpstime_corrected ) const
```

Computes the Coordinated Universal Time (UTC) and returns it in [s](#)

The documentation for this class was generated from the following file:

- [gps_navigation_message.h](#)

10.174 Gps_Utc_Model Class Reference

This class is a storage for the GPS UTC MODEL data as described in IS-GPS-200K.

```
#include <gps_utc_model.h>
```

Public Member Functions

- [Gps_Utc_Model](#) ()=default
- `template<class Archive >`
void **serialize** (Archive &archive, const uint32_t version)

Public Attributes

- double [d_A0](#) {}
Constant of a model that relates GPS and UTC time (ref. 20.3.3.5.2.4 IS-GPS-200K) [s].
- double [d_A1](#) {}
1st order term of a model that relates GPS and UTC time (ref. 20.3.3.5.2.4 IS-GPS-200K) [s/s]
- double [d_A2](#) {}
2nd order term of a model that relates GPS and UTC time (ref. 20.3.3.5.2.4 IS-GPS-200K) [s/s]
- int32_t [d_t_OT](#) {}
Reference time for UTC data (reference 20.3.4.5 and 20.3.3.5.2.4 IS-GPS-200K) [s].
- int32_t [i_WN_T](#) {}
UTC reference week number [weeks].
- int32_t [d_DeltaT_LS](#) {}
delta time due to leap seconds [s]. Number of leap seconds since 6-Jan-1980 as transmitted by the GPS almanac.
- int32_t [i_WN_LSF](#) {}
Week number at the end of which the leap second becomes effective [weeks].
- int32_t [i_DN](#) {}
Day number (DN) at the end of which the leap second becomes effective [days].
- int32_t [d_DeltaT_LSF](#) {}
Scheduled future or recent past (relative to NAV message upload) value of the delta time due to leap seconds [s].
- bool **valid** {}

10.174.1 Detailed Description

This class is a storage for the GPS UTC MODEL data as described in IS-GPS-200K.

See <https://www.gps.gov/technical/icwg/IS-GPS-200K.pdf> Appendix II

Definition at line 32 of file gps_utc_model.h.

10.174.2 Constructor & Destructor Documentation

10.174.2.1 Gps_Utc_Model()

```
Gps_Utc_Model::Gps_Utc_Model ( ) [default]
```

Default constructor

10.174.3 Member Data Documentation

10.174.3.1 d_A0

```
double Gps_Utc_Model::d_A0 {}
```

Constant of a model that relates GPS and UTC time (ref. 20.3.3.5.2.4 IS-GPS-200K) [s].

Definition at line 41 of file gps_utc_model.h.

10.174.3.2 d_A1

```
double Gps_Utc_Model::d_A1 {}
```

1st order term of a model that relates GPS and UTC time (ref. 20.3.3.5.2.4 IS-GPS-200K) [s/s]

Definition at line 42 of file gps_utc_model.h.

10.174.3.3 d_A2

```
double Gps_Utc_Model::d_A2 {}
```

2nd order term of a model that relates GPS and UTC time (ref. 20.3.3.5.2.4 IS-GPS-200K) [s/s]

Definition at line 43 of file gps_utc_model.h.

10.174.3.4 d_DeltaT_LS

```
int32_t Gps_Utc_Model::d_DeltaT_LS {}
```

delta time due to leap seconds [s]. Number of leap seconds since 6-Jan-1980 as transmitted by the GPS almanac.

Definition at line 46 of file `gps_utc_model.h`.

10.174.3.5 d_DeltaT_LSF

```
int32_t Gps_Utc_Model::d_DeltaT_LSF {}
```

Scheduled future or recent past (relative to NAV message upload) value of the delta time due to leap seconds [s].

Definition at line 49 of file `gps_utc_model.h`.

10.174.3.6 d_t_OT

```
int32_t Gps_Utc_Model::d_t_OT {}
```

Reference time for UTC data (reference 20.3.4.5 and 20.3.3.5.2.4 IS-GPS-200K) [s].

Definition at line 44 of file `gps_utc_model.h`.

10.174.3.7 i_DN

```
int32_t Gps_Utc_Model::i_DN {}
```

Day number (DN) at the end of which the leap second becomes effective [days].

Definition at line 48 of file `gps_utc_model.h`.

10.174.3.8 i_WN_LSF

```
int32_t Gps_Utc_Model::i_WN_LSF {}
```

Week number at the end of which the leap second becomes effective [weeks].

Definition at line 47 of file `gps_utc_model.h`.

10.174.3.9 i_WN_T

```
int32_t Gps_Utc_Model::i_WN_T {}
```

UTC reference week number [weeks].

Definition at line 45 of file `gps_utc_model.h`.

The documentation for this class was generated from the following file:

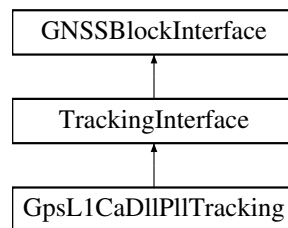
- [gps_utc_model.h](#)

10.175 GpsL1CaDllPllTracking Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <gps_l1_ca_dll_pll_tracking.h>
```

Inheritance diagram for GpsL1CaDllPllTracking:



Public Member Functions

- **GpsL1CaDllPllTracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override

Returns "GPS_L1_CA_DLL_PLL_Tracking".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override

Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override

Stop running tracking.

10.175.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 39 of file `gps_l1_ca_dll_pll_tracking.h`.

10.175.2 Member Function Documentation

10.175.2.1 `implementation()`

```
std::string GpsL1CaDllPllTracking::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L1_CA_DLL_PLL_Tracking".

Implements [GNSSBlockInterface](#).

Definition at line 56 of file `gps_l1_ca_dll_pll_tracking.h`.

10.175.2.2 `set_channel()`

```
void GpsL1CaDllPllTracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.175.2.3 `set_gnss_synchro()`

```
void GpsL1CaDllPllTracking::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.175.2.4 stop_tracking()

```
void GpsL1CaDllPllTracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

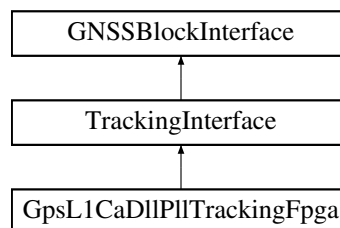
- [gps_l1_ca_dll_pll_tracking.h](#)

10.176 GpsL1CaDllPllTrackingFpga Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <gps_l1_ca_dll_pll_tracking_fpga.h>
```

Inheritance diagram for GpsL1CaDllPllTrackingFpga:



Public Member Functions

- [GpsL1CaDllPllTrackingFpga](#) (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
Constructor.
- virtual [~GpsL1CaDllPllTrackingFpga](#) ()
Destructor.
- std::string [role](#) () override
Role.
- std::string [implementation](#) () override
Returns "GPS_L1_CA_DLL_PLL_Tracking_Fpga".
- size_t [item_size](#) () override
Returns size of lv_16sc_t.
- void [connect](#) (gr::top_block_sptr top_block) override
Connect.
- void [disconnect](#) (gr::top_block_sptr top_block) override
Disconnect.
- gr::basic_block_sptr [get_left_block](#) () override
Get left block.
- gr::basic_block_sptr [get_right_block](#) () override
Get right block.

- void [set_channel](#) (unsigned int channel) override
Set tracking channel unique ID.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void [start_tracking](#) () override
Start the tracking process in the FPGA.
- void [stop_tracking](#) () override
Stop the tracking process in the FPGA.

10.176.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 38 of file `gps_l1_ca_dll_pll_tracking_fpga.h`.

10.176.2 Constructor & Destructor Documentation

10.176.2.1 GpsL1CaDllPllTrackingFpga()

```
GpsL1CaDllPllTrackingFpga::GpsL1CaDllPllTrackingFpga (
    const ConfigurationInterface * configuration,
    const std::string & role,
    unsigned int in_streams,
    unsigned int out_streams )
```

Constructor.

10.176.2.2 ~GpsL1CaDllPllTrackingFpga()

```
virtual GpsL1CaDllPllTrackingFpga::~~GpsL1CaDllPllTrackingFpga ( ) [virtual]
```

Destructor.

10.176.3 Member Function Documentation

10.176.3.1 connect()

```
void GpsL1CaDllPllTrackingFpga::connect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Connect.

Implements [GNSSBlockInterface](#).

10.176.3.2 disconnect()

```
void GpsL1CaDllPllTrackingFpga::disconnect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Disconnect.

Implements [GNSSBlockInterface](#).

10.176.3.3 get_left_block()

```
gr::basic_block_sptr GpsL1CaDllPllTrackingFpga::get_left_block ( ) [override], [virtual]
```

Get left block.

Implements [GNSSBlockInterface](#).

10.176.3.4 get_right_block()

```
gr::basic_block_sptr GpsL1CaDllPllTrackingFpga::get_right_block ( ) [override], [virtual]
```

Get right block.

Implements [GNSSBlockInterface](#).

10.176.3.5 implementation()

```
std::string GpsL1CaDllPllTrackingFpga::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L1_CA_DLL_PLL_Tracking_Fpga".

Implements [GNSSBlockInterface](#).

Definition at line 66 of file `gps_l1_ca_dll_pll_tracking_fpga.h`.

10.176.3.6 item_size()

```
size_t GpsL1CaDllPllTrackingFpga::item_size ( ) [inline], [override], [virtual]
```

Returns size of `lv_16sc_t`.

Implements [GNSSBlockInterface](#).

Definition at line 74 of file `gps_l1_ca_dll_pll_tracking_fpga.h`.

10.176.3.7 role()

```
std::string GpsL1CaDllPllTrackingFpga::role ( ) [inline], [override], [virtual]
```

Role.

Implements [GNSSBlockInterface](#).

Definition at line 58 of file `gps_l1_ca_dll_pll_tracking_fpga.h`.

10.176.3.8 set_channel()

```
void GpsL1CaDllPllTrackingFpga::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.176.3.9 set_gnss_synchro()

```
void GpsL1CaDllPllTrackingFpga::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.176.3.10 start_tracking()

```
void GpsL1CaDllPllTrackingFpga::start_tracking ( ) [override], [virtual]
```

Start the tracking process in the FPGA.

Implements [TrackingInterface](#).

10.176.3.11 stop_tracking()

```
void GpsL1CaDllPllTrackingFpga::stop_tracking ( ) [override], [virtual]
```

Stop the tracking process in the FPGA.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

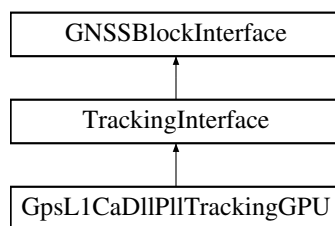
- [gps_l1_ca_dll_pll_tracking_fpga.h](#)

10.177 GpsL1CaDllPllTrackingGPU Class Reference

This class implements a code DLL + carrier PLL tracking loop using GPU accelerated functions.

```
#include <gps_l1_ca_dll_pll_tracking_gpu.h>
```

Inheritance diagram for GpsL1CaDllPllTrackingGPU:



Public Member Functions

- **GpsL1CaDllPllTrackingGPU** (const [ConfigurationInterface](#) *configuration, std::string role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "GPS_L1_CA_DLL_PLL_Tracking_GPU".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override
 - Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override
 - Stop running tracking.

10.177.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop using GPU accelerated functions.

Definition at line 39 of file `gps_l1_ca_dll_pll_tracking_gpu.h`.

10.177.2 Member Function Documentation

10.177.2.1 `implementation()`

```
std::string GpsL1CaDllPllTrackingGPU::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L1_CA_DLL_PLL_Tracking_GPU".

Implements [GNSSBlockInterface](#).

Definition at line 56 of file `gps_l1_ca_dll_pll_tracking_gpu.h`.

10.177.2.2 `set_channel()`

```
void GpsL1CaDllPllTrackingGPU::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.177.2.3 `set_gnss_synchro()`

```
void GpsL1CaDllPllTrackingGPU::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.177.2.4 stop_tracking()

```
void GpsL1CaDllPllTrackingGPU::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

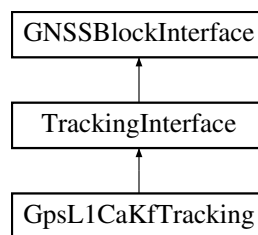
- [gps_l1_ca_dll_pll_tracking_gpu.h](#)

10.178 GpsL1CaKfTracking Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <gps_l1_ca_kf_tracking.h>
```

Inheritance diagram for GpsL1CaKfTracking:



Public Member Functions

- **GpsL1CaKfTracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in←_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "GPS_L1_CA_KF_Tracking".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override
Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override
Stop running tracking.

10.178.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 41 of file `gps_l1_ca_kf_tracking.h`.

10.178.2 Member Function Documentation

10.178.2.1 `implementation()`

```
std::string GpsL1CaKfTracking::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L1_CA_KF_Tracking".

Implements [GNSSBlockInterface](#).

Definition at line 58 of file `gps_l1_ca_kf_tracking.h`.

10.178.2.2 `set_channel()`

```
void GpsL1CaKfTracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.178.2.3 `set_gnss_synchro()`

```
void GpsL1CaKfTracking::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.178.2.4 stop_tracking()

```
void GpsL1CaKfTracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

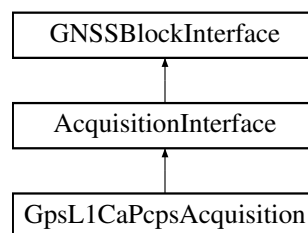
- [gps_l1_ca_kf_tracking.h](#)

10.179 GpsL1CaPcpsAcquisition Class Reference

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

```
#include <gps_l1_ca_pcps_acquisition.h>
```

Inheritance diagram for GpsL1CaPcpsAcquisition:



Public Member Functions

- **GpsL1CaPcpsAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "GPS_L1_CA_PCPS_Acquisition".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **set_channel** (unsigned int channel) override
Set acquisition channel unique ID.
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
Set channel fsm associated to this acquisition instance.
- void **set_threshold** (float threshold) override
Set statistics threshold of PCPS algorithm.
- void **set_doppler_max** (unsigned int doppler_max) override

- *Set maximum Doppler off grid search.*
• void [set_doppler_step](#) (unsigned int doppler_step) override
- *Set Doppler steps for the grid search.*
• void [set_doppler_center](#) (int doppler_center) override
- *Set Doppler center for the grid search.*
• void [init](#) () override
- *Initializes acquisition algorithm.*
• void [set_local_code](#) () override
- *Sets local code for GPS L1/CA PCPS acquisition algorithm.*
• signed int [mag](#) () override
- *Returns the maximum peak of grid search.*
• void [reset](#) () override
- *Restart acquisition algorithm.*
• void [set_state](#) (int state) override
- *If state = 1, it forces the block to start acquiring from the first sample.*
• void [stop_acquisition](#) () override
- *Stop running acquisition.*
• void [set_resampler_latency](#) (uint32_t latency_samples) override
- *Sets the resampler latency to account it in the acquisition code delay estimation.*

10.179.1 Detailed Description

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Definition at line 45 of file `gps_l1_ca_pcps_acquisition.h`.

10.179.2 Member Function Documentation

10.179.2.1 implementation()

```
std::string GpsL1CaPcpsAcquisition::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L1_CA_PCPS_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 64 of file `gps_l1_ca_pcps_acquisition.h`.

10.179.2.2 init()

```
void GpsL1CaPcpsAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.179.2.3 mag()

```
signed int GpsL1CaPcpsAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.179.2.4 reset()

```
void GpsL1CaPcpsAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.179.2.5 set_channel()

```
void GpsL1CaPcpsAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 89 of file `gps_l1_ca_pcps_acquisition.h`.

10.179.2.6 set_channel_fsm()

```
void GpsL1CaPcpsAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 98 of file `gps_l1_ca_pcps_acquisition.h`.

10.179.2.7 set_doppler_center()

```
void GpsL1CaPcpsAcquisition::set_doppler_center (
    int doppler_center ) [override]
```

Set Doppler center for the grid search.

10.179.2.8 `set_doppler_max()`

```
void GpsL1CaPcpsAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.179.2.9 `set_doppler_step()`

```
void GpsL1CaPcpsAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.179.2.10 `set_gnss_synchro()`

```
void GpsL1CaPcpsAcquisition::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.179.2.11 `set_local_code()`

```
void GpsL1CaPcpsAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local code for GPS L1/CA PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.179.2.12 `set_resampler_latency()`

```
void GpsL1CaPcpsAcquisition::set_resampler_latency (
    uint32_t latency_samples ) [override], [virtual]
```

Sets the resampler latency to account it in the acquisition code delay estimation.

Implements [AcquisitionInterface](#).

10.179.2.13 set_state()

```
void GpsL1CaPcpsAcquisition::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.179.2.14 set_threshold()

```
void GpsL1CaPcpsAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.179.2.15 stop_acquisition()

```
void GpsL1CaPcpsAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

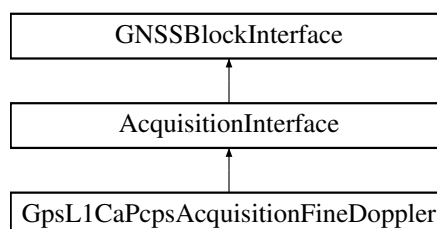
- [gps_l1_ca_pcps_acquisition.h](#)

10.180 GpsL1CaPcpsAcquisitionFineDoppler Class Reference

This class Adapts a PCPS acquisition block with fine Doppler estimation to an [AcquisitionInterface](#) for GPS L1 C/A signals.

```
#include <gps_l1_ca_pcps_acquisition_fine_doppler.h>
```

Inheritance diagram for GpsL1CaPcpsAcquisitionFineDoppler:



Public Member Functions

- **GpsL1CaPcpsAcquisitionFineDoppler** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "GPS_L1_CA_PCPS_Acquisition_Fine_Doppler".*
- size_t **item_size** () override
- void **connect** (boost::shared_ptr< gr::top_block > top_block) override
- void **disconnect** (boost::shared_ptr< gr::top_block > top_block) override
- boost::shared_ptr< gr::basic_block > **get_left_block** () override
- boost::shared_ptr< gr::basic_block > **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.*
- void **set_channel** (unsigned int channel) override
 - Set acquisition channel unique ID.*
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
 - Set channel fsm associated to this acquisition instance.*
- void **set_threshold** (float threshold) override
 - Set statistics threshold of PCPS algorithm.*
- void **set_doppler_max** (unsigned int doppler_max) override
 - Set maximum Doppler off grid search.*
- void **set_doppler_step** (unsigned int doppler_step) override
 - Set Doppler steps for the grid search.*
- void **init** () override
 - Initializes acquisition algorithm.*
- void **set_local_code** () override
- signed int **mag** () override
 - Returns the maximum peak of grid search.*
- void **reset** () override
 - Restart acquisition algorithm.*
- void **set_state** (int state) override
 - If state = 1, it forces the block to start acquiring from the first sample.*
- void **stop_acquisition** () override
 - Stop running acquisition.*
- void **set_resampler_latency** (uint32_t latency_samples __attribute__((unused))) override

10.180.1 Detailed Description

This class Adapts a PCPS acquisition block with fine Doppler estimation to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Definition at line 45 of file `gps_l1_ca_pcps_acquisition_fine_doppler.h`.

10.180.2 Member Function Documentation

10.180.2.1 implementation()

```
std::string GpsL1CaPcpsAcquisitionFineDoppler::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L1_CA_PCPS_Acquisition_Fine_Doppler".

Implements [GNSSBlockInterface](#).

Definition at line 63 of file `gps_l1_ca_pcps_acquisition_fine_doppler.h`.

10.180.2.2 init()

```
void GpsL1CaPcpsAcquisitionFineDoppler::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.180.2.3 mag()

```
signed int GpsL1CaPcpsAcquisitionFineDoppler::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.180.2.4 reset()

```
void GpsL1CaPcpsAcquisitionFineDoppler::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.180.2.5 set_channel()

```
void GpsL1CaPcpsAcquisitionFineDoppler::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 94 of file `gps_l1_ca_pcps_acquisition_fine_doppler.h`.

10.180.2.6 set_channel_fsm()

```
void GpsL1CaPcpsAcquisitionFineDoppler::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 103 of file `gps_l1_ca_pcps_acquisition_fine_doppler.h`.

10.180.2.7 set_doppler_max()

```
void GpsL1CaPcpsAcquisitionFineDoppler::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.180.2.8 set_doppler_step()

```
void GpsL1CaPcpsAcquisitionFineDoppler::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.180.2.9 set_gnss_synchro()

```
void GpsL1CaPcpsAcquisitionFineDoppler::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.180.2.10 set_state()

```
void GpsL1CaPcpsAcquisitionFineDoppler::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.180.2.11 set_threshold()

```
void GpsL1CaPcpsAcquisitionFineDoppler::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.180.2.12 stop_acquisition()

```
void GpsL1CaPcpsAcquisitionFineDoppler::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

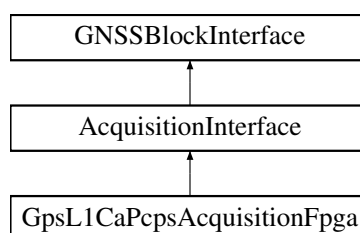
- [gps_l1_ca_pcps_acquisition_fine_doppler.h](#)

10.181 GpsL1CaPcpsAcquisitionFpga Class Reference

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for GPS L1 C/A signals.

```
#include <gps_l1_ca_pcps_acquisition_fpga.h>
```

Inheritance diagram for GpsL1CaPcpsAcquisitionFpga:



Public Member Functions

- [GpsL1CaPcpsAcquisitionFpga](#) (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
Constructor.
- [~GpsL1CaPcpsAcquisitionFpga](#) ()=default
Destructor.
- std::string [role](#) () override
Role.
- std::string [implementation](#) () override
Returns "GPS_L1_CA_PCPS_Acquisition_Fpga".
- size_t [item_size](#) () override
Returns size of lv_16sc_t.
- void [connect](#) (gr::top_block_sptr top_block) override
Connect.
- void [disconnect](#) (gr::top_block_sptr top_block) override
Disconnect.
- gr::basic_block_sptr [get_left_block](#) () override
Get left block.
- gr::basic_block_sptr [get_right_block](#) () override
Get right block.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void [set_channel](#) (unsigned int channel) override
Set acquisition channel unique ID.
- void [set_channel_fsm](#) (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
Set channel fsm associated to this acquisition instance.
- void [set_threshold](#) (float threshold) override
Set statistics threshold of PCPS algorithm.
- void [set_doppler_max](#) (unsigned int doppler_max) override
Set maximum Doppler off grid search.
- void [set_doppler_step](#) (unsigned int doppler_step) override
Set Doppler steps for the grid search.
- void [set_doppler_center](#) (int doppler_center) override
Set Doppler center for the grid search.
- void [init](#) () override
Initializes acquisition algorithm.
- void [set_local_code](#) () override
Sets local code for GPS L1/CA PCPS acquisition algorithm.
- signed int [mag](#) () override
Returns the maximum peak of grid search.
- void [reset](#) () override
Restart acquisition algorithm.
- void [set_state](#) (int state) override
If state = 1, it forces the block to start acquiring from the first sample.
- void [stop_acquisition](#) () override
Stop running acquisition.
- void [set_resampler_latency](#) (uint32_t latency_samples __attribute__((unused))) override
Set Resampler Latency.

10.181.1 Detailed Description

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Definition at line 42 of file `gps_l1_ca_pcps_acquisition_fpga.h`.

10.181.2 Constructor & Destructor Documentation

10.181.2.1 GpsL1CaPcpsAcquisitionFpga()

```
GpsL1CaPcpsAcquisitionFpga::GpsL1CaPcpsAcquisitionFpga (
    const ConfigurationInterface * configuration,
    const std::string & role,
    unsigned int in_streams,
    unsigned int out_streams )
```

Constructor.

10.181.2.2 ~GpsL1CaPcpsAcquisitionFpga()

```
GpsL1CaPcpsAcquisitionFpga::~GpsL1CaPcpsAcquisitionFpga ( ) [default]
```

Destructor.

10.181.3 Member Function Documentation

10.181.3.1 connect()

```
void GpsL1CaPcpsAcquisitionFpga::connect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Connect.

Implements [GNSSBlockInterface](#).

10.181.3.2 disconnect()

```
void GpsL1CaPcpsAcquisitionFpga::disconnect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Disconnect.

Implements [GNSSBlockInterface](#).

10.181.3.3 get_left_block()

```
gr::basic_block_sptr GpsL1CaPcpsAcquisitionFpga::get_left_block ( ) [override], [virtual]
```

Get left block.

Implements [GNSSBlockInterface](#).

10.181.3.4 get_right_block()

```
gr::basic_block_sptr GpsL1CaPcpsAcquisitionFpga::get_right_block ( ) [override], [virtual]
```

Get right block.

Implements [GNSSBlockInterface](#).

10.181.3.5 implementation()

```
std::string GpsL1CaPcpsAcquisitionFpga::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L1_CA_PCPS_Acquisition_Fpga".

Implements [GNSSBlockInterface](#).

Definition at line 69 of file `gps_l1_ca_pcps_acquisition_fpga.h`.

10.181.3.6 init()

```
void GpsL1CaPcpsAcquisitionFpga::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.181.3.7 item_size()

```
size_t GpsL1CaPcpsAcquisitionFpga::item_size ( ) [inline], [override], [virtual]
```

Returns size of lv_16sc_t.

Implements [GNSSBlockInterface](#).

Definition at line 77 of file gps_l1_ca_pcps_acquisition_fpga.h.

10.181.3.8 mag()

```
signed int GpsL1CaPcpsAcquisitionFpga::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.181.3.9 reset()

```
void GpsL1CaPcpsAcquisitionFpga::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.181.3.10 role()

```
std::string GpsL1CaPcpsAcquisitionFpga::role ( ) [inline], [override], [virtual]
```

Role.

Implements [GNSSBlockInterface](#).

Definition at line 61 of file gps_l1_ca_pcps_acquisition_fpga.h.

10.181.3.11 set_channel()

```
void GpsL1CaPcpsAcquisitionFpga::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 112 of file gps_l1_ca_pcps_acquisition_fpga.h.

10.181.3.12 `set_channel_fsm()`

```
void GpsL1CaPcpsAcquisitionFpga::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 121 of file `gps_l1_ca_pcps_acquisition_fpga.h`.

10.181.3.13 `set_doppler_center()`

```
void GpsL1CaPcpsAcquisitionFpga::set_doppler_center (
    int doppler_center ) [override]
```

Set Doppler center for the grid search.

10.181.3.14 `set_doppler_max()`

```
void GpsL1CaPcpsAcquisitionFpga::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.181.3.15 `set_doppler_step()`

```
void GpsL1CaPcpsAcquisitionFpga::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.181.3.16 `set_gnss_synchro()`

```
void GpsL1CaPcpsAcquisitionFpga::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.181.3.17 set_local_code()

```
void GpsL1CaPcpsAcquisitionFpga::set_local_code ( ) [override], [virtual]
```

Sets local code for GPS L1/CA PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.181.3.18 set_resampler_latency()

```
void GpsL1CaPcpsAcquisitionFpga::set_resampler_latency (
    uint32_t latency_samples __attribute__((unused)) ) [inline], [override]
```

Set Resampler Latency.

Definition at line 180 of file `gps_l1_ca_pcps_acquisition_fpga.h`.

10.181.3.19 set_state()

```
void GpsL1CaPcpsAcquisitionFpga::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.181.3.20 set_threshold()

```
void GpsL1CaPcpsAcquisitionFpga::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.181.3.21 stop_acquisition()

```
void GpsL1CaPcpsAcquisitionFpga::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

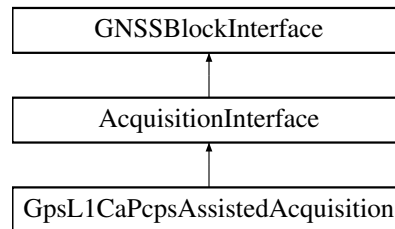
- [gps_l1_ca_pcps_acquisition_fpga.h](#)

10.182 GpsL1CaPcpsAssistedAcquisition Class Reference

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

```
#include <gps_l1_ca_pcps_assisted_acquisition.h>
```

Inheritance diagram for GpsL1CaPcpsAssistedAcquisition:



Public Member Functions

- **GpsL1CaPcpsAssistedAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "GPS_L1_CA_PCPS_Assisted_Acquisition".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.*
- void **set_channel** (unsigned int channel) override
 - Set acquisition channel unique ID.*
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
 - Set channel fsm associated to this acquisition instance.*
- void **set_threshold** (float threshold) override
 - Set statistics threshold of PCPS algorithm.*
- void **set_doppler_max** (unsigned int doppler_max) override
 - Set maximum Doppler off grid search.*
- void **set_doppler_step** (unsigned int doppler_step) override
 - Set Doppler steps for the grid search.*
- void **init** () override
 - Initializes acquisition algorithm.*
- void **set_local_code** () override
- signed int **mag** () override
 - Returns the maximum peak of grid search.*
- void **reset** () override
 - Restart acquisition algorithm.*
- void **set_state** (int state __attribute__((unused))) override
- void **stop_acquisition** () override
 - Stop running acquisition.*
- void **set_resampler_latency** (uint32_t latency_samples __attribute__((unused))) override

10.182.1 Detailed Description

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Definition at line 39 of file `gps_l1_ca_pcps_assisted_acquisition.h`.

10.182.2 Member Function Documentation

10.182.2.1 implementation()

```
std::string GpsL1CaPcpsAssistedAcquisition::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L1_CA_PCPS_Assisted_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 58 of file `gps_l1_ca_pcps_assisted_acquisition.h`.

10.182.2.2 init()

```
void GpsL1CaPcpsAssistedAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.182.2.3 mag()

```
signed int GpsL1CaPcpsAssistedAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.182.2.4 reset()

```
void GpsL1CaPcpsAssistedAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.182.2.5 `set_channel()`

```
void GpsL1CaPcpsAssistedAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 83 of file `gps_l1_ca_pcps_assisted_acquisition.h`.

10.182.2.6 `set_channel_fsm()`

```
void GpsL1CaPcpsAssistedAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 92 of file `gps_l1_ca_pcps_assisted_acquisition.h`.

10.182.2.7 `set_doppler_max()`

```
void GpsL1CaPcpsAssistedAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.182.2.8 `set_doppler_step()`

```
void GpsL1CaPcpsAssistedAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.182.2.9 set_gnss_synchro()

```
void GpsL1CaPcpsAssistedAcquisition::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.182.2.10 set_threshold()

```
void GpsL1CaPcpsAssistedAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.182.2.11 stop_acquisition()

```
void GpsL1CaPcpsAssistedAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

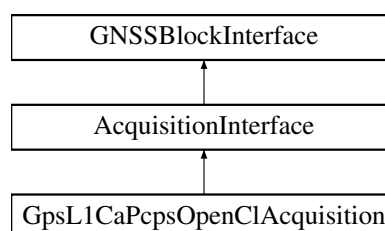
- [gps_l1_ca_pcps_assisted_acquisition.h](#)

10.183 GpsL1CaPcpsOpenCIAcquisition Class Reference

This class adapts an OpenCL PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

```
#include <gps_l1_ca_pcps_openc1_acquisition.h>
```

Inheritance diagram for GpsL1CaPcpsOpenCIAcquisition:



Public Member Functions

- **GpsL1CaPcpsOpenClAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "GPS_L1_CA_PCPS_OpenCl_Acquisition".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.*
- void **set_channel** (unsigned int channel) override
 - Set acquisition channel unique ID.*
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
 - Set channel fsm associated to this acquisition instance.*
- void **set_threshold** (float threshold) override
 - Set statistics threshold of PCPS algorithm.*
- void **set_doppler_max** (unsigned int doppler_max) override
 - Set maximum Doppler off grid search.*
- void **set_doppler_step** (unsigned int doppler_step) override
 - Set Doppler steps for the grid search.*
- void **init** () override
 - Initializes acquisition algorithm.*
- void **set_local_code** () override
 - Sets local code for GPS L1/CA PCPS acquisition algorithm.*
- signed int **mag** () override
 - Returns the maximum peak of grid search.*
- void **reset** () override
 - Restart acquisition algorithm.*
- void **set_state** (int state __attribute__((unused))) override
- void **stop_acquisition** () override
 - Stop running acquisition.*
- void **set_resampler_latency** (uint32_t latency_samples __attribute__((unused))) override
- bool **opencl_ready** () const

10.183.1 Detailed Description

This class adapts an OpenCL PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Definition at line 38 of file `gps_l1_ca_pcps_opencl_acquisition.h`.

10.183.2 Member Function Documentation

10.183.2.1 implementation()

```
std::string GpsL1CaPcpsOpenClAcquisition::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L1_CA_PCPS_OpenCl_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 56 of file `gps_l1_ca_pcps_opencl_acquisition.h`.

10.183.2.2 init()

```
void GpsL1CaPcpsOpenClAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.183.2.3 mag()

```
signed int GpsL1CaPcpsOpenClAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.183.2.4 reset()

```
void GpsL1CaPcpsOpenClAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.183.2.5 set_channel()

```
void GpsL1CaPcpsOpenClAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 81 of file `gps_l1_ca_pcps_opencl_acquisition.h`.

10.183.2.6 set_channel_fsm()

```
void GpsL1CaPcpsOpenClAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 90 of file `gps_l1_ca_pcps_openc1_acquisition.h`.

10.183.2.7 set_doppler_max()

```
void GpsL1CaPcpsOpenClAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.183.2.8 set_doppler_step()

```
void GpsL1CaPcpsOpenClAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.183.2.9 set_gnss_synchro()

```
void GpsL1CaPcpsOpenClAcquisition::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.183.2.10 set_local_code()

```
void GpsL1CaPcpsOpenClAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local code for GPS L1/CA PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.183.2.11 set_threshold()

```
void GpsL1CaPcpsOpenClAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.183.2.12 stop_acquisition()

```
void GpsL1CaPcpsOpenClAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

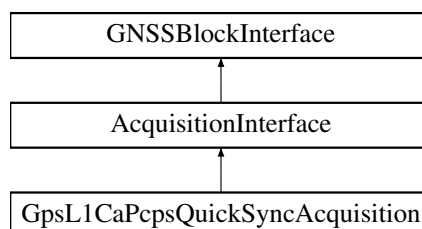
- [gps_l1_ca_pcps_openc1_acquisition.h](#)

10.184 GpsL1CaPcpsQuickSyncAcquisition Class Reference

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

```
#include <gps_l1_ca_pcps_quicksync_acquisition.h>
```

Inheritance diagram for GpsL1CaPcpsQuickSyncAcquisition:



Public Member Functions

- **GpsL1CaPcpsQuickSyncAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "GPS_L1_CA_PCPS_QuickSync_Acquisition".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.*
- void **set_channel** (unsigned int channel) override
 - Set acquisition channel unique ID.*
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
 - Set channel fsm associated to this acquisition instance.*
- void **set_threshold** (float threshold) override
 - Set statistics threshold of PCPS algorithm.*
- void **set_doppler_max** (unsigned int doppler_max) override
 - Set maximum Doppler off grid search.*
- void **set_doppler_step** (unsigned int doppler_step) override
 - Set Doppler steps for the grid search.*
- void **init** () override
 - Initializes acquisition algorithm.*
- void **set_local_code** () override
 - Sets local code for GPS L1/CA PCPS acquisition algorithm.*
- signed int **mag** () override
 - Returns the maximum peak of grid search.*
- void **reset** () override
 - Restart acquisition algorithm.*
- void **set_state** (int state) override
 - If state = 1, it forces the block to start acquiring from the first sample.*
- void **stop_acquisition** () override
 - Stop running acquisition.*
- void **set_resampler_latency** (uint32_t latency_samples __attribute__((unused))) override

10.184.1 Detailed Description

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Definition at line 40 of file `gps_l1_ca_pcps_quicksync_acquisition.h`.

10.184.2 Member Function Documentation

10.184.2.1 implementation()

```
std::string GpsL1CaPcpsQuickSyncAcquisition::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L1_CA_PCPS_QuickSync_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 59 of file `gps_l1_ca_pcps_quicksync_acquisition.h`.

10.184.2.2 init()

```
void GpsL1CaPcpsQuickSyncAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.184.2.3 mag()

```
signed int GpsL1CaPcpsQuickSyncAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.184.2.4 reset()

```
void GpsL1CaPcpsQuickSyncAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.184.2.5 set_channel()

```
void GpsL1CaPcpsQuickSyncAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 84 of file `gps_l1_ca_pcps_quicksync_acquisition.h`.

10.184.2.6 set_channel_fsm()

```
void GpsL1CaPcpsQuickSyncAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 93 of file gps_l1_ca_pcps_quicksync_acquisition.h.

10.184.2.7 set_doppler_max()

```
void GpsL1CaPcpsQuickSyncAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.184.2.8 set_doppler_step()

```
void GpsL1CaPcpsQuickSyncAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.184.2.9 set_gnss_synchro()

```
void GpsL1CaPcpsQuickSyncAcquisition::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.184.2.10 `set_local_code()`

```
void GpsL1CaPcpsQuickSyncAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local code for GPS L1/CA PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.184.2.11 `set_state()`

```
void GpsL1CaPcpsQuickSyncAcquisition::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.184.2.12 `set_threshold()`

```
void GpsL1CaPcpsQuickSyncAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.184.2.13 `stop_acquisition()`

```
void GpsL1CaPcpsQuickSyncAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

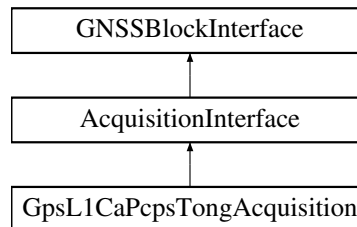
- [gps_l1_ca_pcps_quicksync_acquisition.h](#)

10.185 GpsL1CaPcpsTongAcquisition Class Reference

This class adapts a PCPS Tong acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

```
#include <gps_l1_ca_pcps_tong_acquisition.h>
```

Inheritance diagram for GpsL1CaPcpsTongAcquisition:



Public Member Functions

- **GpsL1CaPcpsTongAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "GPS_L1_CA_PCPS_Tong_Acquisition".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.*
- void **set_channel** (unsigned int channel) override
 - Set acquisition channel unique ID.*
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
 - Set channel fsm associated to this acquisition instance.*
- void **set_threshold** (float threshold) override
 - Set statistics threshold of TONG algorithm.*
- void **set_doppler_max** (unsigned int doppler_max) override
 - Set maximum Doppler off grid search.*
- void **set_doppler_step** (unsigned int doppler_step) override
 - Set Doppler steps for the grid search.*
- void **init** () override
 - Initializes acquisition algorithm.*
- void **set_local_code** () override
 - Sets local code for GPS L1/CA TONG acquisition algorithm.*
- signed int **mag** () override
 - Returns the maximum peak of grid search.*
- void **reset** () override
 - Restart acquisition algorithm.*
- void **set_state** (int state) override
 - If state = 1, it forces the block to start acquiring from the first sample.*
- void **stop_acquisition** () override
 - Stop running acquisition.*
- void **set_resampler_latency** (uint32_t latency_samples __attribute__((unused))) override

10.185.1 Detailed Description

This class adapts a PCPS Tong acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Definition at line 39 of file `gps_l1_ca_pcps_tong_acquisition.h`.

10.185.2 Member Function Documentation

10.185.2.1 implementation()

```
std::string GpsL1CaPcpsTongAcquisition::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L1_CA_PCPS_Tong_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 57 of file `gps_l1_ca_pcps_tong_acquisition.h`.

10.185.2.2 init()

```
void GpsL1CaPcpsTongAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.185.2.3 mag()

```
signed int GpsL1CaPcpsTongAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.185.2.4 reset()

```
void GpsL1CaPcpsTongAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.185.2.5 `set_channel()`

```
void GpsL1CaPcpsTongAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 82 of file `gps_l1_ca_pcps_tong_acquisition.h`.

10.185.2.6 `set_channel_fsm()`

```
void GpsL1CaPcpsTongAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 91 of file `gps_l1_ca_pcps_tong_acquisition.h`.

10.185.2.7 `set_doppler_max()`

```
void GpsL1CaPcpsTongAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.185.2.8 `set_doppler_step()`

```
void GpsL1CaPcpsTongAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.185.2.9 `set_gnss_synchro()`

```
void GpsL1CaPcpsTongAcquisition::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.185.2.10 `set_local_code()`

```
void GpsL1CaPcpsTongAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local code for GPS L1/CA TONG acquisition algorithm.

Implements [AcquisitionInterface](#).

10.185.2.11 `set_state()`

```
void GpsL1CaPcpsTongAcquisition::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.185.2.12 `set_threshold()`

```
void GpsL1CaPcpsTongAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of TONG algorithm.

Implements [AcquisitionInterface](#).

10.185.2.13 `stop_acquisition()`

```
void GpsL1CaPcpsTongAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

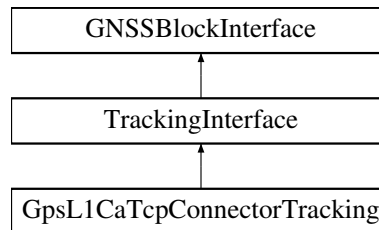
- [gps_l1_ca_pcps_tong_acquisition.h](#)

10.186 GpsL1CaTcpConnectorTracking Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <gps_l1_ca_tcp_connector_tracking.h>
```

Inheritance diagram for GpsL1CaTcpConnectorTracking:



Public Member Functions

- **GpsL1CaTcpConnectorTracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "GPS_L1_CA_TCP_CONNECTOR_Tracking".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override
Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override
Stop running tracking.

10.186.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 40 of file `gps_l1_ca_tcp_connector_tracking.h`.

10.186.2 Member Function Documentation

10.186.2.1 implementation()

```
std::string GpsL1CaTcpConnectorTracking::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L1_CA_TCP_CONNECTOR_Tracking".

Implements [GNSSBlockInterface](#).

Definition at line 57 of file `gps_l1_ca_tcp_connector_tracking.h`.

10.186.2.2 set_channel()

```
void GpsL1CaTcpConnectorTracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.186.2.3 set_gnss_synchro()

```
void GpsL1CaTcpConnectorTracking::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.186.2.4 stop_tracking()

```
void GpsL1CaTcpConnectorTracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

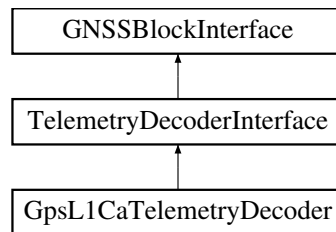
- [gps_l1_ca_tcp_connector_tracking.h](#)

10.187 GpsL1CaTelemetryDecoder Class Reference

This class implements a NAV data decoder for GPS L1 C/A.

```
#include <gps_l1_ca_telemetry_decoder.h>
```

Inheritance diagram for GpsL1CaTelemetryDecoder:



Public Member Functions

- **GpsL1CaTelemetryDecoder** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_satellite** (const [Gnss_Satellite](#) &satellite) override
- std::string **role** () override
- std::string **implementation** () override
 - Returns "GPS_L1_CA_Telemetry_Decoder".*
- void **set_channel** (int channel) override
- void **reset** () override
- size_t **item_size** () override

10.187.1 Detailed Description

This class implements a NAV data decoder for GPS L1 C/A.

Definition at line 38 of file `gps_l1_ca_telemetry_decoder.h`.

10.187.2 Member Function Documentation

10.187.2.1 implementation()

```
std::string GpsL1CaTelemetryDecoder::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L1_CA_Telemetry_Decoder".

Implements [GNSSBlockInterface](#).

Definition at line 62 of file `gps_l1_ca_telemetry_decoder.h`.

The documentation for this class was generated from the following file:

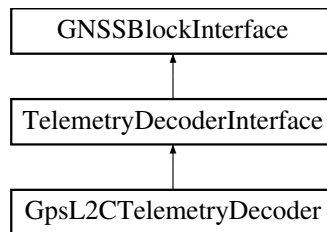
- [gps_l1_ca_telemetry_decoder.h](#)

10.188 GpsL2CTelemetryDecoder Class Reference

This class implements a NAV data decoder for GPS L2 M.

```
#include <gps_l2c_telemetry_decoder.h>
```

Inheritance diagram for GpsL2CTelemetryDecoder:



Public Member Functions

- **GpsL2CTelemetryDecoder** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "GPS_L2C_Telemetry_Decoder".
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_satellite** (const [Gnss_Satellite](#) &satellite) override
- void **set_channel** (int channel) override
- void **reset** () override
- size_t **item_size** () override

10.188.1 Detailed Description

This class implements a NAV data decoder for GPS L2 M.

Definition at line 38 of file `gps_l2c_telemetry_decoder.h`.

10.188.2 Member Function Documentation

10.188.2.1 implementation()

```
std::string GpsL2CTelemetryDecoder::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L2C_Telemetry_Decoder".

Implements [GNSSBlockInterface](#).

Definition at line 55 of file `gps_l2c_telemetry_decoder.h`.

The documentation for this class was generated from the following file:

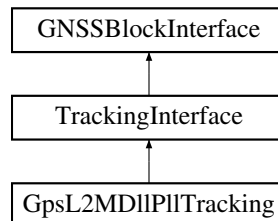
- [gps_l2c_telemetry_decoder.h](#)

10.189 GpsL2MDIIPITracking Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <gps_l2_m_dll_pll_tracking.h>
```

Inheritance diagram for GpsL2MDIIPITracking:



Public Member Functions

- **GpsL2MDIIPITracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "GPS_L2_M_DLL_PLL_Tracking".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override
Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override
Stop running tracking.

10.189.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 39 of file `gps_l2_m_dll_pll_tracking.h`.

10.189.2 Member Function Documentation

10.189.2.1 implementation()

```
std::string GpsL2MD1lP1lTracking::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L2_M_DLL_PLL_Tracking".

Implements [GNSSBlockInterface](#).

Definition at line 56 of file `gps_l2_m_dll_pll_tracking.h`.

10.189.2.2 set_channel()

```
void GpsL2MD1lP1lTracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.189.2.3 set_gnss_synchro()

```
void GpsL2MD1lP1lTracking::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.189.2.4 stop_tracking()

```
void GpsL2MD1lP1lTracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

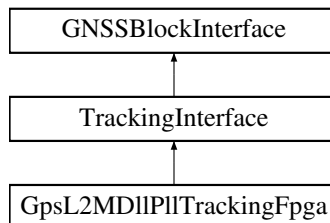
- [gps_l2_m_dll_pll_tracking.h](#)

10.190 GpsL2MDIIPITrackingFpga Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <gps_l2_m_dll_pll_tracking_fpga.h>
```

Inheritance diagram for GpsL2MDIIPITrackingFpga:



Public Member Functions

- **GpsL2MDIIPITrackingFpga** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "GPS_L2_M_DLL_PLL_Tracking_Fpga".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override
Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override
Stop running tracking.

10.190.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 41 of file `gps_l2_m_dll_pll_tracking_fpga.h`.

10.190.2 Member Function Documentation

10.190.2.1 implementation()

```
std::string GpsL2MDIIPllTrackingFpga::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L2_M_DLL_PLL_Tracking_Fpga".

Implements [GNSSBlockInterface](#).

Definition at line 58 of file `gps_l2_m_dll_pll_tracking_fpga.h`.

10.190.2.2 set_channel()

```
void GpsL2MDIIPllTrackingFpga::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.190.2.3 set_gnss_synchro()

```
void GpsL2MDIIPllTrackingFpga::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.190.2.4 stop_tracking()

```
void GpsL2MDIIPllTrackingFpga::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

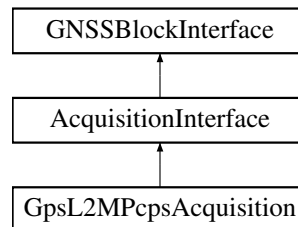
- [gps_l2_m_dll_pll_tracking_fpga.h](#)

10.191 GpsL2MPcpsAcquisition Class Reference

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L2 M signals.

```
#include <gps_l2_m_pcps_acquisition.h>
```

Inheritance diagram for GpsL2MPcpsAcquisition:



Public Member Functions

- **GpsL2MPcpsAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "GPS_L2_M_PCPS_Acquisition".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.*
- void **set_channel** (unsigned int channel) override
 - Set acquisition channel unique ID.*
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
 - Set channel fsm associated to this acquisition instance.*
- void **set_threshold** (float threshold) override
 - Set statistics threshold of PCPS algorithm.*
- void **set_doppler_max** (unsigned int doppler_max) override
 - Set maximum Doppler off grid search.*
- void **set_doppler_step** (unsigned int doppler_step) override
 - Set Doppler steps for the grid search.*
- void **set_doppler_center** (int doppler_center) override
 - Set Doppler center for the grid search.*
- void **init** () override
 - Initializes acquisition algorithm.*
- void **set_local_code** () override
 - Sets local code for GPS L2/M PCPS acquisition algorithm.*
- signed int **mag** () override
 - Returns the maximum peak of grid search.*
- void **reset** () override
 - Restart acquisition algorithm.*

- void [set_state](#) (int state) override
If state = 1, it forces the block to start acquiring from the first sample.
- void [stop_acquisition](#) () override
Stop running acquisition.
- void [set_resampler_latency](#) (uint32_t latency_samples) override
Sets the resampler latency to account it in the acquisition code delay estimation.

10.191.1 Detailed Description

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L2 M signals.

Definition at line 42 of file `gps_l2_m_pcps_acquisition.h`.

10.191.2 Member Function Documentation

10.191.2.1 implementation()

```
std::string GpsL2MPcpsAcquisition::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L2_M_PCPS_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 61 of file `gps_l2_m_pcps_acquisition.h`.

10.191.2.2 init()

```
void GpsL2MPcpsAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.191.2.3 mag()

```
signed int GpsL2MPcpsAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.191.2.4 reset()

```
void GpsL2MPcpsAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.191.2.5 set_channel()

```
void GpsL2MPcpsAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 86 of file `gps_l2_m_pcps_acquisition.h`.

10.191.2.6 set_channel_fsm()

```
void GpsL2MPcpsAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 95 of file `gps_l2_m_pcps_acquisition.h`.

10.191.2.7 set_doppler_center()

```
void GpsL2MPcpsAcquisition::set_doppler_center (
    int doppler_center ) [override]
```

Set Doppler center for the grid search.

10.191.2.8 set_doppler_max()

```
void GpsL2MPcpsAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.191.2.9 set_doppler_step()

```
void GpsL2MPcpsAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.191.2.10 set_gnss_synchro()

```
void GpsL2MPcpsAcquisition::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.191.2.11 set_local_code()

```
void GpsL2MPcpsAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local code for GPS L2/M PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.191.2.12 set_resampler_latency()

```
void GpsL2MPcpsAcquisition::set_resampler_latency (
    uint32_t latency_samples ) [override], [virtual]
```

Sets the resampler latency to account it in the acquisition code delay estimation.

Implements [AcquisitionInterface](#).

10.191.2.13 set_state()

```
void GpsL2MPcpsAcquisition::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.191.2.14 `set_threshold()`

```
void GpsL2MPcpsAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.191.2.15 `stop_acquisition()`

```
void GpsL2MPcpsAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

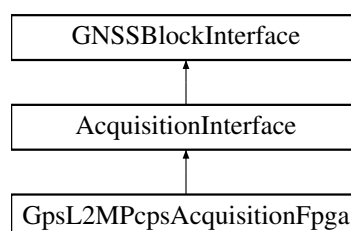
- [gps_l2_m_pcps_acquisition.h](#)

10.192 GpsL2MPcpsAcquisitionFpga Class Reference

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for GPS L2 M signals.

```
#include <gps_l2_m_pcps_acquisition_fpga.h>
```

Inheritance diagram for GpsL2MPcpsAcquisitionFpga:



Public Member Functions

- **GpsL2MPcpsAcquisitionFpga** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "GPS_L2_M_PCPS_Acquisition_Fpga".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
 - Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.*
- void **set_channel** (unsigned int channel) override
 - Set acquisition channel unique ID.*
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
 - Set channel fsm associated to this acquisition instance.*
- void **set_threshold** (float threshold) override
 - Set statistics threshold of PCPS algorithm.*
- void **set_doppler_max** (unsigned int doppler_max) override
 - Set maximum Doppler off grid search.*
- void **set_doppler_step** (unsigned int doppler_step) override
 - Set Doppler steps for the grid search.*
- void **init** () override
 - Initializes acquisition algorithm.*
- void **set_local_code** () override
 - Sets local code for GPS L2/M PCPS acquisition algorithm.*
- signed int **mag** () override
 - Returns the maximum peak of grid search.*
- void **reset** () override
 - Restart acquisition algorithm.*
- void **set_state** (int state) override
 - If state = 1, it forces the block to start acquiring from the first sample.*
- void **stop_acquisition** () override
 - Stop running acquisition.*
- void **set_resampler_latency** (uint32_t latency_samples __attribute__((unused))) override

10.192.1 Detailed Description

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for GPS L2 M signals.

Definition at line 41 of file `gps_l2_m_pcps_acquisition_fpga.h`.

10.192.2 Member Function Documentation

10.192.2.1 implementation()

```
std::string GpsL2MPcpsAcquisitionFpga::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L2_M_PCPS_Acquisition_Fpga".

Implements [GNSSBlockInterface](#).

Definition at line 60 of file `gps_l2_m_pcps_acquisition_fpga.h`.

10.192.2.2 init()

```
void GpsL2MPcpsAcquisitionFpga::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.192.2.3 mag()

```
signed int GpsL2MPcpsAcquisitionFpga::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.192.2.4 reset()

```
void GpsL2MPcpsAcquisitionFpga::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.192.2.5 set_channel()

```
void GpsL2MPcpsAcquisitionFpga::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 85 of file `gps_l2_m_pcps_acquisition_fpga.h`.

10.192.2.6 set_channel_fsm()

```
void GpsL2MPcpsAcquisitionFpga::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 94 of file `gps_l2_m_pcps_acquisition_fpga.h`.

10.192.2.7 set_doppler_max()

```
void GpsL2MPcpsAcquisitionFpga::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.192.2.8 set_doppler_step()

```
void GpsL2MPcpsAcquisitionFpga::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.192.2.9 set_gnss_synchro()

```
void GpsL2MPcpsAcquisitionFpga::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.192.2.10 `set_local_code()`

```
void GpsL2MPcpsAcquisitionFpga::set_local_code ( ) [override], [virtual]
```

Sets local code for GPS L2/M PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.192.2.11 `set_state()`

```
void GpsL2MPcpsAcquisitionFpga::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.192.2.12 `set_threshold()`

```
void GpsL2MPcpsAcquisitionFpga::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.192.2.13 `stop_acquisition()`

```
void GpsL2MPcpsAcquisitionFpga::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

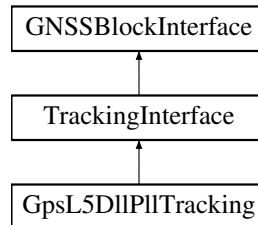
- [gps_l2_m_pcps_acquisition_fpga.h](#)

10.193 GpsL5DIIPIITracking Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <gps_l5_dll_pll_tracking.h>
```

Inheritance diagram for GpsL5DIIPIITracking:



Public Member Functions

- **GpsL5DIIPIITracking** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "GPS_L5_DLL_PLL_Tracking".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_channel** (unsigned int channel) override
Set tracking channel unique ID.
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **start_tracking** () override
- void **stop_tracking** () override
Stop running tracking.

10.193.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 38 of file `gps_l5_dll_pll_tracking.h`.

10.193.2 Member Function Documentation

10.193.2.1 implementation()

```
std::string GpsL5D11P11Tracking::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L5_DLL_PLL_Tracking".

Implements [GNSSBlockInterface](#).

Definition at line 55 of file `gps_l5_dll_pll_tracking.h`.

10.193.2.2 set_channel()

```
void GpsL5D11P11Tracking::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.193.2.3 set_gnss_synchro()

```
void GpsL5D11P11Tracking::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.193.2.4 stop_tracking()

```
void GpsL5D11P11Tracking::stop_tracking ( ) [override], [virtual]
```

Stop running tracking.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

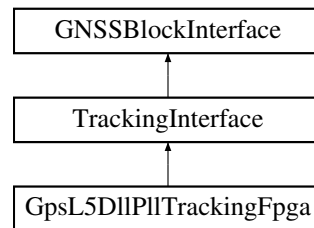
- [gps_l5_dll_pll_tracking.h](#)

10.194 GpsL5DIIPITrackingFpga Class Reference

This class implements a code DLL + carrier PLL tracking loop.

```
#include <gps_l5_dll_pll_tracking_fpga.h>
```

Inheritance diagram for GpsL5DIIPITrackingFpga:



Public Member Functions

- [GpsL5DIIPITrackingFpga](#) (const [ConfigurationInterface](#) *configuration, const std::string &[role](#), unsigned int in_streams, unsigned int out_streams)
Constructor.
- virtual [~GpsL5DIIPITrackingFpga](#) ()
Destructor.
- std::string [role](#) () override
Role.
- std::string [implementation](#) () override
Returns "GPS_L5_DLL_PLL_Tracking_Fpga".
- size_t [item_size](#) () override
Returns size of lv_16sc_t.
- void [connect](#) (gr::top_block_sptr top_block) override
Connect.
- void [disconnect](#) (gr::top_block_sptr top_block) override
Disconnect.
- gr::basic_block_sptr [get_left_block](#) () override
Get left block.
- gr::basic_block_sptr [get_right_block](#) () override
Get right block.
- void [set_channel](#) (unsigned int channel) override
Set tracking channel unique ID.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void [start_tracking](#) () override
Start the tracking process in the FPGA.
- void [stop_tracking](#) () override
Stop the tracking process in the FPGA.

10.194.1 Detailed Description

This class implements a code DLL + carrier PLL tracking loop.

Definition at line 39 of file `gps_l5_dll_pll_tracking_fpga.h`.

10.194.2 Constructor & Destructor Documentation

10.194.2.1 GpsL5D11P11TrackingFpga()

```
GpsL5D11P11TrackingFpga::GpsL5D11P11TrackingFpga (
    const ConfigurationInterface * configuration,
    const std::string & role,
    unsigned int in_streams,
    unsigned int out_streams )
```

Constructor.

10.194.2.2 ~GpsL5D11P11TrackingFpga()

```
virtual GpsL5D11P11TrackingFpga::~~GpsL5D11P11TrackingFpga ( ) [virtual]
```

Destructor.

10.194.3 Member Function Documentation

10.194.3.1 connect()

```
void GpsL5D11P11TrackingFpga::connect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Connect.

Implements [GNSSBlockInterface](#).

10.194.3.2 disconnect()

```
void GpsL5D11P11TrackingFpga::disconnect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Disconnect.

Implements [GNSSBlockInterface](#).

10.194.3.3 get_left_block()

```
gr::basic_block_sptr GpsL5D11P11TrackingFpga::get_left_block ( ) [override], [virtual]
```

Get left block.

Implements [GNSSBlockInterface](#).

10.194.3.4 get_right_block()

```
gr::basic_block_sptr GpsL5D11P11TrackingFpga::get_right_block ( ) [override], [virtual]
```

Get right block.

Implements [GNSSBlockInterface](#).

10.194.3.5 implementation()

```
std::string GpsL5D11P11TrackingFpga::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L5_DLL_PLL_Tracking_Fpga".

Implements [GNSSBlockInterface](#).

Definition at line 67 of file `gps_l5_dll_pll_tracking_fpga.h`.

10.194.3.6 item_size()

```
size_t GpsL5D11P11TrackingFpga::item_size ( ) [inline], [override], [virtual]
```

Returns size of `lv_16sc_t`.

Implements [GNSSBlockInterface](#).

Definition at line 75 of file `gps_l5_dll_pll_tracking_fpga.h`.

10.194.3.7 role()

```
std::string GpsL5D11P11TrackingFpga::role ( ) [inline], [override], [virtual]
```

Role.

Implements [GNSSBlockInterface](#).

Definition at line 59 of file `gps_l5_dll_pll_tracking_fpga.h`.

10.194.3.8 set_channel()

```
void GpsL5D11P11TrackingFpga::set_channel (
    unsigned int channel ) [override], [virtual]
```

Set tracking channel unique ID.

Implements [TrackingInterface](#).

10.194.3.9 set_gnss_synchro()

```
void GpsL5D11P11TrackingFpga::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [TrackingInterface](#).

10.194.3.10 start_tracking()

```
void GpsL5D11P11TrackingFpga::start_tracking ( ) [override], [virtual]
```

Start the tracking process in the FPGA.

Implements [TrackingInterface](#).

10.194.3.11 stop_tracking()

```
void GpsL5D11P11TrackingFpga::stop_tracking ( ) [override], [virtual]
```

Stop the tracking process in the FPGA.

Implements [TrackingInterface](#).

The documentation for this class was generated from the following file:

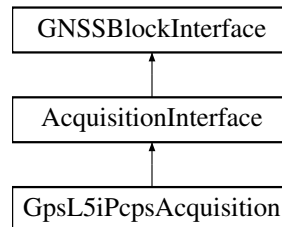
- [gps_l5_dll_pll_tracking_fpga.h](#)

10.195 GpsL5iPcpsAcquisition Class Reference

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L5i signals.

```
#include <gps_l5i_pcps_acquisition.h>
```

Inheritance diagram for GpsL5iPcpsAcquisition:



Public Member Functions

- **GpsL5iPcpsAcquisition** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "GPS_L5i_PCPS_Acquisition".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_gnss_synchro** ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void **set_channel** (unsigned int channel) override
Set acquisition channel unique ID.
- void **set_channel_fsm** (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
Set channel fsm associated to this acquisition instance.
- void **set_threshold** (float threshold) override
Set statistics threshold of PCPS algorithm.
- void **set_doppler_max** (unsigned int doppler_max) override
Set maximum Doppler off grid search.
- void **set_doppler_step** (unsigned int doppler_step) override
Set Doppler steps for the grid search.
- void **set_doppler_center** (int doppler_center) override
Set Doppler center for the grid search.
- void **init** () override
Initializes acquisition algorithm.
- void **set_local_code** () override
Sets local code for GPS L2/M PCPS acquisition algorithm.
- signed int **mag** () override
Returns the maximum peak of grid search.
- void **reset** () override
Restart acquisition algorithm.

- void [set_state](#) (int state) override
If state = 1, it forces the block to start acquiring from the first sample.
- void [stop_acquisition](#) () override
Stop running acquisition.
- void [set_resampler_latency](#) (uint32_t latency_samples) override
Sets the resampler latency to account it in the acquisition code delay estimation.

10.195.1 Detailed Description

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L5i signals.

Definition at line 42 of file `gps_l5i_pcps_acquisition.h`.

10.195.2 Member Function Documentation

10.195.2.1 implementation()

```
std::string GpsL5iPcpsAcquisition::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L5i_PCPS_Acquisition".

Implements [GNSSBlockInterface](#).

Definition at line 61 of file `gps_l5i_pcps_acquisition.h`.

10.195.2.2 init()

```
void GpsL5iPcpsAcquisition::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.195.2.3 mag()

```
signed int GpsL5iPcpsAcquisition::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.195.2.4 reset()

```
void GpsL5iPcpsAcquisition::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.195.2.5 set_channel()

```
void GpsL5iPcpsAcquisition::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 86 of file `gps_l5i_pcps_acquisition.h`.

10.195.2.6 set_channel_fsm()

```
void GpsL5iPcpsAcquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 95 of file `gps_l5i_pcps_acquisition.h`.

10.195.2.7 set_doppler_center()

```
void GpsL5iPcpsAcquisition::set_doppler_center (
    int doppler_center ) [override]
```

Set Doppler center for the grid search.

10.195.2.8 set_doppler_max()

```
void GpsL5iPcpsAcquisition::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.195.2.9 `set_doppler_step()`

```
void GpsL5iPcpsAcquisition::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.195.2.10 `set_gnss_synchro()`

```
void GpsL5iPcpsAcquisition::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.195.2.11 `set_local_code()`

```
void GpsL5iPcpsAcquisition::set_local_code ( ) [override], [virtual]
```

Sets local code for GPS L2/M PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.195.2.12 `set_resampler_latency()`

```
void GpsL5iPcpsAcquisition::set_resampler_latency (
    uint32_t latency_samples ) [override], [virtual]
```

Sets the resampler latency to account it in the acquisition code delay estimation.

Implements [AcquisitionInterface](#).

10.195.2.13 `set_state()`

```
void GpsL5iPcpsAcquisition::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.195.2.14 set_threshold()

```
void GpsL5iPcpsAcquisition::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.195.2.15 stop_acquisition()

```
void GpsL5iPcpsAcquisition::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

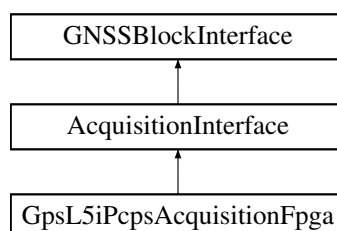
- [gps_l5i_pcps_acquisition.h](#)

10.196 GpsL5iPcpsAcquisitionFpga Class Reference

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for GPS L5i signals.

```
#include <gps_l5i_pcps_acquisition_fpga.h>
```

Inheritance diagram for GpsL5iPcpsAcquisitionFpga:



Public Member Functions

- [GpsL5iPcpsAcquisitionFpga](#) (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
Constructor.
- [~GpsL5iPcpsAcquisitionFpga](#) ()=default
Destructor.
- std::string [role](#) () override
Role.
- std::string [implementation](#) () override
Returns "GPS_L5i_PCPS_Acquisition_Fpga".
- size_t [item_size](#) () override
Returns size of lv_16sc_t.
- void [connect](#) (gr::top_block_sptr top_block) override
Connect.
- void [disconnect](#) (gr::top_block_sptr top_block) override
Disconnect.
- gr::basic_block_sptr [get_left_block](#) () override
Get left block.
- gr::basic_block_sptr [get_right_block](#) () override
Get right block.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro) override
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.
- void [set_channel](#) (unsigned int channel) override
Set acquisition channel unique ID.
- void [set_channel_fsm](#) (std::weak_ptr< [ChannelFsm](#) > channel_fsm) override
Set channel fsm associated to this acquisition instance.
- void [set_threshold](#) (float threshold) override
Set statistics threshold of PCPS algorithm.
- void [set_doppler_max](#) (unsigned int doppler_max) override
Set maximum Doppler off grid search.
- void [set_doppler_step](#) (unsigned int doppler_step) override
Set Doppler steps for the grid search.
- void [set_doppler_center](#) (int doppler_center) override
Set Doppler center for the grid search.
- void [init](#) () override
Initializes acquisition algorithm.
- void [set_local_code](#) () override
Sets local code for GPS L5 PCPS acquisition algorithm.
- signed int [mag](#) () override
Returns the maximum peak of grid search.
- void [reset](#) () override
Restart acquisition algorithm.
- void [set_state](#) (int state) override
If state = 1, it forces the block to start acquiring from the first sample.
- void [stop_acquisition](#) () override
Stop running acquisition.
- void [set_resampler_latency](#) (uint32_t latency_samples __attribute__((unused))) override
Set resampler latency.

10.196.1 Detailed Description

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for GPS L5i signals.

Definition at line 40 of file `gps_l5i_pcps_acquisition_fpga.h`.

10.196.2 Constructor & Destructor Documentation

10.196.2.1 GpsL5iPcpsAcquisitionFpga()

```
GpsL5iPcpsAcquisitionFpga::GpsL5iPcpsAcquisitionFpga (
    const ConfigurationInterface * configuration,
    const std::string & role,
    unsigned int in_streams,
    unsigned int out_streams )
```

Constructor.

10.196.2.2 ~GpsL5iPcpsAcquisitionFpga()

```
GpsL5iPcpsAcquisitionFpga::~GpsL5iPcpsAcquisitionFpga ( ) [default]
```

Destructor.

10.196.3 Member Function Documentation

10.196.3.1 connect()

```
void GpsL5iPcpsAcquisitionFpga::connect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Connect.

Implements [GNSSBlockInterface](#).

10.196.3.2 disconnect()

```
void GpsL5iPcpsAcquisitionFpga::disconnect (
    gr::top_block_sptr top_block ) [override], [virtual]
```

Disconnect.

Implements [GNSSBlockInterface](#).

10.196.3.3 get_left_block()

```
gr::basic_block_sptr GpsL5iPcpsAcquisitionFpga::get_left_block ( ) [override], [virtual]
```

Get left block.

Implements [GNSSBlockInterface](#).

10.196.3.4 get_right_block()

```
gr::basic_block_sptr GpsL5iPcpsAcquisitionFpga::get_right_block ( ) [override], [virtual]
```

Get right block.

Implements [GNSSBlockInterface](#).

10.196.3.5 implementation()

```
std::string GpsL5iPcpsAcquisitionFpga::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L5i_PCPS_Acquisition_Fpga".

Implements [GNSSBlockInterface](#).

Definition at line 68 of file `gps_l5i_pcps_acquisition_fpga.h`.

10.196.3.6 init()

```
void GpsL5iPcpsAcquisitionFpga::init ( ) [override], [virtual]
```

Initializes acquisition algorithm.

Implements [AcquisitionInterface](#).

10.196.3.7 item_size()

```
size_t GpsL5iPcpsAcquisitionFpga::item_size ( ) [inline], [override], [virtual]
```

Returns size of lv_16sc_t.

Implements [GNSSBlockInterface](#).

Definition at line 76 of file gps_l5i_pcps_acquisition_fpga.h.

10.196.3.8 mag()

```
signed int GpsL5iPcpsAcquisitionFpga::mag ( ) [override], [virtual]
```

Returns the maximum peak of grid search.

Implements [AcquisitionInterface](#).

10.196.3.9 reset()

```
void GpsL5iPcpsAcquisitionFpga::reset ( ) [override], [virtual]
```

Restart acquisition algorithm.

Implements [AcquisitionInterface](#).

10.196.3.10 role()

```
std::string GpsL5iPcpsAcquisitionFpga::role ( ) [inline], [override], [virtual]
```

Role.

Implements [GNSSBlockInterface](#).

Definition at line 60 of file gps_l5i_pcps_acquisition_fpga.h.

10.196.3.11 set_channel()

```
void GpsL5iPcpsAcquisitionFpga::set_channel (
    unsigned int channel ) [inline], [override], [virtual]
```

Set acquisition channel unique ID.

Implements [AcquisitionInterface](#).

Definition at line 111 of file gps_l5i_pcps_acquisition_fpga.h.

10.196.3.12 `set_channel_fsm()`

```
void GpsL5iPcpsAcquisitionFpga::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline], [override], [virtual]
```

Set channel fsm associated to this acquisition instance.

Implements [AcquisitionInterface](#).

Definition at line 120 of file `gps_l5i_pcps_acquisition_fpga.h`.

10.196.3.13 `set_doppler_center()`

```
void GpsL5iPcpsAcquisitionFpga::set_doppler_center (
    int doppler_center ) [override]
```

Set Doppler center for the grid search.

10.196.3.14 `set_doppler_max()`

```
void GpsL5iPcpsAcquisitionFpga::set_doppler_max (
    unsigned int doppler_max ) [override], [virtual]
```

Set maximum Doppler off grid search.

Implements [AcquisitionInterface](#).

10.196.3.15 `set_doppler_step()`

```
void GpsL5iPcpsAcquisitionFpga::set_doppler_step (
    unsigned int doppler_step ) [override], [virtual]
```

Set Doppler steps for the grid search.

Implements [AcquisitionInterface](#).

10.196.3.16 `set_gnss_synchro()`

```
void GpsL5iPcpsAcquisitionFpga::set_gnss_synchro (
    Gnss\_Synchro * p_gnss_synchro ) [override], [virtual]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to efficiently exchange synchronization data between acquisition and tracking blocks.

Implements [AcquisitionInterface](#).

10.196.3.17 set_local_code()

```
void GpsL5iPcpsAcquisitionFpga::set_local_code ( ) [override], [virtual]
```

Sets local code for GPS L5 PCPS acquisition algorithm.

Implements [AcquisitionInterface](#).

10.196.3.18 set_resampler_latency()

```
void GpsL5iPcpsAcquisitionFpga::set_resampler_latency (
    uint32_t latency_samples __attribute__((unused)) ) [inline], [override]
```

Set resampler latency.

Definition at line 179 of file `gps_l5i_pcps_acquisition_fpga.h`.

10.196.3.19 set_state()

```
void GpsL5iPcpsAcquisitionFpga::set_state (
    int state ) [override], [virtual]
```

If state = 1, it forces the block to start acquiring from the first sample.

Implements [AcquisitionInterface](#).

10.196.3.20 set_threshold()

```
void GpsL5iPcpsAcquisitionFpga::set_threshold (
    float threshold ) [override], [virtual]
```

Set statistics threshold of PCPS algorithm.

Implements [AcquisitionInterface](#).

10.196.3.21 stop_acquisition()

```
void GpsL5iPcpsAcquisitionFpga::stop_acquisition ( ) [override], [virtual]
```

Stop running acquisition.

Implements [AcquisitionInterface](#).

The documentation for this class was generated from the following file:

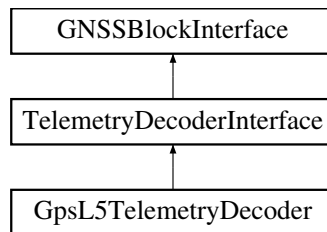
- [gps_l5i_pcps_acquisition_fpga.h](#)

10.197 GpsL5TelemetryDecoder Class Reference

This class implements a NAV data decoder for GPS L5.

```
#include <gps_l5_telemetry_decoder.h>
```

Inheritance diagram for GpsL5TelemetryDecoder:



Public Member Functions

- **GpsL5TelemetryDecoder** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "GPS_L5_Telemetry_Decoder".
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_satellite** (const [Gnss_Satellite](#) &satellite) override
- void **set_channel** (int channel) override
- void **reset** () override
- size_t **item_size** () override

10.197.1 Detailed Description

This class implements a NAV data decoder for GPS L5.

Definition at line 40 of file `gps_l5_telemetry_decoder.h`.

10.197.2 Member Function Documentation

10.197.2.1 implementation()

```
std::string GpsL5TelemetryDecoder::implementation ( ) [inline], [override], [virtual]
```

Returns "GPS_L5_Telemetry_Decoder".

Implements [GNSSBlockInterface](#).

Definition at line 57 of file `gps_l5_telemetry_decoder.h`.

The documentation for this class was generated from the following file:

- [gps_l5_telemetry_decoder.h](#)

10.198 GPU_Complex Struct Reference

Public Member Functions

- CUDA_CALLABLE_MEMBER_DEVICE **GPU_Complex** (float a, float b)
- CUDA_CALLABLE_MEMBER_DEVICE float **magnitude2** (void)
- CUDA_CALLABLE_MEMBER_DEVICE **GPU_Complex operator*** (const [GPU_Complex](#) &a)
- CUDA_CALLABLE_MEMBER_DEVICE **GPU_Complex operator+** (const [GPU_Complex](#) &a)
- CUDA_CALLABLE_MEMBER_DEVICE void **operator+=** (const [GPU_Complex](#) &a)
- CUDA_CALLABLE_MEMBER_DEVICE void **multiply_acc** (const [GPU_Complex](#) &a, const [GPU_Complex](#) &b)

Public Attributes

- float **r**
- float **i**

10.198.1 Detailed Description

Definition at line 42 of file `cuda_multicorrelator.h`.

The documentation for this struct was generated from the following file:

- [cuda_multicorrelator.h](#)

10.199 GPU_Complex_Short Struct Reference

Public Member Functions

- CUDA_CALLABLE_MEMBER_DEVICE **GPU_Complex_Short** (short int a, short int b)
- CUDA_CALLABLE_MEMBER_DEVICE float **magnitude2** (void)
- CUDA_CALLABLE_MEMBER_DEVICE **GPU_Complex_Short operator*** (const [GPU_Complex_Short](#) &a)
- CUDA_CALLABLE_MEMBER_DEVICE **GPU_Complex_Short operator+** (const [GPU_Complex_Short](#) &a)

Public Attributes

- float **r**
- float **i**

10.199.1 Detailed Description

Definition at line 85 of file `cuda_multicorrelator.h`.

The documentation for this struct was generated from the following file:

- [cuda_multicorrelator.h](#)

10.200 Gpx_Printer Class Reference

Prints PVT information to GPX format file.

```
#include <gpx_printer.h>
```

Public Member Functions

- **Gpx_Printer** (const std::string &base_path=".")
- bool **set_headers** (const std::string &filename, bool time_tag_name=true)
- bool **print_position** (const [Pvt_Solution](#) *const position, bool print_average_values)
- bool **close_file** ()

10.200.1 Detailed Description

Prints PVT information to GPX format file.

See <https://www.topografix.com/gpx.asp>

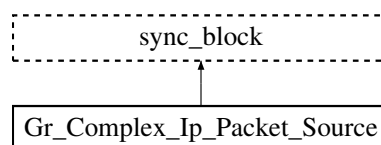
Definition at line 36 of file gpx_printer.h.

The documentation for this class was generated from the following file:

- [gpx_printer.h](#)

10.201 Gr_Complex_Ip_Packet_Source Class Reference

Inheritance diagram for Gr_Complex_Ip_Packet_Source:



Public Types

- typedef boost::shared_ptr< [Gr_Complex_Ip_Packet_Source](#) > **sptr**

Public Member Functions

- **Gr_Complex_Ip_Packet_Source** (std::string src_device, const std::string &origin_address, int udp_port, int udp_packet_size, int n_baseband_channels, const std::string &wire_sample_type, size_t item_size, bool IQ_swap_)
- bool **start** ()
- bool **stop** ()
- int **work** (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Static Public Member Functions

- static sptr **make** (std::string src_device, const std::string &origin_address, int udp_port, int udp_packet_size, int n_baseband_channels, const std::string &wire_sample_type, size_t item_size, bool IQ_swap_)

10.201.1 Detailed Description

Definition at line 40 of file gr_complex_ip_packet_source.h.

The documentation for this class was generated from the following file:

- [gr_complex_ip_packet_source.h](#)

10.202 gtime_t Struct Reference

Public Attributes

- time_t **time**
- double **sec**

10.202.1 Detailed Description

Definition at line 354 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.203 half_cyc_tag Struct Reference

Public Attributes

- unsigned char **sat**
- unsigned char **freq**
- unsigned char **valid**
- char **corr**
- [gtime_t](#) **ts**
- [gtime_t](#) **te**
- struct [half_cyc_tag](#) * **next**

10.203.1 Detailed Description

Definition at line 1076 of file rtklib.h.

The documentation for this struct was generated from the following file:

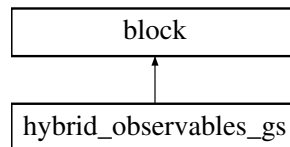
- [rtklib.h](#)

10.204 hybrid_observables_gs Class Reference

This class implements a block that computes observables.

```
#include <hybrid_observables_gs.h>
```

Inheritance diagram for hybrid_observables_gs:



Public Member Functions

- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- hybrid_observables_gs_sptr **hybrid_observables_gs_make** (const [Obs_Conf](#) &conf_)

10.204.1 Detailed Description

This class implements a block that computes observables.

Definition at line 60 of file `hybrid_observables_gs.h`.

The documentation for this class was generated from the following file:

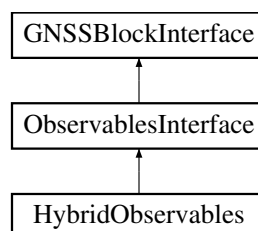
- [hybrid_observables_gs.h](#)

10.205 HybridObservables Class Reference

This class implements an [ObservablesInterface](#) for observables of all kind of GNSS signals.

```
#include <hybrid_observables.h>
```

Inheritance diagram for HybridObservables:



Public Member Functions

- **HybridObservables** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override

Returns "Hybrid_Observables".
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **reset** () override
- size_t **item_size** () override

All blocks must have an [item_size\(\)](#) function implementation.

10.205.1 Detailed Description

This class implements an [ObservablesInterface](#) for observables of all kind of GNSS signals.

Definition at line 39 of file `hybrid_observables.h`.

10.205.2 Member Function Documentation

10.205.2.1 implementation()

```
std::string HybridObservables::implementation ( ) [inline], [override], [virtual]
```

Returns "Hybrid_Observables".

Implements [GNSSBlockInterface](#).

Definition at line 55 of file `hybrid_observables.h`.

10.205.2.2 item_size()

```
size_t HybridObservables::item_size ( ) [inline], [override], [virtual]
```

All blocks must have an [item_size\(\)](#) function implementation.

Implements [GNSSBlockInterface](#).

Definition at line 71 of file `hybrid_observables.h`.

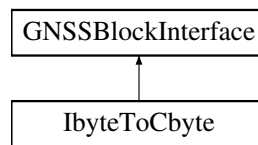
The documentation for this class was generated from the following file:

- [hybrid_observables.h](#)

10.206 IbyteToCbyte Class Reference

```
#include <ibyte_to_cbyte.h>
```

Inheritance diagram for IbyteToCbyte:



Public Member Functions

- **IbyteToCbyte** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "Ibyte_To_Cbyte".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.206.1 Detailed Description

an I/Q interleaved byte (unsigned char) sample stream into a std::complex<unsigned char> stream

Definition at line 37 of file ibyte_to_cbyte.h.

10.206.2 Member Function Documentation

10.206.2.1 implementation()

```
std::string IbyteToCbyte::implementation ( ) [inline], [override], [virtual]
```

Returns "Ibyte_To_Cbyte".

Implements [GNSSBlockInterface](#).

Definition at line 52 of file ibyte_to_cbyte.h.

The documentation for this class was generated from the following file:

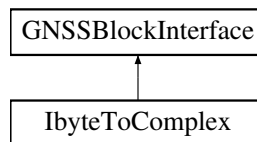
- [ibyte_to_cbyte.h](#)

10.207 IbyteToComplex Class Reference

Adapts an I/Q interleaved byte integer sample stream to a gr_complex (float) stream.

```
#include <ibyte_to_complex.h>
```

Inheritance diagram for IbyteToComplex:



Public Member Functions

- **IbyteToComplex** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "Ibyte_To_Complex".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.207.1 Detailed Description

Adapts an I/Q interleaved byte integer sample stream to a gr_complex (float) stream.

Definition at line 36 of file `ibyte_to_complex.h`.

10.207.2 Member Function Documentation

10.207.2.1 implementation()

```
std::string IbyteToComplex::implementation ( ) [inline], [override], [virtual]
```

Returns "Ibyte_To_Complex".

Implements [GNSSBlockInterface](#).

Definition at line 51 of file `ibyte_to_complex.h`.

The documentation for this class was generated from the following file:

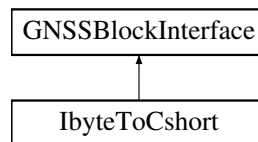
- [ibyte_to_complex.h](#)

10.208 IbyteToCshort Class Reference

Adapts a short integer (16 bits) interleaved sample stream into a `std::complex<short>` stream.

```
#include <ibyte_to_cshort.h>
```

Inheritance diagram for IbyteToCshort:



Public Member Functions

- **IbyteToCshort** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "Ibyte_To_Cshort".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.208.1 Detailed Description

Adapts a short integer (16 bits) interleaved sample stream into a `std::complex<short>` stream.

Definition at line 37 of file `ibyte_to_cshort.h`.

10.208.2 Member Function Documentation

10.208.2.1 implementation()

```
std::string IbyteToCshort::implementation ( ) [inline], [override], [virtual]
```

Returns "Ibyte_To_Cshort".

Implements [GNSSBlockInterface](#).

Definition at line 52 of file `ibyte_to_cshort.h`.

The documentation for this class was generated from the following file:

- [ibyte_to_cshort.h](#)

10.209 INIReader Class Reference

Read an INI file into easy-to-access name/value pairs. (Note that I've gone for simplicity here rather than speed, but it should be pretty decent.)

```
#include <INIReader.h>
```

Public Member Functions

- **INIReader** (const std::string &filename)
*Construct **INIReader** and parse given filename. See [ini.h](#) for more info about the parsing.*
- int **ParseError** ()
Return the result of `ini_parse()`, i.e., 0 on success, line number of first error on parse error, or -1 on file open error.
- std::string **Get** (const std::string §ion, const std::string &name, const std::string &default_value)
Get a string value from INI file, returning default_value if not found.
- int64_t **GetInteger** (const std::string §ion, const std::string &name, int64_t default_value)
Get an integer (long) value from INI file, returning default_value if not found.

10.209.1 Detailed Description

Read an INI file into easy-to-access name/value pairs. (Note that I've gone for simplicity here rather than speed, but it should be pretty decent.)

Definition at line 39 of file INIReader.h.

10.209.2 Constructor & Destructor Documentation

10.209.2.1 INIReader()

```
INIReader::INIReader (  
    const std::string & filename ) [explicit]
```

Construct **INIReader** and parse given filename. See [ini.h](#) for more info about the parsing.

10.209.3 Member Function Documentation

10.209.3.1 Get()

```
std::string INIReader::Get (  
    const std::string & section,  
    const std::string & name,  
    const std::string & default_value )
```

Get a string value from INI file, returning default_value if not found.

10.209.3.2 GetInteger()

```
int64_t INIReader::GetInteger (
    const std::string & section,
    const std::string & name,
    int64_t default_value )
```

Get an integer (long) value from INI file, returning default_value if not found.

10.209.3.3 ParseError()

```
int INIReader::ParseError ( )
```

Return the result of ini_parse(), i.e., 0 on success, line number of first error on parse error, or -1 on file open error.

The documentation for this class was generated from the following file:

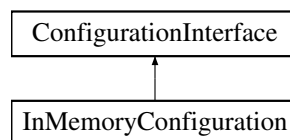
- [INIReader.h](#)

10.210 InMemoryConfiguration Class Reference

This class is an implementation of the interface [ConfigurationInterface](#).

```
#include <in_memory_configuration.h>
```

Inheritance diagram for InMemoryConfiguration:



Public Member Functions

- **std::string property** (std::string property_name, std::string default_value) const override
- **bool property** (std::string property_name, bool default_value) const override
- **int64_t property** (std::string property_name, int64_t default_value) const override
- **uint64_t property** (std::string property_name, uint64_t default_value) const override
- **int32_t property** (std::string property_name, int32_t default_value) const override
- **uint32_t property** (std::string property_name, uint32_t default_value) const override
- **int16_t property** (std::string property_name, int16_t default_value) const override
- **uint16_t property** (std::string property_name, uint16_t default_value) const override
- **float property** (std::string property_name, float default_value) const override
- **double property** (std::string property_name, double default_value) const override
- **void set_property** (std::string property_name, std::string value) override
- **void supersede_property** (const std::string &property_name, const std::string &value)
- **bool is_present** (const std::string &property_name) const

10.210.1 Detailed Description

This class is an implementation of the interface [ConfigurationInterface](#).

This implementation accepts configuration parameters upon instantiation and it is intended to be used in unit testing.

Definition at line 41 of file `in_memory_configuration.h`.

The documentation for this class was generated from the following file:

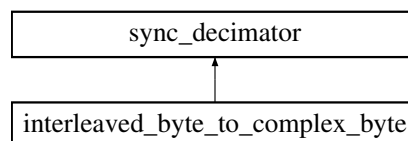
- [in_memory_configuration.h](#)

10.211 interleaved_byte_to_complex_byte Class Reference

This class adapts an 8-bits interleaved sample stream into a 16-bits complex stream (`std::complex<unsigned char>`)

```
#include <interleaved_byte_to_complex_byte.h>
```

Inheritance diagram for `interleaved_byte_to_complex_byte`:



Public Member Functions

- `int work (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)`

Friends

- `interleaved_byte_to_complex_byte_sptr make_interleaved_byte_to_complex_byte ()`

10.211.1 Detailed Description

This class adapts an 8-bits interleaved sample stream into a 16-bits complex stream (`std::complex<unsigned char>`)

Definition at line 44 of file `interleaved_byte_to_complex_byte.h`.

The documentation for this class was generated from the following file:

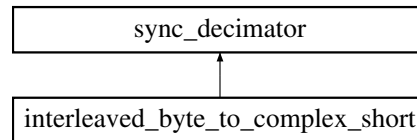
- [interleaved_byte_to_complex_byte.h](#)

10.212 interleaved_byte_to_complex_short Class Reference

This class adapts a short (16-bits) interleaved sample stream into a `std::complex<short>` stream.

```
#include <interleaved_byte_to_complex_short.h>
```

Inheritance diagram for `interleaved_byte_to_complex_short`:



Public Member Functions

- `int work` (`int` noutput_items, `gr_vector_const_void_star` &input_items, `gr_vector_void_star` &output_items)

Friends

- `interleaved_byte_to_complex_short_sptr make_interleaved_byte_to_complex_short` ()

10.212.1 Detailed Description

This class adapts a short (16-bits) interleaved sample stream into a `std::complex<short>` stream.

Definition at line 44 of file `interleaved_byte_to_complex_short.h`.

The documentation for this class was generated from the following file:

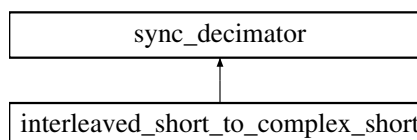
- [interleaved_byte_to_complex_short.h](#)

10.213 interleaved_short_to_complex_short Class Reference

This class adapts a short (16-bits) interleaved sample stream into a `std::complex<short>` stream.

```
#include <interleaved_short_to_complex_short.h>
```

Inheritance diagram for `interleaved_short_to_complex_short`:



Public Member Functions

- int **work** (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- interleaved_short_to_complex_short_sptr **make_interleaved_short_to_complex_short** ()

10.213.1 Detailed Description

This class adapts a short (16-bits) interleaved sample stream into a `std::complex<short>` stream.

Definition at line 44 of file `interleaved_short_to_complex_short.h`.

The documentation for this class was generated from the following file:

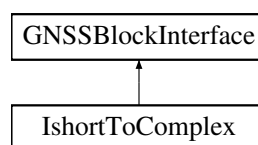
- [interleaved_short_to_complex_short.h](#)

10.214 IshortToComplex Class Reference

Adapts an I/Q interleaved short integer sample stream to a `gr_complex` (float) stream.

```
#include <ishort_to_complex.h>
```

Inheritance diagram for IshortToComplex:



Public Member Functions

- **IshortToComplex** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "Ishort_To_Complex".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.214.1 Detailed Description

Adapts an I/Q interleaved short integer sample stream to a `gr_complex` (float) stream.

Definition at line 36 of file `ishort_to_complex.h`.

10.214.2 Member Function Documentation

10.214.2.1 `implementation()`

```
std::string IshortToComplex::implementation ( ) [inline], [override], [virtual]
```

Returns "`Ishort_To_Complex`".

Implements [GNSSBlockInterface](#).

Definition at line 51 of file `ishort_to_complex.h`.

The documentation for this class was generated from the following file:

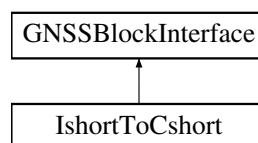
- [ishort_to_complex.h](#)

10.215 IshortToCshort Class Reference

Adapts a short integer (16 bits) interleaved sample stream into a `std::complex<short>` stream.

```
#include <ishort_to_cshort.h>
```

Inheritance diagram for `IshortToCshort`:



Public Member Functions

- **IshortToCshort** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "`Ishort_To_Cshort`".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.215.1 Detailed Description

Adapts a short integer (16 bits) interleaved sample stream into a `std::complex<short>` stream.

Definition at line 37 of file `ishort_to_cshort.h`.

10.215.2 Member Function Documentation

10.215.2.1 implementation()

```
std::string IshortToCshort::implementation ( ) [inline], [override], [virtual]
```

Returns "Ishort_To_Cshort".

Implements [GNSSBlockInterface](#).

Definition at line 52 of file `ishort_to_cshort.h`.

The documentation for this class was generated from the following file:

- [ishort_to_cshort.h](#)

10.216 kernel_info_t Struct Reference

Public Attributes

- `cl_kernel` **kernel**
- `char *` **kernel_name**
- `unsigned` **lmem_size**
- `unsigned` **num_workgroups**
- `unsigned` **num_xforms_per_workgroup**
- `unsigned` **num_workitems_per_workgroup**
- `cl_fft_kernel_dir` **dir**
- `int` **in_place_possible**
- `kernel_info_t *` **next**

10.216.1 Detailed Description

Definition at line 32 of file `fft_internal.h`.

The documentation for this struct was generated from the following file:

- [fft_internal.h](#)

10.217 Kml_Printer Class Reference

Prints PVT information to OGC KML format file (can be viewed with Google Earth)

```
#include <kml_printer.h>
```

Public Member Functions

- **Kml_Printer** (const std::string &base_path=std::string("."))
- bool **set_headers** (const std::string &filename, bool time_tag_name=true)
- bool **print_position** (const [Pvt_Solution](#) *const position, bool print_average_values)
- bool **close_file** ()

10.217.1 Detailed Description

Prints PVT information to OGC KML format file (can be viewed with Google Earth)

See <https://www.opengeospatial.org/standards/kml>

Definition at line 35 of file kml_printer.h.

The documentation for this class was generated from the following file:

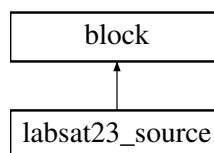
- [kml_printer.h](#)

10.218 labsat23_source Class Reference

This class implements conversion between Labsat2 and 3 format byte packet samples to gr_complex.

```
#include <labsat23_source.h>
```

Inheritance diagram for labsat23_source:



Public Member Functions

- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- labsat23_source_sptr **labsat23_make_source_sptr** (const char *signal_file_basename, int channel_selector, [Concurrent_Queue](#)< pmt::pmt_t > *queue)

10.218.1 Detailed Description

This class implements conversion between Labsat2 and 3 format byte packet samples to `gr_complex`.

Definition at line 51 of file `labsat23_source.h`.

The documentation for this class was generated from the following file:

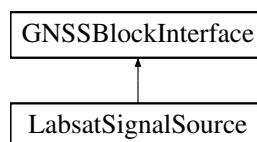
- [labsat23_source.h](#)

10.219 LabsatSignalSource Class Reference

This class reads samples stored by a LabSat 2 or LabSat 3 device.

```
#include <labsat_signal_source.h>
```

Inheritance diagram for LabsatSignalSource:



Public Member Functions

- **LabsatSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_stream, unsigned int out_stream, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "Labsat_Signal_Source".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.219.1 Detailed Description

This class reads samples stored by a LabSat 2 or LabSat 3 device.

Definition at line 37 of file `labsat_signal_source.h`.

10.219.2 Member Function Documentation

10.219.2.1 implementation()

```
std::string LabsatSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "Labsat_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 54 of file labsat_signal_source.h.

The documentation for this class was generated from the following file:

- [labsat_signal_source.h](#)

10.220 lex_t Struct Reference

Public Attributes

- int **n**
- int **nmax**
- [lexmsg_t](#) * **msgs**

10.220.1 Detailed Description

Definition at line 682 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.221 lexeph_t Struct Reference

Public Attributes

- [gtime_t](#) **toe**
- [gtime_t](#) **tof**
- int **sat**
- unsigned char **health**
- unsigned char **ura**
- double **pos** [3]
- double **vel** [3]
- double **acc** [3]
- double **jerk** [3]
- double **af0**
- double **af1**
- double **tgd**
- double **isc** [8]

10.221.1 Detailed Description

Definition at line 689 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.222 `lexion_t` Struct Reference

Public Attributes

- [gtime_t](#) **t0**
- double **tspan**
- double **pos0** [2]
- double **coef** [3][2]

10.222.1 Detailed Description

Definition at line 706 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.223 `lexmsg_t` Struct Reference

Public Attributes

- int **prn**
- int **type**
- int **alert**
- unsigned char **stat**
- unsigned char **snr**
- unsigned int **ttt**
- unsigned char **msg** [212]

10.223.1 Detailed Description

Definition at line 670 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

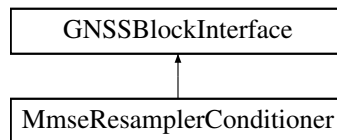
- [rtklib.h](#)

10.224 MmseResamplerConditioner Class Reference

Interface of a MMSE resampler block adapter to a SignalConditionerInterface.

```
#include <mmse_resampler_conditioner.h>
```

Inheritance diagram for MmseResamplerConditioner:



Public Member Functions

- **MmseResamplerConditioner** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_stream, unsigned int out_stream)
- std::string **role** () override
- std::string **implementation** () override
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.224.1 Detailed Description

Interface of a MMSE resampler block adapter to a SignalConditionerInterface.

Definition at line 43 of file mmse_resampler_conditioner.h.

The documentation for this class was generated from the following file:

- [mmse_resampler_conditioner.h](#)

10.225 ModelFunction Class Reference

Public Member Functions

- virtual arma::vec **operator()** (const arma::vec &input)=0

10.225.1 Detailed Description

Definition at line 43 of file nonlinear_tracking.h.

The documentation for this class was generated from the following file:

- [nonlinear_tracking.h](#)

10.226 Monitor_Pvt Class Reference

This class contains parameters and outputs of the PVT block.

```
#include <monitor_pvt.h>
```

Public Member Functions

- `template<class Archive >`
`void serialize (Archive &ar, const unsigned int version)`
This member function serializes and restores [Monitor_Pvt](#) objects from a byte stream.

Public Attributes

- `uint32_t TOW_at_current_symbol_ms`
- `uint32_t week`
- `double RX_time`
- `double user_clk_offset`
- `double pos_x`
- `double pos_y`
- `double pos_z`
- `double vel_x`
- `double vel_y`
- `double vel_z`
- `double cov_xx`
- `double cov_yy`
- `double cov_zz`
- `double cov_xy`
- `double cov_yz`
- `double cov_zx`
- `double latitude`
- `double longitude`
- `double height`
- `uint8_t valid_sats`
- `uint8_t solution_status`
- `uint8_t solution_type`
- `float AR_ratio_factor`
- `float AR_ratio_threshold`
- `double gdop`
- `double pdop`
- `double hdop`
- `double vdop`
- `double user_clk_drift_ppm`

10.226.1 Detailed Description

This class contains parameters and outputs of the PVT block.

Definition at line 29 of file `monitor_pvt.h`.

10.226.2 Member Function Documentation

10.226.2.1 serialize()

```
template<class Archive >
void Monitor_Pvt::serialize (
    Archive & ar,
    const unsigned int version ) [inline]
```

This member function serializes and restores [Monitor_Pvt](#) objects from a byte stream.

Definition at line 90 of file monitor_pvt.h.

The documentation for this class was generated from the following file:

- [monitor_pvt.h](#)

10.227 Monitor_Pvt_Udp_Sink Class Reference

Public Member Functions

- **Monitor_Pvt_Udp_Sink** (const std::vector< std::string > &addresses, const uint16_t &port, bool protobuf←_enabled)
- bool **write_monitor_pvt** (const [Monitor_Pvt](#) *const monitor_pvt)

10.227.1 Detailed Description

Definition at line 37 of file monitor_pvt_udp_sink.h.

The documentation for this class was generated from the following file:

- [monitor_pvt_udp_sink.h](#)

10.228 msm_h_t Struct Reference

Public Attributes

- unsigned char **iod**
- unsigned char **time_s**
- unsigned char **clk_str**
- unsigned char **clk_ext**
- unsigned char **smooth**
- unsigned char **tint_s**
- unsigned char **nsat**
- unsigned char **nsig**
- unsigned char **sats** [64]
- unsigned char **sigs** [32]
- unsigned char **cellmask** [64]

10.228.1 Detailed Description

Definition at line 1269 of file rtklib.h.

The documentation for this struct was generated from the following file:

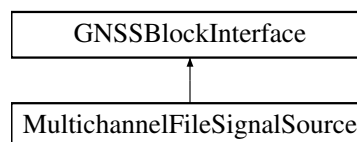
- [rtklib.h](#)

10.229 MultichannelFileSignalSource Class Reference

Class that reads signals samples from files at different frequency bands and adapts it to a SignalSourceInterface.

```
#include <multichannel_file_signal_source.h>
```

Inheritance diagram for MultichannelFileSignalSource:



Public Member Functions

- **MultichannelFileSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "Multichannel_File_Signal_Source".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- std::string **filename** () const
- std::string **item_type** () const
- bool **repeat** () const
- int64_t **sampling_frequency** () const
- uint64_t **samples** () const

10.229.1 Detailed Description

Class that reads signals samples from files at different frequency bands and adapts it to a SignalSourceInterface.

Definition at line 49 of file multichannel_file_signal_source.h.

10.229.2 Member Function Documentation

10.229.2.1 implementation()

```
std::string MultichannelFileSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "Multichannel_File_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 66 of file multichannel_file_signal_source.h.

The documentation for this class was generated from the following file:

- [multichannel_file_signal_source.h](#)

10.230 nav_t Struct Reference

Public Attributes

- int **n**
- int **nmax**
- int **ng**
- int **ngmax**
- int **ns**
- int **nsmax**
- int **ne**
- int **nemax**
- int **nc**
- int **ncmax**
- int **na**
- int **namax**
- int **nt**
- int **ntmax**
- int **nf**
- int **nfmax**
- [eph_t](#) * **eph**
- [geph_t](#) * **geph**
- [seph_t](#) * **seph**
- [peph_t](#) * **peph**
- [pclk_t](#) * **pclk**
- [alm_t](#) * **alm**
- [tec_t](#) * **tec**
- [fcbd_t](#) * **fcb**
- [erp_t](#) **erp**
- double **utc_gps** [4]
- double **utc_glo** [4]
- double **utc_gal** [4]
- double **utc_qzs** [4]
- double **utc_cmp** [4]
- double **utc_irn** [4]
- double **utc_sbs** [4]
- double **ion_gps** [8]
- double **ion_gal** [4]

- double **ion_qzs** [8]
- double **ion_cmp** [8]
- double **ion_irn** [8]
- int **leaps**
- double **lam** [MAXSAT][[NFREQ](#)]
- double **cbias** [MAXSAT][3]
- double **rbias** [[MAXRCV](#)][2][3]
- double **wlbias** [MAXSAT]
- double **glo_cpbias** [4]
- char **glo_fcn** [[MAXPRNGLO](#)+1]
- [pcv_t](#) **pcvs** [MAXSAT]
- [sbssat_t](#) **sbssat**
- [sbsion_t](#) **sbsion** [[MAXBAND](#)+1]
- [dgps_t](#) **dgps** [MAXSAT]
- [ssr_t](#) **ssr** [MAXSAT]
- [lexeph_t](#) **lexeph** [MAXSAT]
- [lexion_t](#) **lexion**
- [pppcorr_t](#) **pppcorr**

10.230.1 Detailed Description

Definition at line 746 of file [rtklib.h](#).

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.231 Nmea_Printer Class Reference

This class provides a implementation of a subset of the NMEA-0183 standard for interfacing marine electronic devices as defined by the National Marine Electronics Association (NMEA).

```
#include <nmea_printer.h>
```

Public Member Functions

- [Nmea_Printer](#) (const std::string &filename, bool flag_nmea_output_file, bool flag_nmea_tty_port, std::string nmea_dump_devname, const std::string &base_path=".")
Default constructor.
- [~Nmea_Printer](#) ()
Default destructor.
- bool [Print_Nmea_Line](#) (const [Rtklib_Solver](#) *const pvt_data, bool print_average_values)
Print NMEA PVT and satellite info to the initialized device.

10.231.1 Detailed Description

This class provides a implementation of a subset of the NMEA-0183 standard for interfacing marine electronic devices as defined by the National Marine Electronics Association (NMEA).

See https://en.wikipedia.org/wiki/NMEA_0183

Definition at line 41 of file [nmea_printer.h](#).

10.231.2 Constructor & Destructor Documentation

10.231.2.1 Nmea_Printer()

```
Nmea_Printer::Nmea_Printer (
    const std::string & filename,
    bool flag_nmea_output_file,
    bool flag_nmea_tty_port,
    std::string nmea_dump_devname,
    const std::string & base_path = "." )
```

Default constructor.

10.231.2.2 ~Nmea_Printer()

```
Nmea_Printer::~Nmea_Printer ( )
```

Default destructor.

10.231.3 Member Function Documentation

10.231.3.1 Print_Nmea_Line()

```
bool Nmea_Printer::Print_Nmea_Line (
    const Rtklib_Solver *const pvt_data,
    bool print_average_values )
```

Print NMEA PVT and satellite info to the initialized device.

The documentation for this class was generated from the following file:

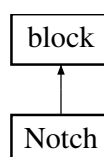
- [nmea_printer.h](#)

10.232 Notch Class Reference

This class implements a real-time software-defined multi state notch filter.

```
#include <notch_cc.h>
```

Inheritance diagram for Notch:



Public Member Functions

- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- notch_sptr **make_notch_filter** (float pfa, float p_c_factor, int32_t length_, int32_t n_segments_est, int32_t n_segments_reset)

10.232.1 Detailed Description

This class implements a real-time software-defined multi state notch filter.

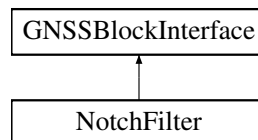
Definition at line 51 of file notch_cc.h.

The documentation for this class was generated from the following file:

- [notch_cc.h](#)

10.233 NotchFilter Class Reference

Inheritance diagram for NotchFilter:



Public Member Functions

- **NotchFilter** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** ()
- std::string **implementation** ()
Returns "Notch_Filter".
- size_t **item_size** ()
- void **connect** (gr::top_block_sptr top_block)
- void **disconnect** (gr::top_block_sptr top_block)
- gr::basic_block_sptr **get_left_block** ()
- gr::basic_block_sptr **get_right_block** ()

10.233.1 Detailed Description

Definition at line 32 of file notch_filter.h.

10.233.2 Member Function Documentation

10.233.2.1 implementation()

```
std::string NotchFilter::implementation ( ) [inline], [virtual]
```

Returns "Notch_Filter".

Implements [GNSSBlockInterface](#).

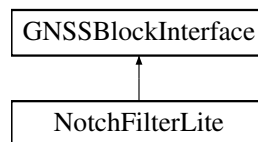
Definition at line 47 of file notch_filter.h.

The documentation for this class was generated from the following file:

- [notch_filter.h](#)

10.234 NotchFilterLite Class Reference

Inheritance diagram for NotchFilterLite:



Public Member Functions

- **NotchFilterLite** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** ()
- std::string **implementation** ()
Returns "Notch_Filter_Lite".
- size_t **item_size** ()
- void **connect** (gr::top_block_sptr top_block)
- void **disconnect** (gr::top_block_sptr top_block)
- gr::basic_block_sptr **get_left_block** ()
- gr::basic_block_sptr **get_right_block** ()

10.234.1 Detailed Description

Definition at line 32 of file notch_filter_lite.h.

10.234.2 Member Function Documentation

10.234.2.1 implementation()

```
std::string NotchFilterLite::implementation ( ) [inline], [virtual]
```

Returns "Notch_Filter_Lite".

Implements [GNSSBlockInterface](#).

Definition at line 47 of file notch_filter_lite.h.

The documentation for this class was generated from the following file:

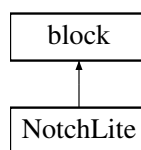
- [notch_filter_lite.h](#)

10.235 NotchLite Class Reference

This class implements a real-time software-defined multi state notch filter light version.

```
#include <notch_lite_cc.h>
```

Inheritance diagram for NotchLite:



Public Member Functions

- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- notch_lite_sptr **make_notch_filter_lite** (float p_c_factor, float pfa, int32_t length_, int32_t n_segments_est, int32_t n_segments_reset, int32_t n_segments_coeff)

10.235.1 Detailed Description

This class implements a real-time software-defined multi state notch filter light version.

Definition at line 52 of file notch_lite_cc.h.

The documentation for this class was generated from the following file:

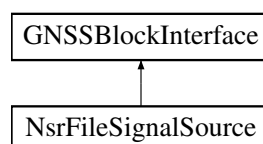
- [notch_lite_cc.h](#)

10.236 NsrFileSignalSource Class Reference

Class that reads signals samples from a file and adapts it to a SignalSourceInterface.

```
#include <nsrc_file_signal_source.h>
```

Inheritance diagram for NsrFileSignalSource:



Public Member Functions

- **NsrFileSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
Returns "Nsr_File_Signal_Source".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- std::string **filename** () const
- std::string **item_type** () const
- bool **repeat** () const
- int64_t **sampling_frequency** () const
- uint64_t **samples** () const

10.236.1 Detailed Description

Class that reads signals samples from a file and adapts it to a SignalSourceInterface.

Definition at line 48 of file nsrc_file_signal_source.h.

10.236.2 Member Function Documentation

10.236.2.1 implementation()

```
std::string NsrFileSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "Nsr_File_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 64 of file `nsr_file_signal_source.h`.

The documentation for this class was generated from the following file:

- [nsr_file_signal_source.h](#)

10.237 ntrip_t Struct Reference

Public Attributes

- int **state**
- int **type**
- int **nb**
- char **url** [256]
- char **mntpnt** [256]
- char **user** [256]
- char **passwd** [256]
- char **str** [NTRIP_MAXSTR]
- unsigned char **buff** [NTRIP_MAXRSP]
- [tcpcli_t](#) * **tcp**

10.237.1 Detailed Description

Definition at line 1162 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.238 Obs_Conf Class Reference

Public Attributes

- std::string **dump_filename**
- int32_t **smoothing_factor**
- uint32_t **nchannels_in**
- uint32_t **nchannels_out**
- bool **enable_carrier_smoothing**
- bool **dump**
- bool **dump_mat**

10.238.1 Detailed Description

Definition at line 27 of file obs_conf.h.

The documentation for this class was generated from the following file:

- [obs_conf.h](#)

10.239 obs_t Struct Reference

Public Attributes

- int **n**
- int **nmax**
- [obsd_t](#) * **data**

10.239.1 Detailed Description

Definition at line 374 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.240 obsd_t Struct Reference

Public Attributes

- [gtime_t](#) **time**
- unsigned char **sat**
- unsigned char **rcv**
- unsigned char **SNR** [[NFREQ+NEXOBS](#)]
- unsigned char **LLI** [[NFREQ+NEXOBS](#)]
- unsigned char **code** [[NFREQ+NEXOBS](#)]
- double **L** [[NFREQ+NEXOBS](#)]
- double **P** [[NFREQ+NEXOBS](#)]
- float **D** [[NFREQ+NEXOBS](#)]

10.240.1 Detailed Description

Definition at line 361 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.241 Observables_Dump_Reader Class Reference

Public Member Functions

- **Observables_Dump_Reader** (int n_channels)
- bool **read_binary_obs** ()
- bool **restart** ()
- int64_t **num_epochs** ()
- bool **open_obs_file** (std::string out_file)
- void **close_obs_file** ()

Public Attributes

- std::vector< double > **RX_time**
- std::vector< double > **TOW_at_current_symbol_s**
- std::vector< double > **Carrier_Doppler_hz**
- std::vector< double > **Acc_carrier_phase_hz**
- std::vector< double > **Pseudorange_m**
- std::vector< double > **PRN**
- std::vector< double > **valid**

10.241.1 Detailed Description

Definition at line 28 of file `observables_dump_reader.h`.

The documentation for this class was generated from the following file:

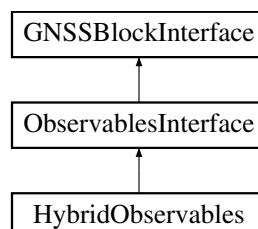
- [observables_dump_reader.h](#)

10.242 ObservablesInterface Class Reference

This abstract class represents an interface to an observables block.

```
#include <observables_interface.h>
```

Inheritance diagram for ObservablesInterface:



Public Member Functions

- virtual void **reset** ()=0

10.242.1 Detailed Description

This abstract class represents an interface to an observables block.

Abstract class for pseudorange_interfaces, derived from [GNSSBlockInterface](#). Since all its methods are virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

Definition at line 40 of file `observables_interface.h`.

The documentation for this class was generated from the following file:

- [observables_interface.h](#)

10.243 opt_t Struct Reference

Public Attributes

- const char * **name**
- int **format**
- void * **var**
- const char * **comment**

10.243.1 Detailed Description

Definition at line 910 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

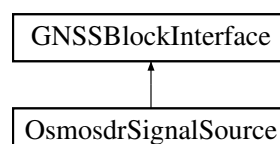
- [rtklib.h](#)

10.244 OsmosdrSignalSource Class Reference

This class reads samples OmoSDR-compatible front-ends, such as HackRF or Realtek's RTL2832U-based USB dongle DVB-T receivers (see <https://osmocom.org/projects/rtl-sdr/wiki>)

```
#include <osmosdr_signal_source.h>
```

Inheritance diagram for OsmosdrSignalSource:



Public Member Functions

- **OsmosdrSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_stream, unsigned int out_stream, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
Returns "Osmosdr_Signal_Source".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.244.1 Detailed Description

This class reads samples OsmoSDR-compatible front-ends, such as HackRF or Realtek's RTL2832U-based USB dongle DVB-T receivers (see <https://osmocom.org/projects/rtl-sdr/wiki>)

Definition at line 46 of file osmosdr_signal_source.h.

10.244.2 Member Function Documentation

10.244.2.1 implementation()

```
std::string OsmosdrSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "Osmosdr_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 63 of file osmosdr_signal_source.h.

The documentation for this class was generated from the following file:

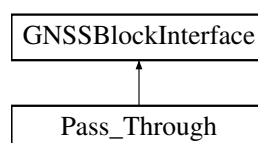
- [osmosdr_signal_source.h](#)

10.245 Pass_Through Class Reference

This class implements a block that connects input and output (does nothing)

```
#include <pass_through.h>
```

Inheritance diagram for Pass_Through:



Public Member Functions

- **Pass_Through** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_stream, unsigned int out_stream)
- std::string **role** () override
- std::string **implementation** () override
returns "Pass_Through"
- std::string **item_type** () const
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.245.1 Detailed Description

This class implements a block that connects input and output (does nothing)

Definition at line 40 of file `pass_through.h`.

10.245.2 Member Function Documentation

10.245.2.1 implementation()

```
std::string Pass_Through::implementation ( ) [inline], [override], [virtual]
```

returns "Pass_Through"

Implements [GNSSBlockInterface](#).

Definition at line 56 of file `pass_through.h`.

The documentation for this class was generated from the following file:

- [pass_through.h](#)

10.246 pclk_t Struct Reference

Public Attributes

- [gtime_t](#) **time**
- int **index**
- double **clk** [MAXSAT][1]
- float **std** [MAXSAT][1]

10.246.1 Detailed Description

Definition at line 485 of file rtklib.h.

The documentation for this struct was generated from the following file:

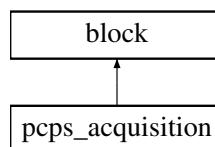
- [rtklib.h](#)

10.247 pcps_acquisition Class Reference

This class implements a Parallel Code Phase Search Acquisition.

```
#include <pcps_acquisition.h>
```

Inheritance diagram for pcps_acquisition:



Public Member Functions

- void [init](#) ()
Initializes acquisition algorithm and reserves memory.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro)
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.
- void [set_local_code](#) (std::complex< float > *code)
Sets local code for PCPS acquisition algorithm.
- void [set_state](#) (int32_t state)
If set to 1, ensures that acquisition starts at the first available sample.
- void [set_resampler_latency](#) (uint32_t latency_samples)
- uint32_t [mag](#) () const
Returns the maximum peak of grid search.
- void [set_active](#) (bool active)
Starts acquisition algorithm, turning from standby mode to active mode.
- void [set_channel](#) (uint32_t channel)
Set acquisition channel unique ID.
- void [set_channel_fsm](#) (std::weak_ptr< [ChannelFsm](#) > channel_fsm)
Set channel fsm associated to this acquisition instance.
- void [set_threshold](#) (float threshold)
Set statistics threshold of PCPS algorithm.
- void [set_doppler_max](#) (uint32_t doppler_max)
Set maximum Doppler grid search.
- void [set_doppler_step](#) (uint32_t doppler_step)
Set Doppler steps for the grid search.
- void [set_doppler_center](#) (int32_t doppler_center)
Set Doppler center frequency for the grid search. It will refresh the Doppler grid.
- int [general_work](#) (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
Parallel Code Phase Search Acquisition signal processing.

Friends

- pcps_acquisition_sptr **pcps_make_acquisition** (const [Acq_Conf](#) &conf_)

10.247.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition.

Check [An Open Source Galileo E1 Software Receiver](#), Algorithm 1, for a pseudocode description of this implementation.

Definition at line 95 of file pcps_acquisition.h.

10.247.2 Member Function Documentation

10.247.2.1 general_work()

```
int pcps_acquisition::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

Parallel Code Phase Search Acquisition signal processing.

10.247.2.2 init()

```
void pcps_acquisition::init ( )
```

Initializes acquisition algorithm and reserves memory.

10.247.2.3 mag()

```
uint32_t pcps_acquisition::mag ( ) const [inline]
```

Returns the maximum peak of grid search.

Definition at line 134 of file pcps_acquisition.h.

10.247.2.4 set_active()

```
void pcps_acquisition::set_active (
    bool active ) [inline]
```

Starts acquisition algorithm, turning from standby mode to active mode.

Parameters

<i>active</i>	- bool that activates/deactivates the block.
---------------	--

Definition at line 144 of file pcps_acquisition.h.

10.247.2.5 set_channel()

```
void pcps_acquisition::set_channel (
    uint32_t channel ) [inline]
```

Set acquisition channel unique ID.

Parameters

<i>channel</i>	- receiver channel.
----------------	---------------------

Definition at line 154 of file pcps_acquisition.h.

10.247.2.6 set_channel_fsm()

```
void pcps_acquisition::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline]
```

Set channel fsm associated to this acquisition instance.

Definition at line 162 of file pcps_acquisition.h.

10.247.2.7 set_doppler_center()

```
void pcps_acquisition::set_doppler_center (
    int32_t doppler_center ) [inline]
```

Set Doppler center frequency for the grid search. It will refresh the Doppler grid.

Parameters

<i>doppler_center</i>	- Frequency center of the search grid [Hz].
-----------------------	---

Definition at line 202 of file pcps_acquisition.h.

10.247.2.8 set_doppler_max()

```
void pcps_acquisition::set_doppler_max (
    uint32_t doppler_max ) [inline]
```

Set maximum Doppler grid search.

Parameters

<i>doppler_max</i>	- Maximum Doppler shift considered in the grid search [Hz].
--------------------	---

Definition at line 182 of file pcps_acquisition.h.

10.247.2.9 set_doppler_step()

```
void pcps_acquisition::set_doppler_step (
    uint32_t doppler_step ) [inline]
```

Set Doppler steps for the grid search.

Parameters

<i>doppler_step</i>	- Frequency bin of the search grid [Hz].
---------------------	--

Definition at line 192 of file pcps_acquisition.h.

10.247.2.10 set_gnss_synchro()

```
void pcps_acquisition::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [inline]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.

Parameters

<i>p_gnss_synchro</i>	Satellite information shared by the processing blocks.
-----------------------	--

Definition at line 110 of file pcps_acquisition.h.

10.247.2.11 set_local_code()

```
void pcps_acquisition::set_local_code (
    std::complex< float > * code )
```


Sets local code for PCPS acquisition algorithm.

Parameters

<i>code</i>	- Pointer to the PRN code.
-------------	----------------------------

10.247.2.12 set_state()

```
void pcps_acquisition::set_state (
    int32_t state )
```

If set to 1, ensures that acquisition starts at the first available sample.

Parameters

<i>state</i>	- int=1 forces start of acquisition
--------------	-------------------------------------

10.247.2.13 set_threshold()

```
void pcps_acquisition::set_threshold (
    float threshold ) [inline]
```

Set statistics threshold of PCPS algorithm.

Parameters

<i>threshold</i>	- Threshold for signal detection (check Navitec2012 , Algorithm 1, for a definition of this threshold).
------------------	---

Definition at line 172 of file pcps_acquisition.h.

The documentation for this class was generated from the following file:

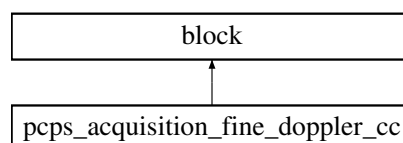
- [pcps_acquisition.h](#)

10.248 pcps_acquisition_fine_doppler_cc Class Reference

This class implements a Parallel Code Phase Search Acquisition.

```
#include <pcps_acquisition_fine_doppler_cc.h>
```

Inheritance diagram for pcps_acquisition_fine_doppler_cc:



Public Member Functions

- [~pcps_acquisition_fine_doppler_cc](#) ()=default
Default destructor.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro)
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.
- unsigned int [mag](#) () const
Returns the maximum peak of grid search.
- void [init](#) ()
Initializes acquisition algorithm.
- void [set_local_code](#) (std::complex< float > *code)
Sets local code for PCPS acquisition algorithm.
- void [set_active](#) (bool active)
Starts acquisition algorithm, turning from standby mode to active mode.
- void [set_channel](#) (unsigned int channel)
Set acquisition channel unique ID.
- void [set_channel_fsm](#) (std::weak_ptr< [ChannelFsm](#) > channel_fsm)
Set channel fsm associated to this acquisition instance.
- void [set_threshold](#) (float threshold)
Set statistics threshold of PCPS algorithm.
- void [set_doppler_max](#) (unsigned int doppler_max)
Set maximum Doppler grid search.
- void [set_doppler_step](#) (unsigned int doppler_step)
Set Doppler steps for the grid search.
- void [set_state](#) (int state)
If set to 1, ensures that acquisition starts at the first available sample.
- unsigned int [nextPowerOf2](#) (unsigned int n)
Obtains the next power of 2 greater or equal to the input parameter.
- void [dump_results](#) (int effective_fft_size)
- void [forecast](#) (int noutput_items, gr_vector_int &ninput_items_required)
- int [general_work](#) (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
Parallel Code Phase Search Acquisition signal processing.

Friends

- [pcps_acquisition_fine_doppler_cc_sptr](#) [pcps_make_acquisition_fine_doppler_cc](#) (const [Acq_Conf](#) &conf_)

10.248.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition.

Definition at line 77 of file [pcps_acquisition_fine_doppler_cc.h](#).

10.248.2 Constructor & Destructor Documentation

10.248.2.1 `~pcps_acquisition_fine_doppler_cc()`

```
pcps_acquisition_fine_doppler_cc::~pcps_acquisition_fine_doppler_cc ( ) [default]
```

Default destructor.

10.248.3 Member Function Documentation

10.248.3.1 `general_work()`

```
int pcps_acquisition_fine_doppler_cc::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

Parallel Code Phase Search Acquisition signal processing.

10.248.3.2 `init()`

```
void pcps_acquisition_fine_doppler_cc::init ( )
```

Initializes acquisition algorithm.

10.248.3.3 `mag()`

```
unsigned int pcps_acquisition_fine_doppler_cc::mag ( ) const [inline]
```

Returns the maximum peak of grid search.

Definition at line 98 of file `pcps_acquisition_fine_doppler_cc.h`.

10.248.3.4 `nextPowerOf2()`

```
unsigned int pcps_acquisition_fine_doppler_cc::nextPowerOf2 (
    unsigned int n )
```

Obtains the next power of 2 greater or equal to the input parameter.

Parameters

<i>n</i>	- Integer value to obtain the next power of 2.
----------	--

10.248.3.5 set_active()

```
void pcps_acquisition_fine_doppler_cc::set_active (  
    bool active ) [inline]
```

Starts acquisition algorithm, turning from standby mode to active mode.

Parameters

<i>active</i>	- bool that activates/deactivates the block.
---------------	--

Definition at line 119 of file pcps_acquisition_fine_doppler_cc.h.

10.248.3.6 set_channel()

```
void pcps_acquisition_fine_doppler_cc::set_channel (  
    unsigned int channel ) [inline]
```

Set acquisition channel unique ID.

Parameters

<i>channel</i>	- receiver channel.
----------------	---------------------

Definition at line 128 of file pcps_acquisition_fine_doppler_cc.h.

10.248.3.7 set_channel_fsm()

```
void pcps_acquisition_fine_doppler_cc::set_channel_fsm (  
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline]
```

Set channel fsm associated to this acquisition instance.

Definition at line 137 of file pcps_acquisition_fine_doppler_cc.h.

10.248.3.8 set_doppler_max()

```
void pcps_acquisition_fine_doppler_cc::set_doppler_max (
    unsigned int doppler_max ) [inline]
```

Set maximum Doppler grid search.

Parameters

<i>doppler_max</i>	- Maximum Doppler shift considered in the grid search [Hz].
--------------------	---

Definition at line 156 of file pcps_acquisition_fine_doppler_cc.h.

10.248.3.9 `set_doppler_step()`

```
void pcps_acquisition_fine_doppler_cc::set_doppler_step (
    unsigned int doppler_step )
```

Set Doppler steps for the grid search.

Parameters

<i>doppler_step</i>	- Frequency bin of the search grid [Hz].
---------------------	--

10.248.3.10 `set_gnss_synchro()`

```
void pcps_acquisition_fine_doppler_cc::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [inline]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.

Parameters

<i>p_gnss_synchro</i>	Satellite information shared by the processing blocks.
-----------------------	--

Definition at line 90 of file pcps_acquisition_fine_doppler_cc.h.

10.248.3.11 `set_local_code()`

```
void pcps_acquisition_fine_doppler_cc::set_local_code (
    std::complex< float > * code )
```

Sets local code for PCPS acquisition algorithm.

Parameters

<i>code</i>	- Pointer to the PRN code.
-------------	----------------------------

10.248.3.12 set_state()

```
void pcps_acquisition_fine_doppler_cc::set_state (
    int state )
```

If set to 1, ensures that acquisition starts at the first available sample.

Parameters

<i>state</i>	- int=1 forces start of acquisition
--------------	-------------------------------------

10.248.3.13 set_threshold()

```
void pcps_acquisition_fine_doppler_cc::set_threshold (
    float threshold ) [inline]
```

Set statistics threshold of PCPS algorithm.

Parameters

<i>threshold</i>	- Threshold for signal detection (check Navitec2012 , Algorithm 1, for a definition of this threshold).
------------------	---

Definition at line 147 of file pcps_acquisition_fine_doppler_cc.h.

The documentation for this class was generated from the following file:

- [pcps_acquisition_fine_doppler_cc.h](#)

10.249 pcps_acquisition_fpga Class Reference

This class implements a Parallel Code Phase Search Acquisition that uses the FPGA.

```
#include <pcps_acquisition_fpga.h>
```

Public Member Functions

- [~pcps_acquisition_fpga](#) ()=default
Destructor.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro)
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.
- uint32_t [mag](#) () const

- Returns the maximum peak of grid search.*
- void `init` ()
Initializes acquisition algorithm.
- void `set_local_code` ()
Sets local code for PCPS acquisition algorithm.
- void `set_state` (int32_t state)
If set to 1, ensures that acquisition starts at the first available sample.
- void `set_active` (bool active)
Starts acquisition algorithm, turning from standby mode to active mode.
- void `set_channel` (uint32_t channel)
Set acquisition channel unique ID.
- void `set_channel_fsm` (std::weak_ptr< [ChannelFsm](#) > channel_fsm)
Set channel fsm associated to this acquisition instance.
- void `set_threshold` (float threshold)
Set statistics threshold of PCPS algorithm.
- void `set_doppler_max` (uint32_t doppler_max)
Set maximum Doppler grid search.
- void `set_doppler_step` (uint32_t doppler_step)
Set Doppler steps for the grid search.
- void `set_doppler_center` (int32_t doppler_center)
Set Doppler center frequency for the grid search. It will refresh the Doppler grid.
- void `reset_acquisition` ()
This function triggers a HW reset of the FPGA PL.

Friends

- pcps_acquisition_fpga_sptr `pcps_make_acquisition_fpga` ([pcpsconf_fpga_t](#) conf_)

10.249.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition that uses the FPGA.

Check [An Open Source Galileo E1 Software Receiver](#), Algorithm 1, for a pseudocode description of this implementation.

Definition at line 74 of file `pcps_acquisition_fpga.h`.

10.249.2 Constructor & Destructor Documentation

10.249.2.1 `~pcps_acquisition_fpga()`

```
pcps_acquisition_fpga::~pcps_acquisition_fpga ( ) [default]
```

Destructor.

10.249.3 Member Function Documentation

10.249.3.1 init()

```
void pcps_acquisition_fpga::init ( )
```

Initializes acquisition algorithm.

10.249.3.2 mag()

```
uint32_t pcps_acquisition_fpga::mag ( ) const [inline]
```

Returns the maximum peak of grid search.

Definition at line 95 of file pcps_acquisition_fpga.h.

10.249.3.3 reset_acquisition()

```
void pcps_acquisition_fpga::reset_acquisition ( )
```

This function triggers a HW reset of the FPGA PL.

10.249.3.4 set_active()

```
void pcps_acquisition_fpga::set_active (
    bool active )
```

Starts acquisition algorithm, turning from standby mode to active mode.

Parameters

<i>active</i>	- bool that activates/deactivates the block.
---------------	--

10.249.3.5 set_channel()

```
void pcps_acquisition_fpga::set_channel (
    uint32_t channel ) [inline]
```

Set acquisition channel unique ID.

Parameters

<i>channel</i>	- receiver channel.
----------------	---------------------

Definition at line 128 of file pcps_acquisition_fpga.h.

10.249.3.6 set_channel_fsm()

```
void pcps_acquisition_fpga::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline]
```

Set channel fsm associated to this acquisition instance.

Definition at line 136 of file pcps_acquisition_fpga.h.

10.249.3.7 set_doppler_center()

```
void pcps_acquisition_fpga::set_doppler_center (
    int32_t doppler_center ) [inline]
```

Set Doppler center frequency for the grid search. It will refresh the Doppler grid.

Parameters

<i>doppler_center</i>	- Frequency center of the search grid [Hz].
-----------------------	---

Definition at line 175 of file pcps_acquisition_fpga.h.

10.249.3.8 set_doppler_max()

```
void pcps_acquisition_fpga::set_doppler_max (
    uint32_t doppler_max ) [inline]
```

Set maximum Doppler grid search.

Parameters

<i>doppler_max</i>	- Maximum Doppler shift considered in the grid search [Hz].
--------------------	---

Definition at line 155 of file pcps_acquisition_fpga.h.

10.249.3.9 set_doppler_step()

```
void pcps_acquisition_fpga::set_doppler_step (
    uint32_t doppler_step ) [inline]
```

Set Doppler steps for the grid search.

Parameters

<i>doppler_step</i>	- Frequency bin of the search grid [Hz].
---------------------	--

Definition at line 165 of file pcps_acquisition_fpga.h.

10.249.3.10 set_gnss_synchro()

```
void pcps_acquisition_fpga::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [inline]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.

Parameters

<i>p_gnss_synchro</i>	Satellite information shared by the processing blocks.
-----------------------	--

Definition at line 87 of file pcps_acquisition_fpga.h.

10.249.3.11 set_local_code()

```
void pcps_acquisition_fpga::set_local_code ( )
```

Sets local code for PCPS acquisition algorithm.

10.249.3.12 set_state()

```
void pcps_acquisition_fpga::set_state (
    int32_t state )
```

If set to 1, ensures that acquisition starts at the first available sample.

Parameters

<i>state</i>	- int=1 forces start of acquisition
--------------	-------------------------------------

10.249.3.13 set_threshold()

```
void pcps_acquisition_fpga::set_threshold (
    float threshold ) [inline]
```

Set statistics threshold of PCPS algorithm.

Parameters

<i>threshold</i>	- Threshold for signal detection (check Navitec2012 , Algorithm 1, for a definition of this threshold).
------------------	---

Definition at line 146 of file pcps_acquisition_fpga.h.

The documentation for this class was generated from the following file:

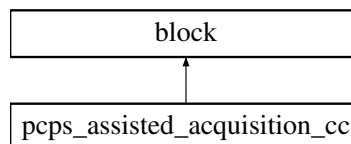
- [pcps_acquisition_fpga.h](#)

10.250 pcps_assisted_acquisition_cc Class Reference

This class implements a Parallel Code Phase Search Acquisition.

```
#include <pcps_assisted_acquisition_cc.h>
```

Inheritance diagram for pcps_assisted_acquisition_cc:



Public Member Functions

- [~pcps_assisted_acquisition_cc](#) ()
Default destructor.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro)
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.
- uint32_t [mag](#) () const
Returns the maximum peak of grid search.
- void [init](#) ()
Initializes acquisition algorithm.
- void [set_local_code](#) (std::complex< float > *code)
Sets local code for PCPS acquisition algorithm.
- void [set_active](#) (bool active)
Starts acquisition algorithm, turning from standby mode to active mode.

- void [set_channel](#) (uint32_t channel)
Set acquisition channel unique ID.
- void [set_channel_fsm](#) (std::weak_ptr< [ChannelFsm](#) > channel_fsm)
Set channel fsm associated to this acquisition instance.
- void [set_threshold](#) (float threshold)
Set statistics threshold of PCPS algorithm.
- void [set_state](#) (int32_t state)
- void [set_doppler_max](#) (uint32_t doppler_max)
Set maximum Doppler grid search.
- void [set_doppler_step](#) (uint32_t doppler_step)
Set Doppler steps for the grid search.
- int [general_work](#) (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
Parallel Code Phase Search Acquisition signal processing.
- void [forecast](#) (int noutput_items, gr_vector_int &ninput_items_required)

Friends

- pcps_assisted_acquisition_cc_sptr [pcps_make_assisted_acquisition_cc](#) (int32_t max_dwells, uint32_t sampled_ms, int32_t doppler_max, int32_t doppler_min, int64_t fs_in, int32_t samples_per_ms, bool dump, const std::string &dump_filename)

10.250.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition.

Check [An Open Source Galileo E1 Software Receiver](#), Algorithm 1, for a pseudocode description of this implementation.

Definition at line 78 of file pcps_assisted_acquisition_cc.h.

10.250.2 Constructor & Destructor Documentation

10.250.2.1 ~pcps_assisted_acquisition_cc()

```
pcps_assisted_acquisition_cc::~pcps_assisted_acquisition_cc ( )
```

Default destructor.

10.250.3 Member Function Documentation

10.250.3.1 general_work()

```
int pcps_assisted_acquisition_cc::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

Parallel Code Phase Search Acquisition signal processing.

10.250.3.2 init()

```
void pcps_assisted_acquisition_cc::init ( )
```

Initializes acquisition algorithm.

10.250.3.3 mag()

```
uint32_t pcps_assisted_acquisition_cc::mag ( ) const [inline]
```

Returns the maximum peak of grid search.

Definition at line 99 of file pcps_assisted_acquisition_cc.h.

10.250.3.4 set_active()

```
void pcps_assisted_acquisition_cc::set_active (
    bool active ) [inline]
```

Starts acquisition algorithm, turning from standby mode to active mode.

Parameters

<i>active</i>	- bool that activates/deactivates the block.
---------------	--

Definition at line 120 of file pcps_assisted_acquisition_cc.h.

10.250.3.5 set_channel()

```
void pcps_assisted_acquisition_cc::set_channel (
    uint32_t channel ) [inline]
```

Set acquisition channel unique ID.

Parameters

<i>channel</i>	- receiver channel.
----------------	---------------------

Definition at line 129 of file pcps_assisted_acquisition_cc.h.

10.250.3.6 set_channel_fsm()

```
void pcps_assisted_acquisition_cc::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline]
```

Set channel fsm associated to this acquisition instance.

Definition at line 137 of file pcps_assisted_acquisition_cc.h.

10.250.3.7 set_doppler_max()

```
void pcps_assisted_acquisition_cc::set_doppler_max (
    uint32_t doppler_max ) [inline]
```

Set maximum Doppler grid search.

Parameters

<i>doppler_max</i>	- Maximum Doppler shift considered in the grid search [Hz].
--------------------	---

Definition at line 161 of file pcps_assisted_acquisition_cc.h.

10.250.3.8 set_doppler_step()

```
void pcps_assisted_acquisition_cc::set_doppler_step (
    uint32_t doppler_step )
```

Set Doppler steps for the grid search.

Parameters

<i>doppler_step</i>	- Frequency bin of the search grid [Hz].
---------------------	--

10.250.3.9 set_gnss_synchro()

```
void pcps_assisted_acquisition_cc::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [inline]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.

Parameters

<i>p_gnss_synchro</i>	Satellite information shared by the processing blocks.
-----------------------	--

Definition at line 91 of file pcps_assisted_acquisition_cc.h.

10.250.3.10 set_local_code()

```
void pcps_assisted_acquisition_cc::set_local_code (
    std::complex< float > * code )
```

Sets local code for PCPS acquisition algorithm.

Parameters

<i>code</i>	- Pointer to the PRN code.
-------------	----------------------------

10.250.3.11 set_threshold()

```
void pcps_assisted_acquisition_cc::set_threshold (
    float threshold ) [inline]
```

Set statistics threshold of PCPS algorithm.

Parameters

<i>threshold</i>	- Threshold for signal detection (check Navitec2012 , Algorithm 1, for a definition of this threshold).
------------------	---

Definition at line 147 of file pcps_assisted_acquisition_cc.h.

The documentation for this class was generated from the following file:

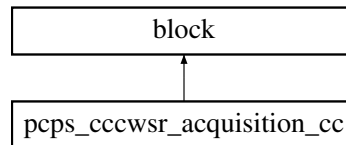
- [pcps_assisted_acquisition_cc.h](#)

10.251 pcps_cccwsr_acquisition_cc Class Reference

This class implements a Parallel Code Phase Search Acquisition with Coherent [Channel](#) Combining With Sign Recovery scheme.

```
#include <pcps_cccwsr_acquisition_cc.h>
```

Inheritance diagram for pcps_cccwsr_acquisition_cc:



Public Member Functions

- [~pcps_cccwsr_acquisition_cc](#) ()
Default destructor.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro)
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.
- uint32_t [mag](#) () const
Returns the maximum peak of grid search.
- void [init](#) ()
Initializes acquisition algorithm.
- void [set_local_code](#) (std::complex< float > *code_data, std::complex< float > *code_pilot)
Sets local code for CCCWSR acquisition algorithm.
- void [set_active](#) (bool active)
Starts acquisition algorithm, turning from standby mode to active mode.
- void [set_state](#) (int32_t state)
If set to 1, ensures that acquisition starts at the first available sample.
- void [set_channel](#) (uint32_t channel)
Set acquisition channel unique ID.
- void [set_channel_fsm](#) (std::weak_ptr< [ChannelFsm](#) > channel_fsm)
Set channel fsm associated to this acquisition instance.
- void [set_threshold](#) (float threshold)
Set statistics threshold of CCCWSR algorithm.
- void [set_doppler_max](#) (uint32_t doppler_max)
Set maximum Doppler grid search.
- void [set_doppler_step](#) (uint32_t doppler_step)
Set Doppler steps for the grid search.
- int [general_work](#) (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
Coherent [Channel](#) Combining With Sign Recovery Acquisition signal processing.

Friends

- pcps_cccwsr_acquisition_cc_sptr [pcps_cccwsr_make_acquisition_cc](#) (uint32_t sampled_ms, uint32_t max_dwells, uint32_t doppler_max, int64_t fs_in, int32_t samples_per_ms, int32_t samples_per_code, bool dump, const std::string &dump_filename)

10.251.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition with Coherent [Channel](#) Combining With Sign Recovery scheme.

Definition at line 67 of file pcps_cccwsr_acquisition_cc.h.

10.251.2 Constructor & Destructor Documentation

10.251.2.1 `~pcps_cccwsr_acquisition_cc()`

```
pcps_cccwsr_acquisition_cc::~~pcps_cccwsr_acquisition_cc ( )
```

Default destructor.

10.251.3 Member Function Documentation

10.251.3.1 `general_work()`

```
int pcps_cccwsr_acquisition_cc::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

Coherent [Channel](#) Combining With Sign Recovery Acquisition signal processing.

10.251.3.2 `init()`

```
void pcps_cccwsr_acquisition_cc::init ( )
```

Initializes acquisition algorithm.

10.251.3.3 `mag()`

```
uint32_t pcps_cccwsr_acquisition_cc::mag ( ) const [inline]
```

Returns the maximum peak of grid search.

Definition at line 88 of file pcps_cccwsr_acquisition_cc.h.

10.251.3.4 `set_active()`

```
void pcps_cccwsr_acquisition_cc::set_active (
    bool active ) [inline]
```

Starts acquisition algorithm, turning from standby mode to active mode.

Parameters

<i>active</i>	- bool that activates/deactivates the block.
---------------	--

Definition at line 110 of file pcps_cccwsr_acquisition_cc.h.

10.251.3.5 set_channel()

```
void pcps_cccwsr_acquisition_cc::set_channel (
    uint32_t channel ) [inline]
```

Set acquisition channel unique ID.

Parameters

<i>channel</i>	- receiver channel.
----------------	---------------------

Definition at line 126 of file pcps_cccwsr_acquisition_cc.h.

10.251.3.6 set_channel_fsm()

```
void pcps_cccwsr_acquisition_cc::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline]
```

Set channel fsm associated to this acquisition instance.

Definition at line 134 of file pcps_cccwsr_acquisition_cc.h.

10.251.3.7 set_doppler_max()

```
void pcps_cccwsr_acquisition_cc::set_doppler_max (
    uint32_t doppler_max ) [inline]
```

Set maximum Doppler grid search.

Parameters

<i>doppler_max</i>	- Maximum Doppler shift considered in the grid search [Hz].
--------------------	---

Definition at line 153 of file pcps_cccwsr_acquisition_cc.h.

10.251.3.8 `set_doppler_step()`

```
void pcps_cccwsr_acquisition_cc::set_doppler_step (
    uint32_t doppler_step ) [inline]
```

Set Doppler steps for the grid search.

Parameters

<i>doppler_step</i>	- Frequency bin of the search grid [Hz].
---------------------	--

Definition at line 162 of file `pcps_cccwsr_acquisition_cc.h`.

10.251.3.9 `set_gnss_synchro()`

```
void pcps_cccwsr_acquisition_cc::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [inline]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.

Parameters

<i>p_gnss_synchro</i>	Satellite information shared by the processing blocks.
-----------------------	--

Definition at line 80 of file `pcps_cccwsr_acquisition_cc.h`.

10.251.3.10 `set_local_code()`

```
void pcps_cccwsr_acquisition_cc::set_local_code (
    std::complex< float > * code_data,
    std::complex< float > * code_pilot )
```

Sets local code for CCCWSR acquisition algorithm.

Parameters

<i>data_code</i>	- Pointer to the data PRN code.
<i>pilot_code</i>	- Pointer to the pilot PRN code.

10.251.3.11 `set_state()`

```
void pcps_cccwsr_acquisition_cc::set_state (
```

```
int32_t state )
```

If set to 1, ensures that acquisition starts at the first available sample.

Parameters

<i>state</i>	- int=1 forces start of acquisition
--------------	-------------------------------------

10.251.3.12 set_threshold()

```
void pcps_cccwsr_acquisition_cc::set_threshold (
    float threshold ) [inline]
```

Set statistics threshold of CCCWSR algorithm.

Parameters

<i>threshold</i>	- Threshold for signal detection (check Navitec2012 , Algorithm 1, for a definition of this threshold).
------------------	---

Definition at line 144 of file pcps_cccwsr_acquisition_cc.h.

The documentation for this class was generated from the following file:

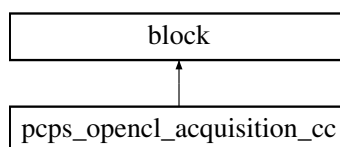
- [pcps_cccwsr_acquisition_cc.h](#)

10.252 pcps_openc1_acquisition_cc Class Reference

This class implements a Parallel Code Phase Search Acquisition.

```
#include <pcps_openc1_acquisition_cc.h>
```

Inheritance diagram for pcps_openc1_acquisition_cc:



Public Member Functions

- [~pcps_openc1_acquisition_cc](#) ()
Default destructor.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro)
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.

- uint32_t **mag** () const
Returns the maximum peak of grid search.
- void **init** ()
Initializes acquisition algorithm.
- void **set_local_code** (std::complex< float > *code)
Sets local code for PCPS acquisition algorithm.
- void **set_active** (bool active)
Starts acquisition algorithm, turning from standby mode to active mode.
- void **set_state** (int state)
If set to 1, ensures that acquisition starts at the first available sample.
- void **set_channel** (uint32_t channel)
Set acquisition channel unique ID.
- void **set_channel_fsm** (std::weak_ptr< ChannelFsm > channel_fsm)
Set channel fsm associated to this acquisition instance.
- void **set_threshold** (float threshold)
Set statistics threshold of PCPS algorithm.
- void **set_doppler_max** (uint32_t doppler_max)
Set maximum Doppler grid search.
- void **set_doppler_step** (uint32_t doppler_step)
Set Doppler steps for the grid search.
- bool **openc1_ready** () const
- void **acquisition_core_volk** ()
- void **acquisition_core_openc1** ()
- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
Parallel Code Phase Search Acquisition signal processing.

Friends

- pcps_openc1_acquisition_cc_sptr **pcps_make_openc1_acquisition_cc** (uint32_t sampled_ms, uint32_t max_dwells, uint32_t doppler_max, int64_t fs_in, int samples_per_ms, int samples_per_code, bool bit_transition_flag, bool dump, const std::string &dump_filename)

10.252.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition.

Check [An Open Source Galileo E1 Software Receiver](#), Algorithm 1, for a pseudocode description of this implementation.

Definition at line 86 of file pcps_openc1_acquisition_cc.h.

10.252.2 Constructor & Destructor Documentation

10.252.2.1 `~pcps_openc1_acquisition_cc()`

```
pcps_openc1_acquisition_cc::~~pcps_openc1_acquisition_cc ( )
```

Default destructor.

10.252.3 Member Function Documentation

10.252.3.1 `general_work()`

```
int pcps_openc1_acquisition_cc::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

Parallel Code Phase Search Acquisition signal processing.

10.252.3.2 `init()`

```
void pcps_openc1_acquisition_cc::init ( )
```

Initializes acquisition algorithm.

10.252.3.3 `mag()`

```
uint32_t pcps_openc1_acquisition_cc::mag ( ) const [inline]
```

Returns the maximum peak of grid search.

Definition at line 107 of file `pcps_openc1_acquisition_cc.h`.

10.252.3.4 `set_active()`

```
void pcps_openc1_acquisition_cc::set_active (
    bool active ) [inline]
```

Starts acquisition algorithm, turning from standby mode to active mode.

Parameters

<i>active</i>	- bool that activates/deactivates the block.
---------------	--

Definition at line 128 of file pcps_openc1_acquisition_cc.h.

10.252.3.5 set_channel()

```
void pcps_openc1_acquisition_cc::set_channel (
    uint32_t channel ) [inline]
```

Set acquisition channel unique ID.

Parameters

<i>channel</i>	- receiver channel.
----------------	---------------------

Definition at line 144 of file pcps_openc1_acquisition_cc.h.

10.252.3.6 set_channel_fsm()

```
void pcps_openc1_acquisition_cc::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline]
```

Set channel fsm associated to this acquisition instance.

Definition at line 152 of file pcps_openc1_acquisition_cc.h.

10.252.3.7 set_doppler_max()

```
void pcps_openc1_acquisition_cc::set_doppler_max (
    uint32_t doppler_max ) [inline]
```

Set maximum Doppler grid search.

Parameters

<i>doppler_max</i>	- Maximum Doppler shift considered in the grid search [Hz].
--------------------	---

Definition at line 171 of file pcps_openc1_acquisition_cc.h.

10.252.3.8 set_doppler_step()

```
void pcps_openc1_acquisition_cc::set_doppler_step (
    uint32_t doppler_step ) [inline]
```

Set Doppler steps for the grid search.

Parameters

<i>doppler_step</i>	- Frequency bin of the search grid [Hz].
---------------------	--

Definition at line 180 of file pcps_openc1_acquisition_cc.h.

10.252.3.9 set_gnss_synchro()

```
void pcps_openc1_acquisition_cc::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [inline]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.

Parameters

<i>p_gnss_synchro</i>	Satellite information shared by the processing blocks.
-----------------------	--

Definition at line 99 of file pcps_openc1_acquisition_cc.h.

10.252.3.10 set_local_code()

```
void pcps_openc1_acquisition_cc::set_local_code (
    std::complex< float > * code )
```

Sets local code for PCPS acquisition algorithm.

Parameters

<i>code</i>	- Pointer to the PRN code.
-------------	----------------------------

10.252.3.11 set_state()

```
void pcps_openc1_acquisition_cc::set_state (
    int state )
```

If set to 1, ensures that acquisition starts at the first available sample.

Parameters

<i>state</i>	- int=1 forces start of acquisition
--------------	-------------------------------------

10.252.3.12 `set_threshold()`

```
void pcps_openc1_acquisition_cc::set_threshold (
    float threshold ) [inline]
```

Set statistics threshold of PCPS algorithm.

Parameters

<i>threshold</i>	- Threshold for signal detection (check Navitec2012 , Algorithm 1, for a definition of this threshold).
------------------	---

Definition at line 162 of file `pcps_openc1_acquisition_cc.h`.

The documentation for this class was generated from the following file:

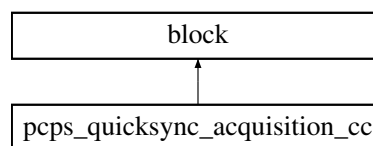
- [pcps_openc1_acquisition_cc.h](#)

10.253 `pcps_quicksync_acquisition_cc` Class Reference

This class implements a Parallel Code Phase Search Acquisition with the implementation of the Sparse QuickSync Algorithm.

```
#include <pcps_quicksync_acquisition_cc.h>
```

Inheritance diagram for `pcps_quicksync_acquisition_cc`:



Public Member Functions

- [~pcps_quicksync_acquisition_cc](#) ()
Default destructor.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro)
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.
- uint32_t [mag](#) () const
Returns the maximum peak of grid search.

- void `init` ()
Initializes acquisition algorithm.
- void `set_local_code` (std::complex< float > *code)
Sets local code for PCPS acquisition algorithm.
- void `set_active` (bool active)
Starts acquisition algorithm, turning from standby mode to active mode.
- void `set_state` (int32_t state)
If set to 1, ensures that acquisition starts at the first available sample.
- void `set_channel` (uint32_t channel)
Set acquisition channel unique ID.
- void `set_channel_fsm` (std::weak_ptr< ChannelFsm > channel_fsm)
Set channel fsm associated to this acquisition instance.
- void `set_threshold` (float threshold)
Set statistics threshold of PCPS algorithm.
- void `set_doppler_max` (uint32_t doppler_max)
Set maximum Doppler grid search.
- void `set_doppler_step` (uint32_t doppler_step)
Set Doppler steps for the grid search.
- int `general_work` (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
Parallel Code Phase Search Acquisition signal processing.

Friends

- pcps_quicksync_acquisition_cc_sptr `pcps_quicksync_make_acquisition_cc` (uint32_t folding_factor, uint32_t sampled_ms, uint32_t max_dwells, uint32_t doppler_max, int64_t fs_in, int32_t samples_per_ms, int32_t samples_per_code, bool bit_transition_flag, bool dump, const std::string &dump_filename)

10.253.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition with the implementation of the Sparse QuickSync Algorithm.

Check [Faster GPS via the Sparse Fourier Transform](#), for details of its implementation and functionality.

Definition at line 88 of file pcps_quicksync_acquisition_cc.h.

10.253.2 Constructor & Destructor Documentation

10.253.2.1 ~pcps_quicksync_acquisition_cc()

```
pcps_quicksync_acquisition_cc::~pcps_quicksync_acquisition_cc ( )
```

Default destructor.

10.253.3 Member Function Documentation

10.253.3.1 `general_work()`

```
int pcps_quicksync_acquisition_cc::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

Parallel Code Phase Search Acquisition signal processing.

10.253.3.2 `init()`

```
void pcps_quicksync_acquisition_cc::init ( )
```

Initializes acquisition algorithm.

10.253.3.3 `mag()`

```
uint32_t pcps_quicksync_acquisition_cc::mag ( ) const [inline]
```

Returns the maximum peak of grid search.

Definition at line 109 of file `pcps_quicksync_acquisition_cc.h`.

10.253.3.4 `set_active()`

```
void pcps_quicksync_acquisition_cc::set_active (
    bool active ) [inline]
```

Starts acquisition algorithm, turning from standby mode to active mode.

Parameters

<i>active</i>	- bool that activates/deactivates the block.
---------------	--

Definition at line 130 of file `pcps_quicksync_acquisition_cc.h`.

10.253.3.5 set_channel()

```
void pcps_quicksync_acquisition_cc::set_channel (
    uint32_t channel ) [inline]
```

Set acquisition channel unique ID.

Parameters

<i>channel</i>	- receiver channel.
----------------	---------------------

Definition at line 146 of file pcps_quicksync_acquisition_cc.h.

10.253.3.6 set_channel_fsm()

```
void pcps_quicksync_acquisition_cc::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline]
```

Set channel fsm associated to this acquisition instance.

Definition at line 154 of file pcps_quicksync_acquisition_cc.h.

10.253.3.7 set_doppler_max()

```
void pcps_quicksync_acquisition_cc::set_doppler_max (
    uint32_t doppler_max ) [inline]
```

Set maximum Doppler grid search.

Parameters

<i>doppler_max</i>	- Maximum Doppler shift considered in the grid search [Hz].
--------------------	---

Definition at line 173 of file pcps_quicksync_acquisition_cc.h.

10.253.3.8 set_doppler_step()

```
void pcps_quicksync_acquisition_cc::set_doppler_step (
    uint32_t doppler_step ) [inline]
```

Set Doppler steps for the grid search.

Parameters

<i>doppler_step</i>	- Frequency bin of the search grid [Hz].
---------------------	--

Definition at line 182 of file pcps_quicksync_acquisition_cc.h.

10.253.3.9 `set_gnss_synchro()`

```
void pcps_quicksync_acquisition_cc::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [inline]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.

Parameters

<i>p_gnss_synchro</i>	Satellite information shared by the processing blocks.
-----------------------	--

Definition at line 101 of file pcps_quicksync_acquisition_cc.h.

10.253.3.10 `set_local_code()`

```
void pcps_quicksync_acquisition_cc::set_local_code (
    std::complex< float > * code )
```

Sets local code for PCPS acquisition algorithm.

Parameters

<i>code</i>	- Pointer to the PRN code.
-------------	----------------------------

10.253.3.11 `set_state()`

```
void pcps_quicksync_acquisition_cc::set_state (
    int32_t state )
```

If set to 1, ensures that acquisition starts at the first available sample.

Parameters

<i>state</i>	- int=1 forces start of acquisition
--------------	-------------------------------------

10.253.3.12 set_threshold()

```
void pcps_quicksync_acquisition_cc::set_threshold (
    float threshold ) [inline]
```

Set statistics threshold of PCPS algorithm.

Parameters

<i>threshold</i>	- Threshold for signal detection (check Navitec2012 , Algorithm 1, for a definition of this threshold).
------------------	---

Definition at line 164 of file pcps_quicksync_acquisition_cc.h.

The documentation for this class was generated from the following file:

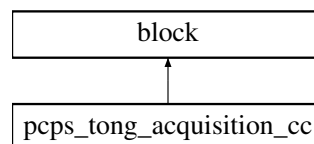
- [pcps_quicksync_acquisition_cc.h](#)

10.254 pcps_tong_acquisition_cc Class Reference

This class implements a Parallel Code Phase Search Acquisition with Tong algorithm.

```
#include <pcps_tong_acquisition_cc.h>
```

Inheritance diagram for pcps_tong_acquisition_cc:



Public Member Functions

- [~pcps_tong_acquisition_cc](#) ()
Default destructor.
- void [set_gnss_synchro](#) ([Gnss_Synchro](#) *p_gnss_synchro)
Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.
- uint32_t [mag](#) () const
Returns the maximum peak of grid search.
- void [init](#) ()
Initializes acquisition algorithm.
- void [set_local_code](#) (std::complex< float > *code)
Sets local code for TONG acquisition algorithm.
- void [set_active](#) (bool active)
Starts acquisition algorithm, turning from standby mode to active mode.

- void [set_state](#) (int32_t state)
If set to 1, ensures that acquisition starts at the first available sample.
- void [set_channel](#) (uint32_t channel)
Set acquisition channel unique ID.
- void [set_channel_fsm](#) (std::weak_ptr< [ChannelFsm](#) > channel_fsm)
Set channel fsm associated to this acquisition instance.
- void [set_threshold](#) (float threshold)
Set statistics threshold of TONG algorithm.
- void [set_doppler_max](#) (uint32_t doppler_max)
Set maximum Doppler grid search.
- void [set_doppler_step](#) (uint32_t doppler_step)
Set Doppler steps for the grid search.
- int [general_work](#) (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
Parallel Code Phase Search Acquisition signal processing.

Friends

- pcps_tong_acquisition_cc_sptr **pcps_tong_make_acquisition_cc** (uint32_t sampled_ms, uint32_t doppler_max, int64_t fs_in, int32_t samples_per_ms, int32_t samples_per_code, uint32_t tong_init_val, uint32_t tong_max_val, uint32_t tong_max_dwells, bool dump, const std::string &dump_filename)

10.254.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition with Tong algorithm.

Definition at line 82 of file pcps_tong_acquisition_cc.h.

10.254.2 Constructor & Destructor Documentation

10.254.2.1 ~pcps_tong_acquisition_cc()

```
pcps_tong_acquisition_cc::~pcps_tong_acquisition_cc ( )
```

Default destructor.

10.254.3 Member Function Documentation

10.254.3.1 general_work()

```
int pcps_tong_acquisition_cc::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

Parallel Code Phase Search Acquisition signal processing.

10.254.3.2 init()

```
void pcps_tong_acquisition_cc::init ( )
```

Initializes acquisition algorithm.

10.254.3.3 mag()

```
uint32_t pcps_tong_acquisition_cc::mag ( ) const [inline]
```

Returns the maximum peak of grid search.

Definition at line 103 of file `pcps_tong_acquisition_cc.h`.

10.254.3.4 set_active()

```
void pcps_tong_acquisition_cc::set_active (
    bool active ) [inline]
```

Starts acquisition algorithm, turning from standby mode to active mode.

Parameters

<i>active</i>	- bool that activates/deactivates the block.
---------------	--

Definition at line 124 of file `pcps_tong_acquisition_cc.h`.

10.254.3.5 set_channel()

```
void pcps_tong_acquisition_cc::set_channel (
    uint32_t channel ) [inline]
```

Set acquisition channel unique ID.

Parameters

<i>channel</i>	- receiver channel.
----------------	---------------------

Definition at line 140 of file pcps_tong_acquisition_cc.h.

10.254.3.6 set_channel_fsm()

```
void pcps_tong_acquisition_cc::set_channel_fsm (
    std::weak_ptr< ChannelFsm > channel_fsm ) [inline]
```

Set channel fsm associated to this acquisition instance.

Definition at line 148 of file pcps_tong_acquisition_cc.h.

10.254.3.7 set_doppler_max()

```
void pcps_tong_acquisition_cc::set_doppler_max (
    uint32_t doppler_max ) [inline]
```

Set maximum Doppler grid search.

Parameters

<i>doppler_max</i>	- Maximum Doppler shift considered in the grid search [Hz].
--------------------	---

Definition at line 167 of file pcps_tong_acquisition_cc.h.

10.254.3.8 set_doppler_step()

```
void pcps_tong_acquisition_cc::set_doppler_step (
    uint32_t doppler_step ) [inline]
```

Set Doppler steps for the grid search.

Parameters

<i>doppler_step</i>	- Frequency bin of the search grid [Hz].
---------------------	--

Definition at line 176 of file pcps_tong_acquisition_cc.h.

10.254.3.9 set_gnss_synchro()

```
void pcps_tong_acquisition_cc::set_gnss_synchro (
    Gnss_Synchro * p_gnss_synchro ) [inline]
```

Set acquisition/tracking common [Gnss_Synchro](#) object pointer to exchange synchronization data between acquisition and tracking blocks.

Parameters

<i>p_gnss_synchro</i>	Satellite information shared by the processing blocks.
-----------------------	--

Definition at line 95 of file `pcps_tong_acquisition_cc.h`.

10.254.3.10 set_local_code()

```
void pcps_tong_acquisition_cc::set_local_code (
    std::complex< float > * code )
```

Sets local code for TONG acquisition algorithm.

Parameters

<i>code</i>	- Pointer to the PRN code.
-------------	----------------------------

10.254.3.11 set_state()

```
void pcps_tong_acquisition_cc::set_state (
    int32_t state )
```

If set to 1, ensures that acquisition starts at the first available sample.

Parameters

<i>state</i>	- int=1 forces start of acquisition
--------------	-------------------------------------

10.254.3.12 set_threshold()

```
void pcps_tong_acquisition_cc::set_threshold (
    float threshold ) [inline]
```

Set statistics threshold of TONG algorithm.

Parameters

<i>threshold</i>	- Threshold for signal detection (check Navitec2012 , Algorithm 1, for a definition of this threshold).
------------------	---

Definition at line 158 of file pcps_tong_acquisition_cc.h.

The documentation for this class was generated from the following file:

- [pcps_tong_acquisition_cc.h](#)

10.255 pcpsconf_fpga_t Struct Reference

Public Attributes

- std::string **device_name**
- int64_t **fs_in**
- float **doppler_step2**
- uint32_t * **all_fft_codes**
- uint32_t **doppler_max**
- uint32_t **select_queue_Fpga**
- uint32_t **downsampling_factor**
- uint32_t **total_block_exp**
- uint32_t **excludelimit**
- uint32_t **num_doppler_bins_step2**
- uint32_t **max_num_acqs**
- int32_t **samples_per_code**
- int32_t **code_length**
- bool **make_2_steps**
- bool **repeat_satellite**

10.255.1 Detailed Description

Definition at line 42 of file pcps_acquisition_fpga.h.

The documentation for this struct was generated from the following file:

- [pcps_acquisition_fpga.h](#)

10.256 pcv_t Struct Reference

Public Attributes

- int **sat**
- char **type** [[MAXANT](#)]
- char **code** [[MAXANT](#)]
- [gtime_t](#) **ts**
- [gtime_t](#) **te**
- double **off** [[NFREQ](#)][3]
- double **var** [[NFREQ](#)][19]

10.256.1 Detailed Description

Definition at line 398 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.257 `pcvs_t` Struct Reference

Public Attributes

- `int n`
- `int nmax`
- `pcv_t * pcv`

10.257.1 Detailed Description

Definition at line 410 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.258 `peph_t` Struct Reference

Public Attributes

- `gtime_t time`
- `int index`
- `double pos [MAXSAT][4]`
- `float std [MAXSAT][4]`
- `double vel [MAXSAT][4]`
- `float vst [MAXSAT][4]`
- `float cov [MAXSAT][3]`
- `float vco [MAXSAT][3]`

10.258.1 Detailed Description

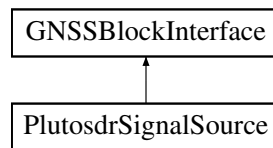
Definition at line 472 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.259 PlutosdrSignalSource Class Reference

Inheritance diagram for PlutosdrSignalSource:



Public Member Functions

- **PlutosdrSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_stream, unsigned int out_stream, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
Returns "Plutosdr_Signal_Source".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.259.1 Detailed Description

Definition at line 47 of file plutosdr_signal_source.h.

10.259.2 Member Function Documentation

10.259.2.1 implementation()

```
std::string PlutosdrSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "Plutosdr_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 64 of file plutosdr_signal_source.h.

The documentation for this class was generated from the following file:

- [plutosdr_signal_source.h](#)

10.260 pppcorr_t Struct Reference

Public Attributes

- int **nsta**
- char **stas** [MAXSTA][8]
- double **rr** [MAXSTA][3]
- int **ns** [MAXSTA]
- int **nsmax** [MAXSTA]
- int **nt** [MAXSTA]
- int **ntmax** [MAXSTA]
- [stec_t](#) * **stec** [MAXSTA]
- [trop_t](#) * **trop** [MAXSTA]

10.260.1 Detailed Description

Definition at line 734 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.261 prcopt_t Struct Reference

Public Attributes

- int **mode**
- int **soltype**
- int **nf**
- int **navsys**
- double **elmin**
- [snrmask_t](#) **snrmask**
- int **sateph**
- int **modear**
- int **glomodear**
- int **bdsmodear**
- int **maxout**
- int **minlock**
- int **minfix**
- int **armaxiter**
- int **ionoopt**
- int **tropopt**
- int **dynamics**
- int **tidecorr**
- int **niter**
- int **codesmooth**
- int **intpref**
- int **sbscorr**
- int **sbassatsel**
- int **rovpos**

- int **refpos**
- double **eratio** [NFREQ]
- double **err** [5]
- double **std** [3]
- double **prn** [6]
- double **sclkstab**
- double **thresar** [8]
- double **elmaskar**
- double **elmaskhold**
- double **thresslip**
- double **maxtdiff**
- double **maxinno**
- double **maxgdop**
- double **baseline** [2]
- double **ru** [3]
- double **rb** [3]
- char **anttype** [2][MAXANT]
- double **antdel** [2][3]
- [pcv_t](#) **pcvr** [2]
- unsigned char **exsats** [MAXSAT]
- int **maxaveep**
- int **initrst**
- int **outsingle**
- char **rnxopt** [2][256]
- int **posopt** [6]
- int **syncsol**
- double **odisp** [2][6 * 11]
- [exterr_t](#) **exterr**
- int **freqopt**
- char **pppopt** [256]

10.261.1 Detailed Description

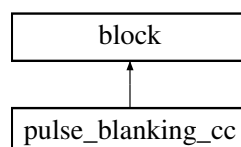
Definition at line 936 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.262 pulse_blanking_cc Class Reference

Inheritance diagram for pulse_blanking_cc:



Public Member Functions

- void **forecast** (int noutput_items, gr_vector_int &ninput_items_required)
- int **general_work** (int noutput_items __attribute__((unused)), gr_vector_int &ninput_items __attribute__((unused)), gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- pulse_blanking_cc_sptr **make_pulse_blanking_cc** (float pfa, int32_t length_, int32_t n_segments_est, int32_t n_segments_reset)

10.262.1 Detailed Description

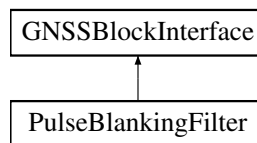
Definition at line 46 of file pulse_blanking_cc.h.

The documentation for this class was generated from the following file:

- [pulse_blanking_cc.h](#)

10.263 PulseBlankingFilter Class Reference

Inheritance diagram for PulseBlankingFilter:



Public Member Functions

- **PulseBlankingFilter** (const [ConfigurationInterface](#) *configuration, std::string role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "Pulse_Blanking_Filter".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.263.1 Detailed Description

Definition at line 36 of file pulse_blanking_filter.h.

10.263.2 Member Function Documentation

10.263.2.1 implementation()

```
std::string PulseBlankingFilter::implementation ( ) [inline], [override], [virtual]
```

Returns "Pulse_Blanking_Filter".

Implements [GNSSBlockInterface](#).

Definition at line 51 of file pulse_blanking_filter.h.

The documentation for this class was generated from the following file:

- [pulse_blanking_filter.h](#)

10.264 Pvt_Conf Class Reference

Public Attributes

- `std::map< int, int > rtcmsg_rate_ms`
- `std::string rinex_name`
- `std::string dump_filename`
- `std::string nmea_dump_filename`
- `std::string nmea_dump_devname`
- `std::string rtcmsg_dump_devname`
- `std::string output_path`
- `std::string rinex_output_path`
- `std::string gpx_output_path`
- `std::string geojson_output_path`
- `std::string nmea_output_file_path`
- `std::string kml_output_path`
- `std::string xml_output_path`
- `std::string rtcmsg_output_file_path`
- `std::string udp_addresses`
- `uint32_t type_of_receiver`
- `int32_t output_rate_ms`
- `int32_t display_rate_ms`
- `int32_t kml_rate_ms`
- `int32_t gpx_rate_ms`
- `int32_t geojson_rate_ms`
- `int32_t nmea_rate_ms`
- `int32_t rinex_version`
- `int32_t rinexobs_rate_ms`
- `int32_t max_obs_block_rx_clock_offset_ms`
- `int udp_port`
- `uint16_t rtcmsg_tcp_port`
- `uint16_t rtcmsg_station_id`
- `bool flag_nmea_tty_port`

- bool **flag_rtcn_server**
- bool **flag_rtcn_tty_port**
- bool **output_enabled**
- bool **rinex_output_enabled**
- bool **gpx_output_enabled**
- bool **geojson_output_enabled**
- bool **nmea_output_file_enabled**
- bool **kml_output_enabled**
- bool **xml_output_enabled**
- bool **rtcn_output_file_enabled**
- bool **monitor_enabled**
- bool **protobuf_enabled**
- bool **enable_rx_clock_correction**
- bool **show_local_time_zone**
- bool **pre_2009_file**
- bool **dump**
- bool **dump_mat**

10.264.1 Detailed Description

Definition at line 27 of file `pvt_conf.h`.

The documentation for this class was generated from the following file:

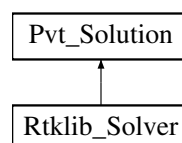
- [pvt_conf.h](#)

10.265 Pvt_Solution Class Reference

Base class for a PVT solution.

```
#include <pvt_solution.h>
```

Inheritance diagram for Pvt_Solution:



Public Member Functions

- void **set_rx_pos** (const std::array< double, 3 > &pos)
Set position: X, Y, Z in Cartesian ECEF coordinates [m].
- void **set_rx_vel** (const std::array< double, 3 > &vel)
Set velocity: East [m/s], North [m/s], Up [m/s].
- void **set_position_UTC_time** (const boost::posix_time::ptime &pt)
- void **set_time_offset_s** (double offset)
Set RX time offset [s].
- void **set_clock_drift_ppm** (double clock_drift_ppm)
Set the Rx clock drift [ppm].
- void **set_speed_over_ground** (double speed_m_s)
Set RX speed over ground [m/s].
- void **set_course_over_ground** (double cog_deg)
Set RX course over ground [deg].
- void **set_valid_position** (bool is_valid)
- void **set_num_valid_observations** (int num)
Set the number of valid pseudorange observations (valid satellites)
- void **set_pre_2009_file** (bool pre_2009_file)
Flag for the week rollover computation in post processing mode for signals older than 2009.
- void **set_averaging_depth** (int depth)
Set length of averaging window.
- void **set_averaging_flag** (bool flag)
- void **perform_pos_averaging** ()
- std::array< double, 3 > **get_rx_pos** () const
- std::array< double, 3 > **get_rx_vel** () const
- boost::posix_time::ptime **get_position_UTC_time** () const
- double **get_latitude** () const
Get RX position Latitude WGS84 [deg].
- double **get_longitude** () const
Get RX position Longitude WGS84 [deg].
- double **get_height** () const
Get RX position height WGS84 [m].
- double **get_time_offset_s** () const
Get RX time offset [s].
- double **get_clock_drift_ppm** () const
Get the Rx clock drift [ppm].
- double **get_speed_over_ground** () const
Get RX speed over ground [m/s].
- double **get_course_over_ground** () const
Get RX course over ground [deg].
- double **get_avg_latitude** () const
Get RX position averaged Latitude WGS84 [deg].
- double **get_avg_longitude** () const
Get RX position averaged Longitude WGS84 [deg].
- double **get_avg_height** () const
Get RX position averaged height WGS84 [m].
- int **get_num_valid_observations** () const
Get the number of valid pseudorange observations (valid satellites)
- bool **is_pre_2009** () const
- bool **is_valid_position** () const
- bool **is_averaging** () const
- virtual double **get_hdop** () const =0
- virtual double **get_vdop** () const =0
- virtual double **get_pdop** () const =0
- virtual double **get_gdop** () const =0

10.265.1 Detailed Description

Base class for a PVT solution.

Definition at line 33 of file pvt_solution.h.

10.265.2 Member Function Documentation

10.265.2.1 get_avg_height()

```
double Pvt_Solution::get_avg_height ( ) const
```

Get RX position averaged height WGS84 [m].

10.265.2.2 get_avg_latitude()

```
double Pvt_Solution::get_avg_latitude ( ) const
```

Get RX position averaged Latitude WGS84 [deg].

10.265.2.3 get_avg_longitude()

```
double Pvt_Solution::get_avg_longitude ( ) const
```

Get RX position averaged Longitude WGS84 [deg].

10.265.2.4 get_clock_drift_ppm()

```
double Pvt_Solution::get_clock_drift_ppm ( ) const
```

Get the Rx clock drift [ppm].

10.265.2.5 get_course_over_ground()

```
double Pvt_Solution::get_course_over_ground ( ) const
```

Get RX course over ground [deg].

10.265.2.6 get_height()

```
double Pvt_Solution::get_height ( ) const
```

Get RX position height WGS84 [m].

10.265.2.7 get_latitude()

```
double Pvt_Solution::get_latitude ( ) const
```

Get RX position Latitude WGS84 [deg].

10.265.2.8 get_longitude()

```
double Pvt_Solution::get_longitude ( ) const
```

Get RX position Longitude WGS84 [deg].

10.265.2.9 get_num_valid_observations()

```
int Pvt_Solution::get_num_valid_observations ( ) const
```

Get the number of valid pseudorange observations (valid satellites)

10.265.2.10 get_speed_over_ground()

```
double Pvt_Solution::get_speed_over_ground ( ) const
```

Get RX speed over ground [m/s].

10.265.2.11 get_time_offset_s()

```
double Pvt_Solution::get_time_offset_s ( ) const
```

Get RX time offset [s].

10.265.2.12 set_averaging_depth()

```
void Pvt_Solution::set_averaging_depth (
    int depth )
```

Set length of averaging window.

10.265.2.13 set_clock_drift_ppm()

```
void Pvt_Solution::set_clock_drift_ppm (
    double clock_drift_ppm )
```

Set the Rx clock drift [ppm].

10.265.2.14 set_course_over_ground()

```
void Pvt_Solution::set_course_over_ground (
    double cog_deg )
```

Set RX course over ground [deg].

10.265.2.15 set_num_valid_observations()

```
void Pvt_Solution::set_num_valid_observations (
    int num )
```

Set the number of valid pseudorange observations (valid satellites)

10.265.2.16 set_pre_2009_file()

```
void Pvt_Solution::set_pre_2009_file (
    bool pre_2009_file )
```

Flag for the week rollover computation in post processing mode for signals older than 2009.

10.265.2.17 set_rx_pos()

```
void Pvt_Solution::set_rx_pos (
    const std::array< double, 3 > & pos )
```

Set position: X, Y, Z in Cartesian ECEF coordinates [m].

10.265.2.18 set_rx_vel()

```
void Pvt_Solution::set_rx_vel (
    const std::array< double, 3 > & vel )
```

Set velocity: East [m/s], North [m/s], Up [m/s].

10.265.2.19 set_speed_over_ground()

```
void Pvt_Solution::set_speed_over_ground (
    double speed_m_s )
```

Set RX speed over ground [m/s].

10.265.2.20 set_time_offset_s()

```
void Pvt_Solution::set_time_offset_s (
    double offset )
```

Set RX time offset [s].

The documentation for this class was generated from the following file:

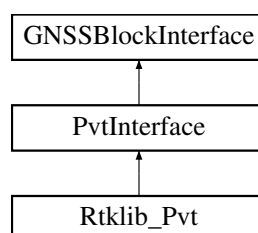
- [pvt_solution.h](#)

10.266 PvtInterface Class Reference

This class represents an interface to a PVT block.

```
#include <pvt_interface.h>
```

Inheritance diagram for PvtInterface:



Public Member Functions

- virtual void **reset** ()=0
- virtual void **clear_ephemeris** ()=0
- virtual std::map< int, [Gps_Ephemeris](#) > **get_gps_ephemeris** () const =0
- virtual std::map< int, [Galileo_Ephemeris](#) > **get_galileo_ephemeris** () const =0
- virtual std::map< int, [Gps_Almanac](#) > **get_gps_almanac** () const =0
- virtual std::map< int, [Galileo_Almanac](#) > **get_galileo_almanac** () const =0
- virtual bool **get_latest_PVT** (double *longitude_deg, double *latitude_deg, double *height_m, double *ground_speed_kmh, double *course_over_ground_deg, time_t *UTC_time)=0

10.266.1 Detailed Description

This class represents an interface to a PVT block.

Abstract class for PVT interfaces, derived from [GNSSBlockInterface](#). Since all its methods are virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

Definition at line 45 of file pvt_interface.h.

The documentation for this class was generated from the following file:

- [pvt_interface.h](#)

10.267 raw_t Struct Reference

Public Attributes

- [gtime_t](#) **time**
- [gtime_t](#) **tobs**
- [obs_t](#) **obs**
- [obs_t](#) **obuf**
- [nav_t](#) **nav**
- [sta_t](#) **sta**
- int **ephsat**
- [sbsmsg_t](#) **sbsmsg**
- char **msgtype** [256]
- unsigned char **subfrm** [MAXSAT][380]
- [lexmsg_t](#) **lexmsg**
- double **lockt** [MAXSAT][[NFREQ](#)+[NEXOBS](#)]
- double **icpp** [MAXSAT]
- double **off** [MAXSAT]
- double **icpc**
- double **prCA** [MAXSAT]
- double **dpCA** [MAXSAT]
- unsigned char **halfc** [MAXSAT][[NFREQ](#)+[NEXOBS](#)]
- char **freqn** [[MAXOBS](#)]
- int **nbyte**
- int **len**
- int **iod**

- int **tod**
- int **tbase**
- int **flag**
- int **outtype**
- unsigned char **buff** [MAXRAWLEN]
- char **opt** [256]
- double **receive_time**
- unsigned int **plen**
- unsigned int **pbyte**
- unsigned int **page**
- unsigned int **reply**
- int **week**
- unsigned char **pbuff** [255+4+2]

10.267.1 Detailed Description

Definition at line 1194 of file rtklib.h.

The documentation for this struct was generated from the following file:

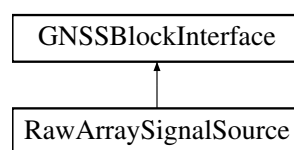
- [rtklib.h](#)

10.268 RawArraySignalSource Class Reference

This class reads samples from a GN3S USB dongle, a RF front-end signal sampler.

```
#include <raw_array_signal_source.h>
```

Inheritance diagram for RawArraySignalSource:



Public Member Functions

- **RawArraySignalSource** (const [ConfigurationInterface](#) *configuration, std::string role, unsigned int in_↔ stream, unsigned int out_stream, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "RawArraySignalSource".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.268.1 Detailed Description

This class reads samples from a GN3S USB dongle, a RF front-end signal sampler.

Definition at line 38 of file `raw_array_signal_source.h`.

10.268.2 Member Function Documentation

10.268.2.1 implementation()

```
std::string RawArraySignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "RawArraySignalSource".

Implements [GNSSBlockInterface](#).

Definition at line 55 of file `raw_array_signal_source.h`.

The documentation for this class was generated from the following file:

- [raw_array_signal_source.h](#)

10.269 Rinex_Printer Class Reference

Class that handles the generation of Receiver INdependent EXchange format (RINEX) files.

```
#include <rinex_printer.h>
```

Public Member Functions

- [Rinex_Printer](#) (int version=0, const std::string &base_path=".", const std::string &base_name="-")
Default constructor. Creates GNSS Navigation and Observables RINEX files and their headers.
- [~Rinex_Printer](#) ()
Default destructor. Closes GNSS Navigation and Observables RINEX files.
- void [rinex_nav_header](#) (std::fstream &out, const [Gps_Iono](#) &iono, const [Gps_Utc_Model](#) &utc_model, const [Gps_Ephemeris](#) &eph) const
Generates the GPS L1 C/A Navigation Data header.
- void [rinex_nav_header](#) (std::fstream &out, const [Gps_CNAV_Iono](#) &iono, const [Gps_CNAV_Utc_Model](#) &utc_model) const
Generates the GPS L2C(M) Navigation Data header.
- void [rinex_nav_header](#) (std::fstream &out, const [Galileo_Iono](#) &iono, const [Galileo_Utc_Model](#) &utc_model) const
Generates the Galileo Navigation Data header.
- void [rinex_nav_header](#) (std::fstream &out, const [Gps_Iono](#) &gps_iono, const [Gps_Utc_Model](#) &gps_utc_↵ model, const [Gps_Ephemeris](#) &eph, const [Galileo_Iono](#) &galileo_iono, const [Galileo_Utc_Model](#) &galileo_↵ utc_model) const

Generates the Mixed (GPS/Galileo) Navigation Data header.

- void [rinex_nav_header](#) (std::fstream &out, const [Gps_CNAV_Iono](#) &iono, const [Gps_CNAV_Utc_Model](#) &utc_model, const [Galileo_Iono](#) &galileo_iono, const [Galileo_Utc_Model](#) &galileo_utc_model) const

Generates the Mixed (GPS CNAV/Galileo) Navigation Data header.

- void [rinex_nav_header](#) (std::fstream &out, const [Glonass_Gnav_Utc_Model](#) &utc_model, const [Glonass_Gnav_Ephemeris](#) &glonass_gnav_eph)

Generates the GLONASS L1, L2 C/A Navigation Data header.

- void [rinex_nav_header](#) (std::fstream &out, const [Galileo_Iono](#) &galileo_iono, const [Galileo_Utc_Model](#) &galileo_utc_model, const [Glonass_Gnav_Utc_Model](#) &glonass_gnav_utc_model, const [Glonass_Gnav_Almanac](#) &glonass_gnav_almanac) const

Generates the Mixed (Galileo/GLONASS) Navigation Data header.

- void [rinex_nav_header](#) (std::fstream &out, const [Gps_Iono](#) &gps_iono, const [Gps_Utc_Model](#) &gps_utc_model, const [Gps_Ephemeris](#) &eph, const [Glonass_Gnav_Utc_Model](#) &glonass_gnav_utc_model, const [Glonass_Gnav_Almanac](#) &glonass_gnav_almanac)

Generates the Mixed (GPS L1 C/A/GLONASS L1, L2) Navigation Data header.

- void [rinex_nav_header](#) (std::fstream &out, const [Gps_CNAV_Iono](#) &gps_iono, const [Gps_CNAV_Utc_Model](#) &gps_utc_model, const [Glonass_Gnav_Utc_Model](#) &glonass_gnav_utc_model, const [Glonass_Gnav_Almanac](#) &glonass_gnav_almanac)

Generates the Mixed (GPS L2C C/A/GLONASS L1, L2) Navigation Data header.

- void [rinex_nav_header](#) (std::fstream &out, const [Beidou_Dnav_Iono](#) &iono, const [Beidou_Dnav_Utc_Model](#) &utc_model) const

Generates the BDS B1I or B3I Navigation Data header.

- void [rinex_nav_header](#) (std::fstream &out, const [Gps_Iono](#) &gps_iono, const [Gps_Utc_Model](#) &gps_utc_model, const [Gps_Ephemeris](#) &eph, const [Beidou_Dnav_Iono](#) &bds_dnav_iono, const [Beidou_Dnav_Utc_Model](#) &bds_dnav_utc_model) const

Generates the Mixed GPS L1,L5 + BDS B1I, B3I Navigation Data header.

- void [rinex_nav_header](#) (std::fstream &out, const [Gps_CNAV_Iono](#) &gps_cnav_iono, const [Gps_CNAV_Utc_Model](#) &gps_cnav_utc_model, const [Beidou_Dnav_Iono](#) &bds_dnav_iono, const [Beidou_Dnav_Utc_Model](#) &bds_dnav_utc_model)

Generates the Mixed GPS L2C + BDS B1I, B3I Navigation Data header.

- void [rinex_nav_header](#) (std::fstream &out, const [Glonass_Gnav_Utc_Model](#) &glo_gnav_utc_model, const [Beidou_Dnav_Iono](#) &bds_dnav_iono, const [Beidou_Dnav_Utc_Model](#) &bds_dnav_utc_model) const

Generates the Mixed GLONASS L1,L2 + BDS B1I, B3I Navigation Data header.

- void [rinex_nav_header](#) (std::fstream &out, const [Galileo_Iono](#) &galileo_iono, const [Galileo_Utc_Model](#) &galileo_utc_model, const [Beidou_Dnav_Iono](#) &bds_dnav_iono, const [Beidou_Dnav_Utc_Model](#) &bds_dnav_utc_model) const

Generates the Mixed (Galileo/BDS B1I, B3I) Navigation Data header.

- void [rinex_obs_header](#) (std::fstream &out, const [Gps_Ephemeris](#) &eph, double d_TOW_first_observation)

Generates the GPS Observation data header.

- void [rinex_obs_header](#) (std::fstream &out, const [Gps_CNAV_Ephemeris](#) &eph, double d_TOW_first_observation, const std::string &gps_bands="2S")

Generates the GPS L2 Observation data header.

- void [rinex_obs_header](#) (std::fstream &out, const [Gps_Ephemeris](#) &eph, const [Gps_CNAV_Ephemeris](#) &eph_cnav, double d_TOW_first_observation, const std::string &gps_bands="1C 2S")

Generates the dual frequency GPS L1 & L2/L5 Observation data header.

- void [rinex_obs_header](#) (std::fstream &out, const [Galileo_Ephemeris](#) &eph, double d_TOW_first_observation, const std::string &bands="1B")

Generates the Galileo Observation data header. Example: bands("1B"), bands("1B 5X"), bands("5X"), ... Default: "1B".

- void [rinex_obs_header](#) (std::fstream &out, const [Gps_Ephemeris](#) &gps_eph, const [Galileo_Ephemeris](#) &galileo_eph, double d_TOW_first_observation, const std::string &galileo_bands="1B")

Generates the Mixed (GPS/Galileo) Observation data header. Example: galileo_bands("1B"), galileo_bands("1B 5X"), galileo_bands("5X"), ... Default: "1B".

- void [rinex_obs_header](#) (std::fstream &out, const [Gps_Ephemeris](#) &gps_eph, const [Gps_CNAV_Ephemeris](#) &eph_cnav, const [Galileo_Ephemeris](#) &galileo_eph, double d_TOW_first_observation, const std::string &gps_bands="1C 2S", const std::string &galileo_bands="1B")
Generates the Mixed (GPS/Galileo) Observation data header. Example: galileo_bands("1B"), galileo_bands("1B 5X"), galileo_bands("5X"), ... Default: "1B".
- void [rinex_obs_header](#) (std::fstream &out, const [Gps_CNAV_Ephemeris](#) &eph_cnav, const [Galileo_Ephemeris](#) &galileo_eph, double d_TOW_first_observation, const std::string &gps_bands="2S", const std::string &galileo_bands="1B")
Generates the Mixed (GPS/Galileo) Observation data header. Example: galileo_bands("1B"), galileo_bands("1B 5X"), galileo_bands("5X"), ... Default: "1B".
- void [rinex_obs_header](#) (std::fstream &out, const [Glonass_Gnav_Ephemeris](#) &eph, double d_TOW_first_observation, const std::string &bands="1G")
Generates the GLONASS GNAV Observation data header. Example: bands("1C"), bands("1C 2C"), bands("2C"), ... Default: "1C".
- void [rinex_obs_header](#) (std::fstream &out, const [Gps_Ephemeris](#) &gps_eph, const [Glonass_Gnav_Ephemeris](#) &glonass_gnav_eph, double d_TOW_first_observation, const std::string &glonass_bands="1C")
Generates the Mixed (GPS L1 C/A /GLONASS) Observation data header. Example: galileo_bands("1C"), galileo_bands("1B 5X"), galileo_bands("5X"), ... Default: "1B".
- void [rinex_obs_header](#) (std::fstream &out, const [Galileo_Ephemeris](#) &galileo_eph, const [Glonass_Gnav_Ephemeris](#) &glonass_gnav_eph, double d_TOW_first_observation, const std::string &galileo_bands="1B", const std::string &glonass_bands="1C")
Generates the Mixed (Galileo/GLONASS) Observation data header. Example: galileo_bands("1C"), galileo_bands("1B 5X"), galileo_bands("5X"), ... Default: "1B".
- void [rinex_obs_header](#) (std::fstream &out, const [Gps_CNAV_Ephemeris](#) &gps_cnav_eph, const [Glonass_Gnav_Ephemeris](#) &glonass_gnav_eph, double d_TOW_first_observation, const std::string &glonass_bands="1G")
Generates the Mixed (GPS L2C/GLONASS) Observation data header. Example: galileo_bands("1G")... Default: "1G".
- void [rinex_obs_header](#) (std::fstream &out, const [Beidou_Dnav_Ephemeris](#) &eph, double d_TOW_first_observation, const std::string &bands)
Generates the a Beidou B1I Observation data header. Example: beidou_bands("B1")
- void [rinex_sbs_header](#) (std::fstream &out) const
Generates the SBAS raw data header.
- boost::posix_time::ptime [compute_BDS_time](#) (const [Beidou_Dnav_Ephemeris](#) &eph, double obs_time) const
Computes the BDS Time and returns a boost::posix_time::ptime object.
- boost::posix_time::ptime [compute_UTC_time](#) (const [Gps_Navigation_Message](#) &nav_msg) const
Computes the UTC time and returns a boost::posix_time::ptime object.
- boost::posix_time::ptime [compute_GPS_time](#) (const [Gps_Ephemeris](#) &eph, double obs_time) const
Computes the GPS time and returns a boost::posix_time::ptime object.
- boost::posix_time::ptime [compute_GPS_time](#) (const [Gps_CNAV_Ephemeris](#) &eph, double obs_time) const
Computes the GPS time and returns a boost::posix_time::ptime object.
- boost::posix_time::ptime [compute_Galileo_time](#) (const [Galileo_Ephemeris](#) &eph, double obs_time) const
Computes the Galileo time and returns a boost::posix_time::ptime object.
- boost::posix_time::ptime [compute_UTC_time](#) (const [Glonass_Gnav_Ephemeris](#) &eph, double obs_time) const
Computes the UTC Time and returns a boost::posix_time::ptime object.
- double [get_leap_second](#) (const [Glonass_Gnav_Ephemeris](#) &eph, double gps_obs_time) const
Computes number of leap seconds of GPS relative to UTC.
- void [log_rinex_nav](#) (std::fstream &out, const std::map< int32_t, [Gps_Ephemeris](#) > &eph_map) const
Writes data from the GPS L1 C/A navigation message into the RINEX file.
- void [log_rinex_nav](#) (std::fstream &out, const std::map< int32_t, [Gps_CNAV_Ephemeris](#) > &eph_map)
Writes data from the GPS L2 navigation message into the RINEX file.
- void [log_rinex_nav](#) (std::fstream &out, const std::map< int32_t, [Galileo_Ephemeris](#) > &eph_map) const

Writes data from the Galileo navigation message into the RINEX file.

- void `log_rinex_nav` (std::fstream &out, const std::map< int32_t, [Gps_Ephemeris](#) > &gps_eph_map, const std::map< int32_t, [Galileo_Ephemeris](#) > &galileo_eph_map)

Writes data from the Mixed (GPS/Galileo) navigation message into the RINEX file.

- void `log_rinex_nav` (std::fstream &out, const std::map< int32_t, [Gps_CNAV_Ephemeris](#) > &gps_cnav_↵
eph_map, const std::map< int32_t, [Galileo_Ephemeris](#) > &galileo_eph_map)

Writes data from the Mixed (GPS/Galileo) navigation message into the RINEX file.

- void `log_rinex_nav` (std::fstream &out, const std::map< int32_t, [Glonass_Gnav_Ephemeris](#) > &eph_map) const

Writes data from the GLONASS GNAV navigation message into the RINEX file.

- void `log_rinex_nav` (std::fstream &out, const std::map< int32_t, [Gps_Ephemeris](#) > &gps_eph_map, const std::map< int32_t, [Glonass_Gnav_Ephemeris](#) > &glonass_gnav_eph_map) const

Writes data from the Mixed (GPS/GLONASS GNAV) navigation message into the RINEX file.

- void `log_rinex_nav` (std::fstream &out, const std::map< int32_t, [Gps_CNAV_Ephemeris](#) > &gps_cnav_↵
eph_map, const std::map< int32_t, [Glonass_Gnav_Ephemeris](#) > &glonass_gnav_eph_map)

Writes data from the Mixed (GPS/GLONASS GNAV) navigation message into the RINEX file.

- void `log_rinex_nav` (std::fstream &out, const std::map< int32_t, [Galileo_Ephemeris](#) > &galileo_eph_map, const std::map< int32_t, [Glonass_Gnav_Ephemeris](#) > &glonass_gnav_eph_map)

Writes data from the Mixed (Galileo/ GLONASS GNAV) navigation message into the RINEX file.

- void `log_rinex_nav` (std::fstream &out, const std::map< int32_t, [Beidou_Dnav_Ephemeris](#) > &eph_map) const

Writes data from the Beidou B1I navigation message into the RINEX file.

- void `log_rinex_obs` (std::fstream &out, const [Gps_Ephemeris](#) &eph, double obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables) const

Writes GPS L1 observables into the RINEX file.

- void `log_rinex_obs` (std::fstream &out, const [Gps_CNAV_Ephemeris](#) &eph, double obs_time, const std_↵
::map< int32_t, [Gnss_Synchro](#) > &observables) const

Writes GPS L2 observables into the RINEX file.

- void `log_rinex_obs` (std::fstream &out, const [Gps_Ephemeris](#) &eph, const [Gps_CNAV_Ephemeris](#) &eph_↵
_cnav, double obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables, bool triple_band=false) const

Writes dual frequency GPS L1 and L2 observables into the RINEX file.

- void `log_rinex_obs` (std::fstream &out, const [Galileo_Ephemeris](#) &eph, double obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables, const std::string &galileo_bands="1B") const

*Writes Galileo observables into the RINEX file. Example: galileo_bands("1B"), galileo_bands("1B 5X"), galileo_↵
bands("5X"), ... Default: "1B".*

- void `log_rinex_obs` (std::fstream &out, const [Gps_Ephemeris](#) &gps_eph, const [Galileo_Ephemeris](#) &galileo_↵
_eph, double gps_obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables) const

Writes Mixed GPS / Galileo observables into the RINEX file.

- void `log_rinex_obs` (std::fstream &out, const [Gps_CNAV_Ephemeris](#) &eph, const [Galileo_Ephemeris](#) &galileo_eph, double gps_obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables) const

Writes Mixed GPS / Galileo observables into the RINEX file.

- void `log_rinex_obs` (std::fstream &out, const [Gps_Ephemeris](#) &gps_eph, const [Gps_CNAV_Ephemeris](#) &gps_cnav_eph, const [Galileo_Ephemeris](#) &galileo_eph, double gps_obs_time, const std::map< int32_↵
t, [Gnss_Synchro](#) > &observables, bool triple_band=false) const

Writes Mixed GPS / Galileo observables into the RINEX file.

- void `log_rinex_obs` (std::fstream &out, const [Glonass_Gnav_Ephemeris](#) &eph, double obs_time, const std_↵
::map< int32_t, [Gnss_Synchro](#) > &observables, const std::string &glonass_bands="1C") const

Writes GLONASS GNAV observables into the RINEX file. Example: glonass_bands("1C"), galileo_bands("1B 5X"), galileo_bands("5X"), ... Default: "1B".

- void `log_rinex_obs` (std::fstream &out, const [Gps_Ephemeris](#) &gps_eph, const [Glonass_Gnav_Ephemeris](#) &glonass_gnav_eph, double gps_obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables) const

Writes Mixed GPS L1 C/A - GLONASS observables into the RINEX file.

- void [log_rinex_obs](#) (std::fstream &out, const [Gps_CNAV_Ephemeris](#) &gps_eph, const [Glonass_Gnav_Ephemeris](#) &glonass_gnav_eph, double gps_obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables) const
Writes Mixed GPS L2C - GLONASS observables into the RINEX file.
- void [log_rinex_obs](#) (std::fstream &out, const [Galileo_Ephemeris](#) &galileo_eph, const [Glonass_Gnav_Ephemeris](#) &glonass_gnav_eph, double galileo_obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables) const
Writes Mixed Galileo/GLONASS observables into the RINEX file.
- void [log_rinex_obs](#) (std::fstream &out, const [Beidou_Dnav_Ephemeris](#) &eph, double obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables, const std::string &bds_bands) const
Writes BDS B1I observables into the RINEX file.
- void [to_date_time](#) (int gps_week, int gps_tow, int &year, int &month, int &day, int &hour, int &minute, int &second) const
Represents GPS time in the date time format. Leap years are considered, but leap seconds are not.
- void [update_nav_header](#) (std::fstream &out, const [Gps_Utc_Model](#) &utc_model, const [Gps_Iono](#) &gps_iono, const [Gps_Ephemeris](#) &eph) const
Writes raw SBAS messages into the RINEX file.
- void [update_nav_header](#) (std::fstream &out, const [Gps_CNAV_Utc_Model](#) &utc_model, const [Gps_CNAV_Iono](#) &iono) const
- void [update_nav_header](#) (std::fstream &out, const [Gps_Iono](#) &gps_iono, const [Gps_Utc_Model](#) &gps_utc_model, const [Gps_Ephemeris](#) &eph, const [Galileo_Iono](#) &galileo_iono, const [Galileo_Utc_Model](#) &galileo_utc_model) const
- void [update_nav_header](#) (std::fstream &out, const [Gps_CNAV_Utc_Model](#) &utc_model, const [Gps_CNAV_Iono](#) &iono, const [Galileo_Iono](#) &galileo_iono, const [Galileo_Utc_Model](#) &galileo_utc_model) const
- void [update_nav_header](#) (std::fstream &out, const [Galileo_Iono](#) &galileo_iono, const [Galileo_Utc_Model](#) &utc_model) const
- void [update_nav_header](#) (std::fstream &out, const [Glonass_Gnav_Utc_Model](#) &glonass_gnav_utc_model, const [Glonass_Gnav_Almanac](#) &glonass_gnav_almanac) const
- void [update_nav_header](#) (std::fstream &out, const [Gps_Iono](#) &gps_iono, const [Gps_Utc_Model](#) &gps_utc_model, const [Gps_Ephemeris](#) &eph, const [Glonass_Gnav_Utc_Model](#) &glonass_gnav_utc_model, const [Glonass_Gnav_Almanac](#) &glonass_gnav_almanac) const
- void [update_nav_header](#) (std::fstream &out, const [Gps_CNAV_Iono](#) &gps_iono, const [Gps_CNAV_Utc_Model](#) &gps_utc_model, const [Glonass_Gnav_Utc_Model](#) &glonass_gnav_utc_model, const [Glonass_Gnav_Almanac](#) &glonass_gnav_almanac) const
- void [update_nav_header](#) (std::fstream &out, const [Galileo_Iono](#) &galileo_iono, const [Galileo_Utc_Model](#) &galileo_utc_model, const [Glonass_Gnav_Utc_Model](#) &glonass_gnav_utc_model, const [Glonass_Gnav_Almanac](#) &glonass_gnav_almanac) const
- void [update_nav_header](#) (std::fstream &out, const [Beidou_Dnav_Utc_Model](#) &utc_model, const [Beidou_Dnav_Iono](#) &beidou_dnav_iono) const
- void [update_obs_header](#) (std::fstream &out, const [Gps_Utc_Model](#) &utc_model) const
- void [update_obs_header](#) (std::fstream &out, const [Gps_CNAV_Utc_Model](#) &utc_model) const
- void [update_obs_header](#) (std::fstream &out, const [Galileo_Utc_Model](#) &galileo_utc_model) const
- void [update_obs_header](#) (std::fstream &out, const [Glonass_Gnav_Utc_Model](#) &glonass_gnav_utc_model) const
- void [update_obs_header](#) (std::fstream &out, const [Beidou_Dnav_Utc_Model](#) &utc_model) const
- void [set_pre_2009_file](#) (bool pre_2009_file)

Public Attributes

- std::fstream [obsFile](#)
Output file stream for RINEX observation file.
- std::fstream [navFile](#)
Output file stream for RINEX navigation data file.
- std::fstream [sbsFile](#)

- *Output file stream for RINEX SBAS raw data file.*
- `std::fstream` [navGalFile](#)
- *Output file stream for RINEX Galileo navigation data file.*
- `std::fstream` [navGloFile](#)
- *Output file stream for RINEX GLONASS navigation data file.*
- `std::fstream` [navBdsFile](#)
- *Output file stream for RINEX Galileo navigation data file.*
- `std::fstream` [navMixFile](#)
- *Output file stream for RINEX Mixed navigation data file.*
- `std::string` **navfilename**
- `std::string` **obsfilename**
- `std::string` **sbsfilename**
- `std::string` **navGalfilename**
- `std::string` **navGlofilename**
- `std::string` **navBdsfilename**
- `std::string` **navMixfilename**

10.269.1 Detailed Description

Class that handles the generation of Receiver INdependent EXchange format (RINEX) files.

Definition at line 75 of file `rinex_printer.h`.

10.269.2 Constructor & Destructor Documentation

10.269.2.1 Rinex_Printer()

```
Rinex_Printer::Rinex_Printer (
    int version = 0,
    const std::string & base_path = ".",
    const std::string & base_name = "-" ) [explicit]
```

Default constructor. Creates GNSS Navigation and Observables RINEX files and their headers.

10.269.2.2 ~Rinex_Printer()

```
Rinex_Printer::~Rinex_Printer ( )
```

Default destructor. Closes GNSS Navigation and Observables RINEX files.

10.269.3 Member Function Documentation

10.269.3.1 compute_BDS_time()

```
boost::posix_time::ptime Rinex_Printer::compute_BDS_time (
    const Beidou_Dnav_Ephemeris & eph,
    double obs_time ) const
```

Computes the BDS Time and returns a `boost::posix_time::ptime` object.

Function used to convert the observation time into BDT time which is used as the default time for RINEX files

Parameters

<i>eph</i>	BeiDou DNAV Ephemeris object
<i>obs_time</i>	Observation time in BDT seconds of week

10.269.3.2 compute_Galileo_time()

```
boost::posix_time::ptime Rinex_Printer::compute_Galileo_time (
    const Galileo_Ephemeris & eph,
    double obs_time ) const
```

Computes the Galileo time and returns a boost::posix_time::ptime object.

10.269.3.3 compute_GPS_time() [1/2]

```
boost::posix_time::ptime Rinex_Printer::compute_GPS_time (
    const Gps_Ephemeris & eph,
    double obs_time ) const
```

Computes the GPS time and returns a boost::posix_time::ptime object.

10.269.3.4 compute_GPS_time() [2/2]

```
boost::posix_time::ptime Rinex_Printer::compute_GPS_time (
    const Gps_CNAV_Ephemeris & eph,
    double obs_time ) const
```

Computes the GPS time and returns a boost::posix_time::ptime object.

10.269.3.5 compute_UTC_time() [1/2]

```
boost::posix_time::ptime Rinex_Printer::compute_UTC_time (
    const Gps_Navigation_Message & nav_msg ) const
```

Computes the UTC time and returns a boost::posix_time::ptime object.

10.269.3.6 compute_UTC_time() [2/2]

```
boost::posix_time::ptime Rinex_Printer::compute_UTC_time (
    const Glonass_Gnav_Ephemeris & eph,
    double obs_time ) const
```

Computes the UTC Time and returns a boost::posix_time::ptime object.

Function used as a method to convert the observation time into UTC time which is used as the default time for RINEX files

Parameters

<i>eph</i>	GLONASS GNAV Ephemeris object
<i>obs_time</i>	Observation time in GPS seconds of week

10.269.3.7 `get_leap_second()`

```
double Rinex_Printer::get_leap_second (
    const Glonass_Gnav_Ephemeris & eph,
    double gps_obs_time ) const
```

Computes number of leap seconds of GPS relative to UTC.

Parameters

<i>eph</i>	GLONASS GNAV Ephemeris object
<i>gps_obs_time</i>	Observation time in GPS seconds of week

10.269.3.8 `log_rinex_nav()` [1/10]

```
void Rinex_Printer::log_rinex_nav (
    std::fstream & out,
    const std::map< int32_t, Gps_Ephemeris > & eph_map ) const
```

Writes data from the GPS L1 C/A navigation message into the RINEX file.

10.269.3.9 `log_rinex_nav()` [2/10]

```
void Rinex_Printer::log_rinex_nav (
    std::fstream & out,
    const std::map< int32_t, Gps_CNAV_Ephemeris > & eph_map )
```

Writes data from the GPS L2 navigation message into the RINEX file.

10.269.3.10 `log_rinex_nav()` [3/10]

```
void Rinex_Printer::log_rinex_nav (
    std::fstream & out,
    const std::map< int32_t, Galileo_Ephemeris > & eph_map ) const
```

Writes data from the Galileo navigation message into the RINEX file.

10.269.3.11 log_rinex_nav() [4/10]

```
void Rinex_Printer::log_rinex_nav (
    std::fstream & out,
    const std::map< int32_t, Gps_Ephemeris > & gps_eph_map,
    const std::map< int32_t, Galileo_Ephemeris > & galileo_eph_map )
```

Writes data from the Mixed (GPS/Galileo) navigation message into the RINEX file.

10.269.3.12 log_rinex_nav() [5/10]

```
void Rinex_Printer::log_rinex_nav (
    std::fstream & out,
    const std::map< int32_t, Gps_CNAV_Ephemeris > & gps_cnav_eph_map,
    const std::map< int32_t, Galileo_Ephemeris > & galileo_eph_map )
```

Writes data from the Mixed (GPS/Galileo) navigation message into the RINEX file.

10.269.3.13 log_rinex_nav() [6/10]

```
void Rinex_Printer::log_rinex_nav (
    std::fstream & out,
    const std::map< int32_t, Glonass_Gnav_Ephemeris > & eph_map ) const
```

Writes data from the GLONASS GNAV navigation message into the RINEX file.

10.269.3.14 log_rinex_nav() [7/10]

```
void Rinex_Printer::log_rinex_nav (
    std::fstream & out,
    const std::map< int32_t, Gps_Ephemeris > & gps_eph_map,
    const std::map< int32_t, Glonass_Gnav_Ephemeris > & glonass_gnav_eph_map ) const
```

Writes data from the Mixed (GPS/GLONASS GNAV) navigation message into the RINEX file.

10.269.3.15 log_rinex_nav() [8/10]

```
void Rinex_Printer::log_rinex_nav (
    std::fstream & out,
    const std::map< int32_t, Gps_CNAV_Ephemeris > & gps_cnav_eph_map,
    const std::map< int32_t, Glonass_Gnav_Ephemeris > & glonass_gnav_eph_map )
```

Writes data from the Mixed (GPS/GLONASS GNAV) navigation message into the RINEX file.

10.269.3.16 log_rinex_nav() [9/10]

```
void Rinex_Printer::log_rinex_nav (
    std::fstream & out,
    const std::map< int32_t, Galileo_Ephemeris > & galileo_eph_map,
    const std::map< int32_t, Glonass_Gnav_Ephemeris > & glonass_gnav_eph_map )
```

Writes data from the Mixed (Galileo/ GLONASS GNAV) navigation message into the RINEX file.

10.269.3.17 log_rinex_nav() [10/10]

```
void Rinex_Printer::log_rinex_nav (
    std::fstream & out,
    const std::map< int32_t, Beidou_Dnav_Ephemeris > & eph_map ) const
```

Writes data from the Beidou B1I navigation message into the RINEX file.

10.269.3.18 log_rinex_obs() [1/12]

```
void Rinex_Printer::log_rinex_obs (
    std::fstream & out,
    const Gps_Ephemeris & eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables ) const
```

Writes GPS L1 observables into the RINEX file.

10.269.3.19 log_rinex_obs() [2/12]

```
void Rinex_Printer::log_rinex_obs (
    std::fstream & out,
    const Gps_CNAV_Ephemeris & eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables ) const
```

Writes GPS L2 observables into the RINEX file.

10.269.3.20 log_rinex_obs() [3/12]

```
void Rinex_Printer::log_rinex_obs (
    std::fstream & out,
    const Gps_Ephemeris & eph,
    const Gps_CNAV_Ephemeris & eph_cnav,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    bool triple_band = false ) const
```

Writes dual frequency GPS L1 and L2 observables into the RINEX file.

10.269.3.21 log_rinex_obs() [4/12]

```
void Rinex_Printer::log_rinex_obs (
    std::fstream & out,
    const Galileo_Ephemeris & eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    const std::string & galileo_bands = "1B" ) const
```

Writes Galileo observables into the RINEX file. Example: galileo_bands("1B"), galileo_bands("1B 5X"), galileo_bands("5X"), ... Default: "1B".

10.269.3.22 log_rinex_obs() [5/12]

```
void Rinex_Printer::log_rinex_obs (
    std::fstream & out,
    const Gps_Ephemeris & gps_eph,
    const Galileo_Ephemeris & galileo_eph,
    double gps_obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables ) const
```

Writes Mixed GPS / Galileo observables into the RINEX file.

10.269.3.23 log_rinex_obs() [6/12]

```
void Rinex_Printer::log_rinex_obs (
    std::fstream & out,
    const Gps_CNAV_Ephemeris & eph,
    const Galileo_Ephemeris & galileo_eph,
    double gps_obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables ) const
```

Writes Mixed GPS / Galileo observables into the RINEX file.

10.269.3.24 log_rinex_obs() [7/12]

```
void Rinex_Printer::log_rinex_obs (
    std::fstream & out,
    const Gps_Ephemeris & gps_eph,
    const Gps_CNAV_Ephemeris & gps_cnav_eph,
    const Galileo_Ephemeris & galileo_eph,
    double gps_obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    bool triple_band = false ) const
```

Writes Mixed GPS / Galileo observables into the RINEX file.

10.269.3.25 log_rinex_obs() [8/12]

```
void Rinex_Printer::log_rinex_obs (
    std::fstream & out,
    const Glonass_Gnav_Ephemeris & eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    const std::string & glonass_bands = "1C" ) const
```

Writes GLONASS GNAV observables into the RINEX file. Example: glonass_bands("1C"), galileo_bands("1B 5X"), galileo_bands("5X"), ... Default: "1B".

10.269.3.26 log_rinex_obs() [9/12]

```
void Rinex_Printer::log_rinex_obs (
    std::fstream & out,
    const Gps_Ephemeris & gps_eph,
    const Glonass_Gnav_Ephemeris & glonass_gnav_eph,
    double gps_obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables ) const
```

Writes Mixed GPS L1 C/A - GLONASS observables into the RINEX file.

10.269.3.27 log_rinex_obs() [10/12]

```
void Rinex_Printer::log_rinex_obs (
    std::fstream & out,
    const Gps_CNAV_Ephemeris & gps_eph,
    const Glonass_Gnav_Ephemeris & glonass_gnav_eph,
    double gps_obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables ) const
```

Writes Mixed GPS L2C - GLONASS observables into the RINEX file.

10.269.3.28 log_rinex_obs() [11/12]

```
void Rinex_Printer::log_rinex_obs (
    std::fstream & out,
    const Galileo_Ephemeris & galileo_eph,
    const Glonass_Gnav_Ephemeris & glonass_gnav_eph,
    double galileo_obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables ) const
```

Writes Mixed Galileo/GLONASS observables into the RINEX file.

10.269.3.29 log_rinex_obs() [12/12]

```
void Rinex_Printer::log_rinex_obs (
    std::fstream & out,
    const Beidou_Dnav_Ephemeris & eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    const std::string & bds_bands ) const
```

Writes BDS B1I observables into the RINEX file.

10.269.3.30 rinex_nav_header() [1/14]

```
void Rinex_Printer::rinex_nav_header (
    std::fstream & out,
    const Gps_Iono & iono,
    const Gps_Utc_Model & utc_model,
    const Gps_Ephemeris & eph ) const
```

Generates the GPS L1 C/A Navigation Data header.

10.269.3.31 rinex_nav_header() [2/14]

```
void Rinex_Printer::rinex_nav_header (
    std::fstream & out,
    const Gps_CNAV_Iono & iono,
    const Gps_CNAV_Utc_Model & utc_model ) const
```

Generates the GPS L2C(M) Navigation Data header.

10.269.3.32 rinex_nav_header() [3/14]

```
void Rinex_Printer::rinex_nav_header (
    std::fstream & out,
    const Galileo_Iono & iono,
    const Galileo_Utc_Model & utc_model ) const
```

Generates the Galileo Navigation Data header.

10.269.3.33 rinex_nav_header() [4/14]

```
void Rinex_Printer::rinex_nav_header (
    std::fstream & out,
    const Gps_Iono & gps_iono,
    const Gps_Utc_Model & gps_utc_model,
    const Gps_Ephemeris & eph,
    const Galileo_Iono & galileo_iono,
    const Galileo_Utc_Model & galileo_utc_model ) const
```

Generates the Mixed (GPS/Galileo) Navigation Data header.

10.269.3.34 rinex_nav_header() [5/14]

```
void Rinex_Printer::rinex_nav_header (
    std::fstream & out,
    const Gps_CNAV_Iono & iono,
    const Gps_CNAV_Utc_Model & utc_model,
    const Galileo_Iono & galileo_iono,
    const Galileo_Utc_Model & galileo_utc_model ) const
```

Generates the Mixed (GPS CNAV/Galileo) Navigation Data header.

10.269.3.35 rinex_nav_header() [6/14]

```
void Rinex_Printer::rinex_nav_header (
    std::fstream & out,
    const Glonass_Gnav_Utc_Model & utc_model,
    const Glonass_Gnav_Ephemeris & glonass_gnav_eph )
```

Generates the GLONASS L1, L2 C/A Navigation Data header.

10.269.3.36 rinex_nav_header() [7/14]

```
void Rinex_Printer::rinex_nav_header (
    std::fstream & out,
    const Galileo_Iono & galileo_iono,
    const Galileo_Utc_Model & galileo_utc_model,
    const Glonass_Gnav_Utc_Model & glonass_gnav_utc_model,
    const Glonass_Gnav_Almanac & glonass_gnav_almanac ) const
```

Generates the Mixed (Galileo/GLONASS) Navigation Data header.

10.269.3.37 rinex_nav_header() [8/14]

```
void Rinex_Printer::rinex_nav_header (
    std::fstream & out,
    const Gps_Iono & gps_iono,
    const Gps_Utc_Model & gps_utc_model,
    const Gps_Ephemeris & eph,
    const Glonass_Gnav_Utc_Model & glonass_gnav_utc_model,
    const Glonass_Gnav_Almanac & glonass_gnav_almanac )
```

Generates the Mixed (GPS L1 C/A/GLONASS L1, L2) Navigation Data header.

10.269.3.38 rinex_nav_header() [9/14]

```
void Rinex_Printer::rinex_nav_header (
    std::fstream & out,
    const Gps_CNAV_Iono & gps_iono,
    const Gps_CNAV_Utc_Model & gps_utc_model,
    const Glonass_Gnav_Utc_Model & glonass_gnav_utc_model,
    const Glonass_Gnav_Almanac & glonass_gnav_almanac )
```

Generates the Mixed (GPS L2C C/A/GLONASS L1, L2) Navigation Data header.

10.269.3.39 rinex_nav_header() [10/14]

```
void Rinex_Printer::rinex_nav_header (
    std::fstream & out,
    const Beidou_Dnav_Iono & iono,
    const Beidou_Dnav_Utc_Model & utc_model ) const
```

Generates the BDS B1I or B3I Navigation Data header.

10.269.3.40 rinex_nav_header() [11/14]

```
void Rinex_Printer::rinex_nav_header (
    std::fstream & out,
    const Gps_Iono & gps_iono,
    const Gps_Utc_Model & gps_utc_model,
    const Gps_Ephemeris & eph,
    const Beidou_Dnav_Iono & bds_dnav_iono,
    const Beidou_Dnav_Utc_Model & bds_dnav_utc_model ) const
```

Generates the Mixed GPS L1,L5 + BDS B1I, B3I Navigation Data header.

10.269.3.41 rinex_nav_header() [12/14]

```
void Rinex_Printer::rinex_nav_header (
    std::fstream & out,
    const Gps_CNAV_Iono & gps_cnav_iono,
    const Gps_CNAV_Utc_Model & gps_cnav_utc_model,
    const Beidou_Dnav_Iono & bds_dnav_iono,
    const Beidou_Dnav_Utc_Model & bds_dnav_utc_model )
```

Generates the Mixed GPS L2C + BDS B1I, B3I Navigation Data header.

10.269.3.42 rinex_nav_header() [13/14]

```
void Rinex_Printer::rinex_nav_header (
    std::fstream & out,
    const Glonass_Gnav_Utc_Model & glo_gnav_utc_model,
    const Beidou_Dnav_Iono & bds_dnav_iono,
    const Beidou_Dnav_Utc_Model & bds_dnav_utc_model ) const
```

Generates the Mixed GLONASS L1,L2 + BDS B1I, B3I Navigation Data header.

10.269.3.43 rinex_nav_header() [14/14]

```
void Rinex_Printer::rinex_nav_header (
    std::fstream & out,
    const Galileo_Iono & galileo_iono,
    const Galileo_Utc_Model & galileo_utc_model,
    const Beidou_Dnav_Iono & bds_dnav_iono,
    const Beidou_Dnav_Utc_Model & bds_dnav_utc_model ) const
```

Generates the Mixed (Galileo/BDS B1I, B3I) Navigation Data header.

10.269.3.44 rinex_obs_header() [1/12]

```
void Rinex_Printer::rinex_obs_header (
    std::fstream & out,
    const Gps_Ephemeris & eph,
    double d_TOW_first_observation )
```

Generates the GPS Observation data header.

10.269.3.45 rinex_obs_header() [2/12]

```
void Rinex_Printer::rinex_obs_header (
    std::fstream & out,
    const Gps_CNAV_Ephemeris & eph,
    double d_TOW_first_observation,
    const std::string & gps_bands = "2S" )
```

Generates the GPS L2 Observation data header.

10.269.3.46 rinex_obs_header() [3/12]

```
void Rinex_Printer::rinex_obs_header (
    std::fstream & out,
    const Gps_Ephemeris & eph,
    const Gps_CNAV_Ephemeris & eph_cnav,
    double d_TOW_first_observation,
    const std::string & gps_bands = "1C 2S" )
```

Generates the dual frequency GPS L1 & L2/L5 Observation data header.

10.269.3.47 rinex_obs_header() [4/12]

```
void Rinex_Printer::rinex_obs_header (
    std::fstream & out,
    const Galileo_Ephemeris & eph,
    double d_TOW_first_observation,
    const std::string & bands = "1B" )
```

Generates the Galileo Observation data header. Example: bands("1B"), bands("1B 5X"), bands("5X"), ... Default: "1B".

10.269.3.48 rinex_obs_header() [5/12]

```
void Rinex_Printer::rinex_obs_header (
    std::fstream & out,
    const Gps_Ephemeris & gps_eph,
    const Galileo_Ephemeris & galileo_eph,
    double d_TOW_first_observation,
    const std::string & galileo_bands = "1B" )
```

Generates the Mixed (GPS/Galileo) Observation data header. Example: galileo_bands("1B"), galileo_bands("1B 5X"), galileo_bands("5X"), ... Default: "1B".

10.269.3.49 rinex_obs_header() [6/12]

```
void Rinex_Printer::rinex_obs_header (
    std::fstream & out,
    const Gps_Ephemeris & gps_eph,
    const Gps_CNAV_Ephemeris & eph_cnav,
    const Galileo_Ephemeris & galileo_eph,
    double d_TOW_first_observation,
    const std::string & gps_bands = "1C 2S",
    const std::string & galileo_bands = "1B" )
```

Generates the Mixed (GPS/Galileo) Observation data header. Example: galileo_bands("1B"), galileo_bands("1B 5X"), galileo_bands("5X"), ... Default: "1B".

10.269.3.50 rinex_obs_header() [7/12]

```
void Rinex_Printer::rinex_obs_header (
    std::fstream & out,
    const Gps_CNAV_Ephemeris & eph_cnav,
    const Galileo_Ephemeris & galileo_eph,
    double d_TOW_first_observation,
    const std::string & gps_bands = "2S",
    const std::string & galileo_bands = "1B" )
```

Generates the Mixed (GPS/Galileo) Observation data header. Example: galileo_bands("1B"), galileo_bands("1B 5X"), galileo_bands("5X"), ... Default: "1B".

10.269.3.51 rinex_obs_header() [8/12]

```
void Rinex_Printer::rinex_obs_header (
    std::fstream & out,
    const Glonass_Gnav_Ephemeris & eph,
    double d_TOW_first_observation,
    const std::string & bands = "1G" )
```

Generates the GLONASS GNAV Observation data header. Example: bands("1C"), bands("1C 2C"), bands("2C"), ... Default: "1C".

10.269.3.52 rinex_obs_header() [9/12]

```
void Rinex_Printer::rinex_obs_header (
    std::fstream & out,
    const Gps_Ephemeris & gps_eph,
    const Glonass_Gnav_Ephemeris & glonass_gnav_eph,
    double d_TOW_first_observation,
    const std::string & glonass_bands = "1C" )
```

Generates the Mixed (GPS L1 C/A /GLONASS) Observation data header. Example: `galileo_bands("1C")`, `galileo_bands("1B 5X")`, `galileo_bands("5X")`, ... Default: "1B".

10.269.3.53 rinex_obs_header() [10/12]

```
void Rinex_Printer::rinex_obs_header (
    std::fstream & out,
    const Galileo_Ephemeris & galileo_eph,
    const Glonass_Gnav_Ephemeris & glonass_gnav_eph,
    double d_TOW_first_observation,
    const std::string & galileo_bands = "1B",
    const std::string & glonass_bands = "1C" )
```

Generates the Mixed (Galileo/GLONASS) Observation data header. Example: `galileo_bands("1C")`, `galileo_bands("1B 5X")`, `galileo_bands("5X")`, ... Default: "1B".

10.269.3.54 rinex_obs_header() [11/12]

```
void Rinex_Printer::rinex_obs_header (
    std::fstream & out,
    const Gps_CNAV_Ephemeris & gps_cnav_eph,
    const Glonass_Gnav_Ephemeris & glonass_gnav_eph,
    double d_TOW_first_observation,
    const std::string & glonass_bands = "1G" )
```

Generates the Mixed (GPS L2C/GLONASS) Observation data header. Example: `galileo_bands("1G")`... Default: "1G".

10.269.3.55 rinex_obs_header() [12/12]

```
void Rinex_Printer::rinex_obs_header (
    std::fstream & out,
    const Beidou_Dnav_Ephemeris & eph,
    double d_TOW_first_observation,
    const std::string & bands )
```

Generates the a Beidou B1I Observation data header. Example: `beidou_bands("B1")`

10.269.3.56 rinex_sbs_header()

```
void Rinex_Printer::rinex_sbs_header (
    std::fstream & out ) const
```

Generates the SBAS raw data header.

10.269.3.57 to_date_time()

```
void Rinex_Printer::to_date_time (
    int  gps_week,
    int  gps_tow,
    int & year,
    int & month,
    int & day,
    int & hour,
    int & minute,
    int & second ) const
```

Represents GPS time in the date time format. Leap years are considered, but leap seconds are not.

10.269.3.58 update_nav_header()

```
void Rinex_Printer::update_nav_header (
    std::fstream & out,
    const Gps_Utc_Model & utc_model,
    const Gps_Iono & gps_iono,
    const Gps_Ephemeris & eph ) const
```

Writes raw SBAS messages into the RINEX file.

10.269.4 Member Data Documentation

10.269.4.1 navBdsFile

```
std::fstream Rinex_Printer::navBdsFile
```

Output file stream for RINEX Galileo navigation data file.

Definition at line 426 of file rinex_printer.h.

10.269.4.2 navFile

```
std::fstream Rinex_Printer::navFile
```

Output file stream for RINEX navigation data file.

Definition at line 422 of file rinex_printer.h.

10.269.4.3 navGalFile

```
std::fstream Rinex_Printer::navGalFile
```

Output file stream for RINEX Galileo navigation data file.

Definition at line 424 of file rinex_printer.h.

10.269.4.4 navGloFile

```
std::fstream Rinex_Printer::navGloFile
```

Output file stream for RINEX GLONASS navigation data file.

Definition at line 425 of file rinex_printer.h.

10.269.4.5 navMixFile

```
std::fstream Rinex_Printer::navMixFile
```

Output file stream for RINEX Mixed navigation data file.

Definition at line 427 of file rinex_printer.h.

10.269.4.6 obsFile

```
std::fstream Rinex_Printer::obsFile
```

Output file stream for RINEX observation file.

Definition at line 421 of file rinex_printer.h.

10.269.4.7 sbsFile

```
std::fstream Rinex_Printer::sbsFile
```

Output file stream for RINEX SBAS raw data file.

Definition at line 423 of file rinex_printer.h.

The documentation for this class was generated from the following file:

- [rinex_printer.h](#)

10.270 RtcM Class Reference

This class implements the generation and reading of some Message Types defined in the RTCM 3.2 Standard, plus some utilities to handle messages.

```
#include <rtcm.h>
```

Public Member Functions

- [RtcM](#) (uint16_t port=2101)
Default constructor that sets TCP port of the RTCM message server and RTCM Station ID. 2101 is the standard RTCM port according to the Internet Assigned Numbers Authority (IANA). See <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml>.
- std::string [print_MT1001](#) (const [Gps_Ephemeris](#) &gps_eph, double obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables, uint16_t station_id)
Prints message type 1001 (L1-Only GPS RTK Observables)
- std::string [print_MT1002](#) (const [Gps_Ephemeris](#) &gps_eph, double obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables, uint16_t station_id)
Prints message type 1002 (Extended L1-Only GPS RTK Observables)
- std::string [print_MT1003](#) (const [Gps_Ephemeris](#) &ephL1, const [Gps_CNAV_Ephemeris](#) &ephL2, double obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables, uint16_t station_id)
Prints message type 1003 (L1 & L2 GPS RTK Observables)
- std::string [print_MT1004](#) (const [Gps_Ephemeris](#) &ephL1, const [Gps_CNAV_Ephemeris](#) &ephL2, double obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables, uint16_t station_id)
Prints message type 1004 (Extended L1 & L2 GPS RTK Observables)
- std::string [print_MT1005](#) (uint32_t ref_id, double ecef_x, double ecef_y, double ecef_z, bool gps, bool glonass, bool galileo, bool non_physical, bool single_oscillator, uint32_t quarter_cycle_indicator)
Prints message type 1005 (Stationary Antenna Reference Point)
- int32_t [read_MT1005](#) (const std::string &message, uint32_t &ref_id, double &ecef_x, double &ecef_y, double &ecef_z, bool &gps, bool &glonass, bool &galileo)
Verifies and reads messages of type 1005 (Stationary Antenna Reference Point). Returns 1 if anything goes wrong, 0 otherwise.
- std::string [print_MT1006](#) (uint32_t ref_id, double ecef_x, double ecef_y, double ecef_z, bool gps, bool glonass, bool galileo, bool non_physical, bool single_oscillator, uint32_t quarter_cycle_indicator, double height)
Prints message type 1006 (Stationary Antenna Reference Point, with Height Information)
- std::string [print_MT1005_test](#) ()
For testing purposes.

- `std::string print_MT1008` (`uint32_t ref_id`, `const std::string &antenna_descriptor`, `uint32_t antenna_setup_id`, `const std::string &antenna_serial_number`)
Prints message type 1008 (Antenna Descriptor & Serial Number)
- `std::string print_MT1009` (`const Glonass_Gnav_Ephemeris &glonass_gnav_eph`, `double obs_time`, `const std::map< int32_t, Gnss_Synchro > &observables`, `uint16_t station_id`)
Prints L1-Only GLONASS RTK Observables.
- `std::string print_MT1010` (`const Glonass_Gnav_Ephemeris &glonass_gnav_eph`, `double obs_time`, `const std::map< int32_t, Gnss_Synchro > &observables`, `uint16_t station_id`)
Prints Extended L1-Only GLONASS RTK Observables.
- `std::string print_MT1011` (`const Glonass_Gnav_Ephemeris &glonass_gnav_ephL1`, `const Glonass_Gnav_Ephemeris &glonass_gnav_ephL2`, `double obs_time`, `const std::map< int32_t, Gnss_Synchro > &observables`, `uint16_t station_id`)
Prints L1&L2 GLONASS RTK Observables.
- `std::string print_MT1012` (`const Glonass_Gnav_Ephemeris &glonass_gnav_ephL1`, `const Glonass_Gnav_Ephemeris &glonass_gnav_ephL2`, `double obs_time`, `const std::map< int32_t, Gnss_Synchro > &observables`, `uint16_t station_id`)
Prints Extended L1&L2 GLONASS RTK Observables.
- `std::string print_MT1019` (`const Gps_Ephemeris &gps_eph`)
Prints message type 1019 (GPS Ephemeris), should be broadcast in the event that the IODC does not match the IODE, and every 2 minutes.
- `int32_t read_MT1019` (`const std::string &message`, `Gps_Ephemeris &gps_eph`)
Verifies and reads messages of type 1019 (GPS Ephemeris). Returns 1 if anything goes wrong, 0 otherwise.
- `std::string print_MT1020` (`const Glonass_Gnav_Ephemeris &glonass_gnav_eph`, `const Glonass_Gnav_Utc_Model &glonass_gnav_utc_model`)
Prints message type 1020 (GLONASS Ephemeris).
- `int32_t read_MT1020` (`const std::string &message`, `Glonass_Gnav_Ephemeris &glonass_gnav_eph`, `Glonass_Gnav_Utc_Model &glonass_gnav_utc_model`)
Verifies and reads messages of type 1020 (GLONASS Ephemeris).
- `std::string print_MT1029` (`uint32_t ref_id`, `const Gps_Ephemeris &gps_eph`, `double obs_time`, `const std::string &message`)
Prints message type 1029 (Unicode Text String)
- `std::string print_MT1045` (`const Galileo_Ephemeris &gal_eph`)
Prints message type 1045 (Galileo Ephemeris), should be broadcast every 2 minutes.
- `int32_t read_MT1045` (`const std::string &message`, `Galileo_Ephemeris &gal_eph`)
Verifies and reads messages of type 1045 (Galileo Ephemeris). Returns 1 if anything goes wrong, 0 otherwise.
- `std::string print_MSM_1` (`const Gps_Ephemeris &gps_eph`, `const Gps_CNAV_Ephemeris &gps_cnav_eph`, `const Galileo_Ephemeris &gal_eph`, `const Glonass_Gnav_Ephemeris &glonass_gnav_eph`, `double obs_time`, `const std::map< int32_t, Gnss_Synchro > &observables`, `uint32_t ref_id`, `uint32_t clock_steering_indicator`, `uint32_t external_clock_indicator`, `int32_t smooth_int`, `bool divergence_free`, `bool more_messages`)
Prints messages of type MSM1 (Compact GNSS observables)
- `std::string print_MSM_2` (`const Gps_Ephemeris &gps_eph`, `const Gps_CNAV_Ephemeris &gps_cnav_eph`, `const Galileo_Ephemeris &gal_eph`, `const Glonass_Gnav_Ephemeris &glonass_gnav_eph`, `double obs_time`, `const std::map< int32_t, Gnss_Synchro > &observables`, `uint32_t ref_id`, `uint32_t clock_steering_indicator`, `uint32_t external_clock_indicator`, `int32_t smooth_int`, `bool divergence_free`, `bool more_messages`)
Prints messages of type MSM2 (Compact GNSS phaseranges)
- `std::string print_MSM_3` (`const Gps_Ephemeris &gps_eph`, `const Gps_CNAV_Ephemeris &gps_cnav_eph`, `const Galileo_Ephemeris &gal_eph`, `const Glonass_Gnav_Ephemeris &glonass_gnav_eph`, `double obs_time`, `const std::map< int32_t, Gnss_Synchro > &observables`, `uint32_t ref_id`, `uint32_t clock_steering_indicator`, `uint32_t external_clock_indicator`, `int32_t smooth_int`, `bool divergence_free`, `bool more_messages`)
Prints messages of type MSM3 (Compact GNSS pseudoranges and phaseranges)
- `std::string print_MSM_4` (`const Gps_Ephemeris &gps_eph`, `const Gps_CNAV_Ephemeris &gps_cnav_eph`, `const Galileo_Ephemeris &gal_eph`, `const Glonass_Gnav_Ephemeris &glonass_gnav_eph`, `double obs_time`, `const std::map< int32_t, Gnss_Synchro > &observables`, `uint32_t ref_id`, `uint32_t clock_steering_indicator`, `uint32_t external_clock_indicator`, `int32_t smooth_int`, `bool divergence_free`, `bool more_messages`)

Prints messages of type MSM4 (Full GNSS pseudoranges and phaseranges plus CNR)

- `std::string print_MSM_5` (const `Gps_Ephemeris` &gps_eph, const `Gps_CNAV_Ephemeris` &gps_cnav_eph, const `Galileo_Ephemeris` &gal_eph, const `Glonass_Gnav_Ephemeris` &glo_gnav_eph, double obs_time, const std::map< int32_t, `Gnss_Synchro` > &observables, uint32_t ref_id, uint32_t clock_steering_indicator, uint32_t external_clock_indicator, int32_t smooth_int, bool divergence_free, bool more_messages)

Prints messages of type MSM5 (Full GNSS pseudoranges, phaseranges, phaserange rate and CNR)

- `std::string print_MSM_6` (const `Gps_Ephemeris` &gps_eph, const `Gps_CNAV_Ephemeris` &gps_cnav_eph, const `Galileo_Ephemeris` &gal_eph, const `Glonass_Gnav_Ephemeris` &glo_gnav_eph, double obs_time, const std::map< int32_t, `Gnss_Synchro` > &observables, uint32_t ref_id, uint32_t clock_steering_indicator, uint32_t external_clock_indicator, int32_t smooth_int, bool divergence_free, bool more_messages)

Prints messages of type MSM6 (Full GNSS pseudoranges and phaseranges plus CNR, high resolution)

- `std::string print_MSM_7` (const `Gps_Ephemeris` &gps_eph, const `Gps_CNAV_Ephemeris` &gps_cnav_eph, const `Galileo_Ephemeris` &gal_eph, const `Glonass_Gnav_Ephemeris` &glo_gnav_eph, double obs_time, const std::map< int32_t, `Gnss_Synchro` > &observables, uint32_t ref_id, uint32_t clock_steering_indicator, uint32_t external_clock_indicator, int32_t smooth_int, bool divergence_free, bool more_messages)

Prints messages of type MSM7 (Full GNSS pseudoranges, phaseranges, phaserange rate and CNR, high resolution)

- `uint32_t lock_time` (const `Gps_Ephemeris` &eph, double obs_time, const `Gnss_Synchro` &gnss_synchro)

Returns the time period in which GPS L1 signals have been continually tracked.

- `uint32_t lock_time` (const `Gps_CNAV_Ephemeris` &eph, double obs_time, const `Gnss_Synchro` &gnss_synchro)

Returns the time period in which GPS L2 signals have been continually tracked.

- `uint32_t lock_time` (const `Galileo_Ephemeris` &eph, double obs_time, const `Gnss_Synchro` &gnss_synchro)

Returns the time period in which Galileo signals have been continually tracked.

- `uint32_t lock_time` (const `Glonass_Gnav_Ephemeris` &eph, double obs_time, const `Gnss_Synchro` &gnss_synchro)

Locks time period in which GLONASS signals have been continually tracked.

- `std::string bin_to_hex` (const std::string &s) const

Returns a string of hexadecimal symbols from a string of binary symbols.

- `std::string hex_to_bin` (const std::string &s) const

Returns a string of binary symbols from a string of hexadecimal symbols.

- `std::string bin_to_binary_data` (const std::string &s) const

Returns a string of binary data from a string of binary symbols.

- `std::string binary_data_to_bin` (const std::string &s) const

Returns a string of binary symbols from a string of binary data.

- `uint32_t bin_to_uint` (const std::string &s) const

Returns an uint32_t from a string of binary symbols.

- `int32_t bin_to_int` (const std::string &s) const
- `double bin_to_double` (const std::string &s) const

Returns double from a string of binary symbols.

- `int32_t bin_to_sint` (const std::string &s) const
- `uint64_t hex_to_uint` (const std::string &s) const

Returns an uint64_t from a string of hexadecimal symbols.

- `int64_t hex_to_int` (const std::string &s) const

Returns an int64_t from a string of hexadecimal symbols.

- `bool check_CRC` (const std::string &message) const

Checks that the CRC of a RTCM package is correct.

- `void run_server` ()

Starts running the server.

- `void stop_server` ()

Stops the server.

- `void send_message` (const std::string &msg)

Sends a message through the server to all connected clients.

- `bool is_server_running` () const

Returns true if the server is running, false otherwise.

10.270.1 Detailed Description

This class implements the generation and reading of some Message Types defined in the RTCM 3.2 Standard, plus some utilities to handle messages.

Generation of the following Message Types: 1001, 1002, 1003, 1004, 1005, 1006, 1008, 1019, 1020, 1029, 1045

Decoding of the following Message Types: 1019, 1045

Generation of the following Multiple Signal Messages: MSM1 (message types 1071, 1091) MSM2 (message types 1072, 1092) MSM3 (message types 1073, 1093) MSM4 (message types 1074, 1094) MSM5 (message types 1075, 1095) MSM6 (message types 1076, 1096) MSM7 (message types 1077, 1097)

RTCM 3 message format (size in bits):

length	data message	parity	preamble	000000
8	24	6	10	

(C) Carles Fernandez-Prades, 2015. cfernandez(at)cttc.es

Definition at line 87 of file rtcM.h.

10.270.2 Constructor & Destructor Documentation

10.270.2.1 RtcM()

```
RtcM::RtcM (
    uint16_t port = 2101 ) [explicit]
```

Default constructor that sets TCP port of the RTCM message server and RTCM Station ID. 2101 is the standard RTCM port according to the Internet Assigned Numbers Authority (IANA). See <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml>.

10.270.3 Member Function Documentation

10.270.3.1 bin_to_binary_data()

```
std::string RtcM::bin_to_binary_data (
    const std::string & s ) const
```

Returns a string of binary data from a string of binary symbols.

10.270.3.2 bin_to_double()

```
double RtcM::bin_to_double (
    const std::string & s ) const
```

Returns double from a string of binary symbols.

10.270.3.3 bin_to_hex()

```
std::string RtcM::bin_to_hex (
    const std::string & s ) const
```

Returns a string of hexadecimal symbols from a string of binary symbols.

10.270.3.4 bin_to_uint()

```
uint32_t RtcM::bin_to_uint (
    const std::string & s ) const
```

Returns an uint32_t from a string of binary symbols.

10.270.3.5 binary_data_to_bin()

```
std::string RtcM::binary_data_to_bin (
    const std::string & s ) const
```

Returns a string of binary symbols from a string of binary data.

10.270.3.6 check_CRC()

```
bool RtcM::check_CRC (
    const std::string & message ) const
```

Checks that the CRC of a RTCM package is correct.

10.270.3.7 hex_to_bin()

```
std::string RtcM::hex_to_bin (
    const std::string & s ) const
```

Returns a string of binary symbols from a string of hexadecimal symbols.

10.270.3.8 hex_to_int()

```
int64_t RtcM::hex_to_int (
    const std::string & s ) const
```

Returns a `int64_t` from a string of hexadecimal symbols.

10.270.3.9 hex_to_uint()

```
uint64_t RtcM::hex_to_uint (
    const std::string & s ) const
```

Returns an `uint64_t` from a string of hexadecimal symbols.

10.270.3.10 is_server_running()

```
bool RtcM::is_server_running ( ) const
```

Returns true if the server is running, false otherwise.

10.270.3.11 lock_time() [1/4]

```
uint32_t RtcM::lock_time (
    const Gps_Ephemeris & eph,
    double obs_time,
    const Gnss_Synchro & gnss_synchro )
```

Returns the time period in which GPS L1 signals have been continually tracked.

10.270.3.12 lock_time() [2/4]

```
uint32_t RtcM::lock_time (
    const Gps_CNAV_Ephemeris & eph,
    double obs_time,
    const Gnss_Synchro & gnss_synchro )
```

Returns the time period in which GPS L2 signals have been continually tracked.

10.270.3.13 lock_time() [3/4]

```
uint32_t RtcM::lock_time (
    const Galileo_Ephemeris & eph,
    double obs_time,
    const Gnss_Synchro & gnss_synchro )
```

Returns the time period in which Galileo signals have been continually tracked.

10.270.3.14 lock_time() [4/4]

```
uint32_t RtcM::lock_time (
    const Glonass_Gnav_Ephemeris & eph,
    double obs_time,
    const Gnss_Synchro & gnss_synchro )
```

Locks time period in which GLONASS signals have been continually tracked.

Note

Code added as part of GSoC 2017 program

Parameters

<i>eph</i>	GLONASS GNAV Broadcast Ephemeris
<i>obs_time</i>	Time of observation at the moment of printing
<i>observables</i>	Set of observables as defined by the platform

Returns

Returns the time period in which GLONASS signals have been continually tracked.

10.270.3.15 print_MSM_1()

```
std::string RtcM::print_MSM_1 (
    const Gps_Ephemeris & gps_eph,
    const Gps_CNAV_Ephemeris & gps_cnav_eph,
    const Galileo_Ephemeris & gal_eph,
    const Glonass_Gnav_Ephemeris & glo_gnav_eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint32_t ref_id,
    uint32_t clock_steering_indicator,
    uint32_t external_clock_indicator,
    int32_t smooth_int,
    bool divergence_free,
    bool more_messages )
```

Prints messages of type MSM1 (Compact GNSS observables)

10.270.3.16 print_MSM_2()

```
std::string RtcM::print_MSM_2 (
    const Gps_Ephemeris & gps_eph,
    const Gps_CNAV_Ephemeris & gps_cnav_eph,
    const Galileo_Ephemeris & gal_eph,
    const Glonass_Gnav_Ephemeris & glo_gnav_eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint32_t ref_id,
    uint32_t clock_steering_indicator,
    uint32_t external_clock_indicator,
    int32_t smooth_int,
    bool divergence_free,
    bool more_messages )
```

Prints messages of type MSM2 (Compact GNSS phaseranges)

10.270.3.17 print_MSM_3()

```
std::string RtcM::print_MSM_3 (
    const Gps_Ephemeris & gps_eph,
    const Gps_CNAV_Ephemeris & gps_cnav_eph,
    const Galileo_Ephemeris & gal_eph,
    const Glonass_Gnav_Ephemeris & glo_gnav_eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint32_t ref_id,
    uint32_t clock_steering_indicator,
    uint32_t external_clock_indicator,
    int32_t smooth_int,
    bool divergence_free,
    bool more_messages )
```

Prints messages of type MSM3 (Compact GNSS pseudoranges and phaseranges)

10.270.3.18 print_MSM_4()

```
std::string RtcM::print_MSM_4 (
    const Gps_Ephemeris & gps_eph,
    const Gps_CNAV_Ephemeris & gps_cnav_eph,
    const Galileo_Ephemeris & gal_eph,
    const Glonass_Gnav_Ephemeris & glo_gnav_eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint32_t ref_id,
    uint32_t clock_steering_indicator,
    uint32_t external_clock_indicator,
    int32_t smooth_int,
    bool divergence_free,
    bool more_messages )
```

Prints messages of type MSM4 (Full GNSS pseudoranges and phaseranges plus CNR)

10.270.3.19 print_MSM_5()

```
std::string RtcM::print_MSM_5 (
    const Gps_Ephemeris & gps_eph,
    const Gps_CNAV_Ephemeris & gps_cnav_eph,
    const Galileo_Ephemeris & gal_eph,
    const Glonass_Gnav_Ephemeris & glo_gnav_eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint32_t ref_id,
    uint32_t clock_steering_indicator,
    uint32_t external_clock_indicator,
    int32_t smooth_int,
    bool divergence_free,
    bool more_messages )
```

Prints messages of type MSM5 (Full GNSS pseudoranges, phaseranges, phaserange rate and CNR)

10.270.3.20 print_MSM_6()

```
std::string RtcM::print_MSM_6 (
    const Gps_Ephemeris & gps_eph,
    const Gps_CNAV_Ephemeris & gps_cnav_eph,
    const Galileo_Ephemeris & gal_eph,
    const Glonass_Gnav_Ephemeris & glo_gnav_eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint32_t ref_id,
    uint32_t clock_steering_indicator,
    uint32_t external_clock_indicator,
    int32_t smooth_int,
    bool divergence_free,
    bool more_messages )
```

Prints messages of type MSM6 (Full GNSS pseudoranges and phaseranges plus CNR, high resolution)

10.270.3.21 print_MSM_7()

```
std::string RtcM::print_MSM_7 (
    const Gps_Ephemeris & gps_eph,
    const Gps_CNAV_Ephemeris & gps_cnav_eph,
    const Galileo_Ephemeris & gal_eph,
    const Glonass_Gnav_Ephemeris & glo_gnav_eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint32_t ref_id,
    uint32_t clock_steering_indicator,
    uint32_t external_clock_indicator,
    int32_t smooth_int,
    bool divergence_free,
    bool more_messages )
```

Prints messages of type MSM7 (Full GNSS pseudoranges, phaseranges, phaserange rate and CNR, high resolution)

10.270.3.22 print_MT1001()

```
std::string RtcM::print_MT1001 (
    const Gps_Ephemeris & gps_eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint16_t station_id )
```

Prints message type 1001 (L1-Only GPS RTK Observables)

10.270.3.23 print_MT1002()

```
std::string RtcM::print_MT1002 (
    const Gps_Ephemeris & gps_eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint16_t station_id )
```

Prints message type 1002 (Extended L1-Only GPS RTK Observables)

10.270.3.24 print_MT1003()

```
std::string RtcM::print_MT1003 (
    const Gps_Ephemeris & ephL1,
    const Gps_CNAV_Ephemeris & ephL2,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint16_t station_id )
```

Prints message type 1003 (L1 & L2 GPS RTK Observables)

10.270.3.25 print_MT1004()

```
std::string RtcM::print_MT1004 (
    const Gps_Ephemeris & ephL1,
    const Gps_CNAV_Ephemeris & ephL2,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint16_t station_id )
```

Prints message type 1004 (Extended L1 & L2 GPS RTK Observables)

10.270.3.26 print_MT1005()

```
std::string RtcM::print_MT1005 (
    uint32_t ref_id,
    double ecef_x,
    double ecef_y,
    double ecef_z,
    bool gps,
    bool glonass,
    bool galileo,
    bool non_physical,
    bool single_oscillator,
    uint32_t quarter_cycle_indicator )
```

Prints message type 1005 (Stationary Antenna Reference Point)

10.270.3.27 print_MT1005_test()

```
std::string RtcM::print_MT1005_test ( )
```

For testing purposes.

10.270.3.28 print_MT1006()

```
std::string RtcM::print_MT1006 (
    uint32_t ref_id,
    double ecef_x,
    double ecef_y,
    double ecef_z,
    bool gps,
    bool glonass,
    bool galileo,
    bool non_physical,
    bool single_oscillator,
    uint32_t quarter_cycle_indicator,
    double height )
```

Prints message type 1006 (Stationary Antenna Reference Point, with Height Information)

10.270.3.29 print_MT1008()

```
std::string RtcM::print_MT1008 (
    uint32_t ref_id,
    const std::string & antenna_descriptor,
    uint32_t antenna_setup_id,
    const std::string & antenna_serial_number )
```

Prints message type 1008 (Antenna Descriptor & Serial Number)

10.270.3.30 print_MT1009()

```
std::string RtcM::print_MT1009 (
    const Glonass_Gnav_Ephemeris & glonass_gnav_eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint16_t station_id )
```

Prints L1-Only GLONASS RTK Observables.

This GLONASS message type is not generally used or supported; type 1012 is to be preferred.

Note

Code added as part of GSoC 2017 program

Parameters

<i>glonass_gnav_eph</i>	GLONASS GNAV Broadcast Ephemeris
<i>obs_time</i>	Time of observation at the moment of printing
<i>observables</i>	Set of observables as defined by the platform

Returns

string with message contents

10.270.3.31 print_MT1010()

```
std::string RtcM::print_MT1010 (
    const Glonass_Gnav_Ephemeris & glonass_gnav_eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint16_t station_id )
```

Prints Extended L1-Only GLONASS RTK Observables.

This GLONASS message type is used when only L1 data is present and bandwidth is very tight, often 1012 is used in such cases.

Note

Code added as part of GSoC 2017 program

Parameters

<i>glonass_gnav_eph</i>	GLONASS GNAV Broadcast Ephemeris
<i>obs_time</i>	Time of observation at the moment of printing
<i>observables</i>	Set of observables as defined by the platform

Returns

string with message contents

10.270.3.32 print_MT1011()

```
std::string RtcM::print_MT1011 (
    const Glonass_Gnav_Ephemeris & glonass_gnav_ephL1,
    const Glonass_Gnav_Ephemeris & glonass_gnav_ephL2,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint16_t station_id )
```

Prints L1&L2 GLONASS RTK Observables.

This GLONASS message type is not generally used or supported; type 1012 is to be preferred

Note

Code added as part of GSoC 2017 program

Parameters

<i>glonass_gnav_eph</i>	GLONASS GNAV Broadcast Ephemeris
<i>obs_time</i>	Time of observation at the moment of printing
<i>observables</i>	Set of observables as defined by the platform

Returns

string with message contents

10.270.3.33 print_MT1012()

```
std::string RtcM::print_MT1012 (
    const Glonass_Gnav_Ephemeris & glonass_gnav_ephL1,
    const Glonass_Gnav_Ephemeris & glonass_gnav_ephL2,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables,
    uint16_t station_id )
```

Prints Extended L1&L2 GLONASS RTK Observables.

This GLONASS message type is the most common observational message type, with L1/L2/SNR content. This is one of the most common messages found.

Note

Code added as part of GSoC 2017 program

Parameters

<i>glonass_gnav_eph</i>	GLONASS GNAV Broadcast Ephemeris
<i>obs_time</i>	Time of observation at the moment of printing
<i>observables</i>	Set of observables as defined by the platform

Returns

string with message contents

10.270.3.34 print_MT1019()

```
std::string RtcM::print_MT1019 (
    const Gps_Ephemeris & gps_eph )
```

Prints message type 1019 (GPS Ephemeris), should be broadcast in the event that the IODC does not match the IODE, and every 2 minutes.

10.270.3.35 print_MT1020()

```
std::string RtcM::print_MT1020 (
    const Glonass_Gnav_Ephemeris & glonass_gnav_eph,
    const Glonass_Gnav_Utc_Model & glonass_gnav_utc_model )
```

Prints message type 1020 (GLONASS Ephemeris).

Note

Code added as part of GSoC 2017 program

Parameters

<i>glonass_gnav_eph</i>	GLONASS GNAV Broadcast Ephemeris
<i>glonass_gnav_utc_model</i>	GLONASS GNAV Clock Information

Returns

Returns message type as a string type

10.270.3.36 print_MT1029()

```
std::string RtcM::print_MT1029 (
    uint32_t ref_id,
```

```
const Gps_Ephemeris & gps_eph,  
double obs_time,  
const std::string & message )
```

Prints message type 1029 (Unicode Text String)

10.270.3.37 print_MT1045()

```
std::string RtcM::print_MT1045 (  
    const Galileo_Ephemeris & gal_eph )
```

Prints message type 1045 (Galileo Ephemeris), should be broadcast every 2 minutes.

10.270.3.38 read_MT1005()

```
int32_t RtcM::read_MT1005 (  
    const std::string & message,  
    uint32_t & ref_id,  
    double & ecef_x,  
    double & ecef_y,  
    double & ecef_z,  
    bool & gps,  
    bool & glonass,  
    bool & galileo )
```

Verifies and reads messages of type 1005 (Stationary Antenna Reference Point). Returns 1 if anything goes wrong, 0 otherwise.

10.270.3.39 read_MT1019()

```
int32_t RtcM::read_MT1019 (  
    const std::string & message,  
    Gps_Ephemeris & gps_eph )
```

Verifies and reads messages of type 1019 (GPS Ephemeris). Returns 1 if anything goes wrong, 0 otherwise.

10.270.3.40 read_MT1020()

```
int32_t RtcM::read_MT1020 (  
    const std::string & message,  
    Glonass_Gnav_Ephemeris & glonass_gnav_eph,  
    Glonass_Gnav_Utc_Model & glonass_gnav_utc_model )
```

Verifies and reads messages of type 1020 (GLONASS Ephemeris).

Note

Code added as part of GSoC 2017 program

Parameters

<i>message</i>	Message to read as a string type
<i>glonass_gnav_eph</i>	GLONASS GNAV Broadcast Ephemeris
<i>glonass_gnav_utc_model</i>	GLONASS GNAV Clock Information

Returns

Returns 1 if anything goes wrong, 0 otherwise.

10.270.3.41 read_MT1045()

```
int32_t RtcM::read_MT1045 (
    const std::string & message,
    Galileo_Ephemeris & gal_eph )
```

Verifies and reads messages of type 1045 (Galileo Ephemeris). Returns 1 if anything goes wrong, 0 otherwise.

10.270.3.42 run_server()

```
void RtcM::run_server ( )
```

Starts running the server.

10.270.3.43 send_message()

```
void RtcM::send_message (
    const std::string & msg )
```

Sends a message through the server to all connected clients.

10.270.3.44 stop_server()

```
void RtcM::stop_server ( )
```

Stops the server.

The documentation for this class was generated from the following file:

- [rtcm.h](#)

10.271 RtcM_Printer Class Reference

This class provides a implementation of a subset of the RTCM Standard 10403.2 messages.

```
#include <rtcm_printer.h>
```

Public Member Functions

- [RtcM_Printer](#) (const std::string &filename, bool flag_rtcM_file_dump, bool flag_rtcM_server, bool flag_rtcM_↵
_tty_port, uint16_t rtcM_tcp_port, uint16_t rtcM_station_id, const std::string &rtcM_dump_devname, bool
time_tag_name=true, const std::string &base_path=".")
Default constructor.
- [~RtcM_Printer](#) ()
Default destructor.
- bool **Print_RtcM_MT1001** (const [Gps_Ephemeris](#) &gps_eph, double obs_time, const std::map< int32_t,
[Gnss_Synchro](#) > &observables)
- bool **Print_RtcM_MT1002** (const [Gps_Ephemeris](#) &gps_eph, double obs_time, const std::map< int32_t,
[Gnss_Synchro](#) > &observables)
- bool **Print_RtcM_MT1003** (const [Gps_Ephemeris](#) &gps_eph, const [Gps_CNAV_Ephemeris](#) &cnav_eph,
double obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables)
- bool **Print_RtcM_MT1004** (const [Gps_Ephemeris](#) &gps_eph, const [Gps_CNAV_Ephemeris](#) &cnav_eph,
double obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables)
- bool **Print_RtcM_MT1009** (const [Glonass_Gnav_Ephemeris](#) &glonass_gnav_eph, double obs_time, const
std::map< int32_t, [Gnss_Synchro](#) > &observables)
Prints L1-Only GLONASS RTK Observables.
- bool **Print_RtcM_MT1010** (const [Glonass_Gnav_Ephemeris](#) &glonass_gnav_eph, double obs_time, const
std::map< int32_t, [Gnss_Synchro](#) > &observables)
Prints Extended L1-Only GLONASS RTK Observables.
- bool **Print_RtcM_MT1011** (const [Glonass_Gnav_Ephemeris](#) &glonass_gnav_ephL1, const [Glonass_Gnav_Ephemeris](#)
&glonass_gnav_ephL2, double obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables)
Prints L1&L2 GLONASS RTK Observables.
- bool **Print_RtcM_MT1012** (const [Glonass_Gnav_Ephemeris](#) &glonass_gnav_ephL1, const [Glonass_Gnav_Ephemeris](#)
&glonass_gnav_ephL2, double obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables)
Prints Extended L1&L2 GLONASS RTK Observables.
- bool **Print_RtcM_MT1019** (const [Gps_Ephemeris](#) &gps_eph)
GPS Ephemeris, should be broadcast in the event that the IODC does not match the IODE, and every 2 minutes.
- bool **Print_RtcM_MT1045** (const [Galileo_Ephemeris](#) &gal_eph)
Galileo Ephemeris, should be broadcast every 2 minutes.
- bool **Print_RtcM_MT1020** (const [Glonass_Gnav_Ephemeris](#) &glonass_gnav_eph, const [Glonass_Gnav_Utc_Model](#)
&utc_model)
Prints GLONASS GNAV Ephemeris.
- bool **Print_RtcM_MSM** (uint32_t msm_number, const [Gps_Ephemeris](#) &gps_eph, const [Gps_CNAV_Ephemeris](#)
&gps_cnav_eph, const [Galileo_Ephemeris](#) &gal_eph, const [Glonass_Gnav_Ephemeris](#) &glo_gnav_eph,
double obs_time, const std::map< int32_t, [Gnss_Synchro](#) > &observables, uint32_t clock_steering_↵
indicator, uint32_t external_clock_indicator, int32_t smooth_int, bool divergence_free, bool more_messages)
- std::string **print_MT1005_test** ()
For testing purposes.
- uint32_t **lock_time** (const [Gps_Ephemeris](#) &eph, double obs_time, const [Gnss_Synchro](#) &gnss_synchro)
- uint32_t **lock_time** (const [Gps_CNAV_Ephemeris](#) &eph, double obs_time, const [Gnss_Synchro](#) &gnss_↵
synchro)
- uint32_t **lock_time** (const [Galileo_Ephemeris](#) &eph, double obs_time, const [Gnss_Synchro](#) &gnss_synchro)
- uint32_t **lock_time** (const [Glonass_Gnav_Ephemeris](#) &eph, double obs_time, const [Gnss_Synchro](#) &gnss_↵
_synchro)
Locks time for logging given GLONASS GNAV Broadcast Ephemeris.

10.271.1 Detailed Description

This class provides a implementation of a subset of the RTCM Standard 10403.2 messages.

Definition at line 43 of file `rtcn_printer.h`.

10.271.2 Constructor & Destructor Documentation

10.271.2.1 Rtcn_Printer()

```
Rtcn_Printer::Rtcn_Printer (
    const std::string & filename,
    bool flag_rtcn_file_dump,
    bool flag_rtcn_server,
    bool flag_rtcn_tty_port,
    uint16_t rtcn_tcp_port,
    uint16_t rtcn_station_id,
    const std::string & rtcn_dump_devname,
    bool time_tag_name = true,
    const std::string & base_path = "." )
```

Default constructor.

10.271.2.2 ~Rtcn_Printer()

```
Rtcn_Printer::~Rtcn_Printer ( )
```

Default destructor.

10.271.3 Member Function Documentation

10.271.3.1 lock_time()

```
uint32_t Rtcn_Printer::lock_time (
    const Glonass_Gnav_Ephemeris & eph,
    double obs_time,
    const Gnss_Synchro & gnss_synchro )
```

Locks time for logging given GLONASS GNAV Broadcast Ephemeris.

Note

Code added as part of GSoC 2017 program `glonass_gnav_eph` GLONASS GNAV Broadcast Ephemeris
`obs_time` Time of observation at the moment of printing observables Set of observables as defined by the platform

Returns

locked time during logging process

10.271.3.2 print_MT1005_test()

```
std::string RtcM_Printer::print_MT1005_test ( )
```

For testing purposes.

10.271.3.3 Print_RtcM_MT1009()

```
bool RtcM_Printer::Print_RtcM_MT1009 (
    const Glonass_Gnav_Ephemeris & glonass_gnav_eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables )
```

Prints L1-Only GLONASS RTK Observables.

This GLONASS message type is not generally used or supported; type 1012 is to be preferred.

Note

Code added as part of GSoC 2017 program

Parameters

<i>glonass_gnav_eph</i>	GLONASS GNAV Broadcast Ephemeris
<i>obs_time</i>	Time of observation at the moment of printing
<i>observables</i>	Set of observables as defined by the platform

Returns

true or false upon operation success

10.271.3.4 Print_RtcM_MT1010()

```
bool RtcM_Printer::Print_RtcM_MT1010 (
    const Glonass_Gnav_Ephemeris & glonass_gnav_eph,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables )
```

Prints Extended L1-Only GLONASS RTK Observables.

This GLONASS message type is used when only L1 data is present and bandwidth is very tight, often 1012 is used in such cases.

Note

Code added as part of GSoC 2017 program

Parameters

<i>glonass_gnav_eph</i>	GLONASS GNAV Broadcast Ephemeris
<i>obs_time</i>	Time of observation at the moment of printing
<i>observables</i>	Set of observables as defined by the platform

Returns

true or false upon operation success

10.271.3.5 Print_Rtcn_MT1011()

```
bool Rtcn_Printer::Print_Rtcn_MT1011 (
    const Glonass_Gnav_Ephemeris & glonass_gnav_ephL1,
    const Glonass_Gnav_Ephemeris & glonass_gnav_ephL2,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables )
```

Prints L1&L2 GLONASS RTK Observables.

This GLONASS message type is not generally used or supported; type 1012 is to be preferred

Note

Code added as part of GSoC 2017 program

Parameters

<i>glonass_gnav_ephL1</i>	GLONASS L1 GNAV Broadcast Ephemeris for satellite
<i>glonass_gnav_ephL2</i>	GLONASS L2 GNAV Broadcast Ephemeris for satellite
<i>obs_time</i>	Time of observation at the moment of printing
<i>observables</i>	Set of observables as defined by the platform

Returns

true or false upon operation success

10.271.3.6 Print_Rtcn_MT1012()

```
bool Rtcn_Printer::Print_Rtcn_MT1012 (
    const Glonass_Gnav_Ephemeris & glonass_gnav_ephL1,
    const Glonass_Gnav_Ephemeris & glonass_gnav_ephL2,
    double obs_time,
    const std::map< int32_t, Gnss_Synchro > & observables )
```

Prints Extended L1&L2 GLONASS RTK Observables.

This GLONASS message type is the most common observational message type, with L1/L2/SNR content. This is one of the most common messages found.

Note

Code added as part of GSoC 2017 program

Parameters

<i>glonass_gnav_ephL1</i>	GLONASS L1 GNAV Broadcast Ephemeris for satellite
<i>glonass_gnav_ephL2</i>	GLONASS L2 GNAV Broadcast Ephemeris for satellite
<i>obs_time</i>	Time of observation at the moment of printing
<i>observables</i>	Set of observables as defined by the platform

Returns

true or false upon operation success

10.271.3.7 Print_Rtcm_MT1019()

```
bool RtcM_Printer::Print_Rtcm_MT1019 (
    const Gps_Ephemeris & gps_eph )
```

GPS Ephemeris, should be broadcast in the event that the IODC does not match the IODE, and every 2 minutes.

10.271.3.8 Print_Rtcm_MT1020()

```
bool RtcM_Printer::Print_Rtcm_MT1020 (
    const Glonass_Gnav_Ephemeris & glonass_gnav_eph,
    const Glonass_Gnav_Utc_Model & utc_model )
```

Prints GLONASS GNAV Ephemeris.

This GLONASS message should be broadcast every 2 minutes

Note

Code added as part of GSoC 2017 program

Parameters

<i>glonass_gnav_eph</i>	GLONASS GNAV Broadcast Ephemeris
<i>utc_model</i>	GLONASS GNAV Clock Information broadcast in string 5

Returns

true or false upon operation success

10.271.3.9 Print_Rtcn_MT1045()

```
bool Rtcn_Printer::Print_Rtcn_MT1045 (
    const Galileo\_Ephemeris & gal_eph )
```

Galileo Ephemeris, should be broadcast every 2 minutes.

The documentation for this class was generated from the following file:

- [rtcn_printer.h](#)

10.272 rtcn_t Struct Reference

Public Attributes

- int **staid**
- int **stah**
- int **seqno**
- int **outtype**
- [gtime_t](#) **time**
- [gtime_t](#) **time_s**
- [obs_t](#) **obs**
- [nav_t](#) **nav**
- [sta_t](#) **sta**
- [dgps_t](#) * **dgps**
- [ssr_t](#) **ssr** [MAXSAT]
- char **msg** [128]
- char **msgtype** [256]
- char **msmtype** [6][128]
- int **obsflag**
- int **ephsat**
- double **cp** [MAXSAT][[NFREQ+NEXOBS](#)]
- unsigned short **lock** [MAXSAT][[NFREQ+NEXOBS](#)]
- unsigned short **loss** [MAXSAT][[NFREQ+NEXOBS](#)]
- [gtime_t](#) **lltime** [MAXSAT][[NFREQ+NEXOBS](#)]
- int **nbyte**
- int **nbit**
- int **len**
- unsigned char **buff** [1200]
- unsigned int **word**
- unsigned int **nmsg2** [100]
- unsigned int **nmsg3** [400]
- char **opt** [256]

10.272.1 Detailed Description

Definition at line 868 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.273 rtk_t Struct Reference

Public Attributes

- [sol_t](#) **sol**
- double **rb** [6]
- int **nx**
- int **na**
- double **tt**
- double * **x**
- double * **P**
- double * **xa**
- double * **Pa**
- int **nfix**
- [ambc_t](#) **ambc** [MAXSAT]
- [ssat_t](#) **ssat** [MAXSAT]
- int **neb**
- char **errbuf** [MAXERRMSG]
- [prcopt_t](#) **opt**

10.273.1 Detailed Description

Definition at line 1059 of file rtklib.h.

The documentation for this struct was generated from the following file:

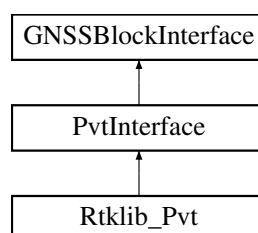
- [rtklib.h](#)

10.274 Rtklib_Pvt Class Reference

This class implements a [PvtInterface](#) for the RTKLIB PVT block.

```
#include <rtklib_pvt.h>
```

Inheritance diagram for Rtklib_Pvt:



Public Member Functions

- **Rtklib_Pvt** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override

Returns "RTKLIB_PVT".
- void **clear_ephemeris** () override
- std::map< int, [Gps_Ephemeris](#) > **get_gps_ephemeris** () const override
- std::map< int, [Galileo_Ephemeris](#) > **get_galileo_ephemeris** () const override
- std::map< int, [Gps_Almanac](#) > **get_gps_almanac** () const override
- std::map< int, [Galileo_Almanac](#) > **get_galileo_almanac** () const override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **reset** () override
- size_t **item_size** () override

All blocks must have an [item_size\(\)](#) function implementation.
- bool **get_latest_PVT** (double *longitude_deg, double *latitude_deg, double *height_m, double *ground_speed_kmh, double *course_over_ground_deg, time_t *UTC_time) override

10.274.1 Detailed Description

This class implements a [PvtInterface](#) for the RTKLIB PVT block.

Definition at line 44 of file rtklib_pvt.h.

10.274.2 Member Function Documentation

10.274.2.1 implementation()

```
std::string Rtklib_Pvt::implementation ( ) [inline], [override], [virtual]
```

Returns "RTKLIB_PVT".

Implements [GNSSBlockInterface](#).

Definition at line 60 of file rtklib_pvt.h.

10.274.2.2 item_size()

```
size_t Rtklib_Pvt::item_size ( ) [inline], [override], [virtual]
```

All blocks must have an [item_size\(\)](#) function implementation.

Implements [GNSSBlockInterface](#).

Definition at line 82 of file `rtklib_pvt.h`.

The documentation for this class was generated from the following file:

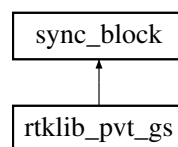
- [rtklib_pvt.h](#)

10.275 rtklib_pvt_gs Class Reference

This class implements a block that computes the PVT solution using the RTKLIB integrated library.

```
#include <rtklib_pvt_gs.h>
```

Inheritance diagram for `rtklib_pvt_gs`:



Public Member Functions

- [~rtklib_pvt_gs](#) ()
Default destructor.
- `std::map< int, Gps_Ephemeris > get_gps_ephemeris_map () const`
Get latest set of GPS ephemeris from PVT block.
- `std::map< int, Gps_Almanac > get_gps_almanac_map () const`
Get latest set of GPS almanac from PVT block.
- `std::map< int, Galileo_Ephemeris > get_galileo_ephemeris_map () const`
Get latest set of Galileo ephemeris from PVT block.
- `std::map< int, Galileo_Almanac > get_galileo_almanac_map () const`
Get latest set of Galileo almanac from PVT block.
- `std::map< int, Beidou_Dnav_Ephemeris > get_beidou_dnav_ephemeris_map () const`
Get latest set of BeiDou DNAV ephemeris from PVT block.
- `std::map< int, Beidou_Dnav_Almanac > get_beidou_dnav_almanac_map () const`
Get latest set of BeiDou DNAV almanac from PVT block.
- `void clear_ephemeris ()`
Clear all ephemeris information and the almanacs for GPS and Galileo.
- `bool get_latest_PVT (double *longitude_deg, double *latitude_deg, double *height_m, double *ground_speed_kmh, double *course_over_ground_deg, time_t *UTC_time) const`
Get the latest Position WGS84 [deg], Ground Velocity, Course over Ground, and UTC Time, if available.
- `int work (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)`
PVT Signal Processing.

Friends

- rtklib_pvt_gs_sptr **rtklib_make_pvt_gs** (uint32_t nchannels, const [Pvt_Conf](#) &conf_, const [rtk_t](#) &rtk)

10.275.1 Detailed Description

This class implements a block that computes the PVT solution using the RTKLIB integrated library.

Definition at line 74 of file rtklib_pvt_gs.h.

10.275.2 Constructor & Destructor Documentation

10.275.2.1 ~rtklib_pvt_gs()

```
rtklib_pvt_gs::~~rtklib_pvt_gs ( )
```

Default destructor.

10.275.3 Member Function Documentation

10.275.3.1 clear_ephemeris()

```
void rtklib_pvt_gs::clear_ephemeris ( )
```

Clear all ephemeris information and the almanacs for GPS and Galileo.

10.275.3.2 get_beidou_dnav_almanac_map()

```
std::map<int, Beidou\_Dnav\_Almanac> rtklib_pvt_gs::get_beidou_dnav_almanac_map ( ) const
```

Get latest set of BeiDou DNAV almanac from PVT block.

10.275.3.3 get_beidou_dnav_ephemeris_map()

```
std::map<int, Beidou\_Dnav\_Ephemeris> rtklib_pvt_gs::get_beidou_dnav_ephemeris_map ( ) const
```

Get latest set of BeiDou DNAV ephemeris from PVT block.

10.275.3.4 `get_galileo_almanac_map()`

```
std::map<int, Galileo_Almanac> rtklib_pvt_gs::get_galileo_almanac_map ( ) const
```

Get latest set of Galileo almanac from PVT block.

10.275.3.5 `get_galileo_ephemeris_map()`

```
std::map<int, Galileo_Ephemeris> rtklib_pvt_gs::get_galileo_ephemeris_map ( ) const
```

Get latest set of Galileo ephemeris from PVT block.

10.275.3.6 `get_gps_almanac_map()`

```
std::map<int, Gps_Almanac> rtklib_pvt_gs::get_gps_almanac_map ( ) const
```

Get latest set of GPS almanac from PVT block.

10.275.3.7 `get_gps_ephemeris_map()`

```
std::map<int, Gps_Ephemeris> rtklib_pvt_gs::get_gps_ephemeris_map ( ) const
```

Get latest set of GPS ephemeris from PVT block.

10.275.3.8 `get_latest_PVT()`

```
bool rtklib_pvt_gs::get_latest_PVT (
    double * longitude_deg,
    double * latitude_deg,
    double * height_m,
    double * ground_speed_kmh,
    double * course_over_ground_deg,
    time_t * UTC_time ) const
```

Get the latest Position WGS84 [deg], Ground Velocity, Course over Ground, and UTC Time, if available.

10.275.3.9 work()

```
int rtklib_pvt_gs::work (
    int noutput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

PVT Signal Processing.

The documentation for this class was generated from the following file:

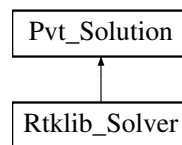
- [rtklib_pvt_gs.h](#)

10.276 Rtklib_Solver Class Reference

This class implements a PVT solution based on RTKLIB.

```
#include <rtklib_solver.h>
```

Inheritance diagram for Rtklib_Solver:



Public Member Functions

- **Rtklib_Solver** (const [rtk_t](#) &rtk, int nchannels, const std::string &dump_filename, bool flag_dump_to_file, bool flag_dump_to_mat)
- bool **get_PVT** (const std::map< int, [Gnss_Synchro](#) > &gnss_observables_map, bool flag_averaging)
- double **get_hdop** () const override
- double **get_vdop** () const override
- double **get_pdop** () const override
- double **get_gdop** () const override
- [Monitor_Pvt](#) **get_monitor_pvt** () const

Public Attributes

- [sol_t](#) **pvt_sol** {}
- std::array< [ssat_t](#), MAXSAT > **pvt_ssat** {}
- std::map< int, [Galileo_Ephemeris](#) > **galileo_ephemeris_map**
Map storing new Galileo_Ephemeris.
- std::map< int, [Gps_Ephemeris](#) > **gps_ephemeris_map**
Map storing new GPS_Ephemeris.
- std::map< int, [Gps_CNAV_Ephemeris](#) > **gps_cnav_ephemeris_map**
Map storing new GPS_CNAV_Ephemeris.
- std::map< int, [Glonass_Gnav_Ephemeris](#) > **glonass_gnav_ephemeris_map**

- Map storing new GLONASS GNAV Ephemeris.*
- `std::map< int, Beidou_Dnav_Ephemeris > beidou_dnav_ephemeris_map`
- Map storing new BeiDou DNAV Ephemeris.*
- `Galileo_Utc_Model galileo_utc_model`
- `Galileo_Iono galileo_iono`
- `std::map< int, Galileo_Almanac > galileo_almanac_map`
- `Gps_Utc_Model gps_utc_model`
- `Gps_Iono gps_iono`
- `std::map< int, Gps_Almanac > gps_almanac_map`
- `Gps_CNAV_Iono gps_cnav_iono`
- `Gps_CNAV_Utc_Model gps_cnav_utc_model`
- `Glonass_Gnav_Utc_Model glonass_gnav_utc_model`
- Map storing GLONASS GNAV UTC Model.*
- `Glonass_Gnav_Almanac glonass_gnav_almanac`
- Map storing GLONASS GNAV Almanac Model.*
- `Beidou_Dnav_Utc_Model beidou_dnav_utc_model`
- `Beidou_Dnav_Iono beidou_dnav_iono`
- `std::map< int, Beidou_Dnav_Almanac > beidou_dnav_almanac_map`

10.276.1 Detailed Description

This class implements a PVT solution based on RTKLIB.

Definition at line 69 of file `rtklib_solver.h`.

10.276.2 Member Data Documentation

10.276.2.1 `beidou_dnav_ephemeris_map`

```
std::map<int, Beidou\_Dnav\_Ephemeris> Rtklib_Solver::beidou_dnav_ephemeris_map
```

Map storing new BeiDou DNAV Ephemeris.

Definition at line 90 of file `rtklib_solver.h`.

10.276.2.2 `galileo_ephemeris_map`

```
std::map<int, Galileo\_Ephemeris> Rtklib_Solver::galileo_ephemeris_map
```

Map storing new [Galileo_Ephemeris](#).

Definition at line 86 of file `rtklib_solver.h`.

10.276.2.3 glonass_gnav_almanac

`Glonass_Gnav_Almanac` `Rtklib_Solver::glonass_gnav_almanac`

Map storing GLONASS GNAV Almanac Model.

Definition at line 104 of file `rtklib_solver.h`.

10.276.2.4 glonass_gnav_ephemeris_map

`std::map<int, Glonass_Gnav_Ephemeris>` `Rtklib_Solver::glonass_gnav_ephemeris_map`

Map storing new GLONASS GNAV Ephemeris.

Definition at line 89 of file `rtklib_solver.h`.

10.276.2.5 glonass_gnav_utc_model

`Glonass_Gnav_Utc_Model` `Rtklib_Solver::glonass_gnav_utc_model`

Map storing GLONASS GNAV UTC Model.

Definition at line 103 of file `rtklib_solver.h`.

10.276.2.6 gps_cnav_ephemeris_map

`std::map<int, Gps_CNAV_Ephemeris>` `Rtklib_Solver::gps_cnav_ephemeris_map`

Map storing new GPS_CNAV_Ephemeris.

Definition at line 88 of file `rtklib_solver.h`.

10.276.2.7 gps_ephemeris_map

`std::map<int, Gps_Ephemeris>` `Rtklib_Solver::gps_ephemeris_map`

Map storing new GPS_Ephemeris.

Definition at line 87 of file `rtklib_solver.h`.

The documentation for this class was generated from the following file:

- [rtklib_solver.h](#)

10.277 Rtklib_Solver_Dump_Reader Class Reference

Public Member Functions

- bool **read_binary_obs** ()
- bool **restart** ()
- int64_t **num_epochs** ()
- bool **open_obs_file** (std::string out_file)

Public Attributes

- uint32_t **TOW_at_current_symbol_ms**
- uint32_t **week**
- double **RX_time**
- double **clk_offset_s**
- double **rr** [6]
- double **qr** [6]
- double **latitude**
- double **longitude**
- double **height**
- uint8_t **ns**
- uint8_t **status**
- uint8_t **type**
- float **AR_ratio**
- float **AR_thres**
- double **dop** [4]

10.277.1 Detailed Description

Definition at line 28 of file rtklib_solver_dump_reader.h.

The documentation for this class was generated from the following file:

- [rtklib_solver_dump_reader.h](#)

10.278 rtksvr_t Struct Reference

Public Attributes

- int **state**
- int **cycle**
- int **nmeacycle**
- int **nmeareq**
- double **nmeapos** [3]
- int **buffsize**
- int **format** [3]
- [solopt_t](#) **solopt** [2]
- int **navsel**
- int **nsbs**

- int **nsol**
- [rtk_t](#) **rtk**
- int **nb** [3]
- int **nsb** [2]
- int **npb** [3]
- unsigned char * **buff** [3]
- unsigned char * **sbuf** [2]
- unsigned char * **pbuf** [3]
- [sol_t](#) **solbuf** [MAXSOLBUF]
- unsigned int **nmsg** [3][10]
- [raw_t](#) **raw** [3]
- [rtcm_t](#) **rtcm** [3]
- [gtime_t](#) **ftime** [3]
- char **files** [3][MAXSTRPATH]
- [obs_t](#) **obs** [3][MAXOBSBUF]
- [nav_t](#) **nav**
- [sbsmsg_t](#) **sbsmsg** [MAXSBSMSG]
- [stream_t](#) **stream** [8]
- [stream_t](#) * **moni**
- unsigned int **tick**
- [pthread_t](#) **thread**
- int **cputime**
- int **prcout**
- [lock_t](#) **lock**

10.278.1 Detailed Description

Definition at line 1231 of file [rtklib.h](#).

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.279 Rtl_Tcp_Dongle_Info Class Reference

This class represents the dongle information which is sent by [rtl_tcp](#).

```
#include <rtl_tcp_dongle_info.h>
```

Public Types

- enum {
TUNER_UNKNOWN = 0, **TUNER_E4000**, **TUNER_FC0012**, **TUNER_FC0013**,
TUNER_FC2580, **TUNER_R820T**, **TUNER_R828D** }

Public Member Functions

- `boost::system::error_code read (boost::asio::ip::tcp::socket &socket)`
- `bool is_valid () const`
- `const char * get_type_name () const`
- `double clip_gain (int gain) const`
- `uint32_t get_tuner_type () const`
- `uint32_t get_tuner_gain_count () const`

10.279.1 Detailed Description

This class represents the dongle information which is sent by rtl_tcp.

Definition at line 31 of file `rtl_tcp_dongle_info.h`.

The documentation for this class was generated from the following file:

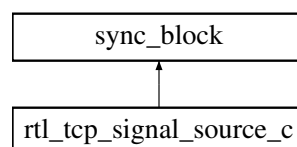
- [rtl_tcp_dongle_info.h](#)

10.280 rtl_tcp_signal_source_c Class Reference

This class reads interleaved I/Q samples from an rtl_tcp server and outputs complex types.

```
#include <rtl_tcp_signal_source_c.h>
```

Inheritance diagram for `rtl_tcp_signal_source_c`:



Public Member Functions

- `int work (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)`
- `void set_frequency (int frequency)`
- `void set_sample_rate (int sample_rate)`
- `void set_agc_mode (bool agc)`
- `void set_gain (int gain)`
- `void set_if_gain (int gain)`

Friends

- `rtl_tcp_signal_source_c_sptr rtl_tcp_make_signal_source_c (const std::string &address, int16_t port, bool flip_iq)`

10.280.1 Detailed Description

This class reads interleaved I/Q samples from an rtl_tcp server and outputs complex types.

Definition at line 69 of file rtl_tcp_signal_source_c.h.

The documentation for this class was generated from the following file:

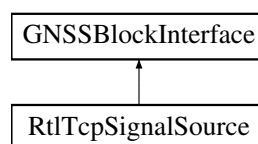
- [rtl_tcp_signal_source_c.h](#)

10.281 RtlTcpSignalSource Class Reference

This class reads from rtl_tcp, which streams interleaved I/Q samples over TCP. (see <https://osmocom.org/projects/rtl-sdr/wiki>)

```
#include <rtl_tcp_signal_source.h>
```

Inheritance diagram for RtlTcpSignalSource:



Public Member Functions

- **RtlTcpSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_stream, unsigned int out_stream, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override

Returns "RtlTcp_Signal_Source".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.281.1 Detailed Description

This class reads from rtl_tcp, which streams interleaved I/Q samples over TCP. (see <https://osmocom.org/projects/rtl-sdr/wiki>)

Definition at line 47 of file rtl_tcp_signal_source.h.

10.281.2 Member Function Documentation

10.281.2.1 implementation()

```
std::string RtlTcpSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "RtlTcp_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 66 of file `rtl_tcp_signal_source.h`.

The documentation for this class was generated from the following file:

- [rtl_tcp_signal_source.h](#)

10.282 Sbas_Ephemeris Class Reference

This class stores SBAS SV ephemeris data.

```
#include <sbas_ephemeris.h>
```

Public Member Functions

- void **print** (std::ostream &out)

Public Attributes

- int [i_prn](#) {}
PRN number.
- int [i_t0](#) {}
Reference epoch time (GPST)
- double [d_tof](#) {}
Time of message frame (GPST)
- int [i_sv_ura](#) {}
SV accuracy (URA index), not standardized.
- bool [b_sv_do_not_use](#) {}
Health status (false:do not use / true:usable)
- double [d_pos](#) [3] {}
Satellite position (m) (ECEF)
- double [d_vel](#) [3] {}
Satellite velocity (m/s) (ECEF)
- double [d_acc](#) [3] {}
Satellite acceleration (m/s²) (ECEF)
- double [d_af0](#) {}
Satellite clock-offset (s)
- double [d_af1](#) {}
Satellite drift (s/s)

10.282.1 Detailed Description

This class stores SBAS SV ephemeris data.

Definition at line 30 of file sbas_ephemeris.h.

10.282.2 Member Data Documentation

10.282.2.1 b_sv_do_not_use

```
bool Sbas_Ephemeris::b_sv_do_not_use {}
```

Health status (false:do not use / true:usable)

Definition at line 39 of file sbas_ephemeris.h.

10.282.2.2 d_acc

```
double Sbas_Ephemeris::d_acc[3] {}
```

Satellite acceleration (m/s²) (ECEF)

Definition at line 42 of file sbas_ephemeris.h.

10.282.2.3 d_af0

```
double Sbas_Ephemeris::d_af0 {}
```

Satellite clock-offset (s)

Definition at line 43 of file sbas_ephemeris.h.

10.282.2.4 d_af1

```
double Sbas_Ephemeris::d_af1 {}
```

Satellite drift (s/s)

Definition at line 44 of file sbas_ephemeris.h.

10.282.2.5 d_pos

```
double Sbas_Ephemeris::d_pos[3] {}
```

Satellite position (m) (ECEF)

Definition at line 40 of file sbas_ephemeris.h.

10.282.2.6 d_tof

```
double Sbas_Ephemeris::d_tof {}
```

Time of message frame (GPST)

Definition at line 37 of file sbas_ephemeris.h.

10.282.2.7 d_vel

```
double Sbas_Ephemeris::d_vel[3] {}
```

Satellite velocity (m/s) (ECEF)

Definition at line 41 of file sbas_ephemeris.h.

10.282.2.8 i_prn

```
int Sbas_Ephemeris::i_prn {}
```

PRN number.

Definition at line 35 of file sbas_ephemeris.h.

10.282.2.9 i_sv_ura

```
int Sbas_Ephemeris::i_sv_ura {}
```

SV accuracy (URA index), not standardized.

Definition at line 38 of file sbas_ephemeris.h.

10.282.2.10 i_t0

```
int Sbas_Ephemeris::i_t0 {}
```

Reference epoch time (GPST)

Definition at line 36 of file sbas_ephemeris.h.

The documentation for this class was generated from the following file:

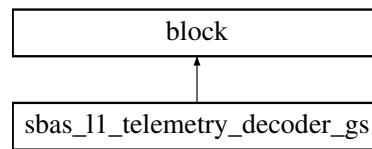
- [sbas_ephemeris.h](#)

10.283 sbas_l1_telemetry_decoder_gs Class Reference

This class implements a block that decodes the SBAS integrity and corrections data defined in RTCA MOPS DO-229.

```
#include <sbas_l1_telemetry_decoder_gs.h>
```

Inheritance diagram for sbas_l1_telemetry_decoder_gs:



Public Member Functions

- void [set_satellite](#) (const [Gnss_Satellite](#) &satellite)
Set satellite PRN.
- void [set_channel](#) (int32_t channel)
Set receiver's channel.
- void [reset](#) ()
- int [general_work](#) (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)
This is where all signal processing takes place.

Friends

- sbas_l1_telemetry_decoder_gs_sptr [sbas_l1_make_telemetry_decoder_gs](#) (const [Gnss_Satellite](#) &satellite, bool dump)

10.283.1 Detailed Description

This class implements a block that decodes the SBAS integrity and corrections data defined in RTCA MOPS DO-229.

Definition at line 59 of file sbas_l1_telemetry_decoder_gs.h.

10.283.2 Member Function Documentation

10.283.2.1 `general_work()`

```
int sbas_l1_telemetry_decoder_gs::general_work (
    int noutput_items,
    gr_vector_int & ninput_items,
    gr_vector_const_void_star & input_items,
    gr_vector_void_star & output_items )
```

This is where all signal processing takes place.

10.283.2.2 `set_channel()`

```
void sbas_l1_telemetry_decoder_gs::set_channel (
    int32_t channel )
```

Set receiver's channel.

10.283.2.3 `set_satellite()`

```
void sbas_l1_telemetry_decoder_gs::set_satellite (
    const Gnss_Satellite & satellite )
```

Set satellite PRN.

The documentation for this class was generated from the following file:

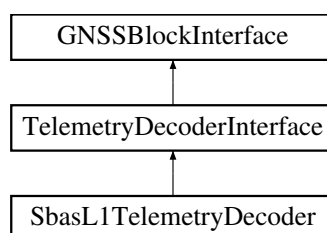
- [sbas_l1_telemetry_decoder_gs.h](#)

10.284 SbasL1TelemetryDecoder Class Reference

This class implements a NAV data decoder for SBAS frames in L1 radio link.

```
#include <sbas_l1_telemetry_decoder.h>
```

Inheritance diagram for SbasL1TelemetryDecoder:



Public Member Functions

- **SbasL1TelemetryDecoder** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams)
- std::string **role** () override
- std::string **implementation** () override
Returns "SBAS_L1_Telemetry_Decoder".
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- void **set_satellite** (const [Gnss_Satellite](#) &satellite) override
- void **set_channel** (int channel) override
- void **reset** () override
- size_t **item_size** () override

10.284.1 Detailed Description

This class implements a NAV data decoder for SBAS frames in L1 radio link.

Definition at line 40 of file `sbas_l1_telemetry_decoder.h`.

10.284.2 Member Function Documentation

10.284.2.1 implementation()

```
std::string SbasL1TelemetryDecoder::implementation ( ) [inline], [override], [virtual]
```

Returns "SBAS_L1_Telemetry_Decoder".

Implements [GNSSBlockInterface](#).

Definition at line 59 of file `sbas_l1_telemetry_decoder.h`.

The documentation for this class was generated from the following file:

- [sbas_l1_telemetry_decoder.h](#)

10.285 sbs_t Struct Reference

Public Attributes

- int **n**
- int **nmax**
- [sbsmsg_t](#) * **msgs**

10.285.1 Detailed Description

Definition at line 567 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.286 sbsfcorr_t Struct Reference

Public Attributes

- [gtime_t](#) **t0**
- double **prc**
- double **rrc**
- double **dt**
- int **iodf**
- short **udre**
- short **ai**

10.286.1 Detailed Description

Definition at line 574 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.287 sbsignp_t Struct Reference

Public Attributes

- [gtime_t](#) **t0**
- short **lat**
- short **lon**
- short **give**
- float **delay**

10.287.1 Detailed Description

Definition at line 613 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.288 sbsigpband_t Struct Reference

Public Attributes

- short **x**
- const short * **y**
- unsigned char **bits**
- unsigned char **bite**

10.288.1 Detailed Description

Definition at line 622 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.289 sbsion_t Struct Reference

Public Attributes

- int **iodi**
- int **nigp**
- [sbsigp_t](#) **igp** [MAXNIGP]

10.289.1 Detailed Description

Definition at line 631 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.290 sbslcorr_t Struct Reference

Public Attributes

- [gtime_t](#) **t0**
- int **iode**
- double **dpos** [3]
- double **dvel** [3]
- double **daf0**
- double **daf1**

10.290.1 Detailed Description

Definition at line 586 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.291 sbsmsg_t Struct Reference

Public Attributes

- int **week**
- int **tow**
- int **prn**
- unsigned char **msg** [29]

10.291.1 Detailed Description

Definition at line 559 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.292 sbssat_t Struct Reference

Public Attributes

- int **iodp**
- int **nsat**
- int **tlat**
- [sbssatp_t](#) **sat** [MAXSAT]

10.292.1 Detailed Description

Definition at line 604 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.293 sbssatp_t Struct Reference

Public Attributes

- int **sat**
- [sbsfcorr_t](#) **fcorr**
- [sbslcorr_t](#) **lcorr**

10.293.1 Detailed Description

Definition at line 596 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.294 seph_t Struct Reference

Public Attributes

- int **sat**
- [gtime_t](#) **t0**
- [gtime_t](#) **tof**
- int **sva**
- int **svh**
- double **pos** [3]
- double **vel** [3]
- double **acc** [3]
- double **af0**
- double **af1**

10.294.1 Detailed Description

Definition at line 494 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.295 Serdes_Gnss_Synchro Class Reference

This class implements serialization and deserialization of [Gnss_Synchro](#) objects using Protocol Buffers.

```
#include <serdes_gnss_synchro.h>
```

Public Member Functions

- [Serdes_Gnss_Synchro](#) (const [Serdes_Gnss_Synchro](#) &other) noexcept
< Copy constructor
- [Serdes_Gnss_Synchro](#) & operator= (const [Serdes_Gnss_Synchro](#) &rhs) noexcept
< Copy assignment operator
- [Serdes_Gnss_Synchro](#) ([Serdes_Gnss_Synchro](#) &&other) noexcept
< Move constructor
- [Serdes_Gnss_Synchro](#) & operator= ([Serdes_Gnss_Synchro](#) &&other) noexcept
< Move assignment operator
- std::string [createProtobuffer](#) (const std::vector< [Gnss_Synchro](#) > &vgs)
- std::vector< [Gnss_Synchro](#) > [readProtobuffer](#) (const gnss_sdr::Observables &obs) const
< Deserialization

10.295.1 Detailed Description

This class implements serialization and deserialization of [Gnss_Synchro](#) objects using Protocol Buffers.

Definition at line 35 of file `serdes_gnss_synchro.h`.

10.295.2 Constructor & Destructor Documentation

10.295.2.1 [Serdes_Gnss_Synchro\(\)](#) [1/2]

```
Serdes_Gnss_Synchro::Serdes_Gnss_Synchro (
    const Serdes\_Gnss\_Synchro & other ) [inline], [noexcept]
```

< Copy constructor

Definition at line 50 of file `serdes_gnss_synchro.h`.

10.295.2.2 [Serdes_Gnss_Synchro\(\)](#) [2/2]

```
Serdes_Gnss_Synchro::Serdes_Gnss_Synchro (
    Serdes\_Gnss\_Synchro && other ) [inline], [noexcept]
```

< Move constructor

Definition at line 61 of file `serdes_gnss_synchro.h`.

10.295.3 Member Function Documentation

10.295.3.1 [createProtobuffer\(\)](#)

```
std::string Serdes_Gnss_Synchro::createProtobuffer (
    const std::vector< Gnss\_Synchro > & vgs ) [inline]
```

Parameters

<code>vgs</code>	Serialization into a string
------------------	-----------------------------

Definition at line 75 of file `serdes_gnss_synchro.h`.

10.295.3.2 `operator=()` [1/2]

```

Serdes_Gnss_Synchro& Serdes_Gnss_Synchro::operator= (
    const Serdes_Gnss_Synchro & rhs ) [inline], [noexcept]

```

< Copy assignment operator

Definition at line 55 of file `serdes_gnss_synchro.h`.

10.295.3.3 `operator=()` [2/2]

```

Serdes_Gnss_Synchro& Serdes_Gnss_Synchro::operator= (
    Serdes_Gnss_Synchro && other ) [inline], [noexcept]

```

< Move assignment operator

Definition at line 66 of file `serdes_gnss_synchro.h`.

10.295.3.4 `readProtobuffer()`

```

std::vector<Gnss_Synchro> Serdes_Gnss_Synchro::readProtobuffer (
    const gnss_sdr::Observables & obs ) const [inline]

```

< Deserialization

Definition at line 123 of file `serdes_gnss_synchro.h`.

References `Gnss_Synchro::Acq_delay_samples`, `Gnss_Synchro::Acq_doppler_hz`, `Gnss_Synchro::Acq_doppler_step`, `Gnss_Synchro::Acq_samplestamp_samples`, `Gnss_Synchro::Carrier_Doppler_hz`, `Gnss_Synchro::Channel_ID`, `Gnss_Synchro::CN0_dB_hz`, `Gnss_Synchro::correlation_length_ms`, `Gnss_Synchro::Flag_valid_acquisition`, `Gnss_Synchro::Flag_valid_pseudorange`, `Gnss_Synchro::Flag_valid_symbol_output`, `Gnss_Synchro::Flag_valid_word`, `Gnss_Synchro::fs`, `Gnss_Synchro::interp_TOW_ms`, `Gnss_Synchro::PRN`, `Gnss_Synchro::Prompt_I`, `Gnss_Synchro::Prompt_Q`, `Gnss_Synchro::Pseudorange_m`, `Gnss_Synchro::RX_time`, `Gnss_Synchro::Signal`, `Gnss_Synchro::System`, `Gnss_Synchro::TOW_at_current_symbol_ms`, and `Gnss_Synchro::Tracking_sample_counter`.

The documentation for this class was generated from the following file:

- [serdes_gnss_synchro.h](#)

10.296 Serdes_Monitor_Pvt Class Reference

This class implements serialization and deserialization of [Monitor_Pvt](#) objects using Protocol Buffers.

```
#include <serdes_monitor_pvt.h>
```

Public Member Functions

- [Serdes_Monitor_Pvt](#) (const [Serdes_Monitor_Pvt](#) &other) noexcept
 < Copy constructor
- [Serdes_Monitor_Pvt](#) & operator= (const [Serdes_Monitor_Pvt](#) &rhs) noexcept
 < Copy assignment operator
- [Serdes_Monitor_Pvt](#) ([Serdes_Monitor_Pvt](#) &&other) noexcept
 < Move constructor
- [Serdes_Monitor_Pvt](#) & operator= ([Serdes_Monitor_Pvt](#) &&other) noexcept
 < Move assignment operator
- std::string [createProtobuffer](#) (const [Monitor_Pvt](#) *const monitor)
- [Monitor_Pvt](#) [readProtobuffer](#) (const gnss_sdr::MonitorPvt &mon) const
 < Deserialization

10.296.1 Detailed Description

This class implements serialization and deserialization of [Monitor_Pvt](#) objects using Protocol Buffers.

Definition at line 34 of file [serdes_monitor_pvt.h](#).

10.296.2 Constructor & Destructor Documentation

10.296.2.1 Serdes_Monitor_Pvt() [1/2]

```
Serdes_Monitor_Pvt::Serdes_Monitor_Pvt (
    const Serdes\_Monitor\_Pvt & other ) [inline], [noexcept]
```

< Copy constructor

Definition at line 49 of file [serdes_monitor_pvt.h](#).

10.296.2.2 Serdes_Monitor_Pvt() [2/2]

```
Serdes_Monitor_Pvt::Serdes_Monitor_Pvt (
    Serdes\_Monitor\_Pvt && other ) [inline], [noexcept]
```

< Move constructor

Definition at line 60 of file [serdes_monitor_pvt.h](#).

10.296.3 Member Function Documentation

10.296.3.1 createProtobuffer()

```
std::string Serdes_Monitor_Pvt::createProtobuffer (  
    const Monitor_Pvt *const monitor ) [inline]
```

Parameters

<i>monitor</i>	Serialization into a string
----------------	-----------------------------

Definition at line 74 of file `serdes_monitor_pvt.h`.

10.296.3.2 `operator=()` [1/2]

```
Serdes_Monitor_Pvt& Serdes_Monitor_Pvt::operator= (
    const Serdes_Monitor_Pvt & rhs ) [inline], [noexcept]
```

< Copy assignment operator

Definition at line 54 of file `serdes_monitor_pvt.h`.

10.296.3.3 `operator=()` [2/2]

```
Serdes_Monitor_Pvt& Serdes_Monitor_Pvt::operator= (
    Serdes_Monitor_Pvt && other ) [inline], [noexcept]
```

< Move assignment operator

Definition at line 65 of file `serdes_monitor_pvt.h`.

10.296.3.4 `readProtobuffer()`

```
Monitor_Pvt Serdes_Monitor_Pvt::readProtobuffer (
    const gnss_sdr::MonitorPvt & mon ) const [inline]
```

< Deserialization

Definition at line 114 of file `serdes_monitor_pvt.h`.

The documentation for this class was generated from the following file:

- [serdes_monitor_pvt.h](#)

10.297 `serial_t` Struct Reference

Public Attributes

- `dev_t` **dev**
- `int` **error**

10.297.1 Detailed Description

Definition at line 1103 of file rtklib.h.

The documentation for this struct was generated from the following file:

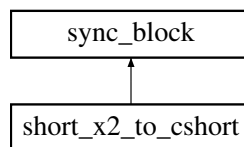
- [rtklib.h](#)

10.298 short_x2_to_cshort Class Reference

This class adapts two short streams into a `std::complex<short>` stream.

```
#include <short_x2_to_cshort.h>
```

Inheritance diagram for `short_x2_to_cshort`:



Public Member Functions

- `int work` (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- `short_x2_to_cshort_sptr make_short_x2_to_cshort ()`

10.298.1 Detailed Description

This class adapts two short streams into a `std::complex<short>` stream.

Definition at line 45 of file `short_x2_to_cshort.h`.

The documentation for this class was generated from the following file:

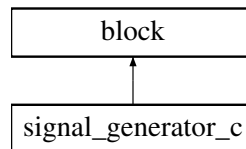
- [short_x2_to_cshort.h](#)

10.299 signal_generator_c Class Reference

This class generates synthesized GNSS signal.

```
#include <signal_generator_c.h>
```

Inheritance diagram for signal_generator_c:



Public Member Functions

- int **general_work** (int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- signal_generator_c_sptr [signal_make_generator_c](#) (const std::vector< std::string > &signal1, const std::vector< std::string > &system, const std::vector< unsigned int > &PRN, const std::vector< float > &C/N0_dB, const std::vector< float > &doppler_Hz, const std::vector< unsigned int > &delay_chips, const std::vector< unsigned int > &delay_sec, bool data_flag, bool noise_flag, unsigned int fs_in, unsigned int vector_length, float BW_BB)

Return a shared_ptr to a new instance of gen_source.

10.299.1 Detailed Description

This class generates synthesized GNSS signal.

See also

[gen_source](#) for a version that subclasses gr_block.

Definition at line 80 of file [signal_generator_c.h](#).

10.299.2 Friends And Related Function Documentation

10.299.2.1 signal_make_generator_c

```

signal_generator_c_sptr signal_make_generator_c (
    const std::vector< std::string > & signall,
    const std::vector< std::string > & system,
    const std::vector< unsigned int > & PRN,
    const std::vector< float > & CNO_dB,
    const std::vector< float > & doppler_Hz,
    const std::vector< unsigned int > & delay_chips,
    const std::vector< unsigned int > & delay_sec,
    bool data_flag,
    bool noise_flag,
    unsigned int fs_in,
    unsigned int vector_length,
    float BW_BB ) [friend]

```

Return a shared_ptr to a new instance of gen_source.

To avoid accidental use of raw pointers, gen_source's constructor is private. signal_make_generator_c is the public interface for creating new instances.

The documentation for this class was generated from the following file:

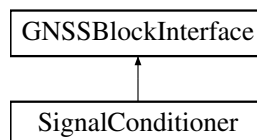
- [signal_generator_c.h](#)

10.300 SignalConditioner Class Reference

This class wraps blocks to change data_type_adapter, input_filter and resampler to be applied to the input flow of sampled signal.

```
#include <signal_conditioner.h>
```

Inheritance diagram for SignalConditioner:



Public Member Functions

- [SignalConditioner](#) (const [ConfigurationInterface](#) *configuration, std::shared_ptr< [GNSSBlockInterface](#) > data_type_adapt, std::shared_ptr< [GNSSBlockInterface](#) > in_filt, std::shared_ptr< [GNSSBlockInterface](#) > res, std::string role, std::string [implementation](#))

Constructor.

- [~SignalConditioner](#) ()=default

Destructor.

- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- std::string **role** () override
- std::string [implementation](#) () override

Returns "Signal_Conditioner".

- size_t **item_size** () override
- std::shared_ptr< [GNSSBlockInterface](#) > **data_type_adapter** ()
- std::shared_ptr< [GNSSBlockInterface](#) > **input_filter** ()
- std::shared_ptr< [GNSSBlockInterface](#) > **resampler** ()

10.300.1 Detailed Description

This class wraps blocks to change `data_type_adapter`, `input_filter` and `resampler` to be applied to the input flow of sampled signal.

Definition at line 36 of file `signal_conditioner.h`.

10.300.2 Constructor & Destructor Documentation

10.300.2.1 SignalConditioner()

```
SignalConditioner::SignalConditioner (
    const ConfigurationInterface * configuration,
    std::shared_ptr< GNSSBlockInterface > data_type_adapt,
    std::shared_ptr< GNSSBlockInterface > in_filt,
    std::shared_ptr< GNSSBlockInterface > res,
    std::string role,
    std::string implementation )
```

Constructor.

10.300.2.2 ~SignalConditioner()

```
SignalConditioner::~SignalConditioner ( ) [default]
```

Destructor.

10.300.3 Member Function Documentation

10.300.3.1 implementation()

```
std::string SignalConditioner::implementation ( ) [inline], [override], [virtual]
```

Returns "Signal_Conditioner".

Implements [GNSSBlockInterface](#).

Definition at line 54 of file `signal_conditioner.h`.

The documentation for this class was generated from the following file:

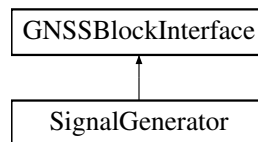
- [signal_conditioner.h](#)

10.301 SignalGenerator Class Reference

This class generates synthesized GNSS signal.

```
#include <signal_generator.h>
```

Inheritance diagram for SignalGenerator:



Public Member Functions

- **SignalGenerator** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_stream, unsigned int out_stream, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
Returns "GNSSSignalGenerator".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override

10.301.1 Detailed Description

This class generates synthesized GNSS signal.

Definition at line 42 of file signal_generator.h.

10.301.2 Member Function Documentation

10.301.2.1 implementation()

```
std::string SignalGenerator::implementation ( ) [inline], [override], [virtual]
```

Returns "GNSSSignalGenerator".

Implements [GNSSBlockInterface](#).

Definition at line 59 of file signal_generator.h.

The documentation for this class was generated from the following file:

- [signal_generator.h](#)

10.302 `snrmask_t` Struct Reference

Public Attributes

- int **ena** [2]
- double **mask** [[NFREQ](#)][9]

10.302.1 Detailed Description

Definition at line 929 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.303 `sol_t` Struct Reference

Public Attributes

- [gtime_t](#) **time**
- double **rr** [6]
- float **qr** [6]
- double **dtr** [6]
- unsigned char **type**
- unsigned char **stat**
- unsigned char **ns**
- float **age**
- float **ratio**
- float **thres**

10.303.1 Detailed Description

Definition at line 813 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.304 `solbuf_t` Struct Reference

Public Attributes

- int **n**
- int **nmax**
- int **cyclic**
- int **start**
- int **end**
- [gtime_t](#) **time**
- [sol_t](#) * **data**
- double **rb** [3]
- unsigned char **buff** [[MAXSOLMSG](#)+1]
- int **nb**

10.304.1 Detailed Description

Definition at line 831 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.305 solopt_t Struct Reference

Public Attributes

- int **posf**
- int **times**
- int **timef**
- int **timeu**
- int **degf**
- int **outhead**
- int **outopt**
- int **datum**
- int **height**
- int **geoid**
- int **solstatic**
- int **sstat**
- int **trace**
- double **nmeaintv** [2]
- char **sep** [64]
- char **prog** [64]
- double **maxsolstd**

10.305.1 Detailed Description

Definition at line 1000 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.306 solstat_t Struct Reference

Public Attributes

- [gtime_t](#) **time**
- unsigned char **sat**
- unsigned char **frq**
- float **az**
- float **el**
- float **resp**
- float **resc**
- unsigned char **flag**
- unsigned char **snr**
- unsigned short **lock**
- unsigned short **outc**
- unsigned short **slipc**
- unsigned short **rejc**

10.306.1 Detailed Description

Definition at line 844 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.307 solstatbuf_t Struct Reference

Public Attributes

- int **n**
- int **nmax**
- [solstat_t](#) * **data**

10.307.1 Detailed Description

Definition at line 861 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.308 Spirent_Motion_Csv_Dump_Reader Class Reference

Public Member Functions

- bool **read_csv_obs** ()
- bool **restart** ()
- int64_t **num_epochs** ()
- bool **open_obs_file** (std::string out_file)
- void **close_obs_file** ()

Public Attributes

- int **header_lines**
- double **TOW_ms**
- double **Pos_X**
- double **Pos_Y**
- double **Pos_Z**
- double **Vel_X**
- double **Vel_Y**
- double **Vel_Z**
- double **Acc_X**
- double **Acc_Y**
- double **Acc_Z**
- double **Jerk_X**
- double **Jerk_Y**
- double **Jerk_Z**
- double **Lat**
- double **Long**
- double **Height**
- double **Heading**
- double **Elevation**
- double **Bank**
- double **Ang_vel_X**
- double **Ang_vel_Y**
- double **Ang_vel_Z**
- double **Ang_acc_X**
- double **Ang_acc_Y**
- double **Ang_acc_Z**
- double **Ant1_Pos_X**
- double **Ant1_Pos_Y**
- double **Ant1_Pos_Z**
- double **Ant1_Vel_X**
- double **Ant1_Vel_Y**
- double **Ant1_Vel_Z**
- double **Ant1_Acc_X**
- double **Ant1_Acc_Y**
- double **Ant1_Acc_Z**
- double **Ant1_Lat**
- double **Ant1_Long**
- double **Ant1_Height**
- double **Ant1_DOP**

10.308.1 Detailed Description

Definition at line 28 of file spirent_motion_csv_dump_reader.h.

The documentation for this class was generated from the following file:

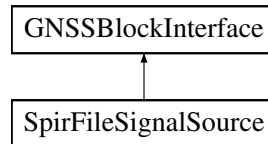
- [spirent_motion_csv_dump_reader.h](#)

10.309 SpirFileSignalSource Class Reference

Class that reads signals samples from a file and adapts it to a `SignalSourceInterface`.

```
#include <spir_file_signal_source.h>
```

Inheritance diagram for `SpirFileSignalSource`:



Public Member Functions

- **SpirFileSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
Returns "*Spir_File_Signal_Source*".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- std::string **filename** () const
- std::string **item_type** () const
- bool **repeat** () const
- int64_t **sampling_frequency** () const
- uint64_t **samples** () const

10.309.1 Detailed Description

Class that reads signals samples from a file and adapts it to a `SignalSourceInterface`.

Definition at line 46 of file `spir_file_signal_source.h`.

10.309.2 Member Function Documentation

10.309.2.1 implementation()

```
std::string SpirFileSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "*Spir_File_Signal_Source*".

Implements [GNSSBlockInterface](#).

Definition at line 63 of file `spir_file_signal_source.h`.

The documentation for this class was generated from the following file:

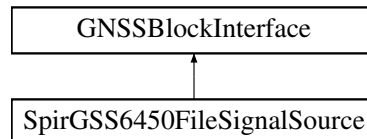
- [spir_file_signal_source.h](#)

10.310 SpirGSS6450FileSignalSource Class Reference

Class that reads signals samples from a file and adapts it to a SignalSourceInterface.

```
#include <spir_gss6450_file_signal_source.h>
```

Inheritance diagram for SpirGSS6450FileSignalSource:



Public Member Functions

- **SpirGSS6450FileSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, uint32_t in_streams, uint32_t out_streams, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** (int RF_channel) override
- gr::basic_block_sptr **get_right_block** () override
- std::string **filename** () const
- std::string **item_type** () const
- bool **repeat** () const
- int64_t **sampling_frequency** () const
- uint64_t **samples** () const

10.310.1 Detailed Description

Class that reads signals samples from a file and adapts it to a SignalSourceInterface.

Definition at line 52 of file spir_gss6450_file_signal_source.h.

The documentation for this class was generated from the following file:

- [spir_gss6450_file_signal_source.h](#)

10.311 `ssat_t` Struct Reference

Public Attributes

- unsigned char **sys**
- unsigned char **vs**
- double **azel** [2]
- double **resp** [NFREQ]
- double **resc** [NFREQ]
- unsigned char **vsat** [NFREQ]
- unsigned char **snr** [NFREQ]
- unsigned char **fix** [NFREQ]
- unsigned char **slip** [NFREQ]
- unsigned char **half** [NFREQ]
- int **lock** [NFREQ]
- unsigned int **outc** [NFREQ]
- unsigned int **slipc** [NFREQ]
- unsigned int **rejc** [NFREQ]
- double **gf**
- double **gf2**
- double **mw**
- double **phw**
- [gtime_t](#) **pt** [2][NFREQ]
- double **ph** [2][NFREQ]

10.311.1 Detailed Description

Definition at line 1023 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.312 `ssr_t` Struct Reference

Public Attributes

- [gtime_t](#) **t0** [6]
- double **udi** [6]
- int **iod** [6]
- int **iode**
- int **iodcrc**
- int **ura**
- int **refd**
- double **deph** [3]
- double **ddeph** [3]
- double **dclk** [3]
- double **hrclk**
- float **cbias** [MAXCODE]
- double **pbias** [MAXCODE]
- float **stdpb** [MAXCODE]
- double **yaw_ang**
- double **yaw_rate**
- unsigned char **update**

10.312.1 Detailed Description

Definition at line 649 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.313 sta_t Struct Reference

Public Attributes

- char **name** [[MAXANT](#)]
- char **marker** [[MAXANT](#)]
- char **antdes** [[MAXANT](#)]
- char **antsno** [[MAXANT](#)]
- char **rectype** [[MAXANT](#)]
- char **recver** [[MAXANT](#)]
- char **recsno** [[MAXANT](#)]
- int **antsetup**
- int **itr**
- int **deltype**
- double **pos** [3]
- double **del** [3]
- double **hgt**

10.313.1 Detailed Description

Definition at line 795 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.314 stec_t Struct Reference

Public Attributes

- [gtime_t](#) **time**
- unsigned char **sat**
- double **ion**
- float **std**
- float **azel** [2]
- unsigned char **flag**

10.314.1 Detailed Description

Definition at line 715 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.315 stream_cfg Struct Reference

Public Attributes

- int64_t **bw_hz**
- int64_t **fs_hz**
- int64_t **lo_hz**
- const char * **rfport**

10.315.1 Detailed Description

Definition at line 39 of file ad9361_manager.h.

The documentation for this struct was generated from the following file:

- [ad9361_manager.h](#)

10.316 stream_t Struct Reference

Public Attributes

- int **type**
- int **mode**
- int **state**
- unsigned int **inb**
- unsigned int **inr**
- unsigned int **outb**
- unsigned int **outr**
- unsigned int **tick**
- unsigned int **tact**
- unsigned int **inbt**
- unsigned int **outbt**
- lock_t **lock**
- void * **port**
- char **path** [[MAXSTRPATH](#)]
- char **msg** [[MAXSTRMSG](#)]

10.316.1 Detailed Description

Definition at line 1087 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.317 StringConverter Class Reference

Class that interprets the contents of a string and converts it into different types.

```
#include <string_converter.h>
```

Public Member Functions

- **bool convert** (const std::string &value, bool default_value)
- **int64_t convert** (const std::string &value, int64_t default_value)
- **uint64_t convert** (const std::string &value, uint64_t default_value)
- **int32_t convert** (const std::string &value, int32_t default_value)
- **uint32_t convert** (const std::string &value, uint32_t default_value)
- **int16_t convert** (const std::string &value, int16_t default_value)
- **uint16_t convert** (const std::string &value, uint16_t default_value)
- **float convert** (const std::string &value, float default_value)
- **double convert** (const std::string &value, double default_value)

10.317.1 Detailed Description

Class that interprets the contents of a string and converts it into different types.

Definition at line 32 of file string_converter.h.

The documentation for this class was generated from the following file:

- [string_converter.h](#)

10.318 Tcp_Communication Class Reference

TCP communication class.

```
#include <tcp_communication.h>
```

Public Member Functions

- int **listen_tcp_connection** (size_t d_port_, size_t d_port_ch0_)
- void **send_receive_tcp_packet_galileo_e1** (boost::array< float, NUM_TX_VARIABLES_GALILEO_E1 > buf, [Tcp_Packet_Data](#) *tcp_data_)
- void **send_receive_tcp_packet_gps_l1_ca** (boost::array< float, NUM_TX_VARIABLES_GPS_L1_CA > buf, [Tcp_Packet_Data](#) *tcp_data_)
- void **close_tcp_connection** (size_t d_port_)

10.318.1 Detailed Description

TCP communication class.

Definition at line 41 of file tcp_communication.h.

The documentation for this class was generated from the following file:

- [tcp_communication.h](#)

10.319 Tcp_Packet_Data Class Reference

Class that implements a TCP data packet.

```
#include <tcp_packet_data.h>
```

Public Attributes

- float **proc_pack_code_error**
- float **proc_pack_carr_error**
- float **proc_pack_carrier_doppler_hz**

10.319.1 Detailed Description

Class that implements a TCP data packet.

Definition at line 27 of file tcp_packet_data.h.

The documentation for this class was generated from the following file:

- [tcp_packet_data.h](#)

10.320 tcp_t Struct Reference

Public Attributes

- int **state**
- char **saddr** [256]
- int **port**
- struct sockaddr_in **addr**
- socket_t **sock**
- int **tcon**
- unsigned int **tact**
- unsigned int **tdis**

10.320.1 Detailed Description

Definition at line 1134 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.321 tcpcli_t Struct Reference

Public Attributes

- [tcp_t](#) **svr**
- int **toinact**
- int **tirecon**

10.321.1 Detailed Description

Definition at line 1154 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.322 TcpCmdInterface Class Reference

Public Member Functions

- void **run_cmd_server** (int tcp_port)
- void **set_msg_queue** (std::shared_ptr< [Concurrent_Queue](#)< pmt::pmt_t >> control_queue)
- time_t **get_utc_time** () const
gets the UTC time parsed from the last TC command issued
- std::array< float, 3 > **get_LLH** () const
gets the Latitude, Longitude and Altitude vector from the last TC command issued
- void **set_pvt** (std::shared_ptr< [PvtInterface](#) > PVT_sptr)

10.322.1 Detailed Description

Definition at line 37 of file tcp_cmd_interface.h.

10.322.2 Member Function Documentation

10.322.2.1 `get_LLH()`

```
std::array<float, 3> TcpCmdInterface::get_LLH ( ) const
```

gets the Latitude, Longitude and Altitude vector from the last TC command issued

10.322.2.2 `get_utc_time()`

```
time_t TcpCmdInterface::get_utc_time ( ) const
```

gets the UTC time parsed from the last TC command issued

The documentation for this class was generated from the following file:

- [tcp_cmd_interface.h](#)

10.323 tcpsvr_t Struct Reference

Public Attributes

- [tcp_t svr](#)
- [tcp_t cli](#) [[MAXCLI](#)]

10.323.1 Detailed Description

Definition at line 1147 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.324 tec_t Struct Reference

Public Attributes

- [gtime_t](#) **time**
- int **ndata** [3]
- double **rb**
- double **lats** [3]
- double **lons** [3]
- double **hgts** [3]
- double * **data**
- float * **rms**

10.324.1 Detailed Description

Definition at line 538 of file rtklib.h.

The documentation for this struct was generated from the following file:

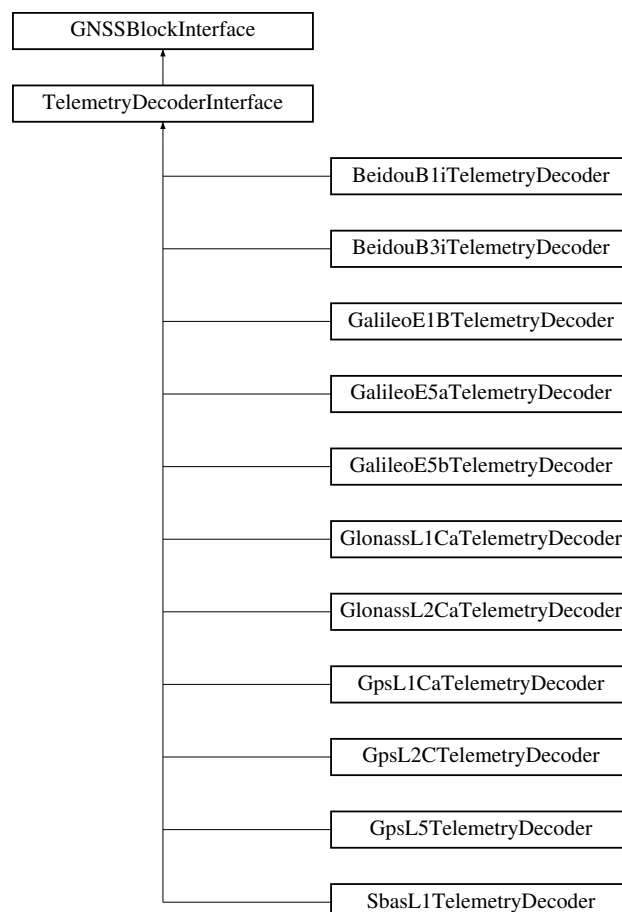
- [rtklib.h](#)

10.325 TelemetryDecoderInterface Class Reference

This abstract class represents an interface to a navigation GNSS block.

```
#include <telemetry_decoder_interface.h>
```

Inheritance diagram for TelemetryDecoderInterface:



Public Member Functions

- virtual void **reset** ()=0
- virtual void **set_satellite** (const [Gnss_Satellite](#) &sat)=0
- virtual void **set_channel** (int channel)=0

10.325.1 Detailed Description

This abstract class represents an interface to a navigation GNSS block.

Abstract class for navigation interfaces. Since all its methods are virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

Definition at line 40 of file `telemetry_decoder_interface.h`.

The documentation for this class was generated from the following file:

- [telemetry_decoder_interface.h](#)

10.326 tle_t Struct Reference

Public Attributes

- int **n**
- int **nmax**
- [tled_t](#) * **data**

10.326.1 Detailed Description

Definition at line 531 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.327 tled_t Struct Reference

Public Attributes

- char **name** [32]
- char **alias** [32]
- char **satno** [16]
- char **satclass**
- char **desig** [16]
- [gtime_t](#) **epoch**
- double **ndot**
- double **nddot**
- double **bstar**
- int **etype**
- int **eleno**
- double **inc**
- double **OMG**
- double **ecc**
- double **omg**
- double **M**
- double **n**
- int **rev**

10.327.1 Detailed Description

Definition at line 508 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.328 Tlm_Dump_Reader Class Reference

Public Member Functions

- bool **read_binary_obs** ()
- bool **restart** ()
- int64_t **num_epochs** ()
- bool **open_obs_file** (std::string out_file)

Public Attributes

- double **TOW_at_current_symbol**
- uint64_t **Tracking_sample_counter**
- double **d_TOW_at_Preamble**

10.328.1 Detailed Description

Definition at line 28 of file tlm_dump_reader.h.

The documentation for this class was generated from the following file:

- [tlm_dump_reader.h](#)

10.329 Tracking_2nd_DLL_filter Class Reference

This class implements a 2nd order DLL filter for code tracking loop.

```
#include <tracking_2nd_DLL_filter.h>
```

Public Member Functions

- **Tracking_2nd_DLL_filter** (float pdi_code)
- void **set_DLL_BW** (float dll_bw_hz)
Set DLL filter bandwidth [Hz].
- void **set_pdi** (float pdi_code)
Set Summation interval for code [s].
- void **initialize** ()
Start tracking with acquisition information.
- float **get_code_nco** (float DLL_discriminator)
Numerically controlled oscillator.

10.329.1 Detailed Description

This class implements a 2nd order DLL filter for code tracking loop.

The algorithm is described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S. H. Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007, Applied and Numerical Harmonic Analysis.

Definition at line 37 of file tracking_2nd_DLL_filter.h.

10.329.2 Member Function Documentation

10.329.2.1 get_code_nco()

```
float Tracking_2nd_DLL_filter::get_code_nco (
    float DLL_discriminator )
```

Numerically controlled oscillator.

10.329.2.2 initialize()

```
void Tracking_2nd_DLL_filter::initialize ( )
```

Start tracking with acquisition information.

10.329.2.3 set_DLL_BW()

```
void Tracking_2nd_DLL_filter::set_DLL_BW (
    float dll_bw_hz )
```

Set DLL filter bandwidth [Hz].

10.329.2.4 set_pdi()

```
void Tracking_2nd_DLL_filter::set_pdi (
    float pdi_code )
```

Set Summation interval for code [s].

The documentation for this class was generated from the following file:

- [tracking_2nd_DLL_filter.h](#)

10.330 Tracking_2nd_PLL_filter Class Reference

This class implements a 2nd order PLL filter for carrier tracking loop.

```
#include <tracking_2nd_PLL_filter.h>
```

Public Member Functions

- **Tracking_2nd_PLL_filter** (float pdi_carr)
- void **set_PLL_BW** (float pll_bw_hz)
Set PLL loop bandwidth [Hz].
- void **set_pdi** (float pdi_carr)
Set Summation interval for code [s].
- void **initialize** ()
- float **get_carrier_nco** (float PLL_discriminator)

10.330.1 Detailed Description

This class implements a 2nd order PLL filter for carrier tracking loop.

The algorithm is described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S. H. Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007, Applied and Numerical Harmonic Analysis.

Definition at line 36 of file tracking_2nd_PLL_filter.h.

10.330.2 Member Function Documentation

10.330.2.1 set_pdi()

```
void Tracking_2nd_PLL_filter::set_pdi (  
    float pdi_carr )
```

Set Summation interval for code [s].

10.330.2.2 set_PLL_BW()

```
void Tracking_2nd_PLL_filter::set_PLL_BW (  
    float pll_bw_hz )
```

Set PLL loop bandwidth [Hz].

The documentation for this class was generated from the following file:

- [tracking_2nd_PLL_filter.h](#)

10.331 Tracking_Dump_Reader Class Reference

Public Member Functions

- bool **read_binary_obs** ()
- bool **restart** ()
- int64_t **num_epochs** ()
- bool **open_obs_file** (std::string out_file)

Public Attributes

- float **abs_VE**
- float **abs_E**
- float **abs_P**
- float **abs_L**
- float **abs_VL**
- float **prompt_I**
- float **prompt_Q**
- uint64_t **PRN_start_sample_count**
- float **acc_carrier_phase_rad**
- float **carrier_doppler_hz**
- float **carrier_doppler_rate_hz_s**
- float **code_freq_chips**
- float **code_freq_rate_chips**
- float **carr_error_hz**
- float **carr_error_filt_hz**
- float **code_error_chips**
- float **code_error_filt_chips**
- float **CN0_SNV_dB_Hz**
- float **carrier_lock_test**
- float **aux1**
- double **aux2**
- unsigned int **PRN**

10.331.1 Detailed Description

Definition at line 28 of file tracking_dump_reader.h.

The documentation for this class was generated from the following file:

- [tracking_dump_reader.h](#)

10.332 Tracking_FLL_PLL_filter Class Reference

This class implements a hybrid FLL and PLL filter for tracking carrier loop.

```
#include <tracking_FLL_PLL_filter.h>
```

Public Member Functions

- void **set_params** (float fill_bw_hz, float pll_bw_hz, int order)
- void **initialize** (float d_acq_carrier_doppler_hz)
- float **get_carrier_error** (float FLL_discriminator, float PLL_discriminator, float correlation_time_s)

10.332.1 Detailed Description

This class implements a hybrid FLL and PLL filter for tracking carrier loop.

Definition at line 26 of file tracking_FLL_PLL_filter.h.

The documentation for this class was generated from the following file:

- [tracking_FLL_PLL_filter.h](#)

10.333 Tracking_loop_filter Class Reference

This class implements a generic 1st, 2nd or 3rd order loop filter.

```
#include <tracking_loop_filter.h>
```

Public Member Functions

- **Tracking_loop_filter** (float update_interval, float noise_bandwidth, int loop_order=2, bool include_last_integrator=false)
- **Tracking_loop_filter** (**Tracking_loop_filter** &&)=default
Move operator.
- **Tracking_loop_filter** & **operator=** (**Tracking_loop_filter** &&)=default
Move assignment operator.
- float **get_noise_bandwidth** () const
- float **get_update_interval** () const
- bool **get_include_last_integrator** () const
- int **get_order** () const
- void **set_noise_bandwidth** (float noise_bandwidth)
- void **set_update_interval** (float update_interval)
- void **set_include_last_integrator** (bool include_last_integrator)
- void **set_order** (int loop_order)
- void **initialize** (float initial_output=0.0)
- float **apply** (float current_input)

10.333.1 Detailed Description

This class implements a generic 1st, 2nd or 3rd order loop filter.

Definition at line 33 of file tracking_loop_filter.h.

10.333.2 Constructor & Destructor Documentation

10.333.2.1 Tracking_loop_filter()

```
Tracking_loop_filter::Tracking_loop_filter (  
    Tracking_loop_filter && ) [default]
```

Move operator.

10.333.3 Member Function Documentation

10.333.3.1 operator=()

```
Tracking_loop_filter& Tracking_loop_filter::operator= (  
    Tracking_loop_filter && ) [default]
```

Move assignment operator.

The documentation for this class was generated from the following file:

- [tracking_loop_filter.h](#)

10.334 Tracking_True_Obs_Reader Class Reference

Public Member Functions

- bool **read_binary_obs** ()
- bool **restart** ()
- int64_t **num_epochs** ()
- bool **open_obs_file** (std::string out_file)
- void **close_obs_file** ()

Public Attributes

- bool **d_dump**
- double **signal_timestamp_s**
- double **acc_carrier_phase_cycles**
- double **doppler_l1_hz**
- double **prn_delay_chips**
- double **tow**

10.334.1 Detailed Description

Definition at line 28 of file `tracking_true_obs_reader.h`.

The documentation for this class was generated from the following file:

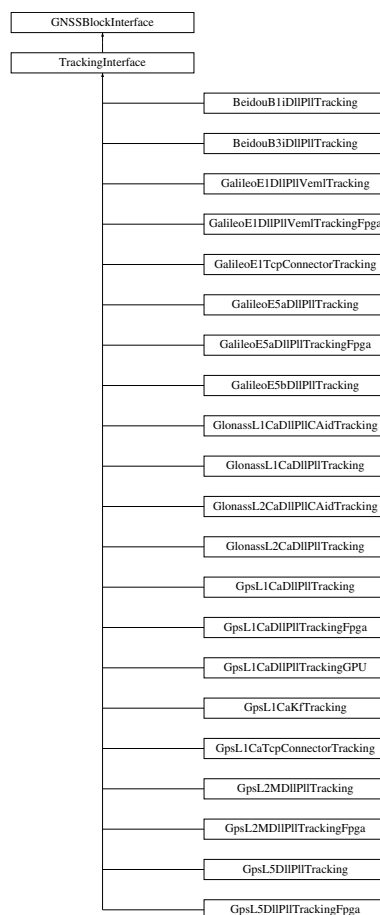
- [tracking_true_obs_reader.h](#)

10.335 TrackingInterface Class Reference

This abstract class represents an interface to a tracking block.

```
#include <tracking_interface.h>
```

Inheritance diagram for TrackingInterface:



Public Member Functions

- virtual void **start_tracking** ()=0
- virtual void **stop_tracking** ()=0
- virtual void **set_gnss_synchro** ([Gnss_Synchro](#) *gnss_synchro)=0
- virtual void **set_channel** (unsigned int channel)=0

10.335.1 Detailed Description

This abstract class represents an interface to a tracking block.

Abstract class for tracking interfaces. Since all its methods are virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

Definition at line 44 of file `tracking_interface.h`.

The documentation for this class was generated from the following file:

- [tracking_interface.h](#)

10.336 trop_t Struct Reference

Public Attributes

- [gtime_t](#) **time**
- double **trp** [3]
- float **std** [3]

10.336.1 Detailed Description

Definition at line 726 of file `rtklib.h`.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.337 True_Observables_Reader Class Reference

Public Member Functions

- bool **read_binary_obs** ()
- bool **restart** ()
- int64_t **num_epochs** ()
- bool **open_obs_file** (std::string out_file)

Public Attributes

- double **gps_time_sec** [12]
- double **doppler_l1_hz** [12]
- double **acc_carrier_phase_l1_cycles** [12]
- double **dist_m** [12]
- double **true_dist_m** [12]
- double **carrier_phase_l1_cycles** [12]
- double **prn** [12]

10.337.1 Detailed Description

Definition at line 28 of file `true_observables_reader.h`.

The documentation for this class was generated from the following file:

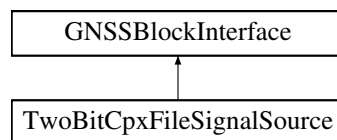
- [true_observables_reader.h](#)

10.338 TwoBitCpxFileSignalSource Class Reference

Class that reads signals samples from a file and adapts it to a `SignalSourceInterface`.

```
#include <two_bit_cpx_file_signal_source.h>
```

Inheritance diagram for `TwoBitCpxFileSignalSource`:



Public Member Functions

- **TwoBitCpxFileSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "Two_Bit_Cpx_File_Signal_Source".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- std::string **filename** () const
- std::string **item_type** () const
- bool **repeat** () const
- int64_t **sampling_frequency** () const
- uint64_t **samples** () const

10.338.1 Detailed Description

Class that reads signals samples from a file and adapts it to a `SignalSourceInterface`.

Definition at line 50 of file `two_bit_cpx_file_signal_source.h`.

10.338.2 Member Function Documentation

10.338.2.1 implementation()

```
std::string TwoBitCpxFileSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "Two_Bit_Cpx_File_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 68 of file two_bit_cpx_file_signal_source.h.

The documentation for this class was generated from the following file:

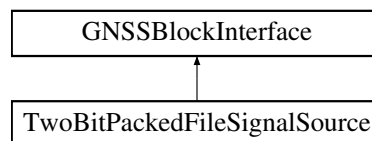
- [two_bit_cpx_file_signal_source.h](#)

10.339 TwoBitPackedFileSignalSource Class Reference

Class that reads signals samples from a file and adapts it to a SignalSourceInterface.

```
#include <two_bit_packed_file_signal_source.h>
```

Inheritance diagram for TwoBitPackedFileSignalSource:



Public Member Functions

- **TwoBitPackedFileSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_streams, unsigned int out_streams, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "Two_Bit_Packed_File_Signal_Source".
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- std::string **filename** () const
- std::string **item_type** () const
- bool **repeat** () const
- int64_t **sampling_frequency** () const
- uint64_t **samples** () const
- bool **big_endian_items** () const
- bool **big_endian_bytes** () const
- bool **is_complex** () const
- bool **reverse_interleaving** () const

10.339.1 Detailed Description

Class that reads signals samples from a file and adapts it to a SignalSourceInterface.

Definition at line 51 of file two_bit_packed_file_signal_source.h.

10.339.2 Member Function Documentation

10.339.2.1 implementation()

```
std::string TwoBitPackedFileSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "Two_Bit_Packed_File_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 67 of file two_bit_packed_file_signal_source.h.

The documentation for this class was generated from the following file:

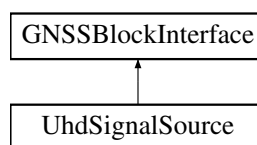
- [two_bit_packed_file_signal_source.h](#)

10.340 UhdSignalSource Class Reference

This class reads samples from a UHD device (see <http://code.ettus.com/redmine/ettus/projects/uhd/wiki>)

```
#include <uhd_signal_source.h>
```

Inheritance diagram for UhdSignalSource:



Public Member Functions

- **UhdSignalSource** (const [ConfigurationInterface](#) *configuration, const std::string &role, unsigned int in_stream, unsigned int out_stream, [Concurrent_Queue](#)< pmt::pmt_t > *queue)
- std::string **role** () override
- std::string **implementation** () override
 - Returns "UHD_Signal_Source".*
- size_t **item_size** () override
- void **connect** (gr::top_block_sptr top_block) override
- void **disconnect** (gr::top_block_sptr top_block) override
- gr::basic_block_sptr **get_left_block** () override
- gr::basic_block_sptr **get_right_block** () override
- gr::basic_block_sptr **get_right_block** (int RF_channel) override

10.340.1 Detailed Description

This class reads samples from a UHD device (see <http://code.ettus.com/redmine/ettus/projects/uhd/wiki>)

Definition at line 44 of file `uhd_signal_source.h`.

10.340.2 Member Function Documentation

10.340.2.1 implementation()

```
std::string UhdSignalSource::implementation ( ) [inline], [override], [virtual]
```

Returns "UHD_Signal_Source".

Implements [GNSSBlockInterface](#).

Definition at line 61 of file `uhd_signal_source.h`.

The documentation for this class was generated from the following file:

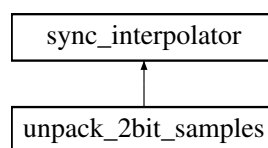
- [uhd_signal_source.h](#)

10.341 unpack_2bit_samples Class Reference

This class takes 2 bit samples that have been packed into bytes or shorts as input and generates a byte for each sample. It generates eight times as much data as is input (every two bits become 16 bits)

```
#include <unpack_2bit_samples.h>
```

Inheritance diagram for `unpack_2bit_samples`:



Public Member Functions

- **unpack_2bit_samples** (bool big_endian_bytes, size_t item_size, bool big_endian_items, bool reverse_interleaving)
- int **work** (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- unpack_2bit_samples_sptr **make_unpack_2bit_samples_sptr** (bool big_endian_bytes, size_t item_size, bool big_endian_items, bool reverse_interleaving)

10.341.1 Detailed Description

This class takes 2 bit samples that have been packed into bytes or shorts as input and generates a byte for each sample. It generates eight times as much data as is input (every two bits become 16 bits)

Definition at line 88 of file unpack_2bit_samples.h.

The documentation for this class was generated from the following file:

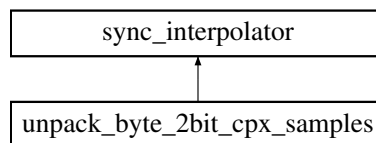
- [unpack_2bit_samples.h](#)

10.342 unpack_byte_2bit_cpx_samples Class Reference

This class implements conversion between byte packet samples to 2bit_cpx samples 1 byte = 2 x complex 2bit I, + 2bit Q samples.

```
#include <unpack_byte_2bit_cpx_samples.h>
```

Inheritance diagram for unpack_byte_2bit_cpx_samples:



Public Member Functions

- `int work (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)`

Friends

- `unpack_byte_2bit_cpx_samples_sptr make_unpack_byte_2bit_cpx_samples_sptr ()`

10.342.1 Detailed Description

This class implements conversion between byte packet samples to 2bit_cpx samples 1 byte = 2 x complex 2bit I, + 2bit Q samples.

Definition at line 48 of file unpack_byte_2bit_cpx_samples.h.

The documentation for this class was generated from the following file:

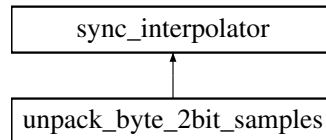
- [unpack_byte_2bit_cpx_samples.h](#)

10.343 unpack_byte_2bit_samples Class Reference

This class implements conversion between byte packet samples to 2bit samples 1 byte = 4 2bit samples.

```
#include <unpack_byte_2bit_samples.h>
```

Inheritance diagram for unpack_byte_2bit_samples:



Public Member Functions

- `int work` (`int` noutput_items, `gr_vector_const_void_star` &input_items, `gr_vector_void_star` &output_items)

Friends

- `unpack_byte_2bit_samples_sptr` **make_unpack_byte_2bit_samples_sptr** ()

10.343.1 Detailed Description

This class implements conversion between byte packet samples to 2bit samples 1 byte = 4 2bit samples.

Definition at line 44 of file `unpack_byte_2bit_samples.h`.

The documentation for this class was generated from the following file:

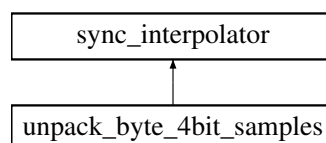
- [unpack_byte_2bit_samples.h](#)

10.344 unpack_byte_4bit_samples Class Reference

This class implements conversion between byte packet samples to 4bit_cpx samples 1 byte = 1 x complex 4bit I, + 4bit Q samples.

```
#include <unpack_byte_4bit_samples.h>
```

Inheritance diagram for unpack_byte_4bit_samples:



Public Member Functions

- int **work** (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- unpack_byte_4bit_samples_sptr **make_unpack_byte_4bit_samples_sptr** ()

10.344.1 Detailed Description

This class implements conversion between byte packet samples to 4bit_cpx samples 1 byte = 1 x complex 4bit I, + 4bit Q samples.

Definition at line 37 of file unpack_byte_4bit_samples.h.

The documentation for this class was generated from the following file:

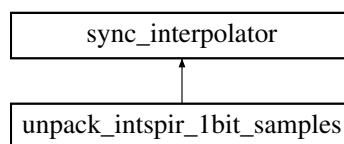
- [unpack_byte_4bit_samples.h](#)

10.345 unpack_intspir_1bit_samples Class Reference

This class implements conversion between byte packet samples to 2bit samples 1 byte = 4 2bit samples.

```
#include <unpack_intspir_1bit_samples.h>
```

Inheritance diagram for unpack_intspir_1bit_samples:



Public Member Functions

- int **work** (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- unpack_intspir_1bit_samples_sptr **make_unpack_intspir_1bit_samples_sptr** ()

10.345.1 Detailed Description

This class implements conversion between byte packet samples to 2bit samples 1 byte = 4 2bit samples.

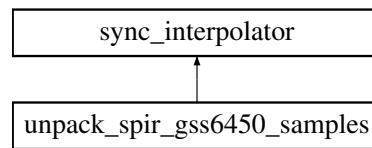
Definition at line 44 of file unpack_intspir_1bit_samples.h.

The documentation for this class was generated from the following file:

- [unpack_intspir_1bit_samples.h](#)

10.346 unpack_spir_gss6450_samples Class Reference

Inheritance diagram for unpack_spir_gss6450_samples:



Public Member Functions

- **unpack_spir_gss6450_samples** (int adc_nbit)
- void **decode_4bits_word** (uint32_t input_uint32, gr_complex *out, int adc_bits_)
- int **work** (int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items)

Friends

- unpack_spir_gss6450_samples_sptr **make_unpack_spir_gss6450_samples_sptr** (int adc_nbit)

10.346.1 Detailed Description

Definition at line 42 of file unpack_spir_gss6450_samples.h.

The documentation for this class was generated from the following file:

- [unpack_spir_gss6450_samples.h](#)

10.347 UnscentedFilter Class Reference

Public Member Functions

- **UnscentedFilter** (int nx)
- **UnscentedFilter** (const arma::vec &x_pred_0, const arma::mat &P_x_pred_0)
- void **initialize** (const arma::mat &x_pred_0, const arma::mat &P_x_pred_0)
- void **predict_sequential** (const arma::vec &x_post, const arma::mat &P_x_post, [ModelFunction](#) *transition_fcn, const arma::mat &noise_covariance)
- void **update_sequential** (const arma::vec &z_upd, const arma::vec &x_pred, const arma::mat &P_x_pred, [ModelFunction](#) *measurement_fcn, const arma::mat &noise_covariance)
- arma::mat **get_x_pred** () const
- arma::mat **get_P_x_pred** () const
- arma::mat **get_x_est** () const
- arma::mat **get_P_x_est** () const

10.347.1 Detailed Description

Definition at line 80 of file nonlinear_tracking.h.

The documentation for this class was generated from the following file:

- [nonlinear_tracking.h](#)

10.348 url_t Struct Reference

Public Attributes

- char **type** [32]
- char **path** [1024]
- char **dir** [1024]
- double **tint**

10.348.1 Detailed Description

Definition at line 901 of file rtklib.h.

The documentation for this struct was generated from the following file:

- [rtklib.h](#)

10.349 v27_decision_t Struct Reference

Public Attributes

- unsigned int **w** [2]

10.349.1 Detailed Description

Definition at line 40 of file fec.h.

The documentation for this struct was generated from the following file:

- [fec.h](#)

10.350 v27_poly_t Struct Reference

Public Attributes

- unsigned char **c0** [32]
- unsigned char **c1** [32]

10.350.1 Detailed Description

Definition at line 34 of file fec.h.

The documentation for this struct was generated from the following file:

- [fec.h](#)

10.351 v27_t Struct Reference

Public Attributes

- unsigned int **metrics1** [64]
- unsigned int **metrics2** [64]
- unsigned int * **old_metrics**
- unsigned int * **new_metrics**
- const [v27_poly_t](#) * **poly**
- [v27_decision_t](#) * **decisions**
- unsigned int **decisions_index**
- unsigned int **decisions_count**

10.351.1 Detailed Description

Definition at line 47 of file fec.h.

The documentation for this struct was generated from the following file:

- [fec.h](#)

10.352 Viterbi_Decoder Class Reference

Class that implements a Viterbi decoder.

```
#include <viterbi_decoder.h>
```

Public Member Functions

- **Viterbi_Decoder** (const int g_encoder[], const int KK, const int nn)
- void **reset** ()
- float **decode_block** (const double input_c[], int *output_u_int, const int LL)
Uses the Viterbi algorithm to perform hard-decision decoding of a convolutional code.
- float **decode_continuous** (const double sym[], const int traceback_depth, int bits[], const int nbits_requested, int &nbits_decoded)

10.352.1 Detailed Description

Class that implements a Viterbi decoder.

Definition at line 31 of file viterbi_decoder.h.

10.352.2 Member Function Documentation

10.352.2.1 decode_block()

```
float Viterbi_Decoder::decode_block (
    const double input_c[],
    int * output_u_int,
    const int LL )
```

Uses the Viterbi algorithm to perform hard-decision decoding of a convolutional code.

Parameters

in	<i>input_c[]</i>	The received signal in LLR-form. For BPSK, must be in form $r = 2*a*y/(sigma^2)$.
in	<i>LL</i>	The number of data bits to be decoded (does not include the mm zero-tail-bits)

Returns

output_u_int[] Hard decisions on the data bits (without the mm zero-tail-bits)

The documentation for this class was generated from the following file:

- [viterbi_decoder.h](#)

Chapter 11

File Documentation

11.1 `acq_conf.h` File Reference

Class that contains all the configuration parameters for generic acquisition block based on the PCPS algorithm.

```
#include "configuration_interface.h"
#include <cstdint>
#include <string>
```

Classes

- class [Acq_Conf](#)

11.1.1 Detailed Description

Class that contains all the configuration parameters for generic acquisition block based on the PCPS algorithm.

Author

Carles Fernandez, 2018. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.2 `acquisition_dump_reader.h` File Reference

Helper file for unit testing.

```
#include <cstdint>
#include <string>
#include <vector>
```

Classes

- class [Acquisition_Dump_Reader](#)

11.2.1 Detailed Description

Helper file for unit testing.

Authors

Carles Fernandez-Prades, 2017. cfernandez(at)cttc.es Antonio Ramos, 2018. antonio.ramos(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.3 acquisition_interface.h File Reference

Header file of the interface to an acquisition GNSS block.

```
#include "gnss_block_interface.h"
#include "gnss_synchro.h"
#include <memory>
```

Classes

- class [Concurrent_Queue< Data >](#)
This class implements a thread-safe std::queue.
- class [AcquisitionInterface](#)
This abstract class represents an interface to an acquisition GNSS block.

11.3.1 Detailed Description

Header file of the interface to an acquisition GNSS block.

Author

Carlos Aviles, 2010. carlos.avilesr(at)gmail.com Luis Esteve, 2011. luis(at)epsilon-formacion.com

This header file contains the interface to an abstract class for acquisition algorithms. Since all its methods are virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.4 acquisition_msg_rx.h File Reference

This is a helper class to catch the asynchronous messages emitted by an acquisition block.

```
#include <gnuradio/block.h>
#include <gnuradio/top_block.h>
#include <pmt/pmt.h>
#include <boost/shared_ptr.hpp>
```

Classes

- class [Acquisition_msg_rx](#)

Typedefs

- using **Acquisition_msg_rx_sptr** = boost::shared_ptr< [Acquisition_msg_rx](#) >

Functions

- Acquisition_msg_rx_sptr **Acquisition_msg_rx_make** ()

11.4.1 Detailed Description

This is a helper class to catch the asynchronous messages emitted by an acquisition block.

Author

Carles Fernandez-Prades, 2018. cfernandez(at)cttc.cat

Copyright (C) 2012-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.5 ad9361_fpga_signal_source.h File Reference

signal source for Analog Devices front-end AD9361 connected directly to FPGA accelerators. This source implements only the AD9361 control. It is NOT compatible with conventional SDR acquisition and tracking blocks. Please use the fmcomms2 source if conventional SDR acquisition and tracking is selected in the configuration file.

```
#include "concurrent_queue.h"
#include "fpga_dynamic_bit_selection.h"
#include "fpga_switch.h"
#include "gnss_block_interface.h"
#include <pmt/pmt.h>
#include <cstdint>
#include <memory>
#include <mutex>
#include <string>
#include <thread>
```

Classes

- class [Ad9361FpgaSignalSource](#)

11.5.1 Detailed Description

signal source for Analog Devices front-end AD9361 connected directly to FPGA accelerators. This source implements only the AD9361 control. It is NOT compatible with conventional SDR acquisition and tracking blocks. Please use the fmcomms2 source if conventional SDR acquisition and tracking is selected in the configuration file.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.6 ad9361_manager.h File Reference

An Analog Devices AD9361 front-end configuration library wrapper for configure some functions via iiod link.

```
#include <iio.h>
#include <stdint>
#include <string>
```

Classes

- struct [stream_cfg](#)

Macros

- `#define FIR_BUF_SIZE 8192`

Enumerations

- enum `iodev` { **RX**, **TX** }

Functions

- void **errchk** (int v, const char *what)
- void **wr_ch_lll** (struct iio_channel *chn, const char *what, int64_t val)
- void **wr_ch_str** (struct iio_channel *chn, const char *what, const char *str)
- struct iio_device * **get_ad9361_phy** (struct iio_context *ctx)
- bool **get_ad9361_stream_dev** (struct iio_context *ctx, enum iodev d, struct iio_device **dev)
- bool **get_ad9361_stream_ch** (struct iio_context *ctx, enum iodev d, struct iio_device *dev, int chid, struct iio_channel **chn)
- bool **get_phy_chan** (struct iio_context *ctx, enum iodev d, int chid, struct iio_channel **chn)
- bool **get_lo_chan** (struct iio_context *ctx, enum iodev d, struct iio_channel **chn)
- bool **cfg_ad9361_streaming_ch** (struct iio_context *ctx, struct [stream_cfg](#) *cfg, enum iodev type, int chid)
- bool **config_ad9361_rx_local** (uint64_t bandwidth_, uint64_t sample_rate_, uint64_t freq_, const std::string &rf_port_select_, bool rx1_enable_, bool rx2_enable_, const std::string &gain_mode_rx1_, const std::string &gain_mode_rx2_, double rf_gain_rx1_, double rf_gain_rx2_, bool quadrature_, bool rfdc_, bool bbdc_, std::string filter_source_, std::string filter_filename_, float Fpass_, float Fstop_)
- bool **config_ad9361_rx_remote** (const std::string &remote_host, uint64_t bandwidth_, uint64_t sample_rate_, uint64_t freq_, const std::string &rf_port_select_, bool rx1_enable_, bool rx2_enable_, const std::string &gain_mode_rx1_, const std::string &gain_mode_rx2_, double rf_gain_rx1_, double rf_gain_rx2_, bool quadrature_, bool rfdc_, bool bbdc_, std::string filter_source_, std::string filter_filename_, float Fpass_, float Fstop_)
- bool **config_ad9361_lo_local** (uint64_t bandwidth_, uint64_t sample_rate_, uint64_t freq_rf_tx_hz_, double tx_attenuation_db_, int64_t freq_dds_tx_hz_, double scale_dds_dbfs_, double phase_dds_deg_)
- bool **config_ad9361_lo_remote** (const std::string &remote_host, uint64_t bandwidth_, uint64_t sample_rate_, uint64_t freq_rf_tx_hz_, double tx_attenuation_db_, int64_t freq_dds_tx_hz_, double scale_dds_dbfs_, double phase_dds_deg_)
- bool **ad9361_disable_lo_remote** (const std::string &remote_host)
- bool **ad9361_disable_lo_local** ()
- bool **load_fir_filter** (std::string &filter, struct iio_device *phy)
- bool **disable_ad9361_rx_local** ()
- bool **disable_ad9361_rx_remote** (const std::string &remote_host)

11.6.1 Detailed Description

An Analog Devices AD9361 front-end configuration library wrapper for configure some functions via iiod link.

Author

Javier Arribas, jarribas(at)cttc.es

This file contains information taken from librtlsdr:

<https://git.osmocom.org/rtl-sdr>

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.7 agnss_ref_location.h File Reference

Interface of an Assisted GNSS REFERENCE LOCATION storage.

```
#include <boost/serialization/nvp.hpp>
```

Classes

- class [Agnss_Ref_Location](#)
Interface of an Assisted GNSS REFERENCE LOCATION storage.

11.7.1 Detailed Description

Interface of an Assisted GNSS REFERENCE LOCATION storage.

Author

Javier Arribas, 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.8 agnss_ref_time.h File Reference

Interface of an Assisted GNSS REFERENCE TIME storage.

```
#include <boost/serialization/nvp.hpp>
```

Classes

- class [Agnss_Ref_Time](#)
Interface of an Assisted GNSS REFERENCE TIME storage.

11.8.1 Detailed Description

Interface of an Assisted GNSS REFERENCE TIME storage.

Author

Javier Arribas, 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.9 array_signal_conditioner.h File Reference

It wraps blocks to change data type, filter and resample input data, adapted to array receiver.

```
#include "gnss_block_interface.h"
#include <gnuradio/block.h>
#include <cstdint>
#include <memory>
#include <string>
```

Classes

- class [ArraySignalConditioner](#)

This class wraps blocks to change data_type_adapter, input_filter and resampler to be applied to the input flow of sampled signal.

11.9.1 Detailed Description

It wraps blocks to change data type, filter and resample input data, adapted to array receiver.

Author

Javier Arribas jarribas (at) cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.10 bayesian_estimation.h File Reference

Interface of a library with Bayesian noise statistic estimation.

```
#include <armadillo>
#include <gnuradio/gr_complex.h>
```

Classes

- class [Bayesian_estimator](#)

Bayesian_estimator is an estimator of noise characteristics (i.e. mean, covariance)

11.10.1 Detailed Description

Interface of a library with Bayesian noise statistic estimation.

[Bayesian_estimator](#) is a Bayesian estimator which attempts to estimate the properties of a stochastic process based on a sequence of discrete samples of the sequence.

[1]: LaMountain, Gerald, Vilà-Valls, Jordi, Closas, Pau, "Bayesian Covariance Estimation for Kalman Filter based Digital Carrier Synchronization," Proceedings of the 31st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2018), Miami, Florida, September 2018, pp. 3575-3586. <https://doi.org/10.33012/2018.15911>

Authors

- Gerald LaMountain, 2018. gerald(at)ece.neu.edu
- Jordi Vila-Valls 2018. jvila(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.11 beamformer.h File Reference

Simple spatial filter using RAW array input and beamforming coefficients.

```
#include <gnuradio/sync_block.h>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [beamformer](#)

This class implements a real-time software-defined spatial filter using the CTTC GNSS experimental antenna array input and a set of dynamically reloadable weights.

Typedefs

- using **beamformer_sptr** = boost::shared_ptr< [beamformer](#) >

Functions

- beamformer_sptr **make_beamformer_sptr** ()

Variables

- const int **GNSS_SDR_BEAMFORMER_CHANNELS** = 8

11.11.1 Detailed Description

Simple spatial filter using RAW array input and beamforming coefficients.

Javier Arribas jarribas (at) cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.12 beamformer_filter.h File Reference

Interface of an adapter of a digital beamformer.

```
#include "gnss_block_interface.h"
#include <gnuradio/hier_block2.h>
#include <stdint>
#include <string>
```

Classes

- class [BeamformerFilter](#)
Interface of an adapter of a digital beamformer block to a [GNSSBlockInterface](#).

11.12.1 Detailed Description

Interface of an adapter of a digital beamformer.

Author

Javier Arribas jarribas (at) cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.13 Beidou_B1I.h File Reference

Defines system parameters for BeiDou B1I signal and DNAV data.

```
#include "gnss_frequencies.h"
#include <stdint>
```

Variables

- constexpr double **BEIDOU_B1I_FREQ_HZ** = **FREQ1_BDS**
B1I [Hz].
- constexpr double **BEIDOU_B1I_CODE_RATE_CPS** = 2.046e6
Beidou B1I code rate [chips/s].
- constexpr double **BEIDOU_B1I_CODE_LENGTH_CHIPS** = 2046.0
Beidou B1I code length [chips].
- constexpr double **BEIDOU_B1I_CODE_PERIOD_S** = 0.001
Beidou B1I code period [seconds].
- constexpr double **BEIDOU_B1I_PREAMBLE_DURATION_S** = 0.220
- constexpr uint32_t **BEIDOU_B1I_CODE_PERIOD_MS** = 1
Beidou B1I code period [ms].
- constexpr uint32_t **BEIDOU_B1I_PREAMBLE_LENGTH_BITS** = 11
- constexpr uint32_t **BEIDOU_B1I_PREAMBLE_LENGTH_SYMBOLS** = 220
- constexpr int32_t **BEIDOU_B1I_SECONDARY_CODE_LENGTH** = 20
- constexpr int32_t **BEIDOU_B1I_GEO_PREAMBLE_LENGTH_SYMBOLS** = 22
- constexpr int32_t **BEIDOU_B1I_PREAMBLE_DURATION_MS** = 220
- constexpr int32_t **BEIDOU_B1I_TELEMETRY_RATE_BITS_SECOND** = 50
- constexpr int32_t **BEIDOU_B1I_TELEMETRY_SYMBOLS_PER_BIT** = 20
- constexpr int32_t **BEIDOU_B1I_GEO_TELEMETRY_SYMBOLS_PER_BIT** = 2
- constexpr int32_t **BEIDOU_B1I_TELEMETRY_SYMBOL_PERIOD_MS** = static_cast<int32_t>(static_cast<uint32_t>(BEIDOU_B1I_TELEMETRY_SYMBOLS_PER_BIT) * **BEIDOU_B1I_CODE_PERIOD_MS**)
- constexpr int32_t **BEIDOU_B1I_TELEMETRY_RATE_SYMBOLS_SECOND** = BEIDOU_B1I_TELEMETRY_RATE_BITS_SECOND * BEIDOU_B1I_TELEMETRY_SYMBOLS_PER_BIT
- constexpr char **BEIDOU_B1I_SECONDARY_CODE_STR** [21] = "00000100110101001110"
- constexpr char **BEIDOU_B1I_GEO_PREAMBLE_SYMBOLS_STR** [23] = "1111110000001100001100"
- constexpr char **BEIDOU_B1I_D2_SECONDARY_CODE_STR** [3] = "00"

11.13.1 Detailed Description

Defines system parameters for BeiDou B1I signal and DNAV data.

Author

Sergi Segura, 2018. sergi.segura.munoz(at)gmail.com

Damian Miralles, 2018. dmiralles2009@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.13.2 Variable Documentation

11.13.2.1 BEIDOU_B1I_CODE_LENGTH_CHIPS

```
constexpr double BEIDOU_B1I_CODE_LENGTH_CHIPS = 2046.0
```

Beidou B1I code length [chips].

Definition at line 30 of file Beidou_B1I.h.

11.13.2.2 BEIDOU_B1I_CODE_PERIOD_MS

```
constexpr uint32_t BEIDOU_B1I_CODE_PERIOD_MS = 1
```

Beidou B1I code period [ms].

Definition at line 33 of file Beidou_B1I.h.

11.13.2.3 BEIDOU_B1I_CODE_PERIOD_S

```
constexpr double BEIDOU_B1I_CODE_PERIOD_S = 0.001
```

Beidou B1I code period [seconds].

Definition at line 31 of file Beidou_B1I.h.

11.13.2.4 BEIDOU_B1I_CODE_RATE_CPS

```
constexpr double BEIDOU_B1I_CODE_RATE_CPS = 2.046e6
```

Beidou B1I code rate [chips/s].

Definition at line 29 of file Beidou_B1I.h.

11.13.2.5 BEIDOU_B1I_FREQ_HZ

```
constexpr double BEIDOU_B1I_FREQ_HZ = FREQ1\_BDS
```

B1I [Hz].

Definition at line 28 of file Beidou_B1I.h.

11.14 beidou_b1i_dll_pll_tracking.h File Reference

Interface of an adapter of a DLL+PLL tracking loop block for Beidou B1I to a [TrackingInterface](#).

```
#include "dll_pll_veml_tracking.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [BeidouB1iDllPlTracking](#)
This class implements a code DLL + carrier PLL tracking loop.

11.14.1 Detailed Description

Interface of an adapter of a DLL+PLL tracking loop block for Beidou B1I to a [TrackingInterface](#).

Author

Sergi Segura, 2018. [sergi.segura.munoz\(at\)gmail.com](mailto:sergi.segura.munoz(at)gmail.com)

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.15 beidou_b1i_pcps_acquisition.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Beidou B1I signals.

```
#include "channel_fsm.h"
#include "complex_byte_to_float_x2.h"
#include "gnss_synchro.h"
#include "pcps_acquisition.h"
#include <gnuradio/blocks/float_to_complex.h>
#include <gnuradio/blocks/stream_to_vector.h>
#include <stdint>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [BeidouB1iPcpsAcquisition](#)

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

11.15.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Beidou B1I signals.

Authors

- Sergi Segura, 2018. [sergi.segura.munoz\(at\)gmail.com](mailto:sergi.segura.munoz(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.16 beidou_b1i_signal_processing.h File Reference

This class implements various functions for BeiDou B1I signals.

```
#include <complex>
#include <cstdint>
#include <gsl/gsl>
```

Functions

- void [beidou_b1i_code_gen_int](#) (own::span< int32_t > _dest, int32_t _prn, uint32_t _chip_shift)
Generates int32_t GPS L1 C/A code for the desired SV ID and code shift.
- void [beidou_b1i_code_gen_float](#) (own::span< float > _dest, int32_t _prn, uint32_t _chip_shift)
Generates float GPS L1 C/A code for the desired SV ID and code shift.
- void [beidou_b1i_code_gen_complex](#) (own::span< std::complex< float >> _dest, int32_t _prn, uint32_t _chip_shift)
Generates complex GPS L1 C/A code for the desired SV ID and code shift, and sampled to specific sampling frequency.
- void [beidou_b1i_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, uint32_t _prn, int32_t _fs, uint32_t _chip_shift, uint32_t _ncodes)
Generates N complex GPS L1 C/A codes for the desired SV ID and code shift.
- void [beidou_b1i_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, uint32_t _prn, int32_t _fs, uint32_t _chip_shift)
Generates complex GPS L1 C/A code for the desired SV ID and code shift.

11.16.1 Detailed Description

This class implements various functions for BeiDou B1I signals.

Author

Sergi Segura, 2018. sergi.segura.munoz(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.16.2 Function Documentation

11.16.2.1 beidou_b1i_code_gen_complex()

```
void beidou_b1i_code_gen_complex (
    own::span< std::complex< float >> _dest,
    int32_t _prn,
    uint32_t _chip_shift )
```

Generates complex GPS L1 C/A code for the desired SV ID and code shift, and sampled to specific sampling frequency.

11.16.2.2 beidou_b1i_code_gen_complex_sampled() [1/2]

```
void beidou_b1i_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    uint32_t _prn,
    int32_t _fs,
    uint32_t _chip_shift,
    uint32_t _ncodes )
```

Generates N complex GPS L1 C/A codes for the desired SV ID and code shift.

11.16.2.3 beidou_b1i_code_gen_complex_sampled() [2/2]

```
void beidou_b1i_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    uint32_t _prn,
    int32_t _fs,
    uint32_t _chip_shift )
```

Generates complex GPS L1 C/A code for the desired SV ID and code shift.

11.16.2.4 beidou_b1i_code_gen_float()

```
void beidou_b1i_code_gen_float (
    own::span< float > _dest,
    int32_t _prn,
    uint32_t _chip_shift )
```

Generates float GPS L1 C/A code for the desired SV ID and code shift.

11.16.2.5 beidou_b1i_code_gen_int()

```
void beidou_b1i_code_gen_int (
    own::span< int32_t > _dest,
    int32_t _prn,
    uint32_t _chip_shift )
```

Generates int32_t GPS L1 C/A code for the desired SV ID and code shift.

11.17 beidou_b1i_telemetry_decoder.h File Reference

Interface of an adapter of a Beidou B1I NAV data decoder block to a [TelemetryDecoderInterface](#).

```
#include "beidou_b1i_telemetry_decoder_gs.h"
#include "gnss_satellite.h"
#include "gnss_synchro.h"
#include "telemetry_decoder_interface.h"
#include <gnuradio/runtime_types.h>
#include <cstdint>
#include <string>
```

Classes

- class [BeidouB1iTelemetryDecoder](#)

This class implements a NAV data decoder for BEIDOU B1I.

11.17.1 Detailed Description

Interface of an adapter of a Beidou B1I NAV data decoder block to a [TelemetryDecoderInterface](#).

Author

Damian Miralles, 2018. dmiralles2009@gmail.com
 Sergi Segura, 2018. [sergi.segura.munoz\(at\)gmail.com](mailto:sergi.segura.munoz(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.18 beidou_b1i_telemetry_decoder_gs.h File Reference

Implementation of a BEIDOU B1I DNAV data decoder block.

```
#include "beidou_dnav_navigation_message.h"
#include "gnss_satellite.h"
#include <boost/circular_buffer.hpp>
#include <gnuradio/block.h>
#include <gnuradio/types.h>
#include <array>
#include <cstdint>
#include <fstream>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [beidou_b1i_telemetry_decoder_gs](#)
This class implements a block that decodes the BeiDou DNAV data.

Typedefs

- using [beidou_b1i_telemetry_decoder_gs_sptr](#) = boost::shared_ptr< [beidou_b1i_telemetry_decoder_gs](#) >

Functions

- beidou_b1i_telemetry_decoder_gs_sptr [beidou_b1i_make_telemetry_decoder_gs](#) (const [Gnss_Satellite](#) &satellite, bool dump)

11.18.1 Detailed Description

Implementation of a BEIDOU B1I DNAV data decoder block.

Code added as part of GSoC 2018 program.

Author

Damian Miralles, 2018. dmiralles2009(at)gmail.com

Sergi Segura, 2018. sergi.segura.munoz(at)gmail.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.19 Beidou_B3I.h File Reference

Defines system parameters for BeiDou B3I signal and DNAV data.

```
#include "gnss_frequencies.h"
#include <stdint>
```

Variables

- constexpr double **BEIDOU_B3I_FREQ_HZ** = **FREQ3_BDS**
BeiDou B3I [Hz].
- constexpr double **BEIDOU_B3I_CODE_RATE_CPS** = 10.23e6
BeiDou B3I code rate [chips/s].
- constexpr double **BEIDOU_B3I_CODE_LENGTH_CHIPS** = 10230.0
BeiDou B3I code length [chips].
- constexpr double **BEIDOU_B3I_CODE_PERIOD_S** = 0.001
BeiDou B3I code period [seconds].
- constexpr double **BEIDOU_B3I_PREAMBLE_DURATION_S** = 0.220
- constexpr uint32_t **BEIDOU_B3I_CODE_PERIOD_MS** = 1
BeiDou B3I code period [ms].
- constexpr uint32_t **BEIDOU_B3I_PREAMBLE_LENGTH_BITS** = 11
- constexpr uint32_t **BEIDOU_B3I_PREAMBLE_LENGTH_SYMBOLS** = 220
- constexpr int32_t **BEIDOU_B3I_SECONDARY_CODE_LENGTH** = 20
- constexpr int32_t **BEIDOU_B3I_GEO_PREAMBLE_LENGTH_SYMBOLS** = 22
- constexpr int32_t **BEIDOU_B3I_PREAMBLE_DURATION_MS** = 220
- constexpr int32_t **BEIDOU_B3I_TELEMETRY_RATE_BITS_SECOND** = 50
D1 NAV message bit rate [bits/s].
- constexpr int32_t **BEIDOU_B3I_TELEMETRY_SYMBOLS_PER_BIT** = 20
- constexpr int32_t **BEIDOU_B3I_GEO_TELEMETRY_SYMBOLS_PER_BIT** = 2
- constexpr int32_t **BEIDOU_B3I_TELEMETRY_SYMBOL_PERIOD_MS** = static_cast<int32_t>(static_cast<uint32_t>(BEIDOU_B3I_TELEMETRY_SYMBOLS_PER_BIT) * **BEIDOU_B3I_CODE_PERIOD_MS**)
- constexpr int32_t **BEIDOU_B3I_TELEMETRY_RATE_SYMBOLS_SECOND** = **BEIDOU_B3I_TELEMETRY_RATE_BITS_SECOND** * **BEIDOU_B3I_TELEMETRY_SYMBOLS_PER_BIT**
- constexpr char **BEIDOU_B3I_SECONDARY_CODE_STR** [21] = "00000100110101001110"
- constexpr char **BEIDOU_B3I_GEO_PREAMBLE_SYMBOLS_STR** [23] = "1111110000001100001100"
- constexpr char **BEIDOU_B3I_D2_SECONDARY_CODE_STR** [3] = "00"

11.19.1 Detailed Description

Defines system parameters for BeiDou B3I signal and DNAV data.

Damian Miralles, 2019. dmiralles2009@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.19.2 Variable Documentation

11.19.2.1 BEIDOU_B3I_CODE_LENGTH_CHIPS

```
constexpr double BEIDOU_B3I_CODE_LENGTH_CHIPS = 10230.0
```

BeiDou B3I code length [chips].

Definition at line 29 of file Beidou_B3I.h.

11.19.2.2 BEIDOU_B3I_CODE_PERIOD_MS

```
constexpr uint32_t BEIDOU_B3I_CODE_PERIOD_MS = 1
```

BeiDou B3I code period [ms].

Definition at line 32 of file Beidou_B3I.h.

11.19.2.3 BEIDOU_B3I_CODE_PERIOD_S

```
constexpr double BEIDOU_B3I_CODE_PERIOD_S = 0.001
```

BeiDou B3I code period [seconds].

Definition at line 30 of file Beidou_B3I.h.

11.19.2.4 BEIDOU_B3I_CODE_RATE_CPS

```
constexpr double BEIDOU_B3I_CODE_RATE_CPS = 10.23e6
```

BeiDou B3I code rate [chips/s].

Definition at line 28 of file Beidou_B3I.h.

11.19.2.5 BEIDOU_B3I_FREQ_HZ

```
constexpr double BEIDOU_B3I_FREQ_HZ = FREQ3\_BDS
```

BeiDou B3I [Hz].

Definition at line 27 of file Beidou_B3I.h.

11.19.2.6 BEIDOU_B3I_TELEMETRY_RATE_BITS_SECOND

```
constexpr int32_t BEIDOU_B3I_TELEMETRY_RATE_BITS_SECOND = 50
```

D1 NAV message bit rate [bits/s].

Definition at line 38 of file Beidou_B3I.h.

11.20 beidou_b3i_dll_pll_tracking.h File Reference

Interface of an adapter of a DLL+PLL tracking loop block for Beidou B3I to a [TrackingInterface](#).

```
#include "dll_pll_veml_tracking.h"  
#include "tracking_interface.h"  
#include <string>
```

Classes

- class [BeidouB3iDllPllTracking](#)

This class implements a code DLL + carrier PLL tracking loop.

11.20.1 Detailed Description

Interface of an adapter of a DLL+PLL tracking loop block for Beidou B3I to a [TrackingInterface](#).

Author

Damian Miralles, 2019. [dmiralles2009\(at\)gmail.com](mailto:dmiralles2009@gmail.com)

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.21 beidou_b3i_pcps_acquisition.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Beidou B3I signals.

```
#include "acq_conf.h"
#include "channel_fsm.h"
#include "complex_byte_to_float_x2.h"
#include "gnss_synchro.h"
#include "pcps_acquisition.h"
#include <gnuradio/blocks/float_to_complex.h>
#include <gnuradio/blocks/stream_to_vector.h>
#include <stdint>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [BeidouB3iPcpsAcquisition](#)

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for BeiDou B3I signals.

11.21.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Beidou B3I signals.

Author

Damian Miralles, 2019. dmiralles2009@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.22 beidou_b3i_signal_processing.h File Reference

This class implements various functions for BeiDou B3I signals.

```
#include <complex>
#include <stdint>
#include <gsl/gsl>
```

Functions

- void [beidou_b3i_code_gen_int](#) (own::span< int > _dest, int32_t _prn, uint32_t _chip_shift)
Generates int BeiDou B3I code for the desired SV ID and code shift.
- void [beidou_b3i_code_gen_float](#) (own::span< float > _dest, int32_t _prn, uint32_t _chip_shift)
Generates float BeiDou B3I code for the desired SV ID and code shift.
- void [beidou_b3i_code_gen_complex](#) (own::span< std::complex< float >> _dest, int32_t _prn, uint32_t _chip_shift)
Generates complex BeiDou B3I code for the desired SV ID and code shift, and sampled to specific sampling frequency.
- void [beidou_b3i_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, uint32_t _prn, int _fs, uint32_t _chip_shift, uint32_t _ncodes)
Generates N complex BeiDou B3I codes for the desired SV ID and code shift.
- void [beidou_b3i_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, uint32_t _prn, int _fs, uint32_t _chip_shift)
Generates complex BeiDou B3I code for the desired SV ID and code shift.

11.22.1 Detailed Description

This class implements various functions for BeiDou B3I signals.

Author

Damian Miralles, 2019. dmiralles2009@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.22.2 Function Documentation

11.22.2.1 beidou_b3i_code_gen_complex()

```
void beidou_b3i_code_gen_complex (
    own::span< std::complex< float >> _dest,
    int32_t _prn,
    uint32_t _chip_shift )
```

Generates complex BeiDou B3I code for the desired SV ID and code shift, and sampled to specific sampling frequency.

11.22.2.2 beidou_b3i_code_gen_complex_sampled() [1/2]

```
void beidou_b3i_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    uint32_t _prn,
    int _fs,
    uint32_t _chip_shift,
    uint32_t _ncodes )
```

Generates N complex BeiDou B3I codes for the desired SV ID and code shift.

11.22.2.3 beidou_b3i_code_gen_complex_sampled() [2/2]

```
void beidou_b3i_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    uint32_t _prn,
    int _fs,
    uint32_t _chip_shift )
```

Generates complex BeiDou B3I code for the desired SV ID and code shift.

11.22.2.4 beidou_b3i_code_gen_float()

```
void beidou_b3i_code_gen_float (
    own::span< float > _dest,
    int32_t _prn,
    uint32_t _chip_shift )
```

Generates float BeiDou B3I code for the desired SV ID and code shift.

11.22.2.5 beidou_b3i_code_gen_int()

```
void beidou_b3i_code_gen_int (
    own::span< int > _dest,
    int32_t _prn,
    uint32_t _chip_shift )
```

Generates int BeiDou B3I code for the desired SV ID and code shift.

11.23 beidou_b3i_telemetry_decoder.h File Reference

Interface of an adapter of a Beidou B3I NAV data decoder block to a [TelemetryDecoderInterface](#).

```
#include "beidou_b3i_telemetry_decoder_gs.h"
#include "gnss_satellite.h"
#include "gnss_synchro.h"
#include "telemetry_decoder_interface.h"
#include <gnuradio/runtime_types.h>
#include <cstdint>
#include <string>
```

Classes

- class [BeidouB3iTelemetryDecoder](#)

This class implements a NAV data decoder for BEIDOU B1I.

11.23.1 Detailed Description

Interface of an adapter of a Beidou B3I NAV data decoder block to a [TelemetryDecoderInterface](#).

Author

Damian Miralles, 2019. dmiralles2009@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.24 beidou_b3i_telemetry_decoder_gs.h File Reference

Implementation of a BEIDOU B3I DNAV data decoder block.

```
#include "beidou_dnav_navigation_message.h"
#include "gnss_satellite.h"
#include <boost/circular_buffer.hpp>
#include <gnuradio/block.h>
#include <gnuradio/types.h>
#include <array>
#include <cstdint>
#include <fstream>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [beidou_b3i_telemetry_decoder_gs](#)
This class implements a block that decodes the BeiDou DNAV data.

Typedefs

- using **beidou_b3i_telemetry_decoder_gs_sptr** = boost::shared_ptr< [beidou_b3i_telemetry_decoder_gs](#) >

Functions

- beidou_b3i_telemetry_decoder_gs_sptr **beidou_b3i_make_telemetry_decoder_gs** (const [Gnss_Satellite](#) &satellite, bool dump)

11.24.1 Detailed Description

Implementation of a BEIDOU B3I DNAV data decoder block.

Author

Damian Miralles, 2019. dmiralles2009(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.25 Beidou_DNAV.h File Reference

Defines system parameters for BeiDou DNAV data processing.

```
#include "MATH_CONSTANTS.h"
#include <stdint>
#include <utility>
#include <vector>
```

Functions

- `const std::vector< std::pair< int32_t, int32_t > > D1_PRE {{{1, 1}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_FRAID {{{16, 3}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_SOW {{{19, 8}, {31, 12}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_PNUM {{{44, 7}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_SAT_H1 {{{43, 1}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_AODC {{{44, 5}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_URAI {{{49, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_WN {{{61, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_TOC {{{74, 9}, {91, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_TGD1 {{{99, 10}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_TGD2 {{{109, 4}, {121, 6}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_ALPHA0 {{{127, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_ALPHA1 {{{135, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_ALPHA2 {{{151, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_ALPHA3 {{{159, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_BETA0 {{{167, 6}, {181, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_BETA1 {{{183, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_BETA2 {{{191, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_BETA3 {{{199, 4}, {211, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_A2 {{{215, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_A0 {{{226, 7}, {241, 17}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_A1 {{{258, 5}, {271, 17}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_AODE {{{288, 5}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_DELTA_N {{{43, 10}, {61, 6}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_CUC {{{67, 16}, {91, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_M0 {{{93, 20}, {121, 12}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_E {{{133, 10}, {151, 22}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_CUS {{{181, 18}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_CRC {{{199, 4}, {211, 14}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_CRS {{{225, 8}, {241, 10}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_SQRT_A {{{251, 12}, {271, 20}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_TOE_SF2 {{{291, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_TOE_SF3 {{{43, 10}, {61, 5}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_I0 {{{66, 17}, {91, 15}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_CIC {{{106, 7}, {121, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_OMEGA_DOT {{{132, 11}, {151, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_CIS {{{164, 9}, {181, 9}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_IDOT {{{190, 13}, {211, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_OMEGA0 {{{212, 21}, {241, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_OMEGA {{{252, 11}, {271, 21}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D1_SQRT_A_ALMANAC {{{51, 2}, {61, 22}}}`

- `const std::vector< std::pair< int32_t, int32_t > > D1_A1_ALMANAC ({91, 11})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_A0_ALMANAC ({102, 11})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_OMEGA0_ALMANAC ({121, 22}, {151, 2})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_E_ALMANAC ({153, 17})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_DELTA_I ({170, 3}, {181, 13})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_TOA ({194, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_OMEGA_DOT_ALMANAC ({202, 1}, {211, 16})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_OMEGA_ALMANAC ({227, 6}, {241, 18})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_M0_ALMANAC ({259, 4}, {271, 20})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA1 ({51, 2}, {61, 7})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA2 ({68, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA3 ({77, 6}, {91, 3})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA4 ({94, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA5 ({103, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA6 ({112, 1}, {121, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA7 ({129, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA8 ({138, 5}, {151, 4})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA9 ({155, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA10 ({164, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA11 ({181, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA12 ({190, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA13 ({199, 4}, {211, 5})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA14 ({216, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA15 ({225, 8}, {241, 1})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA16 ({242, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA17 ({251, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA18 ({260, 3}, {271, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA19 ({277, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA20 ({51, 2}, {61, 7})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA21 ({68, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA22 ({77, 6}, {91, 3})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA23 ({94, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA24 ({103, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA25 ({112, 1}, {121, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA26 ({129, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA27 ({138, 5}, {151, 4})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA28 ({155, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA29 ({164, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_HEA30 ({181, 9})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_WNA ({190, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_TOA2 ({198, 5}, {211, 3})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_A0GPS ({97, 14})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_A1GPS ({111, 2}, {121, 14})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_A0GAL ({135, 8}, {151, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_A1GAL ({157, 16})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_A0GLO ({181, 14})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_A1GLO ({195, 8}, {211, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_DELTA_T_LS ({51, 2}, {61, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_DELTA_T_LSF ({67, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_WN_LSF ({75, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_A0UTC ({91, 22}, {121, 10})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_A1UTC ({131, 12}, {151, 12})`
- `const std::vector< std::pair< int32_t, int32_t > > D1_DN ({163, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > D2_PRE ({1, 11})`
- `const std::vector< std::pair< int32_t, int32_t > > D2_FRAID ({16, 3})`

- `const std::vector< std::pair< int32_t, int32_t > > D2_SOW {{{19, 8}, {31, 12}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_PNUM {{{43, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_SAT_H1 {{{47, 1}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_AODC {{{48, 5}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_URAI {{{61, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_WN {{{65, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_TOC {{{78, 5}, {91, 12}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_TGD1 {{{103, 10}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_TGD2 {{{121, 10}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_ALPHA0 {{{47, 6}, {61, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_ALPHA1 {{{63, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_ALPHA2 {{{71, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_ALPHA3 {{{79, 4}, {91, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_BETA0 {{{95, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_BETA1 {{{103, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_BETA2 {{{111, 2}, {121, 6}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_BETA3 {{{127, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_A0 {{{101, 12}, {121, 12}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_A1_MSB {{{133, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_A1_LSB {{{47, 6}, {61, 12}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_A1 {{{279, 22}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_A2 {{{73, 10}, {91, 1}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_AODE {{{92, 5}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_DELTA_N {{{97, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_CUC_MSB {{{121, 14}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_CUC_LSB {{{47, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_CUC {{{283, 18}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_M0 {{{51, 2}, {61, 22}, {91, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_CUS {{{99, 14}, {121, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_E_MSB {{{125, 10}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_E_LSB {{{47, 6}, {61, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_SQRT_A {{{77, 6}, {91, 22}, {121, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_CIC_MSB {{{125, 10}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_CIC_LSB {{{47, 6}, {61, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_CIC {{{283, 18}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_CIS {{{63, 18}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_TOE {{{81, 2}, {91, 15}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_I0_MSB {{{106, 7}, {121, 14}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_I0_LSB {{{47, 6}, {61, 5}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_I0 {{{269, 32}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_CRC {{{66, 17}, {91, 1}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_CRS {{{92, 18}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_OMEGA_DOT_MSB {{{110, 3}, {121, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_OMEGA_DOT_LSB {{{47, 5}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_OMEGA_DOT {{{277, 24}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_OMEGA0 {{{52, 1}, {61, 22}, {91, 9}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_OMEGA_MSB {{{100, 13}, {121, 14}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_OMEGA_LSB {{{47, 5}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_OMEGA {{{269, 32}}}`
- `const std::vector< std::pair< int32_t, int32_t > > D2_IDOT {{{52, 1}, {61, 13}}}`

Variables

- constexpr double **D1_TOC_LSB** = [TWO_P3](#)
- constexpr double **D1_TGD1_LSB** = 0.1e-9
- constexpr double **D1_TGD2_LSB** = 0.1e-9
- constexpr double **D1_ALPHA0_LSB** = [TWO_N30](#)
- constexpr double **D1_ALPHA1_LSB** = [TWO_N27](#)
- constexpr double **D1_ALPHA2_LSB** = [TWO_N24](#)
- constexpr double **D1_ALPHA3_LSB** = [TWO_N24](#)
- constexpr double **D1_BETA0_LSB** = [TWO_P11](#)
- constexpr double **D1_BETA1_LSB** = [TWO_P14](#)
- constexpr double **D1_BETA2_LSB** = [TWO_P16](#)
- constexpr double **D1_BETA3_LSB** = [TWO_P16](#)
- constexpr double **D1_A2_LSB** = [TWO_N66](#)
- constexpr double **D1_A0_LSB** = [TWO_N33](#)
- constexpr double **D1_A1_LSB** = [TWO_N50](#)
- constexpr double **D1_DELTA_N_LSB** = [PI_TWO_N43](#)
- constexpr double **D1_CUC_LSB** = [TWO_N31](#)
- constexpr double **D1_M0_LSB** = [PI_TWO_N31](#)
- constexpr double **D1_E_LSB** = [TWO_N33](#)
- constexpr double **D1_CUS_LSB** = [TWO_N31](#)
- constexpr double **D1_CRC_LSB** = [TWO_N6](#)
- constexpr double **D1_CRS_LSB** = [TWO_N6](#)
- constexpr double **D1_SQRT_A_LSB** = [TWO_N19](#)
- constexpr double **D1_TOE_LSB** = [TWO_P3](#)
- constexpr double **D1_I0_LSB** = [PI_TWO_N31](#)
- constexpr double **D1_CIC_LSB** = [TWO_N31](#)
- constexpr double **D1_OMEGA_DOT_LSB** = [PI_TWO_N43](#)
- constexpr double **D1_CIS_LSB** = [TWO_N31](#)
- constexpr double **D1_IDOT_LSB** = [PI_TWO_N43](#)
- constexpr double **D1_OMEGA0_LSB** = [PI_TWO_N31](#)
- constexpr double **D1_OMEGA_LSB** = [PI_TWO_N31](#)
- constexpr double **D1_SQRT_A_ALMANAC_LSB** = [TWO_N11](#)
- constexpr double **D1_A1_ALMANAC_LSB** = [TWO_N38](#)
- constexpr double **D1_A0_ALMANAC_LSB** = [TWO_N20](#)
- constexpr double **D1_OMEGA0_ALMANAC_LSB** = [PI_TWO_N23](#)
- constexpr double **D1_E_ALMANAC_LSB** = [TWO_N21](#)
- constexpr double **D1_DELTA_I_LSB** = [PI_TWO_N19](#)
- constexpr double **D1_TOA_LSB** = [TWO_P12](#)
- constexpr double **D1_OMEGA_DOT_ALMANAC_LSB** = [PI_TWO_N38](#)
- constexpr double **D1_OMEGA_ALMANAC_LSB** = [PI_TWO_N23](#)
- constexpr double **D1_M0_ALMANAC_LSB** = [PI_TWO_N23](#)
- constexpr double **D1_A0GPS_LSB** = 0.1e-9
- constexpr double **D1_A1GPS_LSB** = 0.1e-9
- constexpr double **D1_A0GAL_LSB** = 0.1e-9
- constexpr double **D1_A1GAL_LSB** = 0.1e-9
- constexpr double **D1_A0GLO_LSB** = 0.1e-9
- constexpr double **D1_A1GLO_LSB** = 0.1e-9
- constexpr double **D1_A0UTC_LSB** = [TWO_N30](#)
- constexpr double **D1_A1UTC_LSB** = [TWO_N50](#)
- constexpr int32_t **BEIDOU_DNAV_PREAMBLE_LENGTH_BITS** = 11
- constexpr int32_t **BEIDOU_DNAV_PREAMBLE_LENGTH_SYMBOLS** = 11
- constexpr int32_t **BEIDOU_DNAV_PREAMBLE_PERIOD_SYMBOLS** = 300
- constexpr int32_t **BEIDOU_DNAV_SUBFRAME_DATA_BITS** = 300

Number of bits per subframe in the NAV message [bits].

- `constexpr int32_t BEIDOU_DNAV_BDT2GPST_LEAP_SEC_OFFSET = 14`
- `constexpr int32_t BEIDOU_DNAV_BDT2GPST_WEEK_NUM_OFFSET = 1356`
- `constexpr uint32_t BEIDOU_DNAV_SUBFRAME_SYMBOLS = 300`
- `constexpr uint32_t BEIDOU_DNAV_WORDS_SUBFRAME = 10`
- `constexpr uint32_t BEIDOU_DNAV_WORD_LENGTH_BITS = 30`
- `constexpr char BEIDOU_DNAV_PREAMBLE [12] = "11100010010"`

11.25.1 Detailed Description

Defines system parameters for BeiDou DNAV data processing.

Damian Miralles, 2018. dmiralles2009@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.25.2 Variable Documentation

11.25.2.1 BEIDOU_DNAV_SUBFRAME_DATA_BITS

```
constexpr int32_t BEIDOU_DNAV_SUBFRAME_DATA_BITS = 300
```

Number of bits per subframe in the NAV message [bits].

Definition at line 84 of file Beidou_DNAV.h.

11.26 beidou_dnav_almanac.h File Reference

Interface of a Beidou DNAV Almanac storage.

```
#include <boost/serialization/nvp.hpp>
```

Classes

- class [Beidou_Dnav_Almanac](#)

This class is a storage for the BeiDou D1 almanac.

11.26.1 Detailed Description

Interface of a Beidou DNAV Almanac storage.

Author

Sergi Segura, 2018. [sergi.segura.munoz\(at\)gmail.com](mailto:sergi.segura.munoz(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.27 beidou_dnav_ephemeris.h File Reference

Interface of a BEIDOU EPHEMERIS storage.

```
#include <boost/serialization/nvp.hpp>
#include <map>
#include <string>
```

Classes

- class [Beidou_Dnav_Ephemeris](#)

This class is a storage and orbital model functions for the GPS SV ephemeris data as described in BeiDou Navigation Satellite System Signal In Space Interface Control Document Open Service Signal B1I (Version 3.0)

11.27.1 Detailed Description

Interface of a BEIDOU EPHEMERIS storage.

Author

Sergi Segura, 2018. [sergi.segura.munoz\(at\)gmail.com](mailto:sergi.segura.munoz(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.28 beidou_dnav_iono.h File Reference

Interface of a BEIDOU IONOSPHERIC MODEL storage.

```
#include <boost/serialization/nvp.hpp>
```

Classes

- class [Beidou_Dnav_Iono](#)

This class is a storage for the BEIDOU IONOSPHERIC data as described in ICD v2.1.

11.28.1 Detailed Description

Interface of a BEIDOU IONOSPHERIC MODEL storage.

Author

Sergi Segura, 2018. [sergi.segura.munoz\(at\)gmail.com](mailto:sergi.segura.munoz(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.29 beidou_dnav_navigation_message.h File Reference

Interface of a BeiDou DNAV Data message decoder.

```
#include "Beidou_B1I.h"
#include "Beidou_B3I.h"
#include "Beidou_DNAV.h"
#include "beidou_dnav_almanac.h"
#include "beidou_dnav_ephemeris.h"
#include "beidou_dnav_iono.h"
#include "beidou_dnav_utc_model.h"
#include <bitset>
#include <cstdint>
#include <map>
#include <string>
#include <utility>
#include <vector>
```

Classes

- class [Beidou_Dnav_Navigation_Message](#)

This class decodes a BeiDou D1 NAV Data message.

11.29.1 Detailed Description

Interface of a BeiDou DNAV Data message decoder.

Author

Sergi Segura, 2018. [sergi.segura.munoz\(at\)gmail.com](mailto:sergi.segura.munoz(at)gmail.com)
Damian Miralles, 2018. dmiralles2009@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.30 beidou_dnav_utc_model.h File Reference

Interface of a BeiDou UTC MODEL storage.

```
#include <boost/serialization/nvp.hpp>
```

Classes

- class [Beidou_Dnav_Utc_Model](#)
This class is a storage for the BeiDou DNAV UTC Model.

11.30.1 Detailed Description

Interface of a BeiDou UTC MODEL storage.

Author

Damian Miralles, 2018. dmiralles2009@gmail.com
Sergi Segura, 2018. [sergi.segura.munoz\(at\)gmail.com](mailto:sergi.segura.munoz(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.31 bits.h File Reference

Utilities for bit manipulation of the libswiftnav library.

```
#include "swift_common.h"
```

Functions

- `uint8_t parity` (`uint32_t x`)
- `uint32_t getbitu` (`const uint8_t *buff`, `uint32_t pos`, `uint8_t len`)
- `int32_t getbits` (`const uint8_t *buff`, `uint32_t pos`, `uint8_t len`)
- `void setbitu` (`uint8_t *buff`, `uint32_t pos`, `uint32_t len`, `uint32_t data`)
- `void setbits` (`uint8_t *buff`, `uint32_t pos`, `uint32_t len`, `int32_t data`)
- `void bitcopy` (`void *dst`, `uint32_t dst_index`, `const void *src`, `uint32_t src_index`, `uint32_t count`)
- `void bitshl` (`void *buf`, `uint32_t size`, `uint32_t shift`)
- `uint8_t count_bits_u64` (`uint64_t v`, `uint8_t bv`)
- `uint8_t count_bits_u32` (`uint32_t v`, `uint8_t bv`)
- `uint8_t count_bits_u16` (`uint16_t v`, `uint8_t bv`)
- `uint8_t count_bits_u8` (`uint8_t v`, `uint8_t bv`)

11.31.1 Detailed Description

Utilities for bit manipulation of the libswiftnav library.

Author

Fergus Noble fergus@swift-nav.com

This file was originally borrowed from libswiftnav <https://github.com/swift-nav/libswiftnav>, a portable C library implementing GNSS related functions and algorithms, and then modified by J. Arribas and C. Fernandez

Copyright (C) 2013, 2016 Swift Navigation Inc. Contact: Fergus Noble fergus@swift-nav.com

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: LGPL-3.0-only

11.32 byte_to_short.h File Reference

Adapts an 8-bits sample stream (IF) to a short int stream (IF)

```
#include "gnss_block_interface.h"
#include <gnuradio/blocks/char_to_short.h>
#include <gnuradio/blocks/file_sink.h>
#include <cstdint>
#include <string>
```

Classes

- class [ByteToShort](#)
Adapts an 8-bits sample stream (IF) to a short int stream (IF)

11.32.1 Detailed Description

Adapts an 8-bits sample stream (IF) to a short int stream (IF)

Author

Carles Fernandez Prades, cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.33 byte_x2_to_complex_byte.h File Reference

Adapts two signed char streams into a `std::complex<signed char>` stream.

```
#include <boost/shared_ptr.hpp>
#include <gnuradio/sync_block.h>
#include <gnuradio/types.h>
```

Classes

- class [byte_x2_to_complex_byte](#)
This class adapts two signed char streams into a `std::complex<signed char>` stream.

Typedefs

- using `byte_x2_to_complex_byte_sptr` = `boost::shared_ptr< byte_x2_to_complex_byte >`

Functions

- `byte_x2_to_complex_byte_sptr` `make_byte_x2_to_complex_byte ()`

11.33.1 Detailed Description

Adapts two signed char streams into a `std::complex<signed char>` stream.

Author

Carles Fernandez Prades, cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.34 channel.h File Reference

Interface of a GNSS channel.

```
#include "channel_fsm.h"
#include "channel_interface.h"
#include "channel_msg_receiver_cc.h"
#include "concurrent_queue.h"
#include "gnss_signal.h"
#include "gnss_synchro.h"
#include <gnuradio/block.h>
#include <pmt/pmt.h>
#include <cstdint>
#include <cstddef>
#include <memory>
#include <mutex>
#include <string>
```

Classes

- class [Channel](#)

This class represents a GNSS channel. It wraps an [AcquisitionInterface](#), a [Tracking Interface](#) and a [TelemetryDecoderInterface](#), and handles their interaction through a Finite State Machine.

11.34.1 Detailed Description

Interface of a GNSS channel.

Author

Carlos Aviles, 2010. carlos.avilesr(at)gmail.com Luis Esteve, 2011. luis(at)epsilon-formacion.com

It holds blocks for acquisition, tracking, navigation data extraction and pseudorange calculation.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.35 channel_event.h File Reference

Class that defines a channel event.

```
#include <memory>
```

Classes

- class [Channel_Event](#)

Typedefs

- using **channel_event_sptr** = std::shared_ptr< [Channel_Event](#) >

Functions

- channel_event_sptr **channel_event_make** (int channel_id, int event_type)

11.35.1 Detailed Description

Class that defines a channel event.

Author

Javier Arribas, 2019. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.36 channel_fsm.h File Reference

Interface of the State Machine for channel.

```
#include "acquisition_interface.h"
#include "concurrent_queue.h"
#include "telemetry_decoder_interface.h"
#include "tracking_interface.h"
#include <pmt/pmt.h>
#include <cstdint>
#include <memory>
#include <mutex>
```

Classes

- class [ChannelFsm](#)

This class implements a State Machine for channel.

11.36.1 Detailed Description

Interface of the State Machine for channel.

Authors

Javier Arribas, 2019. javiarribas@gmail.com Antonio Ramos, 2017. [antonio.ramos\(at\)cttc.es](mailto:antonio.ramos(at)cttc.es) Luis Esteve, 2011. [luis\(at\)epsilon-formacion.com](mailto:luis(at)epsilon-formacion.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.37 channel_interface.h File Reference

This class represents an interface to a channel GNSS block.

```
#include "gnss_block_interface.h"
#include "gnss_signal.h"
```

Classes

- class [ChannelInterface](#)

This abstract class represents an interface to a channel GNSS block.

11.37.1 Detailed Description

This class represents an interface to a channel GNSS block.

Author

Carlos Aviles, 2010. [carlos.avilesr\(at\)googlemail.com](mailto:carlos.avilesr(at)googlemail.com) Luis Esteve, 2011. [luis\(at\)epsilon-formacion.com](mailto:luis(at)epsilon-formacion.com)

Abstract class for channel blocks. Since all its methods are virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.38 channel_msg_receiver_cc.h File Reference

GNU Radio block that receives asynchronous channel messages from acquisition and tracking blocks.

```
#include "channel_fsm.h"
#include <gnuradio/block.h>
#include <pmt/pmt.h>
#include <memory>
#include <boost/shared_ptr.hpp>
```

Classes

- class [channel_msg_receiver_cc](#)

GNU Radio block that receives asynchronous channel messages from acquisition and tracking blocks.

Typedefs

- using **channel_msg_receiver_cc_sptr** = boost::shared_ptr< [channel_msg_receiver_cc](#) >

Functions

- **channel_msg_receiver_cc_sptr channel_msg_receiver_make_cc** (std::shared_ptr< [ChannelFsm](#) > channel_fsm, bool repeat)

11.38.1 Detailed Description

GNU Radio block that receives asynchronous channel messages from acquisition and tracking blocks.

Author

Javier Arribas, 2016. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.39 channel_status_msg_receiver.h File Reference

GNU Radio block that receives asynchronous channel messages from acquisition and tracking blocks.

```
#include "gnss_synchro.h"
#include "monitor_pvt.h"
#include <gnuradio/block.h>
#include <pmt/pmt.h>
#include <map>
#include <memory>
#include <boost/shared_ptr.hpp>
```


Classes

- class [channel_status_msg_receiver](#)
GNU Radio block that receives asynchronous channel messages from tlm blocks.

Typedefs

- using **channel_status_msg_receiver_sptr** = boost::shared_ptr< [channel_status_msg_receiver](#) >

Functions

- `channel_status_msg_receiver_sptr channel_status_msg_receiver_make ()`

11.39.1 Detailed Description

GNU Radio block that receives asynchronous channel messages from acquisition and tracking blocks.

Author

Javier Arribas, 2019. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.40 cIFFT.h File Reference

FFT in OpenCL.

```
#include <stdio.h>
#include <CL/cl.h>
```

Classes

- struct [cIFFT_Dim3](#)
- struct [cIFFT_SplitComplex](#)
- struct [cIFFT_Complex](#)

Typedefs

- typedef void * **cIFFT_Plan**

Enumerations

- enum **clFFT_Direction** { **clFFT_Forward** = -1, **clFFT_Inverse** = 1 }
- enum **clFFT_Dimension** { **clFFT_1D** = 0, **clFFT_2D** = 1, **clFFT_3D** = 3 }
- enum **clFFT_DataFormat** { **clFFT_SplitComplexFormat** = 0, **clFFT_InterleavedComplexFormat** = 1 }

Functions

- **clFFT_Plan clFFT_CreatePlan** (cl_context context, [clFFT_Dim3](#) n, clFFT_Dimension dim, clFFT_DataFormat dataFormat, cl_int *error_code)
- void **clFFT_DestroyPlan** (clFFT_Plan plan)
- cl_int **clFFT_ExecuteInterleaved** (cl_command_queue queue, clFFT_Plan plan, cl_int batchSize, clFFT_Direction dir, cl_mem data_in, cl_mem data_out, cl_int num_events, cl_event *event_list, cl_event *event)
- cl_int **clFFT_ExecutePlannar** (cl_command_queue queue, clFFT_Plan plan, cl_int batchSize, clFFT_Direction dir, cl_mem data_in_real, cl_mem data_in_imag, cl_mem data_out_real, cl_mem data_out_imag, cl_int num_events, cl_event *event_list, cl_event *event)
- cl_int **clFFT_1DTwistInterleaved** (clFFT_Plan Plan, cl_command_queue queue, cl_mem array, size_t numRows, size_t numCols, size_t startRow, size_t rowsToProcess, clFFT_Direction dir)
- cl_int **clFFT_1DTwistPlannar** (clFFT_Plan Plan, cl_command_queue queue, cl_mem array_real, cl_mem array_imag, size_t numRows, size_t numCols, size_t startRow, size_t rowsToProcess, clFFT_Direction dir)
- void **clFFT_DumpPlan** (clFFT_Plan plan, FILE *file)

11.40.1 Detailed Description

FFT in OpenCL.

Version: <1.0>

Copyright (C) 2008 Apple Inc. All Rights Reserved. SPDX-License-Identifier: LicenseRef-Apple-Permissive

11.41 cnav_msg.h File Reference

Utilities for CNAV message manipulation of the libswiftnav library.

```
#include "fec.h"
#include "swift_common.h"
#include <limits.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdlib.h>
```

Classes

- struct [cnav_msg_t](#)
- struct [cnav_v27_part_t](#)
- struct [cnav_msg_decoder_t](#)

Macros

- `#define GPS_L2_V27_HISTORY_LENGTH_BITS` 64
- `#define GPS_L2C_V27_INIT_BITS` (32)
- `#define GPS_L2C_V27_DECODE_BITS` (32)
- `#define GPS_L2C_V27_DELAY_BITS` (32)

Functions

- `const v27_poly_t * cnav_msg_decoder_get_poly` (void)
- `void cnav_msg_decoder_init` (cnav_msg_decoder_t *dec)
- `bool cnav_msg_decoder_add_symbol` (cnav_msg_decoder_t *dec, unsigned char symbol, cnav_msg_t *msg, uint32_t *delay)

11.41.1 Detailed Description

Utilities for CNAV message manipulation of the libswiftnav library.

Author

Valeri Atamaniouk valeri@swift-nav.com

This file was originally borrowed from libswiftnav <https://github.com/swift-nav/libswiftnav>, a portable C library implementing GNSS related functions and algorithms, and then modified by J. Arribas and C. Fernandez

Copyright (C) 2016 Swift Navigation Inc. Contact: Valeri Atamaniouk valeri@swift-nav.com

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: LGPL-3.0-only

11.42 command_event.h File Reference

Class that defines a receiver command event.

```
#include <memory>
```

Classes

- class [Command_Event](#)

Typedefs

- using `command_event_sptr` = std::shared_ptr< [Command_Event](#) >

Functions

- `command_event_sptr command_event_make (int command_id, int event_type)`

11.42.1 Detailed Description

Class that defines a receiver command event.

Author

Javier Arribas, 2019. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.43 `complex_byte_to_float_x2.h` File Reference

Adapts a `std::complex<signed char>` stream into two 16-bits (short) streams.

```
#include <boost/shared_ptr.hpp>
#include <gnuradio/sync_block.h>
#include <gnuradio/types.h>
```

Classes

- class `complex_byte_to_float_x2`
This class adapts a `std::complex<signed char>` stream into two 16-bits (short) streams.

Typedefs

- using `complex_byte_to_float_x2_sptr` = `boost::shared_ptr< complex_byte_to_float_x2 >`

Functions

- `complex_byte_to_float_x2_sptr make_complex_byte_to_float_x2 ()`

11.43.1 Detailed Description

Adapts a `std::complex<signed char>` stream into two 16-bits (short) streams.

Author

Carles Fernandez Prades, cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.44 complex_float_to_complex_byte.h File Reference

Adapts a `gr_complex` stream into a `std::complex<signed char>` stream.

```
#include <boost/shared_ptr.hpp>
#include <gnuradio/sync_block.h>
#include <gnuradio/types.h>
```

Classes

- class [complex_float_to_complex_byte](#)
This class adapts a `gr_complex` stream into a `std::complex<signed char>` stream.

Typedefs

- using `complex_float_to_complex_byte_sptr` = `boost::shared_ptr< complex_float_to_complex_byte >`

Functions

- `complex_float_to_complex_byte_sptr` `make_complex_float_to_complex_byte ()`

11.44.1 Detailed Description

Adapts a `gr_complex` stream into a `std::complex<signed char>` stream.

Author

Carles Fernandez Prades, cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.45 concurrent_map.h File Reference

Interface of a thread-safe std::map.

```
#include <map>
#include <mutex>
#include <utility>
```

Classes

- class [Concurrent_Map< Data >](#)

This class implements a thread-safe std::map.

11.45.1 Detailed Description

Interface of a thread-safe std::map.

Author

Javier Arribas, 2011. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.46 concurrent_queue.h File Reference

Interface of a thread-safe std::queue.

```
#include <chrono>
#include <condition_variable>
#include <mutex>
#include <queue>
#include <thread>
```

Classes

- class [Concurrent_Queue< Data >](#)

This class implements a thread-safe std::queue.

11.46.1 Detailed Description

Interface of a thread-safe `std::queue`.

Author

Javier Arribas, 2011. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.47 configuration_interface.h File Reference

This class represents an interface to configuration parameters.

```
#include <cstdint>
#include <string>
```

Classes

- class [ConfigurationInterface](#)

This abstract class represents an interface to configuration parameters.

11.47.1 Detailed Description

This class represents an interface to configuration parameters.

Author

Carlos Aviles, 2010. carlos.avilesr(at)gmail.com

The interface defines an accessor method that gets a parameter name as input and returns the value of this parameter, a string, as output. Property names are defined here. This is an abstract class for interfaces.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.48 conjugate_cc.h File Reference

Conjugate a stream of `gr_complex`.

```
#include <boost/shared_ptr.hpp>
#include <gnuradio/sync_block.h>
#include <gnuradio/types.h>
```

Classes

- class [conjugate_cc](#)
This class adapts a `std::complex<short>` stream into two 32-bits (float) streams.

Typedefs

- using **conjugate_cc_sptr** = boost::shared_ptr< [conjugate_cc](#) >

Functions

- conjugate_cc_sptr **make_conjugate_cc** ()

11.48.1 Detailed Description

Conjugate a stream of `gr_complex`.

Author

Carles Fernandez Prades, cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.49 conjugate_ic.h File Reference

Conjugate a stream of `lv_8sc_t` (`std::complex<char>`)

```
#include <gnuradio/sync_block.h>
#include <gnuradio/types.h>
#include <boost/shared_ptr.hpp>
```


Classes

- class [conjugate_ic](#)
This class adapts a `std::complex<short>` stream into two 32-bits (float) streams.

Typedefs

- using **conjugate_ic_sptr** = boost::shared_ptr< [conjugate_ic](#) >

Functions

- [conjugate_ic_sptr](#) **make_conjugate_ic** ()

11.49.1 Detailed Description

Conjugate a stream of `lv_8sc_t (std::complex<char>)`

Author

Carles Fernandez Prades, cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.50 conjugate_sc.h File Reference

Conjugate a stream of `lv_16sc_t (std::complex<short>)`

```
#include <boost/shared_ptr.hpp>
#include <gnuradio/sync_block.h>
#include <gnuradio/types.h>
```

Classes

- class [conjugate_sc](#)
This class adapts a `std::complex<short>` stream into two 32-bits (float) streams.

Typedefs

- using **conjugate_sc_sptr** = boost::shared_ptr< [conjugate_sc](#) >

Functions

- conjugate_sc_sptr **make_conjugate_sc** ()

11.50.1 Detailed Description

Conjugate a stream of lv_16sc_t (std::complex<short>)

Author

Carles Fernandez Prades, cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.51 control_thread.h File Reference

Interface of the receiver control plane.

```
#include "agnss_ref_location.h"
#include "agnss_ref_time.h"
#include "channel_event.h"
#include "command_event.h"
#include "concurrent_queue.h"
#include "gnss_sdr_supl_client.h"
#include "tcp_cmd_interface.h"
#include <pmt/pmt.h>
#include <array>
#include <cstdint>
#include <memory>
#include <string>
#include <thread>
#include <typeinfo>
#include <utility>
#include <vector>
```

Classes

- class [ControlThread](#)

This class represents the main thread of the application, so the name is [ControlThread](#). This is the GNSS Receiver Control Plane: it connects the flowgraph, starts running it, and while it does not stop, reads the control messages generated by the blocks, processes them, and applies the corresponding actions.

11.51.1 Detailed Description

Interface of the receiver control plane.

Author

Carlos Aviles, 2010. carlos.avilesr(at)googlemail.com

GNSS Receiver Control Plane: connects the flowgraph, starts running it, and while it does not stop, reads the control messages generated by the blocks, processes them, and applies the corresponding actions.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.52 convolutional.h File Reference

General functions used to implement convolutional encoding.

```
#include <volk_gnssssdr/volk_gnssssdr.h>
#include <vector>
```

Functions

- int [parity_counter](#) (int symbol, int length)
Determines if a symbol has odd (1) or even (0) parity Output parameters:
- int [nsc_enc_bit](#) (int state_out_p[], int input, int state_in, const int g[], int KK, int nn)
Convolutionally encodes a single bit using a rate 1/n encoder. Takes in one input bit at a time, and produces a n-bit output.
- void [nsc_transit](#) (int output_p[], int trans_p[], int input, int g[], int KK, int nn)
Function that creates the transit and output vectors.
- float [Gamma](#) (const float rec_array[], int symbol, int nn)
Computes the branch metric used for decoding.
- void [Viterbi](#) (int output_u_int[], const int out0[], const int state0[], const int out1[], const int state1[], const float input_c[], int KK, int nn, int LL)
Uses the Viterbi algorithm to perform hard-decision decoding of a convolutional code.

Variables

- const float **MAXLOG** = 1e7

11.52.1 Detailed Description

General functions used to implement convolutional encoding.

Author

Matthew C. Valenti, 2006-2008.
C. Fernandez-Prades, 2019.

Copyright (C) 2006-2008 Matthew C. Valenti Copyright (C) 2019 C. Fernandez-Prades

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

This file is a derived work of the original file, which had this note:

The functions in this file are part of the Iterative Solutions Coded Modulation Library. The Iterative Solutions Coded Modulation Library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

11.52.2 Function Documentation

11.52.2.1 Gamma()

```
float Gamma (
    const float rec_array[],
    int symbol,
    int nn ) [inline]
```

Computes the branch metric used for decoding.

Returns

(returned float) The metric between the hypothetical symbol and the received vector

Parameters

in	<i>rec_array</i>	The received vector, of length nn
in	<i>symbol</i>	The hypothetical symbol
in	<i>nn</i>	The length of the received vector

Definition at line 145 of file convolutional.h.

11.52.2.2 nsc_enc_bit()

```
int nsc_enc_bit (
    int state_out_p[],
    int input,
    int state_in,
    const int g[],
    int KK,
    int nn ) [inline]
```

Convolutionally encodes a single bit using a rate 1/n encoder. Takes in one input bit at a time, and produces a n-bit output.

Parameters

in	<i>input</i>	The input data bit (i.e. a 0 or 1).
in	<i>state_in</i>	The starting state of the encoder (an int from 0 to 2^m-1).
in	<i>g[]</i>	An n-element vector containing the code generators in binary form.
in	<i>KK</i>	The constraint length of the convolutional code.
out	<i>output_p[]</i>	An n-element vector containing the encoded bits.
out	<i>state_out_p[]</i>	An integer containing the final state of the encoder (i.e. the state after encoding this bit)

This function is used by [nsc_transit\(\)](#)

Definition at line 86 of file convolutional.h.

References [parity_counter\(\)](#).

Referenced by [nsc_transit\(\)](#).

11.52.2.3 nsc_transit()

```
void nsc_transit (
    int output_p[],
    int trans_p[],
    int input,
    int g[],
    int KK,
    int nn ) [inline]
```

Function that creates the transit and output vectors.

Definition at line 116 of file convolutional.h.

References [nsc_enc_bit\(\)](#).

11.52.2.4 parity_counter()

```
int parity_counter (
    int symbol,
    int length ) [inline]
```

Determines if a symbol has odd (1) or even (0) parity Output parameters:

Returns

(returned int): The symbol's parity = 1 for odd and 0 for even

Parameters

in	<i>symbol</i>	The integer-valued symbol
in	<i>length</i>	The highest bit position in the symbol

This function is used by [nsc_enc_bit\(\)](#), [rsc_enc_bit\(\)](#), and [rsc_tail\(\)](#)

Definition at line 58 of file convolutional.h.

Referenced by [nsc_enc_bit\(\)](#).

11.52.2.5 Viterbi()

```
void Viterbi (
    int output_u_int[],
    const int out0[],
    const int state0[],
    const int out1[],
    const int state1[],
    const float input_c[],
    int KK,
    int nn,
    int LL ) [inline]
```

Uses the Viterbi algorithm to perform hard-decision decoding of a convolutional code.

Parameters

in	<i>out0[]</i>	The output bits for each state if input is a 0.
in	<i>state0[]</i>	The next state if input is a 0.
in	<i>out1[]</i>	The output bits for each state if input is a 1.
in	<i>state1[]</i>	The next state if input is a 1.
in	<i>r[]</i>	The received signal in LLR-form. For BPSK, must be in form $r = 2*a*y/(\sigma^2)$.
in	<i>KK</i>	The constraint length of the convolutional code.
in	<i>LL</i>	The number of data bits.
out	<i>output_u_int[]</i>	Hard decisions on the data bits

Definition at line 177 of file `convolutional.h`.

11.53 `cpu_multicorrelator.h` File Reference

High optimized CPU vector multiTAP correlator class.

```
#include <complex>
```

Classes

- class [Cpu_Multicorrelator](#)
Class that implements carrier wipe-off and correlators.

11.53.1 Detailed Description

High optimized CPU vector multiTAP correlator class.

Authors

- Javier Arribas, 2015. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)

Class that implements a high optimized vector multiTAP correlator class for CPUs

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.54 `cpu_multicorrelator_16sc.h` File Reference

Highly optimized CPU vector multiTAP correlator class for `lv_16sc_t` (short int complex)

```
#include <volk_gnssdr/volk_gnssdr.h>
```

Classes

- class [Cpu_Multicorrelator_16sc](#)
Class that implements carrier wipe-off and correlators.

11.54.1 Detailed Description

Highly optimized CPU vector multiTAP correlator class for `lv_16sc_t` (short int complex)

Authors

- Javier Arribas, 2016. jarribas(at)cttc.es

Class that implements a highly optimized vector multiTAP correlator class for CPUs

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.55 `cpu_multicorrelator_real_codes.h` File Reference

Highly optimized CPU vector multiTAP correlator class using real-valued local codes.

```
#include <complex>
```

Classes

- class [Cpu_Multicorrelator_Real_Codes](#)
Class that implements carrier wipe-off and correlators.

11.55.1 Detailed Description

Highly optimized CPU vector multiTAP correlator class using real-valued local codes.

Authors

- Javier Arribas, 2015. jarribas(at)cttc.es
- Cillian O'Driscoll, 2017, cillian.odriscoll(at)gmail.com

Class that implements a highly optimized vector multiTAP correlator class for CPUs

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.56 `cshort_to_float_x2.h` File Reference

Adapts a `std::complex<short>` stream into two float streams.

```
#include <boost/shared_ptr.hpp>
#include <gnuradio/sync_block.h>
#include <gnuradio/types.h>
```

Classes

- class [`cshort_to_float_x2`](#)

This class adapts a `std::complex<short>` stream into two 32-bits (float) streams.

Typedefs

- using `cshort_to_float_x2_sptr` = `boost::shared_ptr< cshort_to_float_x2 >`

Functions

- `cshort_to_float_x2_sptr` `make_cshort_to_float_x2` ()

11.56.1 Detailed Description

Adapts a `std::complex<short>` stream into two float streams.

Author

Carles Fernandez Prades, [cfernandez\(at\)cttc.es](mailto:cfernandez(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.57 `cuda_multicorrelator.h` File Reference

Highly optimized CUDA GPU vector multiTAP correlator class.

```
#include <complex>
#include <cuda.h>
#include <cuda_runtime.h>
```

Classes

- struct [GPU_Complex](#)
- struct [GPU_Complex_Short](#)
- class [cuda_multicorrelator](#)

Class that implements carrier wipe-off and correlators using NVIDIA CUDA GPU accelerators.

11.57.1 Detailed Description

Highly optimized CUDA GPU vector multiTAP correlator class.

Authors

- Javier Arribas, 2015. jarribas(at)cttc.es

Class that implements a highly optimized vector multiTAP correlator class for NVIDIA CUDA GPUs

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.58 custom_udp_signal_source.h File Reference

Receives ip frames containing samples in UDP frame encapsulation using a high performance packet capture library (libpcap)

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include "gr_complex_ip_packet_source.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/null_sink.h>
#include <pmt/pmt.h>
#include <memory>
#include <stdexcept>
#include <string>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [CustomUDPSignalSource](#)

This class reads from UDP packets, which streams interleaved I/Q samples over a network.

11.58.1 Detailed Description

Receives ip frames containing samples in UDP frame encapsulation using a high performance packet capture library (libpcap)

Javier Arribas jarribas (at) cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.59 direct_resampler_conditioner.h File Reference

Interface of an adapter of a direct resampler conditioner block to a SignalConditionerInterface.

```
#include "gnss_block_interface.h"
#include <gnuradio/hier_block2.h>
#include <string>
```

Classes

- class [DirectResamplerConditioner](#)
Interface of an adapter of a direct resampler conditioner block to a SignalConditionerInterface.

11.59.1 Detailed Description

Interface of an adapter of a direct resampler conditioner block to a SignalConditionerInterface.

Author

Carlos Aviles, 2010. carlos.avilesr(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.60 `direct_resampler_conditioner_cb.h` File Reference

Nearest neighborhood resampler with `std::complex<signed char>` input and `std::complex<signed char>` output.

```
#include <gnuradio/block.h>
#include <stdint>
#include <boost/shared_ptr.hpp>
```

Classes

- class [direct_resampler_conditioner_cb](#)
This class implements a direct resampler conditioner for `std::complex<signed char>`

Typedefs

- using `direct_resampler_conditioner_cb_sptr` = `boost::shared_ptr< direct_resampler_conditioner_cb >`

Functions

- `direct_resampler_conditioner_cb_sptr` `direct_resampler_make_conditioner_cb` (double sample_freq_in, double sample_freq_out)

11.60.1 Detailed Description

Nearest neighborhood resampler with `std::complex<signed char>` input and `std::complex<signed char>` output.

Author

Luis Esteve, 2011. [luis\(at\)epsilon-formacion.com](mailto:luis(at)epsilon-formacion.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.61 `direct_resampler_conditioner_cc.h` File Reference

Nearest neighborhood resampler with `gr_complex` input and `gr_complex` output.

```
#include <gnuradio/block.h>
#include <stdint>
#include <boost/shared_ptr.hpp>
```

Classes

- class [direct_resampler_conditioner_cc](#)

This class implements a direct resampler conditioner for complex data.

Typedefs

- using **direct_resampler_conditioner_cc_sptr** = boost::shared_ptr< [direct_resampler_conditioner_cc](#) >

Functions

- `direct_resampler_conditioner_cc_sptr` **direct_resampler_make_conditioner_cc** (double sample_freq_in, double sample_freq_out)

11.61.1 Detailed Description

Nearest neighborhood resampler with `gr_complex` input and `gr_complex` output.

Author

Luis Esteve, 2011. [luis\(at\)epsilon-formacion.com](mailto:luis(at)epsilon-formacion.com)

This block takes in a signal stream and performs direct resampling. The theory behind this block can be found in Chapter 7.5 of the following book.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.62 `direct_resampler_conditioner_cs.h` File Reference

Nearest neighborhood resampler with `std::complex<short>` input and `std::complex<short>` output.

```
#include <gnuradio/block.h>
#include <cstdint>
#include <boost/shared_ptr.hpp>
```

Classes

- class [direct_resampler_conditioner_cs](#)

This class implements a direct resampler conditioner for `std::complex<short>`

Typedefs

- using **direct_resampler_conditioner_cs_sptr** = boost::shared_ptr< [direct_resampler_conditioner_cs](#) >

Functions

- **direct_resampler_conditioner_cs_sptr direct_resampler_make_conditioner_cs** (double sample_freq_in, double sample_freq_out)

11.62.1 Detailed Description

Nearest neighborhood resampler with std::complex<short> input and std::complex<short> output.

Author

Luis Esteve, 2011. [luis\(at\)epsilon-formacion.com](mailto:luis(at)epsilon-formacion.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.63 display.h File Reference

Defines useful display constants.

```
#include <string>
```

Macros

- #define **DISPLAY_COLORS** 1

Variables

- const std::string **TEXT_RESET** = "\033[0m"
- const std::string **TEXT_BLACK** = "\033[30m"
- const std::string **TEXT_RED** = "\033[31m"
- const std::string **TEXT_GREEN** = "\033[32m"
- const std::string **TEXT_YELLOW** = "\033[33m"
- const std::string **TEXT_BLUE** = "\033[34m"
- const std::string **TEXT_MAGENTA** = "\033[35m"
- const std::string **TEXT_CYAN** = "\033[36m"
- const std::string **TEXT_WHITE** = "\033[37m"
- const std::string **TEXT_BOLD_BLACK** = "\033[1m\033[30m"
- const std::string **TEXT_BOLD_RED** = "\033[1m\033[31m"
- const std::string **TEXT_BOLD_GREEN** = "\033[1m\033[32m"
- const std::string **TEXT_BOLD_YELLOW** = "\033[1m\033[33m"
- const std::string **TEXT_BOLD_BLUE** = "\033[1m\033[34m"
- const std::string **TEXT_BOLD_MAGENTA** = "\033[1m\033[35m"
- const std::string **TEXT_BOLD_CYAN** = "\033[1m\033[36m"
- const std::string **TEXT_BOLD_WHITE** = "\033[1m\033[37m"

11.63.1 Detailed Description

Defines useful display constants.

Author

Antonio Ramos, 2018. antonio.ramos(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.64 dll_pll_conf.h File Reference

Class that contains all the configuration parameters for generic tracking block based on a DLL and a PLL.

```
#include "configuration_interface.h"
#include <stdint>
#include <string>
```

Classes

- class [Dll_Pll_Conf](#)

11.64.1 Detailed Description

Class that contains all the configuration parameters for generic tracking block based on a DLL and a PLL.

Author

Javier Arribas, 2018. jarribas(at)cttc.es

Class that contains all the configuration parameters for generic tracking block based on a DLL and a PLL.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.65 dll_pll_conf_fpga.h File Reference

Class that contains all the configuration parameters for generic tracking block based on a DLL and a PLL for the FPGA.

```
#include "configuration_interface.h"
#include <stdint>
#include <string>
```

Classes

- class [Dll_Pll_Conf_Fpga](#)

11.65.1 Detailed Description

Class that contains all the configuration parameters for generic tracking block based on a DLL and a PLL for the FPGA.

Author

Marc Majoral, 2019. mmajoral(at)cttc.cat
Javier Arribas, 2018. jarribas(at)cttc.es

Class that contains all the configuration parameters for generic tracking block based on a DLL and a PLL.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.66 dll_pll_veml_tracking.h File Reference

Implementation of a code DLL + carrier PLL tracking block.

```
#include "cpu_multicorrelator_real_codes.h"
#include "dll_pll_conf.h"
#include "exponential_smoother.h"
#include "tracking_FLL_PLL_filter.h"
#include "tracking_loop_filter.h"
#include <boost/circular_buffer.hpp>
#include <gnuradio/block.h>
#include <gnuradio/gr_complex.h>
#include <gnuradio/types.h>
#include <pmt/pmt.h>
#include <volk_gnssssdr/volk_gnssssdr_alloc.h>
#include <stddef>
#include <stdint>
#include <fstream>
#include <string>
#include <typeinfo>
#include <utility>
#include <boost/shared_ptr.hpp>
```


Classes

- class [dll_pll_veml_tracking](#)
This class implements a code DLL + carrier PLL tracking block.

Typedefs

- using [dll_pll_veml_tracking_sptr](#) = boost::shared_ptr< [dll_pll_veml_tracking](#) >

Functions

- [dll_pll_veml_tracking_sptr](#) [dll_pll_veml_make_tracking](#) (const [Dll_Pll_Conf](#) &conf_)

11.66.1 Detailed Description

Implementation of a code DLL + carrier PLL tracking block.

Author

Javier Arribas, 2018. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)
Antonio Ramos, 2018 [antonio.ramosdet\(at\)gmail.com](mailto:antonio.ramosdet(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.67 dll_pll_veml_tracking_fpga.h File Reference

Implementation of a code DLL + carrier PLL tracking block using an FPGA.

```
#include "dll_pll_conf_fpga.h"
#include "exponential_smoother.h"
#include "tracking_FLL_PLL_filter.h"
#include "tracking_loop_filter.h"
#include <boost/circular_buffer.hpp>
#include <gnuradio/block.h>
#include <gnuradio/gr_complex.h>
#include <gnuradio/types.h>
#include <pmt/pmt.h>
#include <volk_gnssdr/volk_gnssdr_alloc.h>
#include <stdint.h>
#include <string>
#include <memory>
#include <string>
#include <typeinfo>
#include <utility>
#include <boost/shared_ptr.hpp>
```

Classes

- class [dll_pll_veml_tracking_fpga](#)
This class implements a code DLL + carrier PLL tracking block.

Typedefs

- using [dll_pll_veml_tracking_fpga_sptr](#) = boost::shared_ptr< [dll_pll_veml_tracking_fpga](#) >

Functions

- [dll_pll_veml_tracking_fpga_sptr](#) [dll_pll_veml_make_tracking_fpga](#) (const [Dll_Pll_Conf_Fpga](#) &conf_)

11.67.1 Detailed Description

Implementation of a code DLL + carrier PLL tracking block using an FPGA.

Author

Marc Majoral, 2019. marc.majoral@cttc.es
Javier Arribas, 2019. jarribas@cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.68 edc.h File Reference

Utilities for CRC computation of the libswiftnav library.

```
#include "swift_common.h"
```

Functions

- [uint32_t](#) [crc24q](#) (const [uint8_t](#) *buf, [uint32_t](#) len, [uint32_t](#) crc)
- [uint32_t](#) [crc24q_bits](#) ([uint32_t](#) crc, const [uint8_t](#) *buf, [uint32_t](#) n_bits, bool invert)

11.68.1 Detailed Description

Utilities for CRC computation of the libswiftnav library.

Author

Fergus Noble fergus@swift-nav.com

This file was originally borrowed from libswiftnav <https://github.com/swift-nav/libswiftnav>, a portable C library implementing GNSS related functions and algorithms, and then modified by J. Arribas and C. Fernandez

Copyright (C) 2010 Swift Navigation Inc. Contact: Fergus Noble fergus@swift-nav.com

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: LGPL-3.0-only

11.69 exponential_smoother.h File Reference

Class that implements an exponential smoother.

```
#include <vector>
```

Classes

- class [Exponential_Smoother](#)
Class that implements a first-order exponential smoother.

11.69.1 Detailed Description

Class that implements an exponential smoother.

Authors

Carles Fernandez, 2019 cfernandez@cttc.es

Class that implements a first-order exponential smoother.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.70 fec.h File Reference

Utilities for the convolutional encoder of the libswiftnav library.

Classes

- struct [v27_poly_t](#)
- struct [v27_decision_t](#)
- struct [v27_t](#)

Macros

- `#define V27POLYA 0x4f`
- `#define V27POLYB 0x6d`

Functions

- void **v27_poly_init** ([v27_poly_t](#) *poly, const signed char polynomial[2])
- void **v27_init** ([v27_t](#) *v, [v27_decision_t](#) *decisions, unsigned int decisions_count, const [v27_poly_t](#) *poly, unsigned char initial_state)
- void **v27_update** ([v27_t](#) *v, const unsigned char *syms, int nbits)
- void **v27_chainback_fixed** ([v27_t](#) *v, unsigned char *data, unsigned int nbits, unsigned char final_state)
- void **v27_chainback_likely** ([v27_t](#) *v, unsigned char *data, unsigned int nbits)

11.70.1 Detailed Description

Utilities for the convolutional encoder of the libswiftnav library.

Author

Phil Karn, KA9Q

This file was originally borrowed from libswiftnav <https://github.com/swift-nav/libswiftnav>, a portable C library implementing GNSS related functions and algorithms, and then modified by J. Arribas and C. Fernandez

Copyright (C) 2004, Phil Karn, KA9Q

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: LGPL-3.0-only

11.71 fft_base_kernels.h File Reference

FFT base kernels for OpenCL.

```
#include <string>
```

11.71.1 Detailed Description

FFT base kernels for OpenCL.

Version: <1.0>

Copyright (C) 2008 Apple Inc. All Rights Reserved. SPDX-License-Identifier: LicenseRef-Apple-Permissive

11.72 `fft_internal.h` File Reference

Internals of FFT for OpenCL.

```
#include "clFFT.h"
#include <iostream>
#include <sstream>
#include <string>
```

Classes

- struct [kernel_info_t](#)
- struct [cl_fft_plan](#)

Typedefs

- typedef enum kernel_dir_t [cl_fft_kernel_dir](#)
- typedef struct [kernel_info_t](#) [cl_fft_kernel_info](#)

Enumerations

- enum [kernel_dir_t](#) { [cl_fft_kernel_x](#), [cl_fft_kernel_y](#), [cl_fft_kernel_z](#) }

Functions

- void **FFT1D** ([cl_fft_plan](#) *plan, [cl_fft_kernel_dir](#) dir)

11.72.1 Detailed Description

Internals of FFT for OpenCL.

Version: <1.0>

Copyright (C) 2008 Apple Inc. All Rights Reserved. SPDX-License-Identifier: LicenseRef-Apple-Permissive

11.73 file_configuration.h File Reference

A [ConfigurationInterface](#) that reads the configuration from a file.

```
#include "INIReader.h"
#include "configuration_interface.h"
#include "in_memory_configuration.h"
#include "string_converter.h"
#include <stdint>
#include <memory>
#include <string>
```

Classes

- class [FileConfiguration](#)

This class is an implementation of the interface [ConfigurationInterface](#).

11.73.1 Detailed Description

A [ConfigurationInterface](#) that reads the configuration from a file.

Author

Carlos Aviles, 2010. carlos.avilesr(at)gmail.com

This implementation has a text file as the source for the values of the parameters. The file is in the INI format, containing sections and pairs of names and values. For more information about the INI format, see https://en.wikipedia.org/wiki/INI_file

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.74 file_signal_source.h File Reference

Interface of a class that reads signals samples from a file and adapts it to a [SignalSourceInterface](#).

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/file_source.h>
#include <gnuradio/blocks/throttle.h>
#include <gnuradio/hier_block2.h>
#include <pmt/pmt.h>
#include <stdint>
#include <memory>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [FileSignalSource](#)

Class that reads signals samples from a file and adapts it to a `SignalSourceInterface`.

11.74.1 Detailed Description

Interface of a class that reads signals samples from a file and adapts it to a `SignalSourceInterface`.

Author

Carlos Aviles, 2010. carlos.avilesr(at)gmail.com

This class represents a file signal source. Internally it uses a GNU Radio's `gr_file_source` as a connector to the data.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.75 fir_filter.h File Reference

Adapts a gnuradio `gr_fir_filter` designed with `pm_remez`.

```
#include "byte_x2_to_complex_byte.h"
#include "complex_byte_to_float_x2.h"
#include "cshort_to_float_x2.h"
#include "gnss_block_interface.h"
#include "short_x2_to_cshort.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/float_to_char.h>
#include <gnuradio/blocks/float_to_complex.h>
#include <gnuradio/blocks/float_to_short.h>
#include <gnuradio/gr_complex.h>
#include <gnuradio/filter/fir_filter_ccf.h>
#include <gnuradio/filter/fir_filter_fff.h>
#include <cmath>
#include <string>
#include <vector>
```

Classes

- class [FirFilter](#)

This class adapts a GNU Radio `gr_fir_filter` designed with `pm_remez`.

11.75.1 Detailed Description

Adapts a gnuradio `gr_fir_filter` designed with `pm_remez`.

Author

Luis Esteve, 2012. [luis\(at\)epsilon-formacion.com](mailto:luis(at)epsilon-formacion.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.76 flexiband_signal_source.h File Reference

Signal Source adapter for the Teleorbit Flexiband front-end device. This adapter requires a Flexiband GNU Radio driver installed (not included with GNSS-SDR)

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include <gnuradio/blocks/char_to_float.h>
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/float_to_complex.h>
#include <gnuradio/blocks/null_sink.h>
#include <gnuradio/hier_block2.h>
#include <pmt/pmt.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [FlexibandSignalSource](#)

This class configures and reads samples from Teleorbit Flexiband front-end. This software requires a Flexiband GNU Radio driver installed (not included with GNSS-SDR).

11.76.1 Detailed Description

Signal Source adapter for the Teleorbit Flexiband front-end device. This adapter requires a Flexiband GNU Radio driver installed (not included with GNSS-SDR)

Author

Javier Arribas, [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.77 fmcomms2_signal_source.h File Reference

Interface to use SDR hardware based in FMCOMMS2 driver from analog devices, for example FMCOMMS4 and ADALM-PLUTO (PlutoSdr)

```
#include "gnss_block_interface.h"
#include <gnuradio/blocks/file_sink.h>
#include <iio/fmcomms2_source.h>
#include "concurrent_queue.h"
#include <pmt/pmt.h>
#include <cstdint>
#include <memory>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [Fmcomms2SignalSource](#)

11.77.1 Detailed Description

Interface to use SDR hardware based in FMCOMMS2 driver from analog devices, for example FMCOMMS4 and ADALM-PLUTO (PlutoSdr)

Author

Rodrigo Muñoz, 2017. [rmunozl\(at\)inacap.cl](mailto:rmunozl@inacap.cl), [rodrigo.munoz\(at\)proteinlab.cl](mailto:rodrigo.munoz@proteinlab.cl)

This class represent a fmcomms2 signal source. It use the gr_iio block

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.78 fpga_acquisition.h File Reference

Highly optimized FPGA vector correlator class.

```
#include <cstdint>
#include <string>
```

Classes

- class [Fpga_Acquisition](#)
Class that implements carrier wipe-off and correlators.

11.78.1 Detailed Description

Highly optimized FPGA vector correlator class.

Authors

- Marc Majoral, 2019. mmajoral(at)cttc.cat

Class that controls and executes a highly optimized acquisition HW accelerator in the FPGA

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.79 fpga_dynamic_bit_selection.h File Reference

Dynamic bit selection in the received signal.

```
#include <stddef>
#include <stdint>
#include <string>
```

Classes

- class [Fpga_dynamic_bit_selection](#)
Class that controls the switch in the FPGA, which connects the FPGA acquisition and multicorrelator modules to either the DMA or the Analog Front-End.

11.79.1 Detailed Description

Dynamic bit selection in the received signal.

Authors

- Marc Majoral, 2020. mmajoral(at)cttc.es

Class that controls the Dynamic Bit Selection in the FPGA.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.80 fpga_multicorrelator.h File Reference

FPGA vector correlator class.

```
#include <gnuradio/block.h>
#include <volk_gnssssdr/volk_gnssssdr_alloc.h>
#include <cstdint>
#include <string>
```

Classes

- class [Fpga_Multicorrelator_8sc](#)
Class that implements carrier wipe-off and correlators.

11.80.1 Detailed Description

FPGA vector correlator class.

Authors

- Marc Majoral, 2019. mmajoral(at)cttc.cat
- Javier Arribas, 2019. jarribas(at)cttc.es

Class that controls and executes a highly optimized vector correlator class in the FPGA

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.81 fpga_switch.h File Reference

Switch that connects the HW accelerator queues to the analog front end or the DMA.

```
#include <string>
```

Classes

- class [Fpga_Switch](#)
Class that controls the switch in the FPGA, which connects the FPGA acquisition and multicorrelator modules to either the DMA or the Analog Front-End.

11.81.1 Detailed Description

Switch that connects the HW accelerator queues to the analog front end or the DMA.

Authors

- Marc Majoral, 2019. mmajoral(at)cttc.cat
- Javier Arribas, 2016. jarribas(at)cttc.es

Class that controls a switch in the FPGA

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.82 freq_xlating_fir_filter.h File Reference

Adapts a gnuradio gr_freq_xlating_fir_filter designed with gr_remez.

```
#include "complex_float_to_complex_byte.h"
#include "gnss_block_interface.h"
#include "short_x2_to_cshort.h"
#include <gnuradio/filter/freq_xlating_fir_filter_ccf.h>
#include <gnuradio/filter/freq_xlating_fir_filter_fcf.h>
#include <gnuradio/filter/freq_xlating_fir_filter_scf.h>
#include <gnuradio/blocks/char_to_short.h>
#include <gnuradio/blocks/complex_to_float.h>
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/float_to_short.h>
#include <string>
#include <vector>
```

Classes

- class [FreqXlatingFirFilter](#)

This class adapts a gnuradio gr_freq_xlating_fir_filter designed with pm_remez.

11.82.1 Detailed Description

Adapts a gnuradio gr_freq_xlating_fir_filter designed with gr_remez.

Author

Luis Esteve, 2012. luis(at)epsilon-formacion.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.83 front_end_cal.h File Reference

Interface of the Front-end calibration program.

```
#include <armadillo>
#include <memory>
#include <string>
```

Classes

- class [FrontEndCal](#)

11.83.1 Detailed Description

Interface of the Front-end calibration program.

Author

Javier Arribas, 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.84 galileo_almanac.h File Reference

Interface of a Galileo ALMANAC storage.

```
#include <boost/serialization/nvp.hpp>
#include <cstdint>
```

Classes

- class [Galileo_Almanac](#)

This class is a storage for the Galileo SV ALMANAC data.

11.84.1 Detailed Description

Interface of a Galileo ALMANAC storage.

Author

Carles Fernandez, 2018. cfernandez(at)cttc.cat

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.85 galileo_almanac_helper.h File Reference

Interface of a Galileo ALMANAC storage helper.

```
#include "galileo_almanac.h"
#include <cstdint>
```

Classes

- class [Galileo_Almanac_Helper](#)

This class is a storage for the GALILEO ALMANAC data as described in GALILEO ICD.

11.85.1 Detailed Description

Interface of a Galileo ALMANAC storage helper.

Author

Javier Arribas, 2013. jarribas(at)cttc.es

Mara Branzanti 2013. mara.branzanti(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.86 Galileo_E1.h File Reference

Defines system parameters for Galileo E1 signal and NAV data.

```
#include "MATH_CONSTANTS.h"
#include "gnss_frequencies.h"
#include <cstdint>
#include <cstdint>
```

Variables

- constexpr double `GALILEO_E1_FREQ_HZ` = `FREQ1`
Galileo E1 carrier frequency [Hz].
- constexpr double `GALILEO_E1_CODE_CHIP_RATE_CPS` = 1.023e6
Galileo E1 code rate [chips/s].
- constexpr double `GALILEO_E1_CODE_PERIOD_S` = 0.004
Galileo E1 code period [s].
- constexpr double `GALILEO_E1_SUB_CARRIER_A_RATE_HZ` = 1.023e6
Galileo E1 sub-carrier 'a' rate [Hz].
- constexpr double `GALILEO_E1_SUB_CARRIER_B_RATE_HZ` = 6.138e6
Galileo E1 sub-carrier 'b' rate [Hz].
- constexpr double `GALILEO_E1_B_CODE_LENGTH_CHIPS` = 4092.0
Galileo E1-B code length [chips].
- constexpr double `GALILEO_E1_B_SYMBOL_RATE_BPS` = 250.0
Galileo E1-B symbol rate [bits/second].
- constexpr int32_t `GALILEO_E1_CODE_PERIOD_MS` = 4
Galileo E1 code period [ms].
- constexpr int32_t `GALILEO_E1_B_SAMPLES_PER_SYMBOL` = 1
(Galileo_E1_CODE_CHIP_RATE_HZ / Galileo_E1_B_CODE_LENGTH_CHIPS) / Galileo_E1_B_SYMBOL_RATE_BPS
- constexpr int32_t `GALILEO_E1_C_SECONDARY_CODE_LENGTH` = 25
Galileo E1-C secondary code length [chips].
- constexpr int32_t `GALILEO_E1_NUMBER_OF_CODES` = 50
- constexpr uint32_t `GALILEO_E1_OPT_ACQ_FS_SPS` = 2000000
Sampling frequency that maximizes the acquisition SNR while using a non-multiple of chip rate.
- constexpr int32_t `GALILEO_E1_HISTORY_DEEP` = 100
Observable history length for interpolation.
- constexpr char `GALILEO_E1_C_SECONDARY_CODE` [26] = "0011100000001010110110010"
- constexpr size_t `GALILEO_E1_B_PRIMARY_CODE_STR_LENGTH` = 1023
- constexpr char `GALILEO_E1_B_PRIMARY_CODE` [GALILEO_E1_NUMBER_OF_CODES][1024]
- constexpr size_t `GALILEO_E1_C_PRIMARY_CODE_STR_LENGTH` = 1023
- constexpr char `GALILEO_E1_C_PRIMARY_CODE` [GALILEO_E1_NUMBER_OF_CODES][1024]

11.86.1 Detailed Description

Defines system parameters for Galileo E1 signal and NAV data.

Author

Luis Esteve, 2012. luis(at)epsilon-formacion.com
Mara Branzanti 2013. mara.branzanti(at)gmail.com
Javier Arribas 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.86.2 Variable Documentation

11.86.2.1 GALILEO_E1_B_CODE_LENGTH_CHIPS

```
constexpr double GALILEO_E1_B_CODE_LENGTH_CHIPS = 4092.0
```

Galileo E1-B code length [chips].

Definition at line 37 of file Galileo_E1.h.

11.86.2.2 GALILEO_E1_B_SAMPLES_PER_SYMBOL

```
constexpr int32_t GALILEO_E1_B_SAMPLES_PER_SYMBOL = 1
```

$(\text{Galileo_E1_CODE_CHIP_RATE_HZ} / \text{Galileo_E1_B_CODE_LENGTH_CHIPS}) / \text{Galileo_E1_B_SYMBOL_RATE_BPS}$

Definition at line 40 of file Galileo_E1.h.

11.86.2.3 GALILEO_E1_B_SYMBOL_RATE_BPS

```
constexpr double GALILEO_E1_B_SYMBOL_RATE_BPS = 250.0
```

Galileo E1-B symbol rate [bits/second].

Definition at line 38 of file Galileo_E1.h.

11.86.2.4 GALILEO_E1_C_SECONDARY_CODE_LENGTH

```
constexpr int32_t GALILEO_E1_C_SECONDARY_CODE_LENGTH = 25
```

Galileo E1-C secondary code length [chips].

Definition at line 41 of file Galileo_E1.h.

11.86.2.5 GALILEO_E1_CODE_CHIP_RATE_CPS

```
constexpr double GALILEO_E1_CODE_CHIP_RATE_CPS = 1.023e6
```

Galileo E1 code rate [chips/s].

Definition at line 33 of file Galileo_E1.h.

11.86.2.6 GALILEO_E1_CODE_PERIOD_MS

```
constexpr int32_t GALILEO_E1_CODE_PERIOD_MS = 4
```

Galileo E1 code period [ms].

Definition at line 39 of file Galileo_E1.h.

11.86.2.7 GALILEO_E1_CODE_PERIOD_S

```
constexpr double GALILEO_E1_CODE_PERIOD_S = 0.004
```

Galileo E1 code period [s].

Definition at line 34 of file Galileo_E1.h.

11.86.2.8 GALILEO_E1_FREQ_HZ

```
constexpr double GALILEO_E1_FREQ_HZ = FREQ1
```

Galileo E1 carrier frequency [Hz].

Definition at line 32 of file Galileo_E1.h.

11.86.2.9 GALILEO_E1_HISTORY_DEEP

```
constexpr int32_t GALILEO_E1_HISTORY_DEEP = 100
```

Observable history length for interpolation.

Definition at line 48 of file Galileo_E1.h.

11.86.2.10 GALILEO_E1_OPT_ACQ_FS_SPS

```
constexpr uint32_t GALILEO_E1_OPT_ACQ_FS_SPS = 2000000
```

Sampling frequency that maximizes the acquisition SNR while using a non-multiple of chip rate.

Definition at line 46 of file Galileo_E1.h.

11.86.2.11 GALILEO_E1_SUB_CARRIER_A_RATE_HZ

```
constexpr double GALILEO_E1_SUB_CARRIER_A_RATE_HZ = 1.023e6
```

Galileo E1 sub-carrier 'a' rate [Hz].

Definition at line 35 of file Galileo_E1.h.

11.86.2.12 GALILEO_E1_SUB_CARRIER_B_RATE_HZ

```
constexpr double GALILEO_E1_SUB_CARRIER_B_RATE_HZ = 6.138e6
```

Galileo E1 sub-carrier 'b' rate [Hz].

Definition at line 36 of file Galileo_E1.h.

11.87 galileo_e1_dll_pll_veml_tracking.h File Reference

Adapts a DLL+PLL VEML (Very Early Minus Late) tracking loop block to a [TrackingInterface](#) for Galileo E1 signals.

```
#include "dll_pll_veml_tracking.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GalileoE1DIPIIVemlTracking](#)

This class Adapts a DLL+PLL VEML (Very Early Minus Late) tracking loop block to a [TrackingInterface](#) for Galileo E1 signals.

11.87.1 Detailed Description

Adapts a DLL+PLL VEML (Very Early Minus Late) tracking loop block to a [TrackingInterface](#) for Galileo E1 signals.

Author

Luis Esteve, 2012. luis(at)epsilon-formacion.com

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.88 galileo_e1_dll_pll_veml_tracking_fpga.h File Reference

Adapts a DLL+PLL VEML (Very Early Minus Late) tracking loop block to a [TrackingInterface](#) for Galileo E1 signals for the FPGA.

```
#include "dll_pll_veml_tracking_fpga.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GalileoE1DllPlLVemlTrackingFpga](#)

This class Adapts a DLL+PLL VEML (Very Early Minus Late) tracking loop block to a [TrackingInterface](#) for Galileo E1 signals.

11.88.1 Detailed Description

Adapts a DLL+PLL VEML (Very Early Minus Late) tracking loop block to a [TrackingInterface](#) for Galileo E1 signals for the FPGA.

Author

Marc Majoral, 2019. mmajoral(at)cttc.cat

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.89 galileo_e1_pcps_8ms_ambiguous_acquisition.h File Reference

Adapts a PCPS 8ms acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

```
#include "acquisition_interface.h"
#include "galileo_pcps_8ms_acquisition_cc.h"
#include "gnss_synchro.h"
#include <gnuradio/blocks/stream_to_vector.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GalileoE1Pcps8msAmbiguousAcquisition](#)

Adapts a PCPS 8ms acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

11.89.1 Detailed Description

Adapts a PCPS 8ms acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

Author

Marc Molina, 2013. marc.molina.pena(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.90 galileo_e1_pcps_ambiguous_acquisition.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

```
#include "acq_conf.h"
#include "channel_fsm.h"
#include "complex_byte_to_float_x2.h"
#include "gnss_synchro.h"
#include "pcps_acquisition.h"
#include <gnuradio/blocks/float_to_complex.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GalileoE1PcpsAmbiguousAcquisition](#)

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

11.90.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

Author

Luis Esteve, 2012. luis(at)epsilon-formacion.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.91 galileo_e1_pcps_ambiguous_acquisition_fpga.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals for the FPGA.

```
#include "acq_conf.h"
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include "pcps_acquisition_fpga.h"
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GalileoE1PcpsAmbiguousAcquisitionFpga](#)

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for Galileo E1 Signals.

11.91.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals for the FPGA.

Author

Marc Majoral, 2019. mmajoral(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.92 galileo_e1_pcps_cccwsr_ambiguous_acquisition.h File Reference

Adapts a PCPS CCCWSR acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include "pcps_cccwsr_acquisition_cc.h"
#include <gnuradio/blocks/stream_to_vector.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GalileoE1PcpsCccwsrAmbiguousAcquisition](#)
Adapts a PCPS CCCWSR acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

11.92.1 Detailed Description

Adapts a PCPS CCCWSR acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

Author

Marc Molina, 2013. marc.molina.pena(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.93 galileo_e1_pcps_quicksync_ambiguous_acquisition.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include "pcps_quicksync_acquisition_cc.h"
#include <gnuradio/blocks/stream_to_vector.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GalileoE1PcpsQuickSyncAmbiguousAcquisition](#)
This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

11.93.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

Date

June, 2014

Author

Damian Miralles Sanchez. dmiralles2009@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.94 galileo_e1_pcps_tong_ambiguous_acquisition.h File Reference

Adapts a PCPS Tong acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include "pcps_tong_acquisition_cc.h"
#include <gnuradio/blocks/stream_to_vector.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GalileoE1PcpsTongAmbiguousAcquisition](#)

Adapts a PCPS Tong acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

11.94.1 Detailed Description

Adapts a PCPS Tong acquisition block to an [AcquisitionInterface](#) for Galileo E1 Signals.

Author

Marc Molina, 2013. [marc.molina.pena\(at\)gmail.com](mailto:marc.molina.pena(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.95 galileo_e1_signal_processing.h File Reference

This library implements various functions for Galileo E1 signals.

```
#include <array>
#include <complex>
#include <cstdint>
#include <gsl/gsl>
```

Functions

- void [galileo_e1_code_gen_sinboc11_float](#) (own::span< float > _dest, const std::array< char, 3 > &_Signal, uint32_t _prn)
This function generates Galileo E1 code (can select E1B or E1C sinboc).
- void [galileo_e1_code_gen_float_sampled](#) (own::span< float > _dest, const std::array< char, 3 > &_Signal, bool _cboc, uint32_t _prn, int32_t _fs, uint32_t _chip_shift, bool _secondary_flag)
This function generates Galileo E1 code (can select E1B or E1C, cboc or sinboc and the sample frequency _fs).
- void [galileo_e1_code_gen_float_sampled](#) (own::span< float > _dest, const std::array< char, 3 > &_Signal, bool _cboc, uint32_t _prn, int32_t _fs, uint32_t _chip_shift)
This function generates Galileo E1 code (can select E1B or E1C, cboc or sinboc and the sample frequency _fs).
- void [galileo_e1_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, const std::array< char, 3 > &_Signal, bool _cboc, uint32_t _prn, int32_t _fs, uint32_t _chip_shift, bool _secondary_flag)
This function generates Galileo E1 code (can select E1B or E1C, cboc or sinboc and the sample frequency _fs).
- void [galileo_e1_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, const std::array< char, 3 > &_Signal, bool _cboc, uint32_t _prn, int32_t _fs, uint32_t _chip_shift)
galileo_e1_code_gen_complex_sampled without _secondary_flag for backward compatibility.

11.95.1 Detailed Description

This library implements various functions for Galileo E1 signals.

Author

Luis Esteve, 2012. luis(at)epsilon-formacion.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.95.2 Function Documentation

11.95.2.1 galileo_e1_code_gen_complex_sampled() [1/2]

```
void galileo_e1_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    const std::array< char, 3 > & _Signal,
    bool _cboc,
    uint32_t _prn,
    int32_t _fs,
    uint32_t _chip_shift,
    bool _secondary_flag )
```

This function generates Galileo E1 code (can select E1B or E1C, cboc or sinboc and the sample frequency _fs).

11.95.2.2 galileo_e1_code_gen_complex_sampled() [2/2]

```
void galileo_e1_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    const std::array< char, 3 > & _Signal,
    bool _cboc,
    uint32_t _prn,
    int32_t _fs,
    uint32_t _chip_shift )
```

galileo_e1_code_gen_complex_sampled without _secondary_flag for backward compatibility.

11.95.2.3 galileo_e1_code_gen_float_sampled() [1/2]

```
void galileo_e1_code_gen_float_sampled (
    own::span< float > _dest,
    const std::array< char, 3 > & _Signal,
    bool _cboc,
    uint32_t _prn,
    int32_t _fs,
    uint32_t _chip_shift,
    bool _secondary_flag )
```

This function generates Galileo E1 code (can select E1B or E1C, cboc or sinboc and the sample frequency _fs).

11.95.2.4 galileo_e1_code_gen_float_sampled() [2/2]

```
void galileo_e1_code_gen_float_sampled (
    own::span< float > _dest,
    const std::array< char, 3 > & _Signal,
    bool _cboc,
    uint32_t _prn,
    int32_t _fs,
    uint32_t _chip_shift )
```

This function generates Galileo E1 code (can select E1B or E1C, cboc or sinboc and the sample frequency _fs).

11.95.2.5 galileo_e1_code_gen_sinboc11_float()

```
void galileo_e1_code_gen_sinboc11_float (
    own::span< float > _dest,
    const std::array< char, 3 > & _Signal,
    uint32_t _prn )
```

This function generates Galileo E1 code (can select E1B or E1C sinboc).

11.96 galileo_e1_tcp_connector_tracking.h File Reference

Interface of an adapter of a TCP connector block based on code DLL + carrier PLL for Galileo E1 to a [TrackingInterface](#).

```
#include "galileo_e1_tcp_connector_tracking_cc.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GalileoE1TcpConnectorTracking](#)
This class implements a code DLL + carrier PLL tracking loop.

11.96.1 Detailed Description

Interface of an adapter of a TCP connector block based on code DLL + carrier PLL for Galileo E1 to a [TrackingInterface](#).

Author

David Pubill, 2012. [dpubill\(at\)cttc.es](mailto:dpubill(at)cttc.es) Luis Esteve, 2012. [luis\(at\)epsilon-formacion.com](mailto:luis(at)epsilon-formacion.com) Javier Arribas, 2011. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2012-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.97 galileo_e1_tcp_connector_tracking_cc.h File Reference

Interface of a TCP connector block based on code DLL + carrier PLL VEML (Very Early Minus Late) tracking block for Galileo E1 signals.

```
#include "cpu_multicorrelator.h"
#include "gnss_synchro.h"
#include "tcp_communication.h"
#include <gnuradio/block.h>
#include <volk_gnssssdr/volk_gnssssdr_alloc.h>
#include <fstream>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [Galileo_E1_Tcp_Connector_Tracking_cc](#)

This class implements a code DLL + carrier PLL VEML (Very Early Minus Late) tracking block for Galileo E1 signals.

Typedefs

- using [galileo_e1_tcp_connector_tracking_cc_sptr](#) = boost::shared_ptr< [Galileo_E1_Tcp_Connector_Tracking_cc](#) >

Functions

- galileo_e1_tcp_connector_tracking_cc_sptr [galileo_e1_tcp_connector_make_tracking_cc](#) (int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float pll_bw_hz, float dll_bw_hz, float early_late_space_chips, float very_early_late_space_chips, size_t port_ch0)

11.97.1 Detailed Description

Interface of a TCP connector block based on code DLL + carrier PLL VEML (Very Early Minus Late) tracking block for Galileo E1 signals.

Author

David Pubill, 2012. dpubill(at)cttc.es Luis Esteve, 2012. luis(at)epsilon-formacion.com Javier Arribas, 2011. jarribas(at)cttc.es

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.98 galileo_e1b_telemetry_decoder.h File Reference

Interface of an adapter of a GALILEO E1B NAV data decoder block to a [TelemetryDecoderInterface](#).

```
#include "galileo_telemetry_decoder_gs.h"
#include "gnss_satellite.h"
#include "gnss_synchro.h"
#include "telemetry_decoder_interface.h"
#include <gnuradio/runtime_types.h>
#include <cstdint>
#include <string>
```

Classes

- class [GalileoE1BTelemetryDecoder](#)

This class implements a NAV data decoder for Galileo INAV frames in E1B radio link.

11.98.1 Detailed Description

Interface of an adapter of a GALILEO E1B NAV data decoder block to a [TelemetryDecoderInterface](#).

Author

Javier Arribas 2013 jarribas(at)cttc.es, Mara Branzanti 2013. mara.branzanti(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.99 galileo_e5_signal_processing.h File Reference

This library implements various functions for Galileo E5 signals such as replica code generation.

```
#include <array>
#include <complex>
#include <cstdint>
#include <gsl/gsl>
```

Functions

- void [galileo_e5_a_code_gen_complex_primary](#) (own::span< std::complex< float >> _dest, int32_t _prn, const std::array< char, 3 > &_Signal)
Generates Galileo E5a code at 1 sample/chip.
- void [galileo_e5_a_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, uint32_t _prn, const std::array< char, 3 > &_Signal, int32_t _fs, uint32_t _chip_shift)
Generates Galileo E5a complex code, shifted to the desired chip and sampled at a frequency fs.
- void [galileo_e5_b_code_gen_complex_primary](#) (own::span< std::complex< float >> _dest, int32_t _prn, const std::array< char, 3 > &_Signal)
Generates Galileo E5b code at 1 sample/chip.
- void [galileo_e5_b_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, uint32_t _prn, const std::array< char, 3 > &_Signal, int32_t _fs, uint32_t _chip_shift)
Generates Galileo E5b complex code, shifted to the desired chip and sampled at a frequency fs.

11.99.1 Detailed Description

This library implements various functions for Galileo E5 signals such as replica code generation.

Author

Marc Sales, 2014. [marcsales92\(at\)gmail.com](mailto:marcsales92(at)gmail.com)
Piyush Gupta, 2020. piyush04111999@gmail.com

Note

Code added as part of GSoC 2020 Program.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.99.2 Function Documentation

11.99.2.1 [galileo_e5_a_code_gen_complex_primary\(\)](#)

```
void galileo_e5_a_code_gen_complex_primary (
    own::span< std::complex< float >> _dest,
    int32_t _prn,
    const std::array< char, 3 > &_Signal )
```

Generates Galileo E5a code at 1 sample/chip.

11.99.2.2 galileo_e5_a_code_gen_complex_sampled()

```
void galileo_e5_a_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    uint32_t _prn,
    const std::array< char, 3 > & _Signal,
    int32_t _fs,
    uint32_t _chip_shift )
```

Generates Galileo E5a complex code, shifted to the desired chip and sampled at a frequency fs.

11.99.2.3 galileo_e5_b_code_gen_complex_primary()

```
void galileo_e5_b_code_gen_complex_primary (
    own::span< std::complex< float >> _dest,
    int32_t _prn,
    const std::array< char, 3 > & _Signal )
```

Generates Galileo E5b code at 1 sample/chip.

11.99.2.4 galileo_e5_b_code_gen_complex_sampled()

```
void galileo_e5_b_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    uint32_t _prn,
    const std::array< char, 3 > & _Signal,
    int32_t _fs,
    uint32_t _chip_shift )
```

Generates Galileo E5b complex code, shifted to the desired chip and sampled at a frequency fs.

11.100 Galileo_E5a.h File Reference

Defines system parameters for Galileo E5a signal and NAV data.

```
#include "MATH_CONSTANTS.h"
#include "gnss_frequencies.h"
#include <cstdint>
#include <cstdint>
```

Variables

- constexpr double `GALILEO_E5A_FREQ_HZ` = `FREQ5`
Galileo E5a carrier frequency [Hz].
- constexpr double `GALILEO_E5A_CODE_CHIP_RATE_CPS` = 1.023e7
Galileo E5a code rate [chips/s].
- constexpr double `GALILEO_E5A_I_TIERED_CODE_PERIOD_S` = 0.020
Galileo E5a-I tiered code period [s].
- constexpr double `GALILEO_E5A_Q_TIERED_CODE_PERIOD_S` = 0.100
Galileo E5a-Q tiered code period [s].
- constexpr double `GALILEO_E5A_CODE_PERIOD_S` = 0.001
Galileo E5a primary code period [s].
- constexpr int32_t `GALILEO_E5A_CODE_LENGTH_CHIPS` = 10230
Galileo E5a primary code length [chips].
- constexpr int32_t `GALILEO_E5A_I_SECONDARY_CODE_LENGTH` = 20
Galileo E5a-I secondary code length [chips].
- constexpr int32_t `GALILEO_E5A_Q_SECONDARY_CODE_LENGTH` = 100
Galileo E5a-Q secondary code length [chips].
- constexpr int32_t `GALILEO_E5A_CODE_PERIOD_MS` = 1
Galileo E5a primary code period [ms].
- constexpr int32_t `GALILEO_E5A_SYMBOL_RATE_BPS` = 50
Galileo E5a symbol rate [bits/second].
- constexpr int32_t `GALILEO_E5A_NUMBER_OF_CODES` = 50
- constexpr int32_t `GALILEO_E5A_HISTORY_DEEP` = 20
- constexpr int32_t `GALILEO_E5A_CRC_ERROR_LIMIT` = 6
- constexpr uint32_t `GALILEO_E5A_OPT_ACQ_FS_SPS` = 10000000
Sampling frequency that maximizes the acquisition SNR while using a non-multiple of chip rate.
- constexpr int32_t `GALILEO_FNAV_PREAMBLE_LENGTH_BITS` = 12
- constexpr int32_t `GALILEO_FNAV_CODES_PER_SYMBOL` = 20
- constexpr int32_t `GALILEO_FNAV_CODES_PER_PREAMBLE` = 240
- constexpr int32_t `GALILEO_FNAV_SYMBOLS_PER_PAGE` = 500
- constexpr int32_t `GALILEO_FNAV_SECONDS_PER_PAGE` = 10
- constexpr int32_t `GALILEO_FNAV_CODES_PER_PAGE` = 10000
- constexpr int32_t `GALILEO_FNAV_INTERLEAVER_ROWS` = 8
- constexpr int32_t `GALILEO_FNAV_INTERLEAVER_COLS` = 61
- constexpr int32_t `GALILEO_FNAV_PAGE_TYPE_BITS` = 6
- constexpr int32_t `GALILEO_FNAV_DATA_FRAME_BITS` = 214
- constexpr int32_t `GALILEO_FNAV_DATA_FRAME_BYTES` = 27
- constexpr char `GALILEO_FNAV_PREAMBLE` [13] = "101101110000"
- constexpr size_t `GALILEO_E5A_I_PRIMARY_CODE_STR_LENGTH` = 2558
- constexpr char `GALILEO_E5A_I_PRIMARY_CODE` [`GALILEO_E5A_NUMBER_OF_CODES`][2559]
- constexpr size_t `GALILEO_E5A_Q_PRIMARY_CODE_STR_LENGTH` = 2558
- constexpr char `GALILEO_E5A_Q_PRIMARY_CODE` [`GALILEO_E5A_NUMBER_OF_CODES`][2559]
- constexpr char `GALILEO_E5A_I_SECONDARY_CODE` [] = "10000100001011101001"
- constexpr size_t `GALILEO_E5A_Q_SECONDARY_CODE_STR_LENGTH` = 100
- constexpr char `GALILEO_E5A_Q_SECONDARY_CODE` [`GALILEO_E5A_NUMBER_OF_CODES`][101]

11.100.1 Detailed Description

Defines system parameters for Galileo E5a signal and NAV data.

Author

Marc Sales, 2014. marcsales92@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.100.2 Variable Documentation

11.100.2.1 GALILEO_E5A_CODE_CHIP_RATE_CPS

```
constexpr double GALILEO_E5A_CODE_CHIP_RATE_CPS = 1.023e7
```

Galileo E5a code rate [chips/s].

Definition at line 31 of file Galileo_E5a.h.

11.100.2.2 GALILEO_E5A_CODE_LENGTH_CHIPS

```
constexpr int32_t GALILEO_E5A_CODE_LENGTH_CHIPS = 10230
```

Galileo E5a primary code length [chips].

Definition at line 35 of file Galileo_E5a.h.

11.100.2.3 GALILEO_E5A_CODE_PERIOD_MS

```
constexpr int32_t GALILEO_E5A_CODE_PERIOD_MS = 1
```

Galileo E5a primary code period [ms].

Definition at line 38 of file Galileo_E5a.h.

11.100.2.4 GALILEO_E5A_CODE_PERIOD_S

```
constexpr double GALILEO_E5A_CODE_PERIOD_S = 0.001
```

Galileo E5a primary code period [s].

Definition at line 34 of file Galileo_E5a.h.

11.100.2.5 GALILEO_E5A_FREQ_HZ

```
constexpr double GALILEO_E5A_FREQ_HZ = FREQ5
```

Galileo E5a carrier frequency [Hz].

Definition at line 30 of file Galileo_E5a.h.

11.100.2.6 GALILEO_E5A_I_SECONDARY_CODE_LENGTH

```
constexpr int32_t GALILEO_E5A_I_SECONDARY_CODE_LENGTH = 20
```

Galileo E5a-I secondary code length [chips].

Definition at line 36 of file Galileo_E5a.h.

11.100.2.7 GALILEO_E5A_I_TIERED_CODE_PERIOD_S

```
constexpr double GALILEO_E5A_I_TIERED_CODE_PERIOD_S = 0.020
```

Galileo E5a-I tiered code period [s].

Definition at line 32 of file Galileo_E5a.h.

11.100.2.8 GALILEO_E5A_OPT_ACQ_FS_SPS

```
constexpr uint32_t GALILEO_E5A_OPT_ACQ_FS_SPS = 10000000
```

Sampling frequency that maximizes the acquisition SNR while using a non-multiple of chip rate.

Definition at line 48 of file Galileo_E5a.h.

11.100.2.9 GALILEO_E5A_Q_SECONDARY_CODE_LENGTH

```
constexpr int32_t GALILEO_E5A_Q_SECONDARY_CODE_LENGTH = 100
```

Galileo E5a-Q secondary code length [chips].

Definition at line 37 of file Galileo_E5a.h.

11.100.2.10 GALILEO_E5A_Q_TIERED_CODE_PERIOD_S

```
constexpr double GALILEO_E5A_Q_TIERED_CODE_PERIOD_S = 0.100
```

Galileo E5a-Q tiered code period [s].

Definition at line 33 of file Galileo_E5a.h.

11.100.2.11 GALILEO_E5A_SYMBOL_RATE_BPS

```
constexpr int32_t GALILEO_E5A_SYMBOL_RATE_BPS = 50
```

Galileo E5a symbol rate [bits/second].

Definition at line 39 of file Galileo_E5a.h.

11.101 galileo_e5a_dll_pll_tracking.h File Reference

Adapts a code DLL + carrier PLL tracking block to a [TrackingInterface](#) for Galileo E5a signals.

```
#include "dll_pll_veml_tracking.h"  
#include "tracking_interface.h"  
#include <string>
```

Classes

- class [GalileoE5aDIIPIITracking](#)

This class implements a code DLL + carrier PLL tracking loop.

11.101.1 Detailed Description

Adapts a code DLL + carrier PLL tracking block to a [TrackingInterface](#) for Galileo E5a signals.

Author

Marc Sales, 2014. marcsales92(at)gmail.com on work from:

- Javier Arribas, 2011. jarribas(at)cttc.es
- Luis Esteve, 2012. luis(at)epsilon-formacion.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.102 galileo_e5a_dll_pll_tracking_fpga.h File Reference

Adapts a code DLL + carrier PLL tracking block to a [TrackingInterface](#) for Galileo E5a signals for the FPGA.

```
#include "dll_pll_veml_tracking_fpga.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GalileoE5aDIIPllTrackingFpga](#)
This class implements a code DLL + carrier PLL tracking loop.

11.102.1 Detailed Description

Adapts a code DLL + carrier PLL tracking block to a [TrackingInterface](#) for Galileo E5a signals for the FPGA.

Author

Marc Majoral, 2019. mmajoral(at)cttc.cat

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.103 galileo_e5a_noncoherent_iq_acquisition_caf.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E5a data and pilot Signals.

```
#include "channel_fsm.h"
#include "galileo_e5a_noncoherent_iq_acquisition_caf_cc.h"
#include "gnss_synchro.h"
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GalileoE5aNoncoherentIQAcquisitionCaf](#)

11.103.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E5a data and pilot Signals.

Author

Marc Sales, 2014. marcsales92(at)gmail.com on work from:

- Javier Arribas, 2011. jarribas(at)cttc.es
- Luis Esteve, 2012. luis(at)epsilon-formacion.com
- Marc Molina, 2013. marc.molina.pena@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.104 galileo_e5a_noncoherent_iq_acquisition_caf_cc.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E5a data and pilot Signals.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include <gnuradio/block.h>
#include <gnuradio/fft/fft.h>
#include <gnuradio/gr_complex.h>
#include <fstream>
#include <memory>
#include <string>
#include <utility>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [galileo_e5a_noncoherentIQ_acquisition_caf_cc](#)
This class implements a Parallel Code Phase Search Acquisition.

Typedefs

- using [galileo_e5a_noncoherentIQ_acquisition_caf_cc_sptr](#) = boost::shared_ptr< [galileo_e5a_noncoherentIQ_acquisition_caf_cc](#) >

Functions

- [galileo_e5a_noncoherentIQ_acquisition_caf_cc_sptr](#) [galileo_e5a_noncoherentIQ_make_acquisition_caf_cc](#) (unsigned int sampled_ms, unsigned int max_dwells, unsigned int doppler_max, int64_t fs_in, int samples_per_ms, int samples_per_code, bool bit_transition_flag, bool dump, const std::string &dump_filename, bool both_signal_components_, int CAF_window_hz_, int Zero_padding_)

11.104.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E5a data and pilot Signals.

Author

Marc Sales, 2014. [marcsales92\(at\)gmail.com](mailto:marcsales92(at)gmail.com) on work from:

- Javier Arribas, 2011. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)
- Luis Esteve, 2012. [luis\(at\)epsilon-formacion.com](mailto:luis(at)epsilon-formacion.com)
- Marc Molina, 2013. marc.molina.pena@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.105 galileo_e5a_pcps_acquisition.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E5a data and pilot Signals.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include "pcps_acquisition.h"
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GalileoE5aPcpsAcquisition](#)

11.105.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E5a data and pilot Signals.

Antonio Ramos, 2018. antonio.ramos(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.106 galileo_e5a_pcps_acquisition_fpga.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E5a data and pilot Signals for the FPGA.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include "pcps_acquisition_fpga.h"
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GalileoE5aPcpsAcquisitionFpga](#)

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for Galileo E5a signals.

11.106.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E5a data and pilot Signals for the FPGA.

Author

Marc Majoral, 2019. mmajoral(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.107 galileo_e5a_telemetry_decoder.h File Reference

Interface of an adapter of a GALILEO E5a FNAV data decoder block to a [TelemetryDecoderInterface](#).

```
#include "galileo_telemetry_decoder_gs.h"
#include "gnss_satellite.h"
#include "gnss_synchro.h"
#include "telemetry_decoder_interface.h"
#include <gnuradio/runtime_types.h>
#include <cstdio>
#include <string>
```

Classes

- class [GalileoE5aTelemetryDecoder](#)

This class implements a NAV data decoder for Galileo INAV frames in E1B radio link.

11.107.1 Detailed Description

Interface of an adapter of a GALILEO E5a FNAV data decoder block to a [TelemetryDecoderInterface](#).

Author

Marc Sales, 2014. marcsales92(at)gmail.com on work from:

- Javier Arribas, 2011. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.108 Galileo_E5b.h File Reference

Defines system parameters for Galileo E5b signal and NAV data.

```
#include "MATH_CONSTANTS.h"
#include "gnss_frequencies.h"
#include <cstdio>
#include <stdint>
```

Variables

- constexpr double `GALILEO_E5B_FREQ_HZ` = `FREQ7`
Galileo E5b carrier frequency [Hz].
- constexpr double `GALILEO_E5B_CODE_CHIP_RATE_CPS` = `1.023e7`
Galileo E5b code rate [chips/s].
- constexpr double `GALILEO_E5B_I_TIERED_CODE_PERIOD_S` = `0.004`
Galileo E5b-I tiered code period [s].
- constexpr double `GALILEO_E5B_Q_TIERED_CODE_PERIOD_S` = `0.100`
Galileo E5b-Q tiered code period [s].
- constexpr double `GALILEO_E5B_CODE_PERIOD_S` = `0.001`
Galileo E5b primary code period [s].
- constexpr int32_t `GALILEO_E5B_CODE_PERIOD_MS` = `1`
Galileo E5b primary code period [ms].
- constexpr int32_t `GALILEO_E5B_CODE_LENGTH_CHIPS` = `10230`
Galileo E5b primary code length [chips].
- constexpr int32_t `GALILEO_E5B_I_SECONDARY_CODE_LENGTH` = `4`
Galileo E5b-I secondary code length [chips].
- constexpr int32_t `GALILEO_E5B_Q_SECONDARY_CODE_LENGTH` = `100`
Galileo E5b-Q secondary code length [chips].
- constexpr int32_t `GALILEO_E5B_SYMBOL_RATE_BPS` = `250`
Galileo E5b symbol rate [bits/second].
- constexpr int32_t `GALILEO_E5B_NUMBER_OF_CODES` = `50`
- constexpr int32_t `GALILEO_E5B_HISTORY_DEEP` = `100`
- constexpr uint32_t `GALILEO_E5B_OPT_ACQ_FS_SPS` = `10000000`
Sampling frequency that maximizes the acquisition SNR while using a non-multiple of chip rate.
- constexpr char `GALILEO_E5B_I_SECONDARY_CODE` [5] = `"1110"`
- constexpr size_t `GALILEO_E5B_I_PRIMARY_CODE_STR_LENGTH` = `2558`
- constexpr char `GALILEO_E5B_I_PRIMARY_CODE` [GALILEO_E5B_NUMBER_OF_CODES][2559]
- constexpr size_t `GALILEO_E5B_Q_PRIMARY_CODE_STR_LENGTH` = `2558`
- constexpr char `GALILEO_E5B_Q_PRIMARY_CODE` [GALILEO_E5B_NUMBER_OF_CODES][2559]
- constexpr size_t `GALILEO_E5B_Q_SECONDARY_CODE_STR_LENGTH` = `100`
- constexpr char `GALILEO_E5B_Q_SECONDARY_CODE` [GALILEO_E5B_NUMBER_OF_CODES][101]

11.108.1 Detailed Description

Defines system parameters for Galileo E5b signal and NAV data.

Author

Piyush Gupta, 2020. piyush04111999@gmail.com

Note

Code added as part of GSoC 2020 program.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.108.2 Variable Documentation

11.108.2.1 GALILEO_E5B_CODE_CHIP_RATE_CPS

```
constexpr double GALILEO_E5B_CODE_CHIP_RATE_CPS = 1.023e7
```

Galileo E5b code rate [chips/s].

Definition at line 32 of file Galileo_E5b.h.

11.108.2.2 GALILEO_E5B_CODE_LENGTH_CHIPS

```
constexpr int32_t GALILEO_E5B_CODE_LENGTH_CHIPS = 10230
```

Galileo E5b primary code length [chips].

Definition at line 37 of file Galileo_E5b.h.

11.108.2.3 GALILEO_E5B_CODE_PERIOD_MS

```
constexpr int32_t GALILEO_E5B_CODE_PERIOD_MS = 1
```

Galileo E5b primary code period [ms].

Definition at line 36 of file Galileo_E5b.h.

11.108.2.4 GALILEO_E5B_CODE_PERIOD_S

```
constexpr double GALILEO_E5B_CODE_PERIOD_S = 0.001
```

Galileo E5b primary code period [s].

Definition at line 35 of file Galileo_E5b.h.

11.108.2.5 GALILEO_E5B_FREQ_HZ

```
constexpr double GALILEO_E5B_FREQ_HZ = FREQ7
```

Galileo E5b carrier frequency [Hz].

Definition at line 31 of file Galileo_E5b.h.

11.108.2.6 GALILEO_E5B_I_SECONDARY_CODE_LENGTH

```
constexpr int32_t GALILEO_E5B_I_SECONDARY_CODE_LENGTH = 4
```

Galileo E5b-I secondary code length [chips].

Definition at line 38 of file Galileo_E5b.h.

11.108.2.7 GALILEO_E5B_I_TIERED_CODE_PERIOD_S

```
constexpr double GALILEO_E5B_I_TIERED_CODE_PERIOD_S = 0.004
```

Galileo E5b-I tiered code period [s].

Definition at line 33 of file Galileo_E5b.h.

11.108.2.8 GALILEO_E5B_OPT_ACQ_FS_SPS

```
constexpr uint32_t GALILEO_E5B_OPT_ACQ_FS_SPS = 10000000
```

Sampling frequency that maximizes the acquisition SNR while using a non-multiple of chip rate.

Definition at line 48 of file Galileo_E5b.h.

11.108.2.9 GALILEO_E5B_Q_SECONDARY_CODE_LENGTH

```
constexpr int32_t GALILEO_E5B_Q_SECONDARY_CODE_LENGTH = 100
```

Galileo E5b-Q secondary code length [chips].

Definition at line 39 of file Galileo_E5b.h.

11.108.2.10 GALILEO_E5B_Q_TIERED_CODE_PERIOD_S

```
constexpr double GALILEO_E5B_Q_TIERED_CODE_PERIOD_S = 0.100
```

Galileo E5b-Q tiered code period [s].

Definition at line 34 of file Galileo_E5b.h.

11.108.2.11 GALILEO_E5B_SYMBOL_RATE_BPS

```
constexpr int32_t GALILEO_E5B_SYMBOL_RATE_BPS = 250
```

Galileo E5b symbol rate [bits/second].

Definition at line 40 of file Galileo_E5b.h.

11.109 galileo_e5b_dll_pll_tracking.h File Reference

Adapts a code DLL + carrier PLL tracking block to a [TrackingInterface](#) for Galileo E5b signals.

```
#include "dll_pll_veml_tracking.h"  
#include "tracking_interface.h"  
#include <string>
```

Classes

- class [GalileoE5bDIIPIITracking](#)
This class implements a code DLL + carrier PLL tracking loop.

11.109.1 Detailed Description

Adapts a code DLL + carrier PLL tracking block to a [TrackingInterface](#) for Galileo E5b signals.

Author

Piyush Gupta, 2020. piyush04111999@gmail.com on work from:

- Javier Arribas, 2011. jarribas@cttc.es
- Luis Esteve, 2012. luis@epsilon-formacion.com
- Marc Sales, 2014. marcsales92@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.110 galileo_e5b_pcps_acquisition.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E5b data and pilot Signals.

```
#include "channel_fsm.h"  
#include "gnss_synchro.h"  
#include "pcps_acquisition.h"  
#include <memory>  
#include <string>  
#include <vector>
```

Classes

- class [GalileoE5bPcpsAcquisition](#)

11.110.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E5b data and pilot Signals.

Author

Piyush Gupta, 2020. piyush04111999@gmail.com

Note

Code added as part of GSoC 2020 program.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.111 galileo_e5b_pcps_acquisition_fpga.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E5b data and pilot Signals for the FPGA.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include "pcps_acquisition_fpga.h"
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GalileoE5bPcpsAcquisitionFpga](#)

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for Galileo E5b signals.

11.111.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Galileo E5b data and pilot Signals for the FPGA.

Author

Piyush Gupta, 2020. piyush04111999@gmail.com

Note

Code added as part of GSoC 2020 Program.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.112 galileo_e5b_telemetry_decoder.h File Reference

Interface of an adapter of a GALILEO E5B NAV data decoder block to a [TelemetryDecoderInterface](#).

```
#include "galileo_telemetry_decoder_gs.h"
#include "gnss_satellite.h"
#include "gnss_synchro.h"
#include "telemetry_decoder_interface.h"
#include <gnuradio/runtime_types.h>
#include <cstdio>
#include <string>
```

Classes

- class [GalileoE5bTelemetryDecoder](#)

This class implements a NAV data decoder for Galileo INAV frames in E5b radio link.

11.112.1 Detailed Description

Interface of an adapter of a GALILEO E5B NAV data decoder block to a [TelemetryDecoderInterface](#).

Author

Piyush Gupta 2020 piyush04111999@gmail.com.

Note

Code added as part of GSoC 2020 Program.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.113 galileo_ephemeris.h File Reference

Interface of a Galileo EPHEMERIS storage.

```
#include <boost/serialization/nvp.hpp>
#include <cstdint>
```

Classes

- class [Galileo_Ephemeris](#)

This class is a storage and orbital model functions for the Galileo SV ephemeris data as described in Galileo ICD paragraph 5.1.1 (See <https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf>)

11.113.1 Detailed Description

Interface of a Galileo EPHEMERIS storage.

Author

Javier Arribas, 2013. [jarribas\(at\)cttc.es](mailto:jarribas@cttc.es),
Mara Branzanti 2013. [mara.branzanti\(at\)gmail.com](mailto:mara.branzanti@gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.114 Galileo_FNAV.h File Reference

Galileo FNAV message constants.

```
#include "MATH_CONSTANTS.h"
#include <cstdint>
#include <utility>
#include <vector>
```

Functions

- `const std::vector< std::pair< int32_t, int32_t > > FNAV_PAGE_TYPE_BIT ({1, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_SV_ID_PRN_1_BIT ({7, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_IO_DNAV_1_BIT ({13, 10})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_T0C_1_BIT ({23, 14})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_AF0_1_BIT ({37, 31})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_AF1_1_BIT ({68, 21})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_AF2_1_BIT ({89, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_SISA_1_BIT ({95, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_AI0_1_BIT ({103, 11})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_AI1_1_BIT ({114, 11})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_AI2_1_BIT ({125, 14})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_REGION1_1_BIT ({139, 1})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_REGION2_1_BIT ({140, 1})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_REGION3_1_BIT ({141, 1})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_REGION4_1_BIT ({142, 1})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_REGION5_1_BIT ({143, 1})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_BGD_1_BIT ({144, 10})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_E5AHS_1_BIT ({154, 2})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_WN_1_BIT ({156, 12})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_TOW_1_BIT ({168, 20})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_E5ADVS_1_BIT ({188, 1})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_IO_DNAV_2_BIT ({7, 10})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_M0_2_BIT ({17, 32})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_OMEGADOT_2_BIT ({49, 24})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_E_2_BIT ({73, 32})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_A12_2_BIT ({105, 32})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_OMEGA0_2_BIT ({137, 32})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_IDOT_2_BIT ({169, 14})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_WN_2_BIT ({183, 12})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_TOW_2_BIT ({195, 20})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_IO_DNAV_3_BIT ({7, 10})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_I0_3_BIT ({17, 32})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_W_3_BIT ({49, 32})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_DELTAN_3_BIT ({81, 16})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_CUC_3_BIT ({97, 16})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_CUS_3_BIT ({113, 16})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_CRC_3_BIT ({129, 16})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_CRS_3_BIT ({145, 16})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_T0E_3_BIT ({161, 14})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_WN_3_BIT ({175, 12})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_TOW_3_BIT ({187, 20})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_IO_DNAV_4_BIT ({7, 10})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_CIC_4_BIT ({17, 16})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_CIS_4_BIT ({33, 16})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_A0_4_BIT ({49, 32})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_A1_4_BIT ({81, 24})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_DELTATLS_4_BIT ({105, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_T0T_4_BIT ({113, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_W_NOT_4_BIT ({121, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_W_NLSF_4_BIT ({129, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_DN_4_BIT ({137, 3})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_DELTATLSF_4_BIT ({140, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_T0G_4_BIT ({148, 8})`

- `const std::vector< std::pair< int32_t, int32_t > > FNAV_A0G_4_BIT {{{156, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_A1G_4_BIT {{{172, 12}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_W_N0G_4_BIT {{{184, 6}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_TOW_4_BIT {{{190, 20}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_IO_DA_5_BIT {{{7, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_W_NA_5_BIT {{{11, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_T0A_5_BIT {{{13, 10}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_SVI_D1_5_BIT {{{23, 6}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_DELTA12_1_5_BIT {{{29, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_E_1_5_BIT {{{42, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_W_1_5_BIT {{{53, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_DELTA1_1_5_BIT {{{69, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_OMEGA0_1_5_BIT {{{80, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_OMEGADOT_1_5_BIT {{{96, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_M0_1_5_BIT {{{107, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_AF0_1_5_BIT {{{123, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_AF1_1_5_BIT {{{139, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_E5AHS_1_5_BIT {{{152, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_SVI_D2_5_BIT {{{154, 6}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_DELTA12_2_5_BIT {{{160, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_E_2_5_BIT {{{173, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_W_2_5_BIT {{{184, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_DELTA1_2_5_BIT {{{200, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_IO_DA_6_BIT {{{7, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_OMEGADOT_2_6_BIT {{{23, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_M0_2_6_BIT {{{34, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_AF0_2_6_BIT {{{50, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_AF1_2_6_BIT {{{66, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_E5AHS_2_6_BIT {{{79, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_SVI_D3_6_BIT {{{81, 6}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_DELTA12_3_6_BIT {{{87, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_E_3_6_BIT {{{100, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_W_3_6_BIT {{{111, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_DELTA1_3_6_BIT {{{127, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_OMEGA0_3_6_BIT {{{138, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_OMEGADOT_3_6_BIT {{{154, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_M0_3_6_BIT {{{165, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_AF0_3_6_BIT {{{181, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_AF1_3_6_BIT {{{197, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > FNAV_E5AHS_3_6_BIT {{{210, 2}}}`

Variables

- `constexpr int32_t FNAV_T0C_1_LSB = 60`
- `constexpr double FNAV_AF0_1_LSB = TWO_N34`
- `constexpr double FNAV_AF1_1_LSB = TWO_N46`
- `constexpr double FNAV_AF2_1_LSB = TWO_N59`
- `constexpr double FNAV_AI0_1_LSB = TWO_N2`
- `constexpr double FNAV_AI1_1_LSB = TWO_N8`
- `constexpr double FNAV_AI2_1_LSB = TWO_N15`
- `constexpr double FNAV_BGD_1_LSB = TWO_N32`
- `constexpr double FNAV_M0_2_LSB = PI_TWO_N31`
- `constexpr double FNAV_OMEGADOT_2_LSB = PI_TWO_N43`
- `constexpr double FNAV_E_2_LSB = TWO_N33`

- constexpr double **FNAV_A12_2_LSB** = [TWO_N19](#)
- constexpr double **FNAV_OMEGA0_2_LSB** = [PI_TWO_N31](#)
- constexpr double **FNAV_IDOT_2_LSB** = [PI_TWO_N43](#)
- constexpr double **FNAV_I0_3_LSB** = [PI_TWO_N31](#)
- constexpr double **FNAV_W_3_LSB** = [PI_TWO_N31](#)
- constexpr double **FNAV_DELTAN_3_LSB** = [PI_TWO_N43](#)
- constexpr double **FNAV_CUC_3_LSB** = [TWO_N29](#)
- constexpr double **FNAV_CUS_3_LSB** = [TWO_N29](#)
- constexpr double **FNAV_CRC_3_LSB** = [TWO_N5](#)
- constexpr double **FNAV_CRS_3_LSB** = [TWO_N5](#)
- constexpr int32_t **FNAV_T0E_3_LSB** = 60
- constexpr double **FNAV_CIC_4_LSB** = [TWO_N29](#)
- constexpr double **FNAV_CIS_4_LSB** = [TWO_N29](#)
- constexpr double **FNAV_A0_4_LSB** = [TWO_N30](#)
- constexpr double **FNAV_A1_4_LSB** = [TWO_N50](#)
- constexpr int32_t **FNAV_T0T_4_LSB** = 3600
- constexpr int32_t **FNAV_T0G_4_LSB** = 3600
- constexpr double **FNAV_A0G_4_LSB** = [TWO_N35](#)
- constexpr double **FNAV_A1G_4_LSB** = [TWO_N51](#)
- constexpr int32_t **FNAV_T0A_5_LSB** = 600
- constexpr double **FNAV_DELTAA12_5_LSB** = [TWO_N9](#)
- constexpr double **FNAV_E_5_LSB** = [TWO_N16](#)
- constexpr double **FNAV_W_5_LSB** = [TWO_N15](#)
- constexpr double **FNAV_DELTAI_5_LSB** = [TWO_N14](#)
- constexpr double **FNAV_OMEGA0_5_LSB** = [TWO_N15](#)
- constexpr double **FNAV_OMEGADOT_5_LSB** = [TWO_N33](#)
- constexpr double **FNAV_M0_5_LSB** = [TWO_N15](#)
- constexpr double **FNAV_AF0_5_LSB** = [TWO_N19](#)
- constexpr double **FNAV_AF1_5_LSB** = [TWO_N38](#)

11.114.1 Detailed Description

Galileo FNAV message constants.

Author

Carles Fernandez, 2020. [cfernandez\(at\)cttc.es](mailto:cfernandez(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.115 galileo_fnav_message.h File Reference

Implementation of a Galileo F/NAV Data message as described in Galileo OS SIS ICD Issue 1.2 (Nov. 2015)

```
#include "Galileo_E5a.h"
#include "Galileo_FNAV.h"
#include "galileo_almanac_helper.h"
#include "galileo_ephemeris.h"
#include "galileo_iono.h"
#include "galileo_utc_model.h"
#include <bitset>
#include <cstdint>
#include <string>
#include <utility>
#include <vector>
```

Classes

- class [Galileo_Fnav_Message](#)

This class handles the Galileo F/NAV Data message, as described in the Galileo Open Service Signal in Space Interface Control Document (OS SIS ICD), Issue 1.2 (Nov 2015). See <https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf>.

11.115.1 Detailed Description

Implementation of a Galileo F/NAV Data message as described in Galileo OS SIS ICD Issue 1.2 (Nov. 2015)

Author

Marc Sales, 2014. marcsales92(at)gmail.com on work from:

- Javier Arribas, 2011. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.116 Galileo_INAV.h File Reference

Galileo INAV message constants.

```
#include "MATH_CONSTANTS.h"
#include <cstdint>
#include <utility>
#include <vector>
```

Functions

- `const std::vector< std::pair< int32_t, int32_t > > TYPE` ({1, 6}}
- `const std::vector< std::pair< int32_t, int32_t > > PAGE_TYPE_BIT` ({1, 6}}
- `const std::vector< std::pair< int32_t, int32_t > > IOD_NAV_1_BIT` ({7, 10}}
- `const std::vector< std::pair< int32_t, int32_t > > T0_E_1_BIT` ({17, 14}}
- `const std::vector< std::pair< int32_t, int32_t > > M0_1_BIT` ({31, 32}}
- `const std::vector< std::pair< int32_t, int32_t > > E_1_BIT` ({63, 32}}
- `const std::vector< std::pair< int32_t, int32_t > > A_1_BIT` ({95, 32}}
- `const std::vector< std::pair< int32_t, int32_t > > IOD_NAV_2_BIT` ({7, 10}}
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA_0_2_BIT` ({17, 32}}
- `const std::vector< std::pair< int32_t, int32_t > > I_0_2_BIT` ({49, 32}}
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA_2_BIT` ({81, 32}}
- `const std::vector< std::pair< int32_t, int32_t > > I_DOT_2_BIT` ({113, 14}}
- `const std::vector< std::pair< int32_t, int32_t > > IOD_NAV_3_BIT` ({7, 10}}
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA_DOT_3_BIT` ({17, 24}}
- `const std::vector< std::pair< int32_t, int32_t > > DELTA_N_3_BIT` ({41, 16}}
- `const std::vector< std::pair< int32_t, int32_t > > C_UC_3_BIT` ({57, 16}}
- `const std::vector< std::pair< int32_t, int32_t > > C_US_3_BIT` ({73, 16}}
- `const std::vector< std::pair< int32_t, int32_t > > C_RC_3_BIT` ({89, 16}}
- `const std::vector< std::pair< int32_t, int32_t > > C_RS_3_BIT` ({105, 16}}
- `const std::vector< std::pair< int32_t, int32_t > > SISA_3_BIT` ({121, 8}}
- `const std::vector< std::pair< int32_t, int32_t > > IOD_NAV_4_BIT` ({7, 10}}
- `const std::vector< std::pair< int32_t, int32_t > > SV_ID_PRN_4_BIT` ({17, 6}}
- `const std::vector< std::pair< int32_t, int32_t > > C_IC_4_BIT` ({23, 16}}
- `const std::vector< std::pair< int32_t, int32_t > > C_IS_4_BIT` ({39, 16}}
- `const std::vector< std::pair< int32_t, int32_t > > T0C_4_BIT` ({55, 14}}
- `const std::vector< std::pair< int32_t, int32_t > > AF0_4_BIT` ({69, 31}}
- `const std::vector< std::pair< int32_t, int32_t > > AF1_4_BIT` ({100, 21}}
- `const std::vector< std::pair< int32_t, int32_t > > AF2_4_BIT` ({121, 6}}
- `const std::vector< std::pair< int32_t, int32_t > > SPARE_4_BIT` ({127, 2}}
- `const std::vector< std::pair< int32_t, int32_t > > AI0_5_BIT` ({7, 11}}
- `const std::vector< std::pair< int32_t, int32_t > > AI1_5_BIT` ({18, 11}}
- `const std::vector< std::pair< int32_t, int32_t > > AI2_5_BIT` ({29, 14}}
- `const std::vector< std::pair< int32_t, int32_t > > REGION1_5_BIT` ({43, 1}}
- `const std::vector< std::pair< int32_t, int32_t > > REGION2_5_BIT` ({44, 1}}
- `const std::vector< std::pair< int32_t, int32_t > > REGION3_5_BIT` ({45, 1}}
- `const std::vector< std::pair< int32_t, int32_t > > REGION4_5_BIT` ({46, 1}}
- `const std::vector< std::pair< int32_t, int32_t > > REGION5_5_BIT` ({47, 1}}
- `const std::vector< std::pair< int32_t, int32_t > > BGD_E1_E5A_5_BIT` ({48, 10}}
- `const std::vector< std::pair< int32_t, int32_t > > BGD_E1_E5B_5_BIT` ({58, 10}}
- `const std::vector< std::pair< int32_t, int32_t > > E5B_HS_5_BIT` ({68, 2}}
- `const std::vector< std::pair< int32_t, int32_t > > E1_B_HS_5_BIT` ({70, 2}}
- `const std::vector< std::pair< int32_t, int32_t > > E5B_DVS_5_BIT` ({72, 1}}
- `const std::vector< std::pair< int32_t, int32_t > > E1_B_DVS_5_BIT` ({73, 1}}
- `const std::vector< std::pair< int32_t, int32_t > > WN_5_BIT` ({74, 12}}
- `const std::vector< std::pair< int32_t, int32_t > > TOW_5_BIT` ({86, 20}}
- `const std::vector< std::pair< int32_t, int32_t > > SPARE_5_BIT` ({106, 23}}
- `const std::vector< std::pair< int32_t, int32_t > > A0_6_BIT` ({7, 32}}
- `const std::vector< std::pair< int32_t, int32_t > > A1_6_BIT` ({39, 24}}
- `const std::vector< std::pair< int32_t, int32_t > > DELTA_T_LS_6_BIT` ({63, 8}}
- `const std::vector< std::pair< int32_t, int32_t > > T0T_6_BIT` ({71, 8}}
- `const std::vector< std::pair< int32_t, int32_t > > W_NOT_6_BIT` ({79, 8}}
- `const std::vector< std::pair< int32_t, int32_t > > WN_LSF_6_BIT` ({87, 8}}
- `const std::vector< std::pair< int32_t, int32_t > > DN_6_BIT` ({95, 3}}

- `const std::vector< std::pair< int32_t, int32_t > > DELTA_T_LSF_6_BIT {{{98, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > TOW_6_BIT {{{106, 20}}}`
- `const std::vector< std::pair< int32_t, int32_t > > IOD_A_7_BIT {{{7, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > WN_A_7_BIT {{{11, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > T0A_7_BIT {{{13, 10}}}`
- `const std::vector< std::pair< int32_t, int32_t > > SVI_D1_7_BIT {{{23, 6}}}`
- `const std::vector< std::pair< int32_t, int32_t > > DELTA_A_7_BIT {{{29, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > E_7_BIT {{{42, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA_7_BIT {{{53, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > DELTA_I_7_BIT {{{69, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA0_7_BIT {{{80, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA_DOT_7_BIT {{{96, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > M0_7_BIT {{{107, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > IOD_A_8_BIT {{{7, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > AF0_8_BIT {{{11, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > AF1_8_BIT {{{27, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > E5B_HS_8_BIT {{{40, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > E1_B_HS_8_BIT {{{42, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > SVI_D2_8_BIT {{{44, 6}}}`
- `const std::vector< std::pair< int32_t, int32_t > > DELTA_A_8_BIT {{{50, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > E_8_BIT {{{63, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA_8_BIT {{{74, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > DELTA_I_8_BIT {{{90, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA0_8_BIT {{{101, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA_DOT_8_BIT {{{117, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > IOD_A_9_BIT {{{7, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > WN_A_9_BIT {{{11, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > T0A_9_BIT {{{13, 10}}}`
- `const std::vector< std::pair< int32_t, int32_t > > M0_9_BIT {{{23, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > AF0_9_BIT {{{39, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > AF1_9_BIT {{{55, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > E5B_HS_9_BIT {{{68, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > E1_B_HS_9_BIT {{{70, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > SVI_D3_9_BIT {{{72, 6}}}`
- `const std::vector< std::pair< int32_t, int32_t > > DELTA_A_9_BIT {{{78, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > E_9_BIT {{{91, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA_9_BIT {{{102, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > DELTA_I_9_BIT {{{118, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > IOD_A_10_BIT {{{7, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA0_10_BIT {{{11, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA_DOT_10_BIT {{{27, 11}}}`
- `const std::vector< std::pair< int32_t, int32_t > > M0_10_BIT {{{38, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > AF0_10_BIT {{{54, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > AF1_10_BIT {{{70, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > E5B_HS_10_BIT {{{83, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > E1_B_HS_10_BIT {{{85, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > A_0_G_10_BIT {{{87, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > A_1_G_10_BIT {{{103, 12}}}`
- `const std::vector< std::pair< int32_t, int32_t > > T_0_G_10_BIT {{{115, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > WN_0_G_10_BIT {{{123, 6}}}`
- `const std::vector< std::pair< int32_t, int32_t > > TIME_0_BIT {{{7, 2}}}`
- `const std::vector< std::pair< int32_t, int32_t > > WN_0_BIT {{{97, 12}}}`
- `const std::vector< std::pair< int32_t, int32_t > > TOW_0_BIT {{{109, 20}}}`

Variables

- constexpr int32_t **GALILEO_INAV_PREAMBLE_LENGTH_BITS** = 10
- constexpr double **GALILEO_INAV_PAGE_PART_WITH_PREAMBLE_SECONDS** = 2.04
Page Duration + (Galileo I/NAV Preamble bits)(Galileo E5b-I tiered Code Period(seconds))*
- constexpr int32_t **GALILEO_INAV_PREAMBLE_PERIOD_SYMBOLS** = 250
- constexpr int32_t **GALILEO_INAV_PAGE_PART_SYMBOLS** = 250
Each Galileo INAV pages are composed of two parts (even and odd) each of 250 symbols, including preamble. See Galileo ICD 4.3.2.
- constexpr int32_t **GALILEO_INAV_PAGE_SYMBOLS** = 500
The complete Galileo INAV page length.
- constexpr int32_t **GALILEO_INAV_PAGE_PART_SECONDS** = 1
- constexpr int32_t **GALILEO_INAV_PAGE_PART_MS** = 1000
- constexpr int32_t **GALILEO_INAV_PAGE_SECONDS** = 2
- constexpr int32_t **GALILEO_INAV_INTERLEAVER_ROWS** = 8
- constexpr int32_t **GALILEO_INAV_INTERLEAVER_COLS** = 30
- constexpr int32_t **GALILEO_TELEMETRY_RATE_BITS_SECOND** = 250
- constexpr int32_t **GALILEO_PAGE_TYPE_BITS** = 6
- constexpr int32_t **GALILEO_DATA_JK_BITS** = 128
- constexpr int32_t **GALILEO_DATA_FRAME_BITS** = 196
- constexpr int32_t **GALILEO_DATA_FRAME_BYTES** = 25
- constexpr char **GALILEO_INAV_PREAMBLE** [11] = "0101100000"
- constexpr int32_t **T0E_1_LSB** = 60
- constexpr double **M0_1_LSB** = **PI_TWO_N31**
- constexpr double **E_1_LSB** = **TWO_N33**
- constexpr double **A_1_LSB_GAL** = **TWO_N19**
- constexpr double **OMEGA_0_2_LSB** = **PI_TWO_N31**
- constexpr double **I_0_2_LSB** = **PI_TWO_N31**
- constexpr double **OMEGA_2_LSB** = **PI_TWO_N31**
- constexpr double **I_DOT_2_LSB** = **PI_TWO_N43**
- constexpr double **OMEGA_DOT_3_LSB** = **PI_TWO_N43**
- constexpr double **DELTA_N_3_LSB** = **PI_TWO_N43**
- constexpr double **C_UC_3_LSB** = **TWO_N29**
- constexpr double **C_US_3_LSB** = **TWO_N29**
- constexpr double **C_RC_3_LSB** = **TWO_N5**
- constexpr double **C_RS_3_LSB** = **TWO_N5**
- constexpr double **C_IC_4_LSB** = **TWO_N29**
- constexpr double **C_IS_4_LSB** = **TWO_N29**
- constexpr int32_t **T0C_4_LSB** = 60
- constexpr double **AF0_4_LSB** = **TWO_N34**
- constexpr double **AF1_4_LSB** = **TWO_N46**
- constexpr double **AF2_4_LSB** = **TWO_N59**
- constexpr double **AI0_5_LSB** = **TWO_N2**
- constexpr double **AI1_5_LSB** = **TWO_N8**
- constexpr double **AI2_5_LSB** = **TWO_N15**
- constexpr double **BGD_E1_E5A_5_LSB** = **TWO_N32**
- constexpr double **BGD_E1_E5B_5_LSB** = **TWO_N32**
- constexpr double **A0_6_LSB** = **TWO_N30**
- constexpr double **A1_6_LSB** = **TWO_N50**
- constexpr int32_t **T0T_6_LSB** = 3600
- constexpr int32_t **T0A_7_LSB** = 600
- constexpr double **DELTA_A_7_LSB** = **TWO_N9**
- constexpr double **E_7_LSB** = **TWO_N16**
- constexpr double **OMEGA_7_LSB** = **TWO_N15**

- `constexpr double DELTA_I_7_LSB = TWO_N14`
- `constexpr double OMEGA0_7_LSB = TWO_N15`
- `constexpr double OMEGA_DOT_7_LSB = TWO_N33`
- `constexpr double M0_7_LSB = TWO_N15`
- `constexpr double AF0_8_LSB = TWO_N19`
- `constexpr double AF1_8_LSB = TWO_N38`
- `constexpr double DELTA_A_8_LSB = TWO_N9`
- `constexpr double E_8_LSB = TWO_N16`
- `constexpr double OMEGA_8_LSB = TWO_N15`
- `constexpr double DELTA_I_8_LSB = TWO_N14`
- `constexpr double OMEGA0_8_LSB = TWO_N15`
- `constexpr double OMEGA_DOT_8_LSB = TWO_N33`
- `constexpr int32_t T0A_9_LSB = 600`
- `constexpr double M0_9_LSB = TWO_N15`
- `constexpr double AF0_9_LSB = TWO_N19`
- `constexpr double AF1_9_LSB = TWO_N38`
- `constexpr double DELTA_A_9_LSB = TWO_N9`
- `constexpr double E_9_LSB = TWO_N16`
- `constexpr double OMEGA_9_LSB = TWO_N15`
- `constexpr double DELTA_I_9_LSB = TWO_N14`
- `constexpr double OMEGA0_10_LSB = TWO_N15`
- `constexpr double OMEGA_DOT_10_LSB = TWO_N33`
- `constexpr double M0_10_LSB = TWO_N15`
- `constexpr double AF0_10_LSB = TWO_N19`
- `constexpr double AF1_10_LSB = TWO_N38`
- `constexpr double A_0G_10_LSB = TWO_N35`
- `constexpr double A_1G_10_LSB = TWO_N51`
- `constexpr int32_t T_0_G_10_LSB = 3600`

11.116.1 Detailed Description

Galileo INAV message constants.

Author

Carles Fernandez, 2020. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.116.2 Variable Documentation

11.116.2.1 GALILEO_INAV_PAGE_PART_SYMBOLS

```
constexpr int32_t GALILEO_INAV_PAGE_PART_SYMBOLS = 250
```

Each Galileo INAV pages are composed of two parts (even and odd) each of 250 symbols, including preamble. See Galileo ICD 4.3.2.

Definition at line 34 of file Galileo_INAV.h.

11.116.2.2 GALILEO_INAV_PAGE_PART_WITH_PREABLE_SECONDS

```
constexpr double GALILEO_INAV_PAGE_PART_WITH_PREABLE_SECONDS = 2.04
```

Page Duration + (Galileo I/NAV Preamble bits)*(Galileo E5b-I tiered Code Period(seconds))

Definition at line 32 of file Galileo_INAV.h.

11.116.2.3 GALILEO_INAV_PAGE_SYMBOLS

```
constexpr int32_t GALILEO_INAV_PAGE_SYMBOLS = 500
```

The complete Galileo INAV page length.

Definition at line 35 of file Galileo_INAV.h.

11.117 galileo_inav_message.h File Reference

Implementation of a Galileo I/NAV Data message as described in Galileo OS SIS ICD Issue 1.2 (Nov. 2015)

```
#include "Galileo_INAV.h"
#include "galileo_almanac_helper.h"
#include "galileo_ephemeris.h"
#include "galileo_iono.h"
#include "galileo_utc_model.h"
#include <bitset>
#include <cstdint>
#include <string>
#include <utility>
#include <vector>
```

Classes

- class [Galileo_Inav_Message](#)

This class handles the Galileo I/NAV Data message, as described in the Galileo Open Service Signal in Space Interface Control Document (OS SIS ICD), Issue 1.2 (Nov 2015). See <https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf>.

11.117.1 Detailed Description

Implementation of a Galileo I/NAV Data message as described in Galileo OS SIS ICD Issue 1.2 (Nov. 2015)

Author

Mara Branzanti 2013. mara.branzanti(at)gmail.com
Javier Arribas, 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.118 galileo_iono.h File Reference

Interface of a Galileo Ionospheric Model storage.

```
#include <boost/serialization/nvp.hpp>
#include <cstdint>
```

Classes

- class [Galileo_iono](#)

This class is a storage for the GALILEO IONOSPHERIC data as described in Galileo ICD paragraph 5.1.6.

11.118.1 Detailed Description

Interface of a Galileo Ionospheric Model storage.

Author

Javier Arribas, 2013. jarribas(at)cttc.es

Mara Branzanti 2013. mara.branzanti(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.119 galileo_pcps_8ms_acquisition_cc.h File Reference

This class implements a Parallel Code Phase Search Acquisition for Galileo E1 signals with coherent integration time = 8 ms (two codes)

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include <gnuradio/block.h>
#include <gnuradio/fft/fft.h>
#include <gnuradio/gr_complex.h>
#include <fstream>
#include <memory>
#include <string>
#include <utility>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [galileo_pcps_8ms_acquisition_cc](#)

This class implements a Parallel Code Phase Search Acquisition for Galileo E1 signals with coherent integration time = 8 ms (two codes)

Typedefs

- using [galileo_pcps_8ms_acquisition_cc_sptr](#) = boost::shared_ptr< [galileo_pcps_8ms_acquisition_cc](#) >

Functions

- [galileo_pcps_8ms_acquisition_cc_sptr](#) [galileo_pcps_8ms_make_acquisition_cc](#) (uint32_t sampled_ms, uint32_t max_dwells, uint32_t doppler_max, int64_t fs_in, int32_t samples_per_ms, int32_t samples_per_↵ code, bool dump, const std::string &dump_filename)

11.119.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition for Galileo E1 signals with coherent integration time = 8 ms (two codes)

Author

Marc Molina, 2013. marc.molina.pena(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.120 galileo_telemetry_decoder_gs.h File Reference

Implementation of a Galileo unified INAV and FNAV message demodulator block.

```
#include "galileo_fnav_message.h"
#include "galileo_inav_message.h"
#include "gnss_satellite.h"
#include <boost/circular_buffer.hpp>
#include <gnuradio/block.h>
#include <gnuradio/types.h>
#include <array>
#include <cstdint>
#include <fstream>
#include <string>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [galileo_telemetry_decoder_gs](#)

This class implements a block that decodes the INAV and FNAV data defined in Galileo ICD.

Typedefs

- using [galileo_telemetry_decoder_gs_sptr](#) = boost::shared_ptr< [galileo_telemetry_decoder_gs](#) >

Functions

- [galileo_telemetry_decoder_gs_sptr](#) [galileo_make_telemetry_decoder_gs](#) (const [Gnss_Satellite](#) &satellite, int frame_type, bool dump)

11.120.1 Detailed Description

Implementation of a Galileo unified INAV and FNAV message demodulator block.

Author

Javier Arribas 2018. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.121 galileo_utc_model.h File Reference

Interface of a Galileo UTC MODEL storage.

```
#include <boost/serialization/nvp.hpp>
#include <cstdint>
```

Classes

- class [Galileo_Utc_Model](#)

This class is a storage for the GALILEO UTC MODEL data as described in Galileo ICD <https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo-OS-SIS-ICD.pdf> paragraph 5.1.7.

11.121.1 Detailed Description

Interface of a Galileo UTC MODEL storage.

Author

Javier Arribas, 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.122 gen_signal_source.h File Reference

It wraps blocks that generates synthesized GNSS signal and filters it.

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include <pmt/pmt.h>
#include <memory>
#include <string>
```

Classes

- class [GenSignalSource](#)

This class wraps blocks that generates synthesized GNSS signal and filters the signal.

11.122.1 Detailed Description

It wraps blocks that generates synthesized GNSS signal and filters it.

Author

Marc Molina, 2013. marc.molina.pena@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.123 geofunctions.h File Reference

A set of coordinate transformations functions and helpers, some of them migrated from MATLAB, for geographic information systems.

```
#include <armadillo>
```

Functions

- arma::mat [Skew_symmetric](#) (const arma::vec &a)
Calculates skew-symmetric matrix.
- double **WGS84_g0** (double Lat_rad)
- double **WGS84_geocentric_radius** (double Lat_geodetic_rad)
- int [topocent](#) (double *Az, double *El, double *D, const arma::vec &x, const arma::vec &dx)
Transformation of vector dx into topocentric coordinate system with origin at x Inputs: x - vector origin coordinates (in ECEF system [X; Y; Z;]) dx - vector ([dX; dY; dZ;]).
- int [togeod](#) (double *dphi, double *dlambda, double *h, double a, double finv, double X, double Y, double Z)
Subroutine to calculate geodetic coordinates latitude, longitude, height given Cartesian coordinates X,Y,Z, and reference ellipsoid values semi-major axis (a) and the inverse of flattening (finv).
- arma::vec [Gravity_ECEF](#) (const arma::vec &r_eb_e)
Calculates acceleration due to gravity resolved about ECEF-frame.
- arma::vec [cart2geo](#) (const arma::vec &XYZ, int ellipsoid_selection)
Conversion of Cartesian coordinates (X,Y,Z) to geographical coordinates (latitude, longitude, h) on a selected reference ellipsoid.
- arma::vec **LLH_to_deg** (const arma::vec &LLH)
- double **degtorad** (double angleInDegrees)
- double **radtodeg** (double angleInRadians)
- double **mstoknotsh** (double MetersPerSeconds)
- double **mstokph** (double MetersPerSeconds)
- arma::vec **CTM_to_Euler** (const arma::mat &C)
- arma::mat **Euler_to_CTM** (const arma::vec &eul)
- void **ECEF_to_Geo** (const arma::vec &r_eb_e, const arma::vec &v_eb_e, const arma::mat &C_b_e, arma::vec &LLH, arma::vec &v_eb_n, arma::mat &C_b_n)
- void [Geo_to_ECEF](#) (const arma::vec &LLH, const arma::vec &v_eb_n, const arma::mat &C_b_n, arma::vec &r_eb_e, arma::vec &v_eb_e, arma::mat &C_b_e)
From Geographic to ECEF coordinates.
- void [pv_Geo_to_ECEF](#) (double L_b, double lambda_b, double h_b, const arma::vec &v_eb_n, arma::vec &r_eb_e, arma::vec &v_eb_e)
Converts curvilinear to Cartesian position and velocity resolving axes from NED to ECEF This function created 11/4/2012 by Paul Groves.
- double [great_circle_distance](#) (double lat1, double lon1, double lat2, double lon2)
The Haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes.
- void [cart2utm](#) (const arma::vec &r_eb_e, int zone, arma::vec &r_enu)
Transformation of ECEF (X,Y,Z) to (E,N,U) in UTM, zone 'zone'.
- int [findUtmZone](#) (double latitude_deg, double longitude_deg)
Function finds the UTM zone number for given longitude and latitude.
- double **clsin** (const arma::colvec &ar, int degree, double argument)
Clenshaw summation of sinus of argument.
- void **clksin** (const arma::colvec &ar, int degree, double arg_real, double arg_imag, double *re, double *im)
Clenshaw summation of sinus with complex argument.

11.123.1 Detailed Description

A set of coordinate transformations functions and helpers, some of them migrated from MATLAB, for geographic information systems.

Author

Javier Arribas, 2018. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.123.2 Function Documentation**11.123.2.1 cart2geo()**

```
arma::vec cart2geo (
    const arma::vec & XYZ,
    int ellipsoid_selection )
```

Conversion of Cartesian coordinates (X,Y,Z) to geographical coordinates (latitude, longitude, h) on a selected reference ellipsoid.

Choices of Reference Ellipsoid for Geographical Coordinates 0. International Ellipsoid 1924

1. International Ellipsoid 1967
2. World Geodetic System 1972
3. Geodetic Reference System 1980
4. World Geodetic System 1984

11.123.2.2 cart2utm()

```
void cart2utm (
    const arma::vec & r_eb_e,
    int zone,
    arma::vec & r_enu )
```

Transformation of ECEF (X,Y,Z) to (E,N,U) in UTM, zone 'zone'.

11.123.2.3 clksin()

```
void clksin (
    const arma::colvec & ar,
    int degree,
    double arg_real,
    double arg_imag,
    double * re,
    double * im )
```

Clenshaw summation of sinus with complex argument.

11.123.2.4 clsin()

```
double clsin (
    const arma::colvec & ar,
    int degree,
    double argument )
```

Clenshaw summation of sinus of argument.

11.123.2.5 findUtmZone()

```
int findUtmZone (
    double latitude_deg,
    double longitude_deg )
```

Function finds the UTM zone number for given longitude and latitude.

11.123.2.6 Geo_to_ECEF()

```
void Geo_to_ECEF (
    const arma::vec & LLH,
    const arma::vec & v_eb_n,
    const arma::mat & C_b_n,
    arma::vec & r_eb_e,
    arma::vec & v_eb_e,
    arma::mat & C_b_e )
```

From Geographic to ECEF coordinates.

Inputs: LLH latitude (rad), longitude (rad), height (m) v_{eb_n} velocity of body frame w.r.t. ECEF frame, resolved along north, east, and down (m/s) C_{b_n} body-to-NED coordinate transformation matrix

Outputs: r_{eb_e} Cartesian position of body frame w.r.t. ECEF frame, resolved along ECEF-frame axes (m) v_{eb_e} velocity of body frame w.r.t. ECEF frame, resolved along ECEF-frame axes (m/s) C_{b_e} body-to-ECEF-frame coordinate transformation matrix

11.123.2.7 Gravity_ECEF()

```
arma::vec Gravity_ECEF (
    const arma::vec & r_eb_e )
```

Calculates acceleration due to gravity resolved about ECEF-frame.

11.123.2.8 great_circle_distance()

```
double great_circle_distance (
    double lat1,
    double lon1,
    double lat2,
    double lon2 )
```

The Haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes.

11.123.2.9 pv_Geo_to_ECEF()

```
void pv_Geo_to_ECEF (
    double L_b,
    double lambda_b,
    double h_b,
    const arma::vec & v_eb_n,
    arma::vec & r_eb_e,
    arma::vec & v_eb_e )
```

Converts curvilinear to Cartesian position and velocity resolving axes from NED to ECEF This function created 11/4/2012 by Paul Groves.

Inputs: L_b latitude (rad) lambda_b longitude (rad) h_b height (m) v_eb_n velocity of body frame w.r.t. ECEF frame, resolved along north, east, and down (m/s)

Outputs: r_eb_e Cartesian position of body frame w.r.t. ECEF frame, resolved along ECEF-frame axes (m) v_eb_e velocity of body frame w.r.t. ECEF frame, resolved along ECEF-frame axes (m/s)

11.123.2.10 Skew_symmetric()

```
arma::mat Skew_symmetric (
    const arma::vec & a )
```

Calculates skew-symmetric matrix.

11.123.2.11 togeod()

```
int togeod (
    double * dphi,
    double * dlambd,
    double * h,
    double a,
    double finv,
    double X,
    double Y,
    double Z )
```

Subroutine to calculate geodetic coordinates latitude, longitude, height given Cartesian coordinates X,Y,Z, and reference ellipsoid values semi-major axis (a) and the inverse of flattening (finv).

The output units of angular quantities will be in decimal degrees (15.5 degrees not 15 deg 30 min). The output units of h will be the same as the units of X,Y,Z,a.

Inputs:

a	- semi-major axis of the reference ellipsoid
finv	- inverse of flattening of the reference ellipsoid
X,Y,Z	- Cartesian coordinates

Outputs:

dphi	- latitude
dlambda	- longitude
h	- height above reference ellipsoid

Based in a Matlab function by Kai Borre

11.123.2.12 topocent()

```
int topocent (
    double * Az,
    double * El,
    double * D,
    const arma::vec & x,
    const arma::vec & dx )
```

Transformation of vector dx into topocentric coordinate system with origin at x Inputs: x - vector origin coordinates (in ECEF system [X; Y; Z;]) dx - vector ([dX; dY; dZ;]).

Outputs: D - vector length. Units like the input Az - azimuth from north positive clockwise, degrees El - elevation angle, degrees

Based on a Matlab function by Kai Borre

11.124 geojson_printer.h File Reference

Interface of a class that prints PVT solutions in GeoJSON format.

```
#include <fstream>
#include <string>
```

Classes

- class [GeoJSON_Printer](#)
Prints PVT solutions in GeoJSON format file.

11.124.1 Detailed Description

Interface of a class that prints PVT solutions in GeoJSON format.

Author

Carles Fernandez-Prades, 2015. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.125 glonass_gnav_almanac.h File Reference

Interface of a GLONASS GNAV ALMANAC storage.

```
#include <boost/serialization/nvp.hpp>
#include <cstdint>
```

Classes

- class [Glonass_Gnav_Almanac](#)
This class is a storage for the GLONASS SV ALMANAC data as described GLONASS ICD (Edition 5.1)

11.125.1 Detailed Description

Interface of a GLONASS GNAV ALMANAC storage.

Note

Code added as part of GSoC 2017 program

Author

Damian Miralles, 2017. dmiralles2009(at)gmail.com

See also

[GLONASS ICD](#)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.126 glonass_gnav_ephemeris.h File Reference

Interface of a GLONASS EPHEMERIS storage.

```
#include "glonass_gnav_utc_model.h"
#include <boost/date_time/posix_time/ptime.hpp>
#include <boost/serialization/nvp.hpp>
#include <cstdint>
```

Classes

- class [Glonass_Gnav_Ephemeris](#)

This class is a storage and orbital model functions for the GLONASS SV ephemeris data as described in GLONASS ICD (Edition 5.1)

11.126.1 Detailed Description

Interface of a GLONASS EPHEMERIS storage.

Note

Code added as part of GSoC 2017 program

Author

Damian Miralles, 2017. dmiralles2009(at)gmail.com

See also

[GLONASS ICD](#)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.127 glonass_gnav_navigation_message.h File Reference

Interface of a GLONASS GNAV Data message decoder as described in GLONASS ICD (Edition 5.1)

```
#include "GLONASS_L1_L2_CA.h"
#include "glonass_gnav_almanac.h"
#include "glonass_gnav_ephemeris.h"
#include "glonass_gnav_utc_model.h"
#include <bitset>
#include <cstdint>
#include <map>
#include <string>
#include <utility>
#include <vector>
```

Classes

- class [Glonass_Gnav_Navigation_Message](#)

This class decodes a GLONASS GNAV Data message as described in GLONASS ICD (Edition 5.1)

11.127.1 Detailed Description

Interface of a GLONASS GNAV Data message decoder as described in GLONASS ICD (Edition 5.1)

Note

Code added as part of GSoC 2017 program

Author

Damian Miralles, 2017. dmiralles2009(at)gmail.com

See also

[GLONASS ICD](#)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.128 glonass_gnav_utc_model.h File Reference

Interface of a GLONASS GNAV UTC MODEL storage.

```
#include <boost/serialization/nvp.hpp>
#include <cstdint>
```

Classes

- class [Glonass_Gnav_Utc_Model](#)

This class is a storage for the GLONASS GNAV UTC MODEL data as described in GLONASS ICD (Edition 5.1)

11.128.1 Detailed Description

Interface of a GLONASS GNAV UTC MODEL storage.

Note

Code added as part of GSoC 2017 program

Author

Damian Miralles, 2017. dmiralles2009(at)gmail.com

See also

[GLONASS ICD](#)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.129 glonass_l1_ca_dll_pll_c_aid_tracking.h File Reference

Interface of an adapter of a DLL+PLL tracking loop block for Glonass L1 C/A to a [TrackingInterface](#).

```
#include "glonass_l1_ca_dll_pll_c_aid_tracking_cc.h"
#include "glonass_l1_ca_dll_pll_c_aid_tracking_sc.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GlonassL1CaDllPllCAidTracking](#)
This class implements a code DLL + carrier PLL tracking loop.

11.129.1 Detailed Description

Interface of an adapter of a DLL+PLL tracking loop block for Glonass L1 C/A to a [TrackingInterface](#).

Author

Gabriel Araujo, 2017. gabriel.araujo.5000(at)gmail.com
Luis Esteve, 2017. luis(at)epsilon-formacion.com
Damian Miralles, 2017. dmiralles2009(at)gmail.com

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkha user, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.130 glonass_l1_ca_dll_pll_c_aid_tracking_cc.h File Reference

Implementation of a code DLL + carrier PLL tracking block.

```
#include "gnss_synchro.h"
#include "tracking_2nd_DLL_filter.h"
#include "tracking_FLL_PLL_filter.h"
#include "cpu_multicorrelator.h"
#include <gnuradio/block.h>
#include <pmt/pmt.h>
#include <volk_gnssssdr/volk_gnssssdr_alloc.h>
#include <deque>
#include <fstream>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [glonass_l1_ca_dll_pll_c_aid_tracking_cc](#)
This class implements a DLL + PLL tracking loop block.

Typedefs

- using [glonass_l1_ca_dll_pll_c_aid_tracking_cc_sptr](#) = boost::shared_ptr< [glonass_l1_ca_dll_pll_c_aid_tracking_cc](#) >

Functions

- [glonass_l1_ca_dll_pll_c_aid_tracking_cc_sptr](#) [glonass_l1_ca_dll_pll_c_aid_make_tracking_cc](#) (int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float pll_bw_hz, float dll_bw_hz, float pll_bw_narrow_hz, float dll_bw_narrow_hz, int32_t extend_correlation_ms, float early_late_space_chips)

11.130.1 Detailed Description

Implementation of a code DLL + carrier PLL tracking block.

Author

Gabriel Araujo, 2017. gabriel.araujo.5000(at)gmail.com
Luis Esteve, 2017. luis(at)epsilon-formacion.com
Damian Miralles, 2017. dmiralles2009(at)gmail.com

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkha user, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.131 glonass_l1_ca_dll_pll_c_aid_tracking_sc.h File Reference

Implementation of a code DLL + carrier PLL tracking block.

```
#include "cpu_multicorrelator_16sc.h"
#include "glonass_l1_signal_processing.h"
#include "gnss_synchro.h"
#include "tracking_2nd_DLL_filter.h"
#include "tracking_FLL_PLL_filter.h"
#include <gnuradio/block.h>
#include <volk_gnssssdr/volk_gnssssdr.h>
#include <volk_gnssssdr/volk_gnssssdr_alloc.h>
#include <deque>
#include <fstream>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [glonass_l1_ca_dll_pll_c_aid_tracking_sc](#)
This class implements a DLL + PLL tracking loop block.

Typedefs

- using [glonass_l1_ca_dll_pll_c_aid_tracking_sc_sptr](#) = boost::shared_ptr< [glonass_l1_ca_dll_pll_c_aid_tracking_sc](#) >

Functions

- [glonass_l1_ca_dll_pll_c_aid_tracking_sc_sptr glonass_l1_ca_dll_pll_c_aid_make_tracking_sc](#) (int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float pll_bw_hz, float dll_bw_hz, float pll_bw_narrow_hz, float dll_bw_narrow_hz, int32_t extend_correlation_ms, float early_late_space_chips)

11.131.1 Detailed Description

Implementation of a code DLL + carrier PLL tracking block.

Author

Gabriel Araujo, 2017. gabriel.araujo.5000(at)gmail.com
Luis Esteve, 2017. luis(at)epsilon-formacion.com
Damian Miralles, 2017. dmiralles2009(at)gmail.com

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkha user, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.132 glonass_l1_ca_dll_pll_tracking.h File Reference

Interface of an adapter of a DLL+PLL tracking loop block for Glonass L1 C/A to a [TrackingInterface](#).

```
#include "glonass_l1_ca_dll_pll_tracking_cc.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GlonassL1CaDllPllTracking](#)
This class implements a code DLL + carrier PLL tracking loop.

11.132.1 Detailed Description

Interface of an adapter of a DLL+PLL tracking loop block for Glonass L1 C/A to a [TrackingInterface](#).

Author

Gabriel Araujo, 2017. gabriel.araujo.5000(at)gmail.com
Luis Esteve, 2017. luis(at)epsilon-formacion.com

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkha user, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.133 glonass_l1_ca_dll_pll_tracking_cc.h File Reference

Implementation of a code DLL + carrier PLL tracking block.

```
#include "cpu_multicorrelator.h"
#include "gnss_synchro.h"
#include "tracking_2nd_DLL_filter.h"
#include "tracking_2nd_PLL_filter.h"
#include <gnuradio/block.h>
#include <volk_gnssdr/volk_gnssdr_alloc.h>
#include <fstream>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
```


Classes

- class [Glonass_L1_Ca_Dll_Pll_Tracking_cc](#)
This class implements a DLL + PLL tracking loop block.

Typedefs

- using **glonass_l1_ca_dll_pll_tracking_cc_sptr** = boost::shared_ptr< [Glonass_L1_Ca_Dll_Pll_Tracking_cc](#) >

Functions

- `glonass_l1_ca_dll_pll_tracking_cc_sptr glonass_l1_ca_dll_pll_make_tracking_cc (int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float pll_bw_hz, float dll_bw_hz, float early_late_space_chips)`

11.133.1 Detailed Description

Implementation of a code DLL + carrier PLL tracking block.

Author

Gabriel Araujo, 2017. gabriel.araujo.5000(at)gmail.com
Luis Esteve, 2017. luis(at)epsilon-formacion.com
Damian Miralles, 2017. dmiralles2009(at)gmail.com

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkha user, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.134 glonass_l1_ca_pcps_acquisition.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Glonass L1 C/A signals.

```
#include "acq_conf.h"
#include "channel_fsm.h"
#include "complex_byte_to_float_x2.h"
#include "gnss_synchro.h"
#include "pcps_acquisition.h"
#include <gnuradio/blocks/float_to_complex.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GlonassL1CaPcpsAcquisition](#)

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

11.134.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Glonass L1 C/A signals.

Author

Gabriel Araujo, 2017. gabriel.araujo.5000(at)gmail.com

Luis Esteve, 2017. luis(at)epsilon-formacion.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.135 glonass_l1_ca_telemetry_decoder.h File Reference

Interface of an adapter of a GLONASS L1 C/A NAV data decoder block to a [TelemetryDecoderInterface](#).

```
#include "glonass_l1_ca_telemetry_decoder_gs.h"
#include "gnss_satellite.h"
#include "gnss_synchro.h"
#include "telemetry_decoder_interface.h"
#include <gnuradio/runtime_types.h>
#include <cstdio>
#include <string>
```

Classes

- class [GlonassL1CaTelemetryDecoder](#)

This class implements a NAV data decoder for GLONASS L1 C/A.

11.135.1 Detailed Description

Interface of an adapter of a GLONASS L1 C/A NAV data decoder block to a [TelemetryDecoderInterface](#).

Note

Code added as part of GSoC 2017 program

Author

Damian Miralles, 2017. dmiralles2009(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.136 glonass_l1_ca_telemetry_decoder_gs.h File Reference

Implementation of a GLONASS L1 C/A NAV data decoder block.

```
#include "GLONASS_L1_L2_CA.h"
#include "glonass_gnav_navigation_message.h"
#include "gnss_satellite.h"
#include "gnss_synchro.h"
#include <boost/circular_buffer.hpp>
#include <gnuradio/block.h>
#include <gnuradio/types.h>
#include <array>
#include <cstdint>
#include <fstream>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [glonass_l1_ca_telemetry_decoder_gs](#)

This class implements a block that decodes the GNAV data defined in GLONASS ICD v5.1.

Typedefs

- using [glonass_l1_ca_telemetry_decoder_gs_sptr](#) = boost::shared_ptr< [glonass_l1_ca_telemetry_decoder_gs](#) >

Functions

- [glonass_l1_ca_telemetry_decoder_gs_sptr](#) [glonass_l1_ca_make_telemetry_decoder_gs](#) (const [Gnss_Satellite](#) &satellite, bool dump)

11.136.1 Detailed Description

Implementation of a GLONASS L1 C/A NAV data decoder block.

Note

Code added as part of GSoC 2017 program

Author

Damian Miralles, 2017. [dmiralles2009\(at\)gmail.com](mailto:dmiralles2009(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.137 GLONASS_L1_L2_CA.h File Reference

Defines system parameters for GLONASS L1 C/A signal and NAV data.

```
#include "gnss_frequencies.h"
#include <stdint>
#include <map>
#include <utility>
#include <vector>
```

Macros

- `#define GLONASS_GNAV_PREAMBLE`

Functions

- `const std::vector< std::pair< int32_t, int32_t > > STRING_ID ({{2, 4}})`
- `const std::vector< std::pair< int32_t, int32_t > > KX ({{78, 8}})`
- `const std::vector< std::pair< int32_t, int32_t > > P1 ({{8, 2}})`
- `const std::vector< std::pair< int32_t, int32_t > > T_K_HR ({{10, 5}})`
- `const std::vector< std::pair< int32_t, int32_t > > T_K_MIN ({{15, 6}})`
- `const std::vector< std::pair< int32_t, int32_t > > T_K_SEC ({{21, 1}})`
- `const std::vector< std::pair< int32_t, int32_t > > X_N_DOT ({{22, 24}})`
- `const std::vector< std::pair< int32_t, int32_t > > X_N_DOT_DOT ({{46, 5}})`
- `const std::vector< std::pair< int32_t, int32_t > > X_N ({{51, 27}})`
- `const std::vector< std::pair< int32_t, int32_t > > B_N ({{6, 3}})`
- `const std::vector< std::pair< int32_t, int32_t > > P2 ({{9, 1}})`
- `const std::vector< std::pair< int32_t, int32_t > > T_B ({{10, 7}})`
- `const std::vector< std::pair< int32_t, int32_t > > Y_N_DOT ({{22, 24}})`
- `const std::vector< std::pair< int32_t, int32_t > > Y_N_DOT_DOT ({{46, 5}})`
- `const std::vector< std::pair< int32_t, int32_t > > Y_N ({{51, 27}})`
- `const std::vector< std::pair< int32_t, int32_t > > P3 ({{6, 1}})`
- `const std::vector< std::pair< int32_t, int32_t > > GAMMA_N ({{7, 11}})`
- `const std::vector< std::pair< int32_t, int32_t > > P ({{19, 2}})`
- `const std::vector< std::pair< int32_t, int32_t > > EPH_L_N ({{21, 1}})`
- `const std::vector< std::pair< int32_t, int32_t > > Z_N_DOT ({{22, 24}})`
- `const std::vector< std::pair< int32_t, int32_t > > Z_N_DOT_DOT ({{46, 5}})`
- `const std::vector< std::pair< int32_t, int32_t > > Z_N ({{51, 27}})`
- `const std::vector< std::pair< int32_t, int32_t > > TAU_N ({{6, 22}})`
- `const std::vector< std::pair< int32_t, int32_t > > DELTA_TAU_N ({{28, 5}})`
- `const std::vector< std::pair< int32_t, int32_t > > E_N ({{33, 5}})`
- `const std::vector< std::pair< int32_t, int32_t > > P4 ({{52, 1}})`
- `const std::vector< std::pair< int32_t, int32_t > > F_T ({{53, 4}})`
- `const std::vector< std::pair< int32_t, int32_t > > N_T ({{60, 11}})`
- `const std::vector< std::pair< int32_t, int32_t > > N ({{71, 5}})`
- `const std::vector< std::pair< int32_t, int32_t > > M ({{76, 2}})`
- `const std::vector< std::pair< int32_t, int32_t > > DAY_NUMBER_A ({{6, 11}})`
- `const std::vector< std::pair< int32_t, int32_t > > TAU_C ({{17, 32}})`
- `const std::vector< std::pair< int32_t, int32_t > > N_4 ({{50, 5}})`
- `const std::vector< std::pair< int32_t, int32_t > > TAU_GPS ({{55, 22}})`
- `const std::vector< std::pair< int32_t, int32_t > > ALM_L_N ({{77, 1}})`

- `const std::vector< std::pair< int32_t, int32_t > > C_N` (`{{6, 1}}`)
- `const std::vector< std::pair< int32_t, int32_t > > M_N_A` (`{{7, 2}}`)
- `const std::vector< std::pair< int32_t, int32_t > > N_A` (`{{9, 5}}`)
- `const std::vector< std::pair< int32_t, int32_t > > TAU_N_A` (`{{14, 10}}`)
- `const std::vector< std::pair< int32_t, int32_t > > LAMBDA_N_A` (`{{24, 21}}`)
- `const std::vector< std::pair< int32_t, int32_t > > DELTA_I_N_A` (`{{45, 18}}`)
- `const std::vector< std::pair< int32_t, int32_t > > EPSILON_N_A` (`{{63, 15}}`)
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA_N_A` (`{{6, 16}}`)
- `const std::vector< std::pair< int32_t, int32_t > > T_LAMBDA_N_A` (`{{22, 21}}`)
- `const std::vector< std::pair< int32_t, int32_t > > DELTA_T_N_A` (`{{43, 22}}`)
- `const std::vector< std::pair< int32_t, int32_t > > DELTA_T_DOT_N_A` (`{{65, 7}}`)
- `const std::vector< std::pair< int32_t, int32_t > > H_N_A` (`{{72, 5}}`)
- `const std::vector< std::pair< int32_t, int32_t > > B1` (`{{6, 11}}`)
- `const std::vector< std::pair< int32_t, int32_t > > B2` (`{{17, 10}}`)

Variables

- `constexpr double GLONASS_F_M_A = 0.35e9`
Gravitational constant of atmosphere [m^3/s^2].
- `constexpr double GLONASS_SEMI_MAJOR_AXIS = 6378136`
Semi-major axis of Earth [m].
- `constexpr double GLONASS_FLATTENING = 1.0 / 29825784.0`
Flattening parameter.
- `constexpr double GLONASS_GRAVITY = 97803284.0`
Equatorial acceleration of gravity [mGal].
- `constexpr double GLONASS_GRAVITY_CORRECTION = 0.87`
Correction to acceleration of gravity at sea-level due to Atmosphere[uGal].
- `constexpr double GLONASS_J2 = 1082625.75e-9`
Second zonal harmonic of the geopotential.
- `constexpr double GLONASS_J4 = -2370.89e-9`
Fourth zonal harmonic of the geopotential.
- `constexpr double GLONASS_J6 = 6.08e-9`
Sixth zonal harmonic of the geopotential.
- `constexpr double GLONASS_J8 = 1.40e-11`
Eighth zonal harmonic of the geopotential.
- `constexpr double GLONASS_U0 = 62636861.4`
Normal potential at surface of common terrestrial ellipsoid [m^2/s^2].
- `constexpr double GLONASS_C20 = -1082.63e-6`
Second zonal coefficient of spherical harmonic expansion.
- `constexpr double GLONASS_EARTH_RADIUS = 6378.136`
Equatorial radius of Earth [km].
- `constexpr double GLONASS_EARTH_INCLINATION = 0.000409148809899e3`
Mean inclination of ecliptic to equator (23 deg 26 min 33 sec) [rad].
- `constexpr double GLONASS_TAU_0 = -0.005835151531174e3`
(-334 deg 19 min 46.40 sec) [rad];
- `constexpr double GLONASS_TAU_1 = 0.071018041257371e3`
(4069 deg 02 min 02.52 sec) [rad];
- `constexpr double GLONASS_MOON_Q0 = -0.001115184961435e3`
(-63 deg 53 min 43.41 sec) [rad]
- `constexpr double GLONASS_MOON_Q1 = 8.328691103668023e3`
(477198 deg 50 min 56.79 sec) [rad]

- constexpr double `GLONASS_MOON_OMEGA_0` = 0.004523601514852e3
(259 deg 10 min 59.79 sec) [rad]
- constexpr double `GLONASS_MOON_OMEGA_1` = -0.033757146246552e3
(-1934 deg 08 min 31.23 sec) [rad]
- constexpr double `GLONASS_MOON_GM` = 4902.835
Lunar gravitational constant [km³/s²].
- constexpr double `GLONASS_MOON_SEMI_MAJOR_AXIS` = 3.84385243e5
Semi-major axis of lunar orbit [km];.
- constexpr double `GLONASS_MOON_ECCENTRICITY` = 0.054900489
Eccentricity of lunar orbit.
- constexpr double `GLONASS_MOON_INCLINATION` = 0.000089803977407e3
Inclination of lunar orbit to ecliptic plane (5 deg 08 min 43.4 sec) [rad].
- constexpr double `GLONASS_SUN_OMEGA` = 0.004908229466869e3
TODO What is this operation in the seconds with T?(281 deg 13 min 15.0 + 6189.03 x T sec) [rad].
- constexpr double `GLONASS_SUN_Q0` = 0.006256583774423e3
(358 deg 28 min 33.04 sec) [rad]
- constexpr double `GLONASS_SUN_Q1` = 0e3
TODO Why is the value greater than 60?(129596579.10 sec) [rad].
- constexpr double `GLONASS_SUN_GM` = 0.1325263e12
Solar gravitational constant [km³/s²].
- constexpr double `GLONASS_SUN_SEMI_MAJOR_AXIS` = 1.49598e8
Semi-major axis of solar orbit [km];.
- constexpr double `GLONASS_SUN_ECCENTRICITY` = 0.016719
Eccentricity of solar orbit.
- constexpr double `GLONASS_L2_CA_FREQ_HZ` = `FREQ2_GLO`
L2 [Hz].
- constexpr double `GLONASS_L2_CA_DFREQ_HZ` = `DFRQ2_GLO`
Freq Bias for GLONASS L1 [Hz].
- constexpr double `GLONASS_L2_CA_CODE_RATE_CPS` = 0.511e6
GLONASS L1 C/A code rate [chips/s].
- constexpr double `GLONASS_L2_CA_CODE_LENGTH_CHIPS` = 511.0
GLONASS L1 C/A code length [chips].
- constexpr double `GLONASS_L2_CA_CODE_PERIOD_S` = 0.001
GLONASS L1 C/A code period [seconds].
- constexpr double `GLONASS_L2_CA_CHIP_PERIOD_S` = 1.9569e-06
GLONASS L1 C/A chip period [seconds].
- constexpr double `GLONASS_L2_CA_SYMBOL_RATE_BPS` = 1000.0
- constexpr double `GLONASS_L1_CA_FREQ_HZ` = `FREQ1_GLO`
L1 [Hz].
- constexpr double `GLONASS_L1_CA_DFREQ_HZ` = `DFRQ1_GLO`
Freq Bias for GLONASS L1 [Hz].
- constexpr double `GLONASS_L1_CA_CODE_RATE_CPS` = 0.511e6
GLONASS L1 C/A code rate [chips/s].
- constexpr double `GLONASS_L1_CA_CODE_LENGTH_CHIPS` = 511.0
GLONASS L1 C/A code length [chips].
- constexpr double `GLONASS_L1_CA_CODE_PERIOD_S` = 0.001
GLONASS L1 C/A code period [seconds].
- constexpr double `GLONASS_L1_CA_CHIP_PERIOD_S` = 1.9569e-06
GLONASS L1 C/A chip period [seconds].
- constexpr double `GLONASS_L1_CA_SYMBOL_RATE_BPS` = 1000.0
- constexpr int32_t `GLONASS_CA_NBR_SATS` = 24

- `constexpr int32_t GLONASS_L1_CA_HISTORY_DEEP = 100`
- `constexpr double GLONASS_GNAV_PREAMBLE_DURATION_S = 0.300`
- `constexpr int32_t GLONASS_GNAV_PREAMBLE_LENGTH_BITS = 30`
- `constexpr int32_t GLONASS_GNAV_PREAMBLE_LENGTH_SYMBOLS = 300`
- `constexpr int32_t GLONASS_GNAV_PREAMBLE_PERIOD_SYMBOLS = 2000`
- `constexpr int32_t GLONASS_GNAV_TELEMETRY_RATE_BITS_SECOND = 50`
NAV message bit rate [bits/s].
- `constexpr int32_t GLONASS_GNAV_TELEMETRY_SYMBOLS_PER_BIT = 10`
- `constexpr int32_t GLONASS_GNAV_TELEMETRY_SYMBOLS_PER_PREAMBLE_BIT = 10`
- `constexpr int32_t GLONASS_GNAV_TELEMETRY_RATE_SYMBOLS_SECOND = GLONASS_GNAV_TELEMETRY_RATE_BITS_SECOND * GLONASS_GNAV_TELEMETRY_SYMBOLS_PER_BIT`
NAV message bit rate [symbols/s].
- `constexpr int32_t GLONASS_GNAV_STRING_SYMBOLS = 2000`
Number of bits per string in the GNAV message (85 data bits + 30 time mark bits) [bits].
- `constexpr int32_t GLONASS_GNAV_STRING_BITS = 85`
Number of bits per string in the GNAV message (85 data bits + 30 time mark bits) [bits].
- `constexpr int32_t GLONASS_GNAV_HAMMING_CODE_BITS = 8`
Number of bits in hamming code sequence of GNAV message.
- `constexpr int32_t GLONASS_GNAV_DATA_SYMBOLS = 1700`
- `constexpr double GLONASS_LEAP_SECONDS [19][7]`
Record of leap seconds definition for GLOT to GPST conversion and vice versa.
- `const std::map< uint32_t, int32_t > GLONASS_PRN`
- `const std::vector< int32_t > GLONASS_GNAV_CRC_I_INDEX {9, 10, 12, 13, 15, 17, 19, 20, 22, 24, 26, 28, 30, 32, 34, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84}`
- `const std::vector< int32_t > GLONASS_GNAV_CRC_J_INDEX {9, 11, 12, 14, 15, 18, 19, 21, 22, 25, 26, 29, 30, 33, 34, 36, 37, 40, 41, 44, 45, 48, 49, 52, 53, 56, 57, 60, 61, 64, 65, 67, 68, 71, 72, 75, 76, 79, 80, 83, 84}`
- `const std::vector< int32_t > GLONASS_GNAV_CRC_K_INDEX {10, 11, 12, 16, 17, 18, 19, 23, 24, 25, 26, 31, 32, 33, 34, 38, 39, 40, 41, 46, 47, 48, 49, 54, 55, 56, 57, 62, 63, 64, 65, 69, 70, 71, 72, 77, 78, 79, 80, 85}`
- `const std::vector< int32_t > GLONASS_GNAV_CRC_L_INDEX {13, 14, 15, 16, 17, 18, 19, 27, 28, 29, 30, 31, 32, 33, 34, 42, 43, 44, 45, 46, 47, 48, 49, 58, 59, 60, 61, 62, 63, 64, 65, 73, 74, 75, 76, 77, 78, 79, 80}`
- `const std::vector< int32_t > GLONASS_GNAV_CRC_M_INDEX {20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 81, 82, 83, 84, 85}`
- `const std::vector< int32_t > GLONASS_GNAV_CRC_N_INDEX {35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65}`
- `const std::vector< int32_t > GLONASS_GNAV_CRC_P_INDEX {66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85}`
- `const std::vector< int32_t > GLONASS_GNAV_CRC_Q_INDEX {9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85}`

11.137.1 Detailed Description

Defines system parameters for GLONASS L1 C/A signal and NAV data.

Note

File renamed from GLONASS_L1_CA.h to [GLONASS_L1_L2_CA.h](#) to accommodate GLO L2 addition

Author

Damian Miralles, 2017. dmiralles2009(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.137.2 Macro Definition Documentation**11.137.2.1 GLONASS_GNAV_PREAMBLE**

```
#define GLONASS_GNAV_PREAMBLE
```

Value:

```
{
    1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0 \
}
```

Definition at line 88 of file GLONASS_L1_L2_CA.h.

11.137.3 Variable Documentation**11.137.3.1 GLONASS_C20**

```
constexpr double GLONASS_C20 = -1082.63e-6
```

Second zonal coefficient of spherical harmonic expansion.

Definition at line 43 of file GLONASS_L1_L2_CA.h.

11.137.3.2 GLONASS_EARTH_INCLINATION

```
constexpr double GLONASS_EARTH_INCLINATION = 0.000409148809899e3
```

Mean inclination of ecliptic to equator (23 deg 26 min 33 sec) [rad].

Definition at line 45 of file GLONASS_L1_L2_CA.h.

11.137.3.3 GLONASS_EARTH_RADIUS

```
constexpr double GLONASS_EARTH_RADIUS = 6378.136
```

Equatorial radius of Earth [km].

Definition at line 44 of file GLONASS_L1_L2_CA.h.

11.137.3.4 GLONASS_F_M_A

```
constexpr double GLONASS_F_M_A = 0.35e9
```

Gravitational constant of atmosphere [m^3/s^2].

Definition at line 33 of file GLONASS_L1_L2_CA.h.

11.137.3.5 GLONASS_FLATTENING

```
constexpr double GLONASS_FLATTENING = 1.0 / 29825784.0
```

Flattening parameter.

Definition at line 35 of file GLONASS_L1_L2_CA.h.

11.137.3.6 GLONASS_GNAV_HAMMING_CODE_BITS

```
constexpr int32_t GLONASS_GNAV_HAMMING_CODE_BITS = 8
```

Number of bits in hamming code sequence of GNAV message.

Definition at line 102 of file GLONASS_L1_L2_CA.h.

11.137.3.7 GLONASS_GNAV_STRING_BITS

```
constexpr int32_t GLONASS_GNAV_STRING_BITS = 85
```

Number of bits per string in the GNAV message (85 data bits + 30 time mark bits) [bits].

Definition at line 101 of file GLONASS_L1_L2_CA.h.

11.137.3.8 GLONASS_GNAV_STRING_SYMBOLS

```
constexpr int32_t GLONASS_GNAV_STRING_SYMBOLS = 2000
```

Number of bits per string in the GNAV message (85 data bits + 30 time mark bits) [bits].

Definition at line 100 of file GLONASS_L1_L2_CA.h.

11.137.3.9 GLONASS_GNAV_TELEMETRY_RATE_BITS_SECOND

```
constexpr int32_t GLONASS_GNAV_TELEMETRY_RATE_BITS_SECOND = 50
```

NAV message bit rate [bits/s].

Definition at line 96 of file GLONASS_L1_L2_CA.h.

11.137.3.10 GLONASS_GNAV_TELEMETRY_RATE_SYMBOLS_SECOND

```
constexpr int32_t GLONASS_GNAV_TELEMETRY_RATE_SYMBOLS_SECOND = GLONASS_GNAV_TELEMETRY_RATE_BITS_SECOND  
* GLONASS_GNAV_TELEMETRY_SYMBOLS_PER_BIT
```

NAV message bit rate [symbols/s].

Definition at line 99 of file GLONASS_L1_L2_CA.h.

11.137.3.11 GLONASS_GRAVITY

```
constexpr double GLONASS_GRAVITY = 97803284.0
```

Equatorial acceleration of gravity [mGal].

Definition at line 36 of file GLONASS_L1_L2_CA.h.

11.137.3.12 GLONASS_GRAVITY_CORRECTION

```
constexpr double GLONASS_GRAVITY_CORRECTION = 0.87
```

Correction to acceleration of gravity at sea-level due to Atmosphere[uGal].

Definition at line 37 of file GLONASS_L1_L2_CA.h.

11.137.3.13 GLONASS_J2

```
constexpr double GLONASS_J2 = 1082625.75e-9
```

Second zonal harmonic of the geopotential.

Definition at line 38 of file GLONASS_L1_L2_CA.h.

11.137.3.14 GLONASS_J4

```
constexpr double GLONASS_J4 = -2370.89e-9
```

Fourth zonal harmonic of the geopotential.

Definition at line 39 of file GLONASS_L1_L2_CA.h.

11.137.3.15 GLONASS_J6

```
constexpr double GLONASS_J6 = 6.08e-9
```

Sixth zonal harmonic of the geopotential.

Definition at line 40 of file GLONASS_L1_L2_CA.h.

11.137.3.16 GLONASS_J8

```
constexpr double GLONASS_J8 = 1.40e-11
```

Eighth zonal harmonic of the geopotential.

Definition at line 41 of file GLONASS_L1_L2_CA.h.

11.137.3.17 GLONASS_L1_CA_CHIP_PERIOD_S

```
constexpr double GLONASS_L1_CA_CHIP_PERIOD_S = 1.9569e-06
```

GLONASS L1 C/A chip period [seconds].

Definition at line 79 of file GLONASS_L1_L2_CA.h.

11.137.3.18 GLONASS_L1_CA_CODE_LENGTH_CHIPS

```
constexpr double GLONASS_L1_CA_CODE_LENGTH_CHIPS = 511.0
```

GLONASS L1 C/A code length [chips].

Definition at line 77 of file GLONASS_L1_L2_CA.h.

11.137.3.19 GLONASS_L1_CA_CODE_PERIOD_S

```
constexpr double GLONASS_L1_CA_CODE_PERIOD_S = 0.001
```

GLONASS L1 C/A code period [seconds].

Definition at line 78 of file GLONASS_L1_L2_CA.h.

11.137.3.20 GLONASS_L1_CA_CODE_RATE_CPS

```
constexpr double GLONASS_L1_CA_CODE_RATE_CPS = 0.511e6
```

GLONASS L1 C/A code rate [chips/s].

Definition at line 76 of file GLONASS_L1_L2_CA.h.

11.137.3.21 GLONASS_L1_CA_DFREQ_HZ

```
constexpr double GLONASS_L1_CA_DFREQ_HZ = DFRQ1_GLO
```

Freq Bias for GLONASS L1 [Hz].

Definition at line 75 of file GLONASS_L1_L2_CA.h.

11.137.3.22 GLONASS_L1_CA_FREQ_HZ

```
constexpr double GLONASS_L1_CA_FREQ_HZ = FREQ1_GLO
```

L1 [Hz].

Definition at line 74 of file GLONASS_L1_L2_CA.h.

11.137.3.23 GLONASS_L2_CA_CHIP_PERIOD_S

```
constexpr double GLONASS_L2_CA_CHIP_PERIOD_S = 1.9569e-06
```

GLONASS L1 C/A chip period [seconds].

Definition at line 71 of file GLONASS_L1_L2_CA.h.

11.137.3.24 GLONASS_L2_CA_CODE_LENGTH_CHIPS

```
constexpr double GLONASS_L2_CA_CODE_LENGTH_CHIPS = 511.0
```

GLONASS L1 C/A code length [chips].

Definition at line 69 of file GLONASS_L1_L2_CA.h.

11.137.3.25 GLONASS_L2_CA_CODE_PERIOD_S

```
constexpr double GLONASS_L2_CA_CODE_PERIOD_S = 0.001
```

GLONASS L1 C/A code period [seconds].

Definition at line 70 of file GLONASS_L1_L2_CA.h.

11.137.3.26 GLONASS_L2_CA_CODE_RATE_CPS

```
constexpr double GLONASS_L2_CA_CODE_RATE_CPS = 0.511e6
```

GLONASS L1 C/A code rate [chips/s].

Definition at line 68 of file GLONASS_L1_L2_CA.h.

11.137.3.27 GLONASS_L2_CA_DFREQ_HZ

```
constexpr double GLONASS_L2_CA_DFREQ_HZ = DFRQ2_GLO
```

Freq Bias for GLONASS L1 [Hz].

Definition at line 67 of file GLONASS_L1_L2_CA.h.

11.137.3.28 GLONASS_L2_CA_FREQ_HZ

```
constexpr double GLONASS_L2_CA_FREQ_HZ = FREQ2_GLO
```

L2 [Hz].

Definition at line 66 of file GLONASS_L1_L2_CA.h.

11.137.3.29 GLONASS_LEAP_SECONDS

```
constexpr double GLONASS_LEAP_SECONDS[19][7]
```

Initial value:

```
= {
    {2017, 1, 1, 0, 0, 0, -18},
    {2015, 7, 1, 0, 0, 0, -17},
    {2012, 7, 1, 0, 0, 0, -16},
    {2009, 1, 1, 0, 0, 0, -15},
    {2006, 1, 1, 0, 0, 0, -14},
    {1999, 1, 1, 0, 0, 0, -13},
    {1997, 7, 1, 0, 0, 0, -12},
    {1996, 1, 1, 0, 0, 0, -11},
    {1994, 7, 1, 0, 0, 0, -10},
    {1993, 7, 1, 0, 0, 0, -9},
    {1992, 7, 1, 0, 0, 0, -8},
    {1991, 1, 1, 0, 0, 0, -7},
    {1990, 1, 1, 0, 0, 0, -6},
    {1988, 1, 1, 0, 0, 0, -5},
    {1985, 7, 1, 0, 0, 0, -4},
    {1983, 7, 1, 0, 0, 0, -3},
    {1982, 7, 1, 0, 0, 0, -2},
    {1981, 7, 1, 0, 0, 0, -1},
    {}
}
```

Record of leap seconds definition for GLOT to GPST conversion and vice versa.

Each entry is defined by an array of 7 elements consisting of yr,month,day,hr,min,sec,utc-gpst

Note

Ideally should use leap seconds definitions of rtklibGLONASS SV's orbital slots PRN = (orbital_slot - 1)

Definition at line 110 of file GLONASS_L1_L2_CA.h.

11.137.3.30 GLONASS_MOON_ECCENTRICITY

```
constexpr double GLONASS_MOON_ECCENTRICITY = 0.054900489
```

Eccentricity of lunar orbit.

Definition at line 56 of file GLONASS_L1_L2_CA.h.

11.137.3.31 GLONASS_MOON_GM

```
constexpr double GLONASS_MOON_GM = 4902.835
```

Lunar gravitational constant [km^3/s^2].

Definition at line 54 of file GLONASS_L1_L2_CA.h.

11.137.3.32 GLONASS_MOON_INCLINATION

```
constexpr double GLONASS_MOON_INCLINATION = 0.000089803977407e3
```

Inclination of lunar orbit to ecliptic plane (5 deg 08 min 43.4 sec) [rad].

Definition at line 57 of file GLONASS_L1_L2_CA.h.

11.137.3.33 GLONASS_MOON_OMEGA_0

```
constexpr double GLONASS_MOON_OMEGA_0 = 0.004523601514852e3
```

(259 deg 10 min 59.79 sec) [rad]

Definition at line 52 of file GLONASS_L1_L2_CA.h.

11.137.3.34 GLONASS_MOON_OMEGA_1

```
constexpr double GLONASS_MOON_OMEGA_1 = -0.033757146246552e3
```

(-1934 deg 08 min 31.23 sec) [rad]

Definition at line 53 of file GLONASS_L1_L2_CA.h.

11.137.3.35 GLONASS_MOON_Q0

```
constexpr double GLONASS_MOON_Q0 = -0.001115184961435e3
```

(-63 deg 53 min 43.41 sec) [rad]

Definition at line 50 of file GLONASS_L1_L2_CA.h.

11.137.3.36 GLONASS_MOON_Q1

```
constexpr double GLONASS_MOON_Q1 = 8.328691103668023e3
```

(477198 deg 50 min 56.79 sec) [rad]

Definition at line 51 of file GLONASS_L1_L2_CA.h.

11.137.3.37 GLONASS_MOON_SEMI_MAJOR_AXIS

```
constexpr double GLONASS_MOON_SEMI_MAJOR_AXIS = 3.84385243e5
```

Semi-major axis of lunar orbit [km];.

Definition at line 55 of file GLONASS_L1_L2_CA.h.

11.137.3.38 GLONASS_SEMI_MAJOR_AXIS

```
constexpr double GLONASS_SEMI_MAJOR_AXIS = 6378136
```

Semi-major axis of Earth [m].

Definition at line 34 of file GLONASS_L1_L2_CA.h.

11.137.3.39 GLONASS_SUN_ECCENTRICITY

```
constexpr double GLONASS_SUN_ECCENTRICITY = 0.016719
```

Eccentricity of solar orbit.

Definition at line 64 of file GLONASS_L1_L2_CA.h.

11.137.3.40 GLONASS_SUN_GM

```
constexpr double GLONASS_SUN_GM = 0.1325263e12
```

Solar gravitational constant [km³/s²].

Definition at line 62 of file GLONASS_L1_L2_CA.h.

11.137.3.41 GLONASS_SUN_OMEGA

```
constexpr double GLONASS_SUN_OMEGA = 0.004908229466869e3
```

TODO What is this operation in the seconds with T?(281 deg 13 min 15.0 + 6189.03 x T sec) [rad].

Definition at line 59 of file GLONASS_L1_L2_CA.h.

11.137.3.42 GLONASS_SUN_Q0

```
constexpr double GLONASS_SUN_Q0 = 0.006256583774423e3
```

(358 deg 28 min 33.04 sec) [rad]

Definition at line 60 of file GLONASS_L1_L2_CA.h.

11.137.3.43 GLONASS_SUN_Q1

```
constexpr double GLONASS_SUN_Q1 = 0e3
```

TODO Why is the value greater than 60?(129596579.10 sec) [rad].

Definition at line 61 of file GLONASS_L1_L2_CA.h.

11.137.3.44 GLONASS_SUN_SEMI_MAJOR_AXIS

```
constexpr double GLONASS_SUN_SEMI_MAJOR_AXIS = 1.49598e8
```

Semi-major axis of solar orbit [km];.

Definition at line 63 of file GLONASS_L1_L2_CA.h.

11.137.3.45 GLONASS_TAU_0

```
constexpr double GLONASS_TAU_0 = -0.005835151531174e3
```

(-334 deg 19 min 46.40 sec) [rad];

Definition at line 47 of file GLONASS_L1_L2_CA.h.

11.137.3.46 GLONASS_TAU_1

```
constexpr double GLONASS_TAU_1 = 0.071018041257371e3
```

(4069 deg 02 min 02.52 sec) [rad];

Definition at line 48 of file GLONASS_L1_L2_CA.h.

11.137.3.47 GLONASS_U0

```
constexpr double GLONASS_U0 = 62636861.4
```

Normal potential at surface of common terrestrial ellipsoid [m^2/s^2].

Definition at line 42 of file GLONASS_L1_L2_CA.h.

11.138 glonass_l1_signal_processing.h File Reference

This class implements various functions for GLONASS L1 CA signals.

```
#include <complex>
#include <cstdint>
#include <gsl/gsl>
```

Functions

- void [glonass_l1_ca_code_gen_complex](#) (own::span< std::complex< float >> _dest, uint32_t _chip_shift)
Generates complex GLONASS L1 C/A code for the desired SV ID and code shift, and sampled to specific sampling frequency.
- void [glonass_l1_ca_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, int32_t _fs, uint32_t _chip_shift, uint32_t _ncodes)
Generates N complex GLONASS L1 C/A codes for the desired SV ID and code shift.
- void [glonass_l1_ca_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, int32_t _fs, uint32_t _chip_shift)
Generates complex GLONASS L1 C/A code for the desired SV ID and code shift.

11.138.1 Detailed Description

This class implements various functions for GLONASS L1 CA signals.

Author

Gabriel Araujo, 2017. [gabriel.araujo\(at\)ieee.org](mailto:gabriel.araujo(at)ieee.org)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.138.2 Function Documentation

11.138.2.1 glonass_l1_ca_code_gen_complex()

```
void glonass_l1_ca_code_gen_complex (
    own::span< std::complex< float >> _dest,
    uint32_t _chip_shift )
```

Generates complex GLONASS L1 C/A code for the desired SV ID and code shift, and sampled to specific sampling frequency.

11.138.2.2 glonass_l1_ca_code_gen_complex_sampled() [1/2]

```
void glonass_l1_ca_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    int32_t _fs,
    uint32_t _chip_shift,
    uint32_t _ncodes )
```

Generates N complex GLONASS L1 C/A codes for the desired SV ID and code shift.

11.138.2.3 glonass_l1_ca_code_gen_complex_sampled() [2/2]

```
void glonass_l1_ca_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    int32_t _fs,
    uint32_t _chip_shift )
```

Generates complex GLONASS L1 C/A code for the desired SV ID and code shift.

11.139 glonass_l2_ca_dll_pll_c_aid_tracking.h File Reference

Interface of an adapter of a DLL+PLL tracking loop block for Glonass L2 C/A to a [TrackingInterface](#).

```
#include "glonass_l2_ca_dll_pll_c_aid_tracking_cc.h"
#include "glonass_l2_ca_dll_pll_c_aid_tracking_sc.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GlonassL2CaDllPllCAidTracking](#)

This class implements a code DLL + carrier PLL tracking loop.

11.139.1 Detailed Description

Interface of an adapter of a DLL+PLL tracking loop block for Glonass L2 C/A to a [TrackingInterface](#).

Author

Damian Miralles, 2018. dmiralles2009(at)gmail.com

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkha user, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.140 glonass_l2_ca_dll_pll_c_aid_tracking_cc.h File Reference

Implementation of a code DLL + carrier PLL tracking block.

```
#include "cpu_multicorrelator.h"
#include "gnss_synchro.h"
#include "tracking_2nd_DLL_filter.h"
#include "tracking_FLL_PLL_filter.h"
#include <gnuradio/block.h>
#include <pmt/pmt.h>
#include <volk_gnssdr/volk_gnssdr_alloc.h>
#include <deque>
#include <fstream>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [glonass_l2_ca_dll_pll_c_aid_tracking_cc](#)
This class implements a DLL + PLL tracking loop block.

Typedefs

- using [glonass_l2_ca_dll_pll_c_aid_tracking_cc_sptr](#) = boost::shared_ptr< [glonass_l2_ca_dll_pll_c_aid_tracking_cc](#) >

Functions

- [glonass_l2_ca_dll_pll_c_aid_tracking_cc_sptr](#) [glonass_l2_ca_dll_pll_c_aid_make_tracking_cc](#) (int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float pll_bw_hz, float dll_bw_hz, float pll_bw_narrow_hz, float dll_bw_narrow_hz, int32_t extend_correlation_ms, float early_late_space_chips)

11.140.1 Detailed Description

Implementation of a code DLL + carrier PLL tracking block.

Author

Damian Miralles, 2018. dmiralles2009(at)gmail.com

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkha user, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.141 glonass_l2_ca_dll_pll_c_aid_tracking_sc.h File Reference

Implementation of a code DLL + carrier PLL tracking block.

```
#include "cpu_multicorrelator_16sc.h"
#include "glonass_l2_signal_processing.h"
#include "gnss_synchro.h"
#include "tracking_2nd_DLL_filter.h"
#include "tracking_FLL_PLL_filter.h"
#include <gnuradio/block.h>
#include <volk_gnssdr/volk_gnssdr_alloc.h>
#include <deque>
#include <fstream>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [glonass_l2_ca_dll_pll_c_aid_tracking_sc](#)
This class implements a DLL + PLL tracking loop block.

Typedefs

- using [glonass_l2_ca_dll_pll_c_aid_tracking_sc_sptr](#) = boost::shared_ptr< [glonass_l2_ca_dll_pll_c_aid_tracking_sc](#) >

Functions

- glonass_l2_ca_dll_pll_c_aid_tracking_sc_sptr [glonass_l2_ca_dll_pll_c_aid_make_tracking_sc](#)(int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float pll_bw_hz, float dll_bw_hz, float pll_bw_narrow_hz, float dll_bw_narrow_hz, int32_t extend_correlation_ms, float early_late_space_chips)

11.141.1 Detailed Description

Implementation of a code DLL + carrier PLL tracking block.

Author

Damian Miralles, 2018. dmiralles2009(at)gmail.com

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkha user, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.142 glonass_l2_ca_dll_pll_tracking.h File Reference

Interface of an adapter of a DLL+PLL tracking loop block for Glonass L2 C/A to a [TrackingInterface](#).

```
#include "glonass_l2_ca_dll_pll_tracking_cc.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GlonassL2CaDllPllTracking](#)

This class implements a code DLL + carrier PLL tracking loop.

11.142.1 Detailed Description

Interface of an adapter of a DLL+PLL tracking loop block for Glonass L2 C/A to a [TrackingInterface](#).

Author

Damian Miralles, 2018, dmiralles2009(at)gmail.com

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkha user, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.143 glonass_l2_ca_dll_pll_tracking_cc.h File Reference

Implementation of a code DLL + carrier PLL tracking block.

```
#include "cpu_multicorrelator.h"
#include "gnss_synchro.h"
#include "tracking_2nd_DLL_filter.h"
#include "tracking_2nd_PLL_filter.h"
#include <gnuradio/block.h>
#include <volk_gnssssdr/volk_gnssssdr_alloc.h>
#include <fstream>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [Glonass_L2_Ca_DLL_Pll_Tracking_cc](#)
This class implements a DLL + PLL tracking loop block.

Typedefs

- using **glonass_l2_ca_dll_pll_tracking_cc_sptr** = boost::shared_ptr< [Glonass_L2_Ca_DLL_Pll_Tracking_cc](#) >

Functions

- glonass_l2_ca_dll_pll_tracking_cc_sptr **glonass_l2_ca_dll_pll_make_tracking_cc** (int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float pll_bw_hz, float dll_bw_hz, float early_late_space_chips)

11.143.1 Detailed Description

Implementation of a code DLL + carrier PLL tracking block.

Author

Damian Miralles, 2018. dmiralles2009(at)gmail.com

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkha user, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.144 glonass_l2_ca_pcps_acquisition.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Glonass L2 C/A signals.

```
#include "acq_conf.h"
#include "channel_fsm.h"
#include "complex_byte_to_float_x2.h"
#include "gnss_synchro.h"
#include "pcps_acquisition.h"
#include <gnuradio/blocks/float_to_complex.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GlonassL2CaPcpsAcquisition](#)

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GLONASS L2 C/A signals.

11.144.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for Glonass L2 C/A signals.

Author

Damian Miralles, 2018, dmiralles2009@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.145 glonass_l2_ca_telemetry_decoder.h File Reference

Interface of an adapter of a GLONASS L2 C/A NAV data decoder block to a [TelemetryDecoderInterface](#).

```
#include "glonass_l2_ca_telemetry_decoder_gs.h"
#include "gnss_satellite.h"
#include "gnss_synchro.h"
#include "telemetry_decoder_interface.h"
#include <gnuradio/runtime_types.h>
#include <cstring>
#include <string>
```


Classes

- class [GlonassL2CaTelemetryDecoder](#)

This class implements a NAV data decoder for GLONASS L2 C/A.

11.145.1 Detailed Description

Interface of an adapter of a GLONASS L2 C/A NAV data decoder block to a [TelemetryDecoderInterface](#).

Author

Damian Miralles, 2018. dmiralles2009(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.146 glonass_l2_ca_telemetry_decoder_gs.h File Reference

Implementation of a GLONASS L2 C/A NAV data decoder block.

```
#include "GLONASS_L1_L2_CA.h"
#include "glonass_gnav_navigation_message.h"
#include "gnss_satellite.h"
#include "gnss_synchro.h"
#include <boost/circular_buffer.hpp>
#include <gnuradio/block.h>
#include <gnuradio/types.h>
#include <array>
#include <cstdint>
#include <fstream>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [glonass_l2_ca_telemetry_decoder_gs](#)

This class implements a block that decodes the GNAV data defined in GLONASS ICD v5.1.

Typedefs

- using [glonass_l2_ca_telemetry_decoder_gs_sptr](#) = boost::shared_ptr< [glonass_l2_ca_telemetry_decoder_gs](#) >

Functions

- `glonass_l2_ca_telemetry_decoder_gs_sptr` `glonass_l2_ca_make_telemetry_decoder_gs` (const [Gnss_Satellite](#) &satellite, bool dump)

11.146.1 Detailed Description

Implementation of a GLONASS L2 C/A NAV data decoder block.

Author

Damian Miralles, 2018. [dmiralles2009\(at\)gmail.com](mailto:dmiralles2009(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.147 `glonass_l2_signal_processing.h` File Reference

This class implements various functions for GLONASS L2 CA signals.

```
#include <complex>
#include <cstdint>
#include <gsl/gsl>
```

Functions

- void `glonass_l2_ca_code_gen_complex` (own::span< std::complex< float >> _dest, uint32_t _chip_shift)
Generates complex GLONASS L2 C/A code for the desired SV ID and code shift, and sampled to specific sampling frequency.
- void `glonass_l2_ca_code_gen_complex_sampled` (own::span< std::complex< float >> _dest, int32_t _fs, uint32_t _chip_shift, uint32_t _ncodes)
Generates N complex GLONASS L2 C/A codes for the desired SV ID and code shift.
- void `glonass_l2_ca_code_gen_complex_sampled` (own::span< std::complex< float >> _dest, int32_t _fs, uint32_t _chip_shift)
Generates complex GLONASS L2 C/A code for the desired SV ID and code shift.

11.147.1 Detailed Description

This class implements various functions for GLONASS L2 CA signals.

Author

Damian Miralles, 2018, [dmiralles2009\(at\)gmail.com](mailto:dmiralles2009(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.147.2 Function Documentation

11.147.2.1 glonass_l2_ca_code_gen_complex()

```
void glonass_l2_ca_code_gen_complex (
    own::span< std::complex< float >> _dest,
    uint32_t _chip_shift )
```

Generates complex GLONASS L2 C/A code for the desired SV ID and code shift, and sampled to specific sampling frequency.

11.147.2.2 glonass_l2_ca_code_gen_complex_sampled() [1/2]

```
void glonass_l2_ca_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    int32_t _fs,
    uint32_t _chip_shift,
    uint32_t _ncodes )
```

Generates N complex GLONASS L2 C/A codes for the desired SV ID and code shift.

11.147.2.3 glonass_l2_ca_code_gen_complex_sampled() [2/2]

```
void glonass_l2_ca_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    int32_t _fs,
    uint32_t _chip_shift )
```

Generates complex GLONASS L2 C/A code for the desired SV ID and code shift.

11.148 gn3s_signal_source.h File Reference

GN3S USB dongle GPS RF front-end signal sampler driver.

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/hier_block2.h>
#include <pmt/pmt.h>
#include <cstdint>
#include <memory>
#include <string>
```

Classes

- class [Gn3sSignalSource](#)

This class reads samples from a GN3S USB dongle, a RF front-end signal sampler.

11.148.1 Detailed Description

GN3S USB dongle GPS RF front-end signal sampler driver.

Author

Javier Arribas, jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.149 gnmax_signal_source.h File Reference

gnMAX2769 USB dongle GPS RF front-end signal sampler driver

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/hier_block2.h>
#include <pmt/pmt.h>
#include <string>
```

Classes

- class [GnMaxSignalSource](#)

This class reads samples from a gnMAX2769 USB dongle, a RF front-end signal sampler.

11.149.1 Detailed Description

gnMAX2769 USB dongle GPS RF front-end signal sampler driver

Author

Wojciech Kazubski, wk(at)ire.pw.edu.pl
Javier Arribas, jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.150 gnss_block_factory.h File Reference

Interface of a factory that returns smart pointers to GNSS blocks.

```
#include "concurrent_queue.h"
#include <pmt/pmt.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GNSSBlockFactory](#)
Class that produces all kinds of GNSS blocks.

11.150.1 Detailed Description

Interface of a factory that returns smart pointers to GNSS blocks.

Author

Carlos Aviles, 2010. carlos.avilesr(at)gmail.com Luis Esteve, 2011. luis(at)epsilon-formacion.com Javier Arribas, 2011. jarribas(at)cttc.es Carles Fernandez-Prades, 2014-2020. cfernandez(at)cttc.es

This class encapsulates the complexity behind the instantiation of GNSS blocks.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.151 gnss_block_interface.h File Reference

This interface represents a GNSS block.

```
#include <gnuradio/top_block.h>
#include <cassert>
#include <string>
```

Classes

- class [GNSSBlockInterface](#)
This abstract class represents an interface to GNSS blocks.

11.151.1 Detailed Description

This interface represents a GNSS block.

Author

Carlos Aviles, 2010. carlos.avilesr(at)gmail.com

Abstract class for GNSS block interfaces. Since all its methods are virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.152 gnss_circular_deque.h File Reference

This class implements a circular deque for [Gnss_Synchro](#).

```
#include <boost/circular_buffer.hpp>
#include <vector>
```

Classes

- class [Gnss_circular_deque< T >](#)

11.152.1 Detailed Description

This class implements a circular deque for [Gnss_Synchro](#).

Author

Antonio Ramos, 2018. antonio.ramosdet(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.153 gnss_flowgraph.h File Reference

Interface of a GNSS receiver flow graph.

```
#include "channel_status_msg_receiver.h"
#include "concurrent_queue.h"
#include "gnss_sdr_sample_counter.h"
#include "gnss_signal.h"
#include "pvt_interface.h"
#include <gnuradio/blocks/null_sink.h>
#include <gnuradio/runtime_types.h>
#include <pmt/pmt.h>
#include <list>
#include <map>
#include <memory>
#include <mutex>
#include <string>
#include <utility>
#include <vector>
```

Classes

- class [GNSSFlowgraph](#)

This class represents a GNSS flow graph.

11.153.1 Detailed Description

Interface of a GNSS receiver flow graph.

Author

Carlos Aviles, 2010. [carlos.avilesr\(at\)gmail.com](mailto:carlos.avilesr(at)gmail.com) Luis Esteve, 2011. [luis\(at\)epsilon-formacion.com](mailto:luis(at)epsilon-formacion.com) Carles Fernandez-Prades, 2014-2020. [cfernandez\(at\)cttc.es](mailto:cfernandez(at)cttc.es) Álvaro Cebrián Juan, 2018. [acebrianjuan\(at\)gmail.com](mailto:acebrianjuan(at)gmail.com)↔

It contains a signal source, a signal conditioner, a set of channels, an observables block and a pvt.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.154 gnss_frequencies.h File Reference

GNSS Frequencies.

Variables

- constexpr double `FREQ1` = 1.57542e9
L1/E1 frequency (Hz)
- constexpr double `FREQ2` = 1.22760e9
L2 frequency (Hz)
- constexpr double `FREQ5` = 1.17645e9
L5/E5a frequency (Hz)
- constexpr double `FREQ6` = 1.27875e9
E6/LEX frequency (Hz)
- constexpr double `FREQ7` = 1.20714e9
E5b frequency (Hz)
- constexpr double `FREQ8` = 1.191795e9
E5a+b frequency (Hz)
- constexpr double `FREQ9` = 2.492028e9
S frequency (Hz)
- constexpr double `FREQ1_GLO` = 1.60200e9
GLONASS G1 base frequency (Hz)
- constexpr double `DFRQ1_GLO` = 0.56250e6
GLONASS G1 bias frequency (Hz/n)
- constexpr double `FREQ2_GLO` = 1.24600e9
GLONASS G2 base frequency (Hz)
- constexpr double `DFRQ2_GLO` = 0.43750e6
GLONASS G2 bias frequency (Hz/n)
- constexpr double `FREQ3_GLO` = 1.202025e9
GLONASS G3 frequency (Hz)
- constexpr double `FREQ1_BDS` = 1.561098e9
BeiDou B1 frequency (Hz)
- constexpr double `FREQ2_BDS` = 1.20714e9
BeiDou B2 frequency (Hz)
- constexpr double `FREQ3_BDS` = 1.26852e9
BeiDou B3 frequency (Hz)

11.154.1 Detailed Description

GNSS Frequencies.

Author

Carles Fernandez, 2017. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.154.2 Variable Documentation

11.154.2.1 DFRQ1_GLO

```
constexpr double DFRQ1_GLO = 0.56250e6
```

GLONASS G1 bias frequency (Hz/n)

Definition at line 33 of file gnss_frequencies.h.

11.154.2.2 DFRQ2_GLO

```
constexpr double DFRQ2_GLO = 0.43750e6
```

GLONASS G2 bias frequency (Hz/n)

Definition at line 35 of file gnss_frequencies.h.

11.154.2.3 FREQ1

```
constexpr double FREQ1 = 1.57542e9
```

L1/E1 frequency (Hz)

Definition at line 25 of file gnss_frequencies.h.

11.154.2.4 FREQ1_BDS

```
constexpr double FREQ1_BDS = 1.561098e9
```

BeiDou B1 frequency (Hz)

Definition at line 37 of file gnss_frequencies.h.

11.154.2.5 FREQ1_GLO

```
constexpr double FREQ1_GLO = 1.60200e9
```

GLONASS G1 base frequency (Hz)

Definition at line 32 of file gnss_frequencies.h.

11.154.2.6 **FREQ2**

```
constexpr double FREQ2 = 1.22760e9
```

L2 frequency (Hz)

Definition at line 26 of file gnss_frequencies.h.

11.154.2.7 **FREQ2_BDS**

```
constexpr double FREQ2_BDS = 1.20714e9
```

BeiDou B2 frequency (Hz)

Definition at line 38 of file gnss_frequencies.h.

11.154.2.8 **FREQ2_GLO**

```
constexpr double FREQ2_GLO = 1.24600e9
```

GLONASS G2 base frequency (Hz)

Definition at line 34 of file gnss_frequencies.h.

11.154.2.9 **FREQ3_BDS**

```
constexpr double FREQ3_BDS = 1.26852e9
```

BeiDou B3 frequency (Hz)

Definition at line 39 of file gnss_frequencies.h.

11.154.2.10 **FREQ3_GLO**

```
constexpr double FREQ3_GLO = 1.202025e9
```

GLONASS G3 frequency (Hz)

Definition at line 36 of file gnss_frequencies.h.

11.154.2.11 **FREQ5**

```
constexpr double FREQ5 = 1.17645e9
```

L5/E5a frequency (Hz)

Definition at line 27 of file gnss_frequencies.h.

11.154.2.12 **FREQ6**

```
constexpr double FREQ6 = 1.27875e9
```

E6/LEX frequency (Hz)

Definition at line 28 of file gnss_frequencies.h.

11.154.2.13 **FREQ7**

```
constexpr double FREQ7 = 1.20714e9
```

E5b frequency (Hz)

Definition at line 29 of file gnss_frequencies.h.

11.154.2.14 **FREQ8**

```
constexpr double FREQ8 = 1.191795e9
```

E5a+b frequency (Hz)

Definition at line 30 of file gnss_frequencies.h.

11.154.2.15 **FREQ9**

```
constexpr double FREQ9 = 2.492028e9
```

S frequency (Hz)

Definition at line 31 of file gnss_frequencies.h.

11.155 gnss_obs_codes.h File Reference

GNSS Observable codes.

```
#include <stdint>
```

Variables

- constexpr uint32_t [CODE_NONE](#) = 0
obs code: none or unknown
- constexpr uint32_t [CODE_L1C](#) = 1
obs code: L1C/A, G1C/A, E1C (GPS, GLO, GAL, QZS, SBS)
- constexpr uint32_t [CODE_L1P](#) = 2
obs code: L1P, G1P (GPS, GLO)
- constexpr uint32_t [CODE_L1W](#) = 3
obs code: L1 Z-track (GPS)
- constexpr uint32_t [CODE_L1Y](#) = 4
obs code: L1Y (GPS)
- constexpr uint32_t [CODE_L1M](#) = 5
obs code: L1M (GPS)
- constexpr uint32_t [CODE_L1N](#) = 6
obs code: L1codeless (GPS)
- constexpr uint32_t [CODE_L1S](#) = 7
obs code: L1C(D) (GPS, QZS)
- constexpr uint32_t [CODE_L1L](#) = 8
obs code: L1C(P) (GPS, QZS)
- constexpr uint32_t [CODE_L1E](#) = 9
(not used)
- constexpr uint32_t [CODE_L1A](#) = 10
obs code: E1A (GAL)
- constexpr uint32_t [CODE_L1B](#) = 11
obs code: E1B (GAL)
- constexpr uint32_t [CODE_L1X](#) = 12
obs code: E1B+C, L1C(D+P) (GAL, QZS)
- constexpr uint32_t [CODE_L1Z](#) = 13
obs code: E1A+B+C, L1SAIF (GAL, QZS)
- constexpr uint32_t [CODE_L2C](#) = 14
obs code: L2C/A, G1C/A (GPS, GLO)
- constexpr uint32_t [CODE_L2D](#) = 15
obs code: L2 L1C/A-(P2-P1) (GPS)
- constexpr uint32_t [CODE_L2S](#) = 16
obs code: L2C(M) (GPS, QZS)
- constexpr uint32_t [CODE_L2L](#) = 17
obs code: L2C(L) (GPS, QZS)
- constexpr uint32_t [CODE_L2X](#) = 18
obs code: L2C(M+L), B1I+Q (GPS, QZS, BDS)
- constexpr uint32_t [CODE_L2P](#) = 19
obs code: L2P, G2P (GPS, GLO)
- constexpr uint32_t [CODE_L2W](#) = 20

- obs code: L2 Z-track (GPS)*
- constexpr uint32_t [CODE_L2Y](#) = 21
- obs code: L2Y (GPS)*
- constexpr uint32_t [CODE_L2M](#) = 22
- obs code: L2M (GPS)*
- constexpr uint32_t [CODE_L2N](#) = 23
- obs code: L2codeless (GPS)*
- constexpr uint32_t [CODE_L5I](#) = 24
- obs code: L5/E5aI (GPS,GAL,QZS,SBS)*
- constexpr uint32_t [CODE_L5Q](#) = 25
- obs code: L5/E5aQ (GPS,GAL,QZS,SBS)*
- constexpr uint32_t [CODE_L5X](#) = 26
- obs code: L5/E5aI+Q/L5B+C (GPS,GAL,QZS,IRN,SBS)*
- constexpr uint32_t [CODE_L7I](#) = 27
- obs code: E5bI,B2I (GAL,BDS)*
- constexpr uint32_t [CODE_L7Q](#) = 28
- obs code: E5bQ,B2Q (GAL,BDS)*
- constexpr uint32_t [CODE_L7X](#) = 29
- obs code: E5bI+Q,B2I+Q (GAL,BDS)*
- constexpr uint32_t [CODE_L6A](#) = 30
- obs code: E6A (GAL)*
- constexpr uint32_t [CODE_L6B](#) = 31
- obs code: E6B (GAL)*
- constexpr uint32_t [CODE_L6C](#) = 32
- obs code: E6C (GAL)*
- constexpr uint32_t [CODE_L6X](#) = 33
- obs code: E6B+C,LEXS+L,B3I+Q (GAL,QZS,BDS)*
- constexpr uint32_t [CODE_L6Z](#) = 34
- obs code: E6A+B+C (GAL)*
- constexpr uint32_t [CODE_L6S](#) = 35
- obs code: LEXS (QZS)*
- constexpr uint32_t [CODE_L6L](#) = 36
- obs code: LEXL (QZS)*
- constexpr uint32_t [CODE_L8I](#) = 37
- obs code: E5(a+b)I (GAL)*
- constexpr uint32_t [CODE_L8Q](#) = 38
- obs code: E5(a+b)Q (GAL)*
- constexpr uint32_t [CODE_L8X](#) = 39
- obs code: E5(a+b)I+Q (GAL)*
- constexpr uint32_t [CODE_L2I](#) = 40
- obs code: B1I (BDS)*
- constexpr uint32_t [CODE_L2Q](#) = 41
- obs code: B1Q (BDS)*
- constexpr uint32_t [CODE_L6I](#) = 42
- obs code: B3I (BDS)*
- constexpr uint32_t [CODE_L6Q](#) = 43
- obs code: B3Q (BDS)*
- constexpr uint32_t [CODE_L3I](#) = 44
- obs code: G3I (GLO)*
- constexpr uint32_t [CODE_L3Q](#) = 45
- obs code: G3Q (GLO)*

- `constexpr uint32_t CODE_L3X = 46`
obs code: G3I+Q (GLO)
- `constexpr uint32_t CODE_L1I = 47`
obs code: B1I (BDS)
- `constexpr uint32_t CODE_L1Q = 48`
obs code: B1Q (BDS)
- `constexpr uint32_t CODE_L5A = 49`
obs code: L5A SPS (IRN)
- `constexpr uint32_t CODE_L5B = 50`
obs code: L5B RS(D) (IRN)
- `constexpr uint32_t CODE_L5C = 51`
obs code: L5C RS(P) (IRN)
- `constexpr uint32_t CODE_L9A = 52`
obs code: SA SPS (IRN)
- `constexpr uint32_t CODE_L9B = 53`
obs code: SB RS(D) (IRN)
- `constexpr uint32_t CODE_L9C = 54`
obs code: SC RS(P) (IRN)
- `constexpr uint32_t CODE_L9X = 55`
obs code: SB+C (IRN)
- `constexpr int32_t MAXCODE = 55`
max number of obs code

11.155.1 Detailed Description

GNSS Observable codes.

Author

Carles Fernandez, 2017. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.155.2 Variable Documentation

11.155.2.1 CODE_L1A

```
constexpr uint32_t CODE_L1A = 10
```

obs code: E1A (GAL)

Definition at line 37 of file `gnss_obs_codes.h`.

11.155.2.2 CODE_L1B

```
constexpr uint32_t CODE_L1B = 11
```

obs code: E1B (GAL)

Definition at line 38 of file gnss_obs_codes.h.

11.155.2.3 CODE_L1C

```
constexpr uint32_t CODE_L1C = 1
```

obs code: L1C/A,G1C/A,E1C (GPS,GLO,GAL,QZS,SBS)

Definition at line 28 of file gnss_obs_codes.h.

11.155.2.4 CODE_L1E

```
constexpr uint32_t CODE_L1E = 9
```

(not used)

Definition at line 36 of file gnss_obs_codes.h.

11.155.2.5 CODE_L1I

```
constexpr uint32_t CODE_L1I = 47
```

obs code: B1I (BDS)

Definition at line 74 of file gnss_obs_codes.h.

11.155.2.6 CODE_L1L

```
constexpr uint32_t CODE_L1L = 8
```

obs code: L1C(P) (GPS,QZS)

Definition at line 35 of file gnss_obs_codes.h.

11.155.2.7 CODE_L1M

```
constexpr uint32_t CODE_L1M = 5
```

obs code: L1M (GPS)

Definition at line 32 of file gnss_obs_codes.h.

11.155.2.8 CODE_L1N

```
constexpr uint32_t CODE_L1N = 6
```

obs code: L1codeless (GPS)

Definition at line 33 of file gnss_obs_codes.h.

11.155.2.9 CODE_L1P

```
constexpr uint32_t CODE_L1P = 2
```

obs code: L1P,G1P (GPS,GLO)

Definition at line 29 of file gnss_obs_codes.h.

11.155.2.10 CODE_L1Q

```
constexpr uint32_t CODE_L1Q = 48
```

obs code: B1Q (BDS)

Definition at line 75 of file gnss_obs_codes.h.

11.155.2.11 CODE_L1S

```
constexpr uint32_t CODE_L1S = 7
```

obs code: L1C(D) (GPS,QZS)

Definition at line 34 of file gnss_obs_codes.h.

11.155.2.12 CODE_L1W

```
constexpr uint32_t CODE_L1W = 3
```

obs code: L1 Z-track (GPS)

Definition at line 30 of file gnss_obs_codes.h.

11.155.2.13 CODE_L1X

```
constexpr uint32_t CODE_L1X = 12
```

obs code: E1B+C,L1C(D+P) (GAL,QZS)

Definition at line 39 of file gnss_obs_codes.h.

11.155.2.14 CODE_L1Y

```
constexpr uint32_t CODE_L1Y = 4
```

obs code: L1Y (GPS)

Definition at line 31 of file gnss_obs_codes.h.

11.155.2.15 CODE_L1Z

```
constexpr uint32_t CODE_L1Z = 13
```

obs code: E1A+B+C,L1SAIF (GAL,QZS)

Definition at line 40 of file gnss_obs_codes.h.

11.155.2.16 CODE_L2C

```
constexpr uint32_t CODE_L2C = 14
```

obs code: L2C/A,G1C/A (GPS,GLO)

Definition at line 41 of file gnss_obs_codes.h.

11.155.2.17 CODE_L2D

```
constexpr uint32_t CODE_L2D = 15
```

obs code: L2 L1C/A-(P2-P1) (GPS)

Definition at line 42 of file gnss_obs_codes.h.

11.155.2.18 CODE_L2I

```
constexpr uint32_t CODE_L2I = 40
```

obs code: B1I (BDS)

Definition at line 67 of file gnss_obs_codes.h.

11.155.2.19 CODE_L2L

```
constexpr uint32_t CODE_L2L = 17
```

obs code: L2C(L) (GPS,QZS)

Definition at line 44 of file gnss_obs_codes.h.

11.155.2.20 CODE_L2M

```
constexpr uint32_t CODE_L2M = 22
```

obs code: L2M (GPS)

Definition at line 49 of file gnss_obs_codes.h.

11.155.2.21 CODE_L2N

```
constexpr uint32_t CODE_L2N = 23
```

obs code: L2codeless (GPS)

Definition at line 50 of file gnss_obs_codes.h.

11.155.2.22 CODE_L2P

```
constexpr uint32_t CODE_L2P = 19
```

obs code: L2P,G2P (GPS,GLO)

Definition at line 46 of file gnss_obs_codes.h.

11.155.2.23 CODE_L2Q

```
constexpr uint32_t CODE_L2Q = 41
```

obs code: B1Q (BDS)

Definition at line 68 of file gnss_obs_codes.h.

11.155.2.24 CODE_L2S

```
constexpr uint32_t CODE_L2S = 16
```

obs code: L2C(M) (GPS,QZS)

Definition at line 43 of file gnss_obs_codes.h.

11.155.2.25 CODE_L2W

```
constexpr uint32_t CODE_L2W = 20
```

obs code: L2 Z-track (GPS)

Definition at line 47 of file gnss_obs_codes.h.

11.155.2.26 CODE_L2X

```
constexpr uint32_t CODE_L2X = 18
```

obs code: L2C(M+L),B1I+Q (GPS,QZS,BDS)

Definition at line 45 of file gnss_obs_codes.h.

11.155.2.27 CODE_L2Y

```
constexpr uint32_t CODE_L2Y = 21
```

obs code: L2Y (GPS)

Definition at line 48 of file gnss_obs_codes.h.

11.155.2.28 CODE_L3I

```
constexpr uint32_t CODE_L3I = 44
```

obs code: G3I (GLO)

Definition at line 71 of file gnss_obs_codes.h.

11.155.2.29 CODE_L3Q

```
constexpr uint32_t CODE_L3Q = 45
```

obs code: G3Q (GLO)

Definition at line 72 of file gnss_obs_codes.h.

11.155.2.30 CODE_L3X

```
constexpr uint32_t CODE_L3X = 46
```

obs code: G3I+Q (GLO)

Definition at line 73 of file gnss_obs_codes.h.

11.155.2.31 CODE_L5A

```
constexpr uint32_t CODE_L5A = 49
```

obs code: L5A SPS (IRN)

Definition at line 76 of file gnss_obs_codes.h.

11.155.2.32 CODE_L5B

```
constexpr uint32_t CODE_L5B = 50
```

obs code: L5B RS(D) (IRN)

Definition at line 77 of file gnss_obs_codes.h.

11.155.2.33 CODE_L5C

```
constexpr uint32_t CODE_L5C = 51
```

obs code: L5C RS(P) (IRN)

Definition at line 78 of file gnss_obs_codes.h.

11.155.2.34 CODE_L5I

```
constexpr uint32_t CODE_L5I = 24
```

obs code: L5/E5aI (GPS,GAL,QZS,SBS)

Definition at line 51 of file gnss_obs_codes.h.

11.155.2.35 CODE_L5Q

```
constexpr uint32_t CODE_L5Q = 25
```

obs code: L5/E5aQ (GPS,GAL,QZS,SBS)

Definition at line 52 of file gnss_obs_codes.h.

11.155.2.36 CODE_L5X

```
constexpr uint32_t CODE_L5X = 26
```

obs code: L5/E5aI+Q/L5B+C (GPS,GAL,QZS,IRN,SBS)

Definition at line 53 of file gnss_obs_codes.h.

11.155.2.37 CODE_L6A

```
constexpr uint32_t CODE_L6A = 30
```

obs code: E6A (GAL)

Definition at line 57 of file gnss_obs_codes.h.

11.155.2.38 CODE_L6B

```
constexpr uint32_t CODE_L6B = 31
```

obs code: E6B (GAL)

Definition at line 58 of file gnss_obs_codes.h.

11.155.2.39 CODE_L6C

```
constexpr uint32_t CODE_L6C = 32
```

obs code: E6C (GAL)

Definition at line 59 of file gnss_obs_codes.h.

11.155.2.40 CODE_L6I

```
constexpr uint32_t CODE_L6I = 42
```

obs code: B3I (BDS)

Definition at line 69 of file gnss_obs_codes.h.

11.155.2.41 CODE_L6L

```
constexpr uint32_t CODE_L6L = 36
```

obs code: LEXL (QZS)

Definition at line 63 of file gnss_obs_codes.h.

11.155.2.42 CODE_L6Q

```
constexpr uint32_t CODE_L6Q = 43
```

obs code: B3Q (BDS)

Definition at line 70 of file gnss_obs_codes.h.

11.155.2.43 CODE_L6S

```
constexpr uint32_t CODE_L6S = 35
```

obs code: LEXS (QZS)

Definition at line 62 of file gnss_obs_codes.h.

11.155.2.44 CODE_L6X

```
constexpr uint32_t CODE_L6X = 33
```

obs code: E6B+C,LEXS+L,B3I+Q (GAL,QZS,BDS)

Definition at line 60 of file gnss_obs_codes.h.

11.155.2.45 CODE_L6Z

```
constexpr uint32_t CODE_L6Z = 34
```

obs code: E6A+B+C (GAL)

Definition at line 61 of file gnss_obs_codes.h.

11.155.2.46 CODE_L7I

```
constexpr uint32_t CODE_L7I = 27
```

obs code: E5bI,B2I (GAL,BDS)

Definition at line 54 of file gnss_obs_codes.h.

11.155.2.47 CODE_L7Q

```
constexpr uint32_t CODE_L7Q = 28
```

obs code: E5bQ,B2Q (GAL,BDS)

Definition at line 55 of file gnss_obs_codes.h.

11.155.2.48 CODE_L7X

```
constexpr uint32_t CODE_L7X = 29
```

obs code: E5bl+Q,B2l+Q (GAL,BDS)

Definition at line 56 of file gnss_obs_codes.h.

11.155.2.49 CODE_L8I

```
constexpr uint32_t CODE_L8I = 37
```

obs code: E5(a+b)I (GAL)

Definition at line 64 of file gnss_obs_codes.h.

11.155.2.50 CODE_L8Q

```
constexpr uint32_t CODE_L8Q = 38
```

obs code: E5(a+b)Q (GAL)

Definition at line 65 of file gnss_obs_codes.h.

11.155.2.51 CODE_L8X

```
constexpr uint32_t CODE_L8X = 39
```

obs code: E5(a+b)I+Q (GAL)

Definition at line 66 of file gnss_obs_codes.h.

11.155.2.52 CODE_L9A

```
constexpr uint32_t CODE_L9A = 52
```

obs code: SA SPS (IRN)

Definition at line 79 of file gnss_obs_codes.h.

11.155.2.53 CODE_L9B

```
constexpr uint32_t CODE_L9B = 53
```

obs code: SB RS(D) (IRN)

Definition at line 80 of file gnss_obs_codes.h.

11.155.2.54 CODE_L9C

```
constexpr uint32_t CODE_L9C = 54
```

obs code: SC RS(P) (IRN)

Definition at line 81 of file gnss_obs_codes.h.

11.155.2.55 CODE_L9X

```
constexpr uint32_t CODE_L9X = 55
```

obs code: SB+C (IRN)

Definition at line 82 of file gnss_obs_codes.h.

11.155.2.56 CODE_NONE

```
constexpr uint32_t CODE_NONE = 0
```

obs code: none or unknown

Definition at line 27 of file gnss_obs_codes.h.

11.155.2.57 MAXCODE

```
constexpr int32_t MAXCODE = 55
```

max number of obs code

Definition at line 83 of file gnss_obs_codes.h.

11.156 gnss_satellite.h File Reference

Interface of the [Gnss_Satellite](#) class.

```
#include <stdint>
#include <map>
#include <ostream>
#include <set>
#include <string>
```

Classes

- class [Gnss_Satellite](#)
This class represents a GNSS satellite.

11.156.1 Detailed Description

Interface of the [Gnss_Satellite](#) class.

Author

Carles Fernandez-Prades, 2012. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.157 gnss_sdr_create_directory.h File Reference

Create a directory.

```
#include <string>
```

Functions

- bool **gnss_sdr_create_directory** (const std::string &foldername)

11.157.1 Detailed Description

Create a directory.

Author

Carles Fernandez-Prades, 2018. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.158 gnss_sdr_flags.h File Reference

Helper file for gnss-sdr commandline flags.

```
#include <gflags/gflags.h>
#include <cstdint>
```

Functions

- **DECLARE_string** (c)
Path to the configuration file.
- **DECLARE_string** (config_file)
Path to the configuration file.
- **DECLARE_string** (log_dir)
Path to the folder in which logging will be stored.
- **DECLARE_string** (s)
Path to the file containing the signal samples.
- **DECLARE_string** (signal_source)
Path to the file containing the signal samples.
- **DECLARE_bool** (rf_shutdown)
Shutdown RF when program exits.
- **DECLARE_int32** (doppler_max)
If defined, maximum Doppler value in the search grid, in Hz (overrides the configuration file).
- **DECLARE_int32** (doppler_step)
If defined, sets the frequency step in the search grid, in Hz, in Hz (overrides the configuration file).
- **DECLARE_int32** (cn0_samples)
Number of correlator outputs used for CN0 estimation.
- **DECLARE_int32** (cn0_min)

- *Minimum valid CN0 (in dB-Hz).*
- `DECLARE_int32` (max_lock_fail)
 - *Maximum number of code lock failures before dropping a satellite.*
- `DECLARE_int32` (max_carrier_lock_fail)
 - *Maximum number of carrier lock failures before dropping a satellite.*
- `DECLARE_double` (carrier_lock_th)
 - *Carrier lock threshold (in rad).*
- `DECLARE_double` (dll_bw_hz)
 - *Bandwidth of the DLL low pass filter, in Hz (overrides the configuration file).*
- `DECLARE_double` (pll_bw_hz)
 - *Bandwidth of the PLL low pass filter, in Hz (overrides the configuration file).*
- `DECLARE_int32` (carrier_smoothing_factor)
 - *Sets carrier smoothing factor M (overrides the configuration file).*
- `DECLARE_string` (RINEX_version)
 - *If defined, specifies the RINEX version (2.11 or 3.02). Overrides the configuration file.*
- `DECLARE_string` (RINEX_name)
 - *If defined, specifies the RINEX files base name.*

Variables

- `const int32_t DEFAULT_CARRIER_SMOOTHING_FACTOR = 200`

11.158.1 Detailed Description

Helper file for gnss-sdr commandline flags.

Author

Carles Fernandez-Prades, 2018. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.158.2 Function Documentation

11.158.2.1 `DECLARE_bool()`

```
DECLARE_bool (
    rf_shutdown )
```

Shutdown RF when program exits.

11.158.2.2 DECLARE_double() [1/3]

```
DECLARE_double (
    carrier_lock_th )
```

Carrier lock threshold (in rad).

11.158.2.3 DECLARE_double() [2/3]

```
DECLARE_double (
    dll_bw_hz )
```

Bandwidth of the DLL low pass filter, in Hz (overrides the configuration file).

11.158.2.4 DECLARE_double() [3/3]

```
DECLARE_double (
    pll_bw_hz )
```

Bandwidth of the PLL low pass filter, in Hz (overrides the configuration file).

11.158.2.5 DECLARE_int32() [1/7]

```
DECLARE_int32 (
    doppler_max )
```

If defined, maximum Doppler value in the search grid, in Hz (overrides the configuration file).

11.158.2.6 DECLARE_int32() [2/7]

```
DECLARE_int32 (
    doppler_step )
```

If defined, sets the frequency step in the search grid, in Hz, in Hz (overrides the configuration file).

11.158.2.7 DECLARE_int32() [3/7]

```
DECLARE_int32 (
    cn0_samples )
```

Number of correlator outputs used for CN0 estimation.

11.158.2.8 DECLARE_int32() [4/7]

```
DECLARE_int32 (
    cn0_min )
```

Minimum valid CN0 (in dB-Hz).

11.158.2.9 DECLARE_int32() [5/7]

```
DECLARE_int32 (
    max_lock_fail )
```

Maximum number of code lock failures before dropping a satellite.

11.158.2.10 DECLARE_int32() [6/7]

```
DECLARE_int32 (
    max_carrier_lock_fail )
```

Maximum number of carrier lock failures before dropping a satellite.

11.158.2.11 DECLARE_int32() [7/7]

```
DECLARE_int32 (
    carrier_smoothing_factor )
```

Sets carrier smoothing factor M (overrides the configuration file).

11.158.2.12 DECLARE_string() [1/7]

```
DECLARE_string (
    c )
```

Path to the configuration file.

11.158.2.13 DECLARE_string() [2/7]

```
DECLARE_string (
    config_file )
```

Path to the configuration file.

11.158.2.14 DECLARE_string() [3/7]

```
DECLARE_string (
    log_dir )
```

Path to the folder in which logging will be stored.

11.158.2.15 DECLARE_string() [4/7]

```
DECLARE_string (
    s )
```

Path to the file containing the signal samples.

11.158.2.16 DECLARE_string() [5/7]

```
DECLARE_string (
    signal_source )
```

Path to the file containing the signal samples.

11.158.2.17 DECLARE_string() [6/7]

```
DECLARE_string (
    RINEX_version )
```

If defined, specifies the RINEX version (2.11 or 3.02). Overrides the configuration file.

11.158.2.18 DECLARE_string() [7/7]

```
DECLARE_string (
    RINEX_name )
```

If defined, specifies the RINEX files base name.

11.159 gnss_sdr_fpga_sample_counter.h File Reference

Simple block to report the current receiver time based on the output of the tracking or telemetry blocks.

```
#include <gnuradio/block.h>
#include <gnuradio/types.h>
#include <cstdint>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [gnss_sdr_fpga_sample_counter](#)

Typedefs

- using **gnss_sdr_fpga_sample_counter_sptr** = boost::shared_ptr< [gnss_sdr_fpga_sample_counter](#) >

Functions

- gnss_sdr_fpga_sample_counter_sptr **gnss_sdr_make_fpga_sample_counter** (double _fs, int32_t _↔ interval_ms)

11.159.1 Detailed Description

Simple block to report the current receiver time based on the output of the tracking or telemetry blocks.

Author

Javier Arribas 2018. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.160 gnss_sdr_make_unique.h File Reference

This file implements std::make_unique for C++11.

11.160.1 Detailed Description

This file implements std::make_unique for C++11.

Author

Carles Fernandez-Prades, 2020. [cfernandez\(at\)cttc.es](mailto:cfernandez(at)cttc.es)

Based on <https://stackoverflow.com/a/17902439>

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.161 gnss_sdr_sample_counter.h File Reference

Simple block to report the current receiver time based on the output of the tracking or telemetry blocks.

```
#include <gnuradio/sync_decimator.h>
#include <gnuradio/types.h>
#include <cstdint>
#include <cstdint>
#include <boost/shared_ptr.hpp>
```

Classes

- class [gnss_sdr_sample_counter](#)

Typedefs

- using **gnss_sdr_sample_counter_sptr** = boost::shared_ptr< [gnss_sdr_sample_counter](#) >

Functions

- gnss_sdr_sample_counter_sptr **gnss_sdr_make_sample_counter** (double _fs, int32_t _interval_ms, size_t _size)

11.161.1 Detailed Description

Simple block to report the current receiver time based on the output of the tracking or telemetry blocks.

Author

Javier Arribas 2018. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.162 gnss_sdr_supl_client.h File Reference

class that implements a C++ interface to external Secure User Location Protocol (SUPL) client library.

```
#include "supl.h"
#include "agnss_ref_location.h"
#include "agnss_ref_time.h"
#include "galileo_almanac.h"
#include "galileo_ephemeris.h"
#include "galileo_iono.h"
#include "galileo_utc_model.h"
#include "glonass_gnav_ephemeris.h"
#include "glonass_gnav_utc_model.h"
#include "gps_acq_assist.h"
#include "gps_almanac.h"
#include "gps_cnav_ephemeris.h"
#include "gps_cnav_utc_model.h"
#include "gps_ephemeris.h"
#include "gps_iono.h"
#include "gps_utc_model.h"
#include <fstream>
#include <map>
#include <string>
```

Classes

- class [Gnss_Sdr_Supl_Client](#)

class that implements a C++ interface to external Secure User Location Protocol (SUPL) client library..

11.162.1 Detailed Description

class that implements a C++ interface to external Secure User Location Protocol (SUPL) client library.

Author

Javier Arribas, 2013. jarribas(at)cttc.es

TODO: put here supl.c author info class that implements a C++ interface to external Secure User Location Protocol (SUPL) client library.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.163 gnss_sdr_time_counter.h File Reference

Simple block to report the current receiver time based on the output of the tracking or telemetry blocks.

```
#include <gnuradio/block.h>
#include <gnuradio/types.h>
#include <cstdint>
#include <memory>
```

Classes

- class [gnss_sdr_time_counter](#)

Typedefs

- using [gnss_sdr_time_counter_sptr](#) = std::shared_ptr< [gnss_sdr_time_counter](#) >

Functions

- [gnss_sdr_time_counter_sptr gnss_sdr_make_time_counter](#) ()

11.163.1 Detailed Description

Simple block to report the current receiver time based on the output of the tracking or telemetry blocks.

Author

Antonio Ramos 2018. antonio.ramosdet(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.164 gnss_sdr_valve.h File Reference

Interface of a GNU Radio block that sends a STOP message to the control queue right after a specific number of samples have passed through it.

```
#include "concurrent_queue.h"
#include <gnuradio/sync_block.h>
#include <gnuradio/types.h>
#include <pmt/pmt.h>
#include <cstddef>
#include <cstdint>
#include <boost/shared_ptr.hpp>
```

Classes

- class [Gnss_Sdr_Valve](#)

Implementation of a GNU Radio block that sends a STOP message to the control queue right after a specific number of samples have passed through it.

Functions

- `boost::shared_ptr< Gnss_Sdr_Valve > gnss_sdr_make_valve` (`size_t sizeof_stream_item`, `uint64_t nitems`, `Concurrent_Queue< pmt::pmt_t > *queue`)
- `boost::shared_ptr< Gnss_Sdr_Valve > gnss_sdr_make_valve` (`size_t sizeof_stream_item`, `uint64_t nitems`, `Concurrent_Queue< pmt::pmt_t > *queue`, `bool stop_flowgraph`)

11.164.1 Detailed Description

Interface of a GNU Radio block that sends a STOP message to the control queue right after a specific number of samples have passed through it.

Author

Javier Arribas, 2018. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)
 Carlos Aviles, 2010. [carlos.avilesr\(at\)gmail.com](mailto:carlos.avilesr(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.165 gnss_signal.h File Reference

Implementation of the [Gnss_Signal](#) class.

```
#include "gnss_satellite.h"
#include <ostream>
#include <string>
```

Classes

- class [Gnss_Signal](#)

This class represents a GNSS signal.

11.165.1 Detailed Description

Implementation of the [Gnss_Signal](#) class.

Author

Luis Esteve, 2012. [luis\(at\)epsilon-formacion.com](mailto:luis(at)epsilon-formacion.com)

Javier Arribas, 2012. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.166 gnss_signal_processing.h File Reference

This library gathers a few functions used by the algorithms of gnss-sdr, regardless of system used.

```
#include <complex>
#include <stdint>
#include <gsl/gsl>
```

Functions

- void [complex_exp_gen](#) (own::span< std::complex< float >> _dest, double _f, double _fs)
This function generates a complex exponential in _dest.
- void [complex_exp_gen_conj](#) (own::span< std::complex< float >> _dest, double _f, double _fs)
This function generates a conjugate complex exponential in _dest.
- void [hex_to_binary_converter](#) (own::span< int32_t > _dest, char _from)
This function makes a conversion from hex (the input is a char) to binary (the output are 4 ints with +1 or -1 values).
- void [resampler](#) (const own::span< float > _from, own::span< float > _dest, float _fs_in, float _fs_out)
This function resamples a sequence of float values.
- void [resampler](#) (own::span< const std::complex< float >> _from, own::span< std::complex< float >> _dest, float _fs_in, float _fs_out)
This function resamples a sequence of complex values.

11.166.1 Detailed Description

This library gathers a few functions used by the algorithms of gnss-sdr, regardless of system used.

Author

Luis Esteve, 2012. [luis\(at\)epsilon-formacion.com](mailto:luis(at)epsilon-formacion.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.166.2 Function Documentation

11.166.2.1 `complex_exp_gen()`

```
void complex_exp_gen (
    own::span< std::complex< float >> _dest,
    double _f,
    double _fs )
```

This function generates a complex exponential in `_dest`.

11.166.2.2 `complex_exp_gen_conj()`

```
void complex_exp_gen_conj (
    own::span< std::complex< float >> _dest,
    double _f,
    double _fs )
```

This function generates a conjugate complex exponential in `_dest`.

11.166.2.3 `hex_to_binary_converter()`

```
void hex_to_binary_converter (
    own::span< int32_t > _dest,
    char _from )
```

This function makes a conversion from hex (the input is a char) to binary (the output are 4 ints with +1 or -1 values).

11.166.2.4 `resampler()` [1/2]

```
void resampler (
    const own::span< float > _from,
    own::span< float > _dest,
    float _fs_in,
    float _fs_out )
```

This function resamples a sequence of float values.

11.166.2.5 resampler() [2/2]

```
void resampler (
    own::span< const std::complex< float >> _from,
    own::span< std::complex< float >> _dest,
    float _fs_in,
    float _fs_out )
```

This function resamples a sequence of complex values.

11.167 gnss_synchro.h File Reference

Interface of the [Gnss_Synchro](#) class.

```
#include <boost/serialization/nvp.hpp>
#include <cstdint>
#include <utility>
```

Classes

- class [Gnss_Synchro](#)

This is the class that contains the information that is shared by the processing blocks.

11.167.1 Detailed Description

Interface of the [Gnss_Synchro](#) class.

Author

Luis Esteve, 2012. [luis\(at\)epsilon-formacion.com](mailto:luis(at)epsilon-formacion.com) Javier Arribas, 2012. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)

Álvaro Cebrián Juan, 2018. [acebrianjuan\(at\)gmail.com](mailto:acebrianjuan(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.168 gnss_synchro_monitor.h File Reference

Interface of a receiver monitoring block which allows sending a data stream with the receiver internal parameters ([Gnss_Synchro](#) objects) to local or remote clients over UDP.

```
#include "gnss_synchro_udp_sink.h"
#include <gnuradio/runtime_types.h>
#include <gnuradio/sync_block.h>
#include <memory>
#include <string>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [gnss_synchro_monitor](#)

This class implements a monitoring block which allows sending a data stream with the receiver internal parameters ([Gnss_Synchro](#) objects) to local or remote clients over UDP.

Typedefs

- using **gnss_synchro_monitor_sptr** = boost::shared_ptr< [gnss_synchro_monitor](#) >

Functions

- `gnss_synchro_monitor_sptr gnss_synchro_make_monitor (int n_channels, int decimation_factor, int udp_port, const std::vector< std::string > &udp_addresses, bool enable_protobuf)`

11.168.1 Detailed Description

Interface of a receiver monitoring block which allows sending a data stream with the receiver internal parameters ([Gnss_Synchro](#) objects) to local or remote clients over UDP.

Author

Álvaro Cebrián Juan, 2018. [acebrianjuan\(at\)gmail.com](mailto:acebrianjuan(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.169 gnss_synchro_udp_sink.h File Reference

Interface of a class that sends serialized [Gnss_Synchro](#) objects over udp to one or multiple endpoints.

```
#include "gnss_synchro.h"
#include "serdes_gnss_synchro.h"
#include <boost/asio.hpp>
#include <boost/system/error_code.hpp>
#include <stdint>
#include <string>
#include <vector>
```

Classes

- class [Gnss_Synchro_Udp_Sink](#)

This class sends serialized [Gnss_Synchro](#) objects over UDP to one or multiple endpoints.

Typedefs

- using **b_io_context** = boost::asio::io_service

11.169.1 Detailed Description

Interface of a class that sends serialized [Gnss_Synchro](#) objects over udp to one or multiple endpoints.

Author

Álvaro Cebrián Juan, 2018. [acebrianjuan\(at\)gmail.com](mailto:acebrianjuan(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.170 gnuplot_i.h File Reference

A C++ interface to gnuplot.

```
#include <gflags/gflags.h>
#include <cmath>
#include <cstdlib>
#include <cstring>
#include <fstream>
#include <iostream>
#include <list>
#include <sstream>
#include <stdexcept>
#include <string>
#include <sys/stat.h>
#include <vector>
```

Classes

- class [GnuplotException](#)
- class [Gnuplot](#)

Functions

- **DEFINE_bool** (show_plots, true, "Show plots on screen. Disable for non-interactive testing.")
- `template<typename Container >`
void **stringtok** (Container &container, std::string const &in, const char *const delimiters=" \\")

11.170.1 Detailed Description

A C++ interface to gnuplot.

Author

Carles Fernandez-Prades, 2017. cfernandez(at)cttc.es

Original source code found at <https://code.google.com/archive/p/gnuplot-cpp/> by Jeremy Conlin jeremit0(at)gmail.com

Version history: 0. C interface by N. Devillard (27/01/03)

1. C++ interface: direct translation from the C interface by Rajarshi Guha (07/03/03)
2. corrections for Win32 compatibility by V. Chyzhdzenka (20/05/03)
3. some member functions added, corrections for Win32 and Linux compatibility by M. Burgis (10/03/08)
4. Some fixes and improvements for Linux and macOS

by C. Fernandez (22/10/17)

Copyright (C) 2013-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.171 `gps_acq_assist.h` File Reference

Interface of a GPS RRLL ACQUISITION ASSISTANCE storage.

```
#include <cstdint>
```

Classes

- class [Gps_Acq_Assist](#)

This class is a storage for the GPS GSM RRLL acquisition assistance data as described in Digital cellular telecommunications system (Phase 2+); Location Services (LCS); Mobile Station (MS) - Serving Mobile Location Centre (SMLC) Radio Resource LCS Protocol (RRLP) (3GPP TS 44.031 version 5.12.0 Release 5)

11.171.1 Detailed Description

Interface of a GPS RRLL ACQUISITION ASSISTANCE storage.

Author

Javier Arribas, 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.172 gps_almanac.h File Reference

Interface of a GPS ALMANAC storage.

```
#include <boost/serialization/nvp.hpp>
#include <cstdint>
```

Classes

- class [Gps_Almanac](#)

This class is a storage for the GPS SV ALMANAC data as described in IS-GPS-200K.

11.172.1 Detailed Description

Interface of a GPS ALMANAC storage.

Author

Javier Arribas, 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.173 GPS_CNAV.h File Reference

Defines parameters for GPS CNAV.

```
#include "MATH_CONSTANTS.h"
#include <cstdint>
#include <utility>
#include <vector>
```

Functions

- `const std::vector< std::pair< int32_t, int32_t > > CNAV_PRN` ({9, 6})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_MSG_TYPE` ({15, 6})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_TOW` ({21, 17})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_ALERT_FLAG` ({38, 1})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_WN` ({39, 13})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_HEALTH` ({52, 3})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_TOP1` ({55, 11})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_URA` ({66, 5})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_TOE1` ({71, 11})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_DELTA_A` ({82, 26})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_A_DOT` ({108, 25})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_DELTA_N0` ({133, 17})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_DELTA_N0_DOT` ({150, 23})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_M0` ({173, 33})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_E_ECCENTRICITY` ({206, 33})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_OMEGA` ({239, 33})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_INTEGRITY_FLAG` ({272, 1})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_L2_PHASING_FLAG` ({273, 1})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_TOE2` ({39, 11})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_OMEGA0` ({50, 33})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_I0` ({83, 33})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_DELTA_OMEGA_DOT` ({116, 17})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_I0_DOT` ({133, 15})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_CIS` ({148, 16})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_CIC` ({164, 16})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_CRS` ({180, 24})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_CRC` ({204, 24})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_CUS` ({228, 21})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_CUC` ({249, 21})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_TOP2` ({39, 11})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_URA_NED0` ({50, 5})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_URA_NED1` ({55, 3})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_URA_NED2` ({58, 3})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_TOC` ({61, 11})
- `const std::vector< std::pair< int, int > > CNAV_AF0` ({72, 26})
- `const std::vector< std::pair< int, int > > CNAV_AF1` ({98, 20})
- `const std::vector< std::pair< int, int > > CNAV_AF2` ({118, 10})
- `const std::vector< std::pair< int, int > > CNAV_TGD` ({128, 13})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_ISCL1` ({141, 13})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_ISCL2` ({154, 13})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_ISCL5I` ({167, 13})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_ISCL5Q` ({180, 13})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_ALPHA0` ({193, 8})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_ALPHA1` ({201, 8})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_ALPHA2` ({209, 8})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_ALPHA3` ({217, 8})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_BETA0` ({225, 8})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_BETA1` ({233, 8})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_BETA2` ({241, 8})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_BETA3` ({249, 8})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_WNOP` ({257, 8})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_A0` ({128, 16})
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_A1` ({144, 13})

- `const std::vector< std::pair< int32_t, int32_t > > CNAV_A2 {{{157, 7}}}`
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_DELTA_TLS {{{164, 8}}}`
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_TOT {{{172, 16}}}`
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_WN_OT {{{188, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_WN_LSF {{{201, 13}}}`
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_DN {{{214, 4}}}`
- `const std::vector< std::pair< int32_t, int32_t > > CNAV_DELTA_TLSF {{{218, 8}}}`

Variables

- `constexpr int32_t GPS_CNAV_DATA_PAGE_BITS = 300`
- `constexpr int32_t CNAV_TOW_LSB = 6`
- `constexpr int32_t CNAV_TOP1_LSB = 300`
- `constexpr int32_t CNAV_TOE1_LSB = 300`
- `constexpr double CNAV_DELTA_A_LSB = TWO_N9`
- `constexpr double CNAV_A_DOT_LSB = TWO_N21`
- `constexpr double CNAV_DELTA_N0_LSB = TWO_N44 * GNSS_PI`
- `constexpr double CNAV_DELTA_N0_DOT_LSB = TWO_N57 * GNSS_PI`
- `constexpr double CNAV_M0_LSB = TWO_N32 * GNSS_PI`
- `constexpr double CNAV_E_ECCENTRICITY_LSB = TWO_N34`
- `constexpr double CNAV_OMEGA_LSB = TWO_N32 * GNSS_PI`
- `constexpr int32_t CNAV_TOE2_LSB = 300`
- `constexpr double CNAV_OMEGA0_LSB = TWO_N32 * GNSS_PI`
- `constexpr double CNAV_I0_LSB = TWO_N32 * GNSS_PI`
- `constexpr double CNAV_DELTA_OMEGA_DOT_LSB = TWO_N44 * GNSS_PI`
- `constexpr double CNAV_I0_DOT_LSB = TWO_N44 * GNSS_PI`
- `constexpr double CNAV_CIS_LSB = TWO_N30`
- `constexpr double CNAV_CIC_LSB = TWO_N30`
- `constexpr double CNAV_CRS_LSB = TWO_N8`
- `constexpr double CNAV_CRC_LSB = TWO_N8`
- `constexpr double CNAV_CUS_LSB = TWO_N30`
- `constexpr double CNAV_CUC_LSB = TWO_N30`
- `constexpr int32_t CNAV_TOP2_LSB = 300`
- `constexpr int32_t CNAV_TOC_LSB = 300`
- `constexpr double CNAV_AF0_LSB = TWO_N35`
- `constexpr double CNAV_AF1_LSB = TWO_N48`
- `constexpr double CNAV_AF2_LSB = TWO_N60`
- `constexpr double CNAV_TGD_LSB = TWO_N35`
- `constexpr double CNAV_ISCL1_LSB = TWO_N35`
- `constexpr double CNAV_ISCL2_LSB = TWO_N35`
- `constexpr double CNAV_ISCL5I_LSB = TWO_N35`
- `constexpr double CNAV_ISCL5Q_LSB = TWO_N35`
- `constexpr double CNAV_ALPHA0_LSB = TWO_N30`
- `constexpr double CNAV_ALPHA1_LSB = TWO_N27`
- `constexpr double CNAV_ALPHA2_LSB = TWO_N24`
- `constexpr double CNAV_ALPHA3_LSB = TWO_N24`
- `constexpr double CNAV_BETA0_LSB = TWO_P11`
- `constexpr double CNAV_BETA1_LSB = TWO_P14`
- `constexpr double CNAV_BETA2_LSB = TWO_P16`
- `constexpr double CNAV_BETA3_LSB = TWO_P16`
- `constexpr double CNAV_A0_LSB = TWO_N35`
- `constexpr double CNAV_A1_LSB = TWO_N51`
- `constexpr double CNAV_A2_LSB = TWO_N68`
- `constexpr int32_t CNAV_DELTA_TLS_LSB = 1`

- `constexpr int32_t CNAV_TOT_LSB = TWO_P4`
- `constexpr int32_t CNAV_WN_OT_LSB = 1`
- `constexpr int32_t CNAV_WN_LSF_LSB = 1`
- `constexpr int32_t CNAV_DN_LSB = 1`
- `constexpr int32_t CNAV_DELTA_TLSF_LSB = 1`

11.173.1 Detailed Description

Defines parameters for GPS CNAV.

Author

Antonio Ramos, 2017. antonio.ramos(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.174 gps_cnav_ephemeris.h File Reference

Interface of a GPS CNAV EPHEMERIS storage.

```
#include <boost/serialization/nvp.hpp>
#include <cstdint>
```

Classes

- class [Gps_CNAV_Ephemeris](#)

This class is a storage and orbital model functions for the GPS SV ephemeris data as described in IS-GPS-200K.

11.174.1 Detailed Description

Interface of a GPS CNAV EPHEMERIS storage.

Author

Javier Arribas, 2015. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.175 gps_cnav_iono.h File Reference

Interface of a GPS CNAV IONOSPHERIC MODEL storage.

```
#include <boost/serialization/nvp.hpp>
```

Classes

- class [Gps_CNAV_Iono](#)

This class is a storage for the GPS IONOSPHERIC data as described in IS-GPS-200K.

11.175.1 Detailed Description

Interface of a GPS CNAV IONOSPHERIC MODEL storage.

Author

Javier Arribas, 2015. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.176 gps_cnav_navigation_message.h File Reference

Interface of a GPS CNAV Data message decoder.

```
#include "GPS_CNAV.h"
#include "gps_cnav_ephemeris.h"
#include "gps_cnav_iono.h"
#include "gps_cnav_utc_model.h"
#include <bitset>
#include <cstdint>
#include <map>
#include <string>
#include <utility>
#include <vector>
```

Classes

- class [Gps_CNAV_Navigation_Message](#)

This class decodes a GPS CNAV Data message as described in IS-GPS-200K.

11.176.1 Detailed Description

Interface of a GPS CNAV Data message decoder.

Author

Javier Arribas, 2015. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.177 gps_cnav_utc_model.h File Reference

Interface of a GPS CNAV UTC MODEL storage.

```
#include <boost/serialization/nvp.hpp>
#include <stdint>
```

Classes

- class [Gps_CNAV_Utc_Model](#)

This class is a storage for the GPS UTC MODEL data as described in in IS-GPS-200K.

11.177.1 Detailed Description

Interface of a GPS CNAV UTC MODEL storage.

Author

Javier Arribas, 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.178 `gps_ephemeris.h` File Reference

Interface of a GPS EPHEMERIS storage.

```
#include <boost/serialization/nvp.hpp>
#include <stdint>
#include <map>
#include <string>
```

Classes

- class [Gps_Ephemeris](#)

This class is a storage and orbital model functions for the GPS SV ephemeris data as described in IS-GPS-200K.

11.178.1 Detailed Description

Interface of a GPS EPHEMERIS storage.

Author

Javier Arribas, 2013. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.179 `gps_iono.h` File Reference

Interface of a GPS IONOSPHERIC MODEL storage.

```
#include <boost/serialization/nvp.hpp>
```

Classes

- class [Gps_Iono](#)

This class is a storage for the GPS IONOSPHERIC data as described in IS-GPS-200K.

11.179.1 Detailed Description

Interface of a GPS IONOSPHERIC MODEL storage.

Author

Javier Arribas, 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.180 GPS_L1_CA.h File Reference

Defines system parameters for GPS L1 C/A signal and NAV data.

```
#include "MATH_CONSTANTS.h"
#include "gnss_frequencies.h"
#include <cstdint>
#include <utility>
#include <vector>
```

Functions

- const std::vector< std::pair< int32_t, int32_t > > **TOW** ({{31, 17}})
- const std::vector< std::pair< int32_t, int32_t > > **INTEGRITY_STATUS_FLAG** ({{23, 1}})
- const std::vector< std::pair< int32_t, int32_t > > **ALERT_FLAG** ({{48, 1}})
- const std::vector< std::pair< int32_t, int32_t > > **ANTI_SPOOFING_FLAG** ({{49, 1}})
- const std::vector< std::pair< int32_t, int32_t > > **SUBFRAME_ID** ({{50, 3}})
- const std::vector< std::pair< int32_t, int32_t > > **GPS_WEEK** ({{61, 10}})
- const std::vector< std::pair< int32_t, int32_t > > **CA_OR_P_ON_L2** ({{71, 2}})
- const std::vector< std::pair< int32_t, int32_t > > **SV_ACCURACY** ({{73, 4}})
- const std::vector< std::pair< int32_t, int32_t > > **SV_HEALTH** ({{77, 6}})
- const std::vector< std::pair< int32_t, int32_t > > **L2_P_DATA_FLAG** ({{91, 1}})
- const std::vector< std::pair< int32_t, int32_t > > **T_GD** ({{197, 8}})
- const std::vector< std::pair< int32_t, int32_t > > **IODC** ({{83, 2}, {211, 8}})
- const std::vector< std::pair< int32_t, int32_t > > **T_OC** ({{219, 16}})
- const std::vector< std::pair< int32_t, int32_t > > **A_F2** ({{241, 8}})
- const std::vector< std::pair< int32_t, int32_t > > **A_F1** ({{249, 16}})
- const std::vector< std::pair< int32_t, int32_t > > **A_F0** ({{271, 22}})
- const std::vector< std::pair< int32_t, int32_t > > **IODE_SF2** ({{61, 8}})
- const std::vector< std::pair< int32_t, int32_t > > **C_RS** ({{69, 16}})
- const std::vector< std::pair< int32_t, int32_t > > **DELTA_N** ({{91, 16}})
- const std::vector< std::pair< int32_t, int32_t > > **M_0** ({{107, 8}, {121, 24}})
- const std::vector< std::pair< int32_t, int32_t > > **C_UC** ({{151, 16}})
- const std::vector< std::pair< int32_t, int32_t > > **ECCENTRICITY** ({{167, 8}, {181, 24}})
- const std::vector< std::pair< int32_t, int32_t > > **C_US** ({{211, 16}})

- `const std::vector< std::pair< int32_t, int32_t > > SQRT_A ({227, 8}, {241, 24})`
- `const std::vector< std::pair< int32_t, int32_t > > T_OE ({271, 16})`
- `const std::vector< std::pair< int32_t, int32_t > > FIT_INTERVAL_FLAG ({271, 1})`
- `const std::vector< std::pair< int32_t, int32_t > > AODO ({272, 5})`
- `const std::vector< std::pair< int32_t, int32_t > > C_IC ({61, 16})`
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA_0 ({77, 8}, {91, 24})`
- `const std::vector< std::pair< int32_t, int32_t > > C_IS ({121, 16})`
- `const std::vector< std::pair< int32_t, int32_t > > I_0 ({137, 8}, {151, 24})`
- `const std::vector< std::pair< int32_t, int32_t > > C_RC ({181, 16})`
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA ({197, 8}, {211, 24})`
- `const std::vector< std::pair< int32_t, int32_t > > OMEGA_DOT ({241, 24})`
- `const std::vector< std::pair< int32_t, int32_t > > IODE_SF3 ({271, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > I_DOT ({279, 14})`
- `const std::vector< std::pair< int32_t, int32_t > > SV_DATA_ID ({61, 2})`
- `const std::vector< std::pair< int32_t, int32_t > > SV_PAGE ({63, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > ALPHA_0 ({69, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > ALPHA_1 ({77, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > ALPHA_2 ({91, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > ALPHA_3 ({99, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > BETA_0 ({107, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > BETA_1 ({121, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > BETA_2 ({129, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > BETA_3 ({137, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > A_1 ({151, 24})`
- `const std::vector< std::pair< int32_t, int32_t > > A_0 ({181, 24}, {211, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > T_OT ({219, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > WN_T ({227, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > DELTAT_LS ({241, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > WN_LSF ({249, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > DN ({257, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > DELTAT_LSF ({271, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV25 ({229, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV26 ({241, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV27 ({247, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV28 ({253, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV29 ({259, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV30 ({271, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV31 ({277, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV32 ({283, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > T_OA ({69, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > WN_A ({77, 8})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV1 ({91, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV2 ({97, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV3 ({103, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV4 ({109, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV5 ({121, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV6 ({127, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV7 ({133, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV8 ({139, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV9 ({151, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV10 ({157, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV11 ({163, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV12 ({169, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV13 ({181, 6})`
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV14 ({187, 6})`

- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV15` (`{{193, 6}}`)
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV16` (`{{199, 6}}`)
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV17` (`{{211, 6}}`)
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV18` (`{{217, 6}}`)
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV19` (`{{223, 6}}`)
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV20` (`{{229, 6}}`)
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV21` (`{{241, 6}}`)
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV22` (`{{247, 6}}`)
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV23` (`{{253, 6}}`)
- `const std::vector< std::pair< int32_t, int32_t > > HEALTH_SV24` (`{{259, 6}}`)

Variables

- `constexpr double GPS_L1_FREQ_HZ = FREQ1`
L1 [Hz].
- `constexpr double GPS_L1_CA_CODE_RATE_CPS = 1.023e6`
GPS L1 C/A code rate [chips/s].
- `constexpr double GPS_L1_CA_CODE_LENGTH_CHIPS = 1023.0`
GPS L1 C/A code length [chips].
- `constexpr double GPS_L1_CA_CODE_PERIOD_S = 0.001`
GPS L1 C/A code period [seconds].
- `constexpr double GPS_L1_CA_CHIP_PERIOD_S = 9.7752e-07`
GPS L1 C/A chip period [seconds].
- `constexpr uint32_t GPS_L1_CA_CODE_PERIOD_MS = 1U`
GPS L1 C/A code period [ms].
- `constexpr uint32_t GPS_L1_CA_BIT_PERIOD_MS = 20U`
GPS L1 C/A bit period [ms].
- `constexpr double MAX_TOA_DELAY_MS = 20.0`
Maximum Time-Of-Arrival (TOA) difference between satellites for a receiver operated on Earth surface is 20 ms.
- `constexpr uint32_t GPS_L1_CA_OPT_ACQ_FS_SPS = 2000000`
Sampling frequency that maximizes the acquisition SNR while using a non-multiple of chip rate.
- `constexpr int32_t GPS_L1_CA_HISTORY_DEEP = 100`
- `constexpr double GPS_CA_PREAMBLE_DURATION_S = 0.160`
- `constexpr int32_t GPS_CA_PREAMBLE_LENGTH_BITS = 8`
- `constexpr int32_t GPS_CA_PREAMBLE_LENGTH_SYMBOLS = 160`
- `constexpr int32_t GPS_CA_PREAMBLE_DURATION_MS = 160`
- `constexpr int32_t GPS_CA_TELEMETRY_RATE_BITS_SECOND = 50`
NAV message bit rate [bits/s].
- `constexpr int32_t GPS_CA_TELEMETRY_SYMBOLS_PER_BIT = 20`
- `constexpr int32_t GPS_CA_TELEMETRY_RATE_SYMBOLS_SECOND = GPS_CA_TELEMETRY_RATE_BITS_SECOND * GPS_CA_TELEMETRY_SYMBOLS_PER_BIT`
NAV message bit rate [symbols/s].
- `constexpr int32_t GPS_WORD_LENGTH = 4`
CRC + GPS WORD (-2 -1 0 ... 29) Bits = 4 bytes.
- `constexpr int32_t GPS_SUBFRAME_LENGTH = 40`
GPS_WORD_LENGTH x 10 = 40 bytes.
- `constexpr int32_t GPS_SUBFRAME_BITS = 300`
Number of bits per subframe in the NAV message [bits].
- `constexpr int32_t GPS_SUBFRAME_SECONDS = 6`
Subframe duration [seconds].
- `constexpr int32_t GPS_SUBFRAME_MS = 6000`

[illegible]

Defines system parameters for GPS L1 C/A signal and NAV data.

Javier Arribas, 2011. jarribas(at)cttc.es

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.180.2 Function Documentation

11.180.2.1 ALPHA_0()

```
const std::vector<std::pair<int32_t, int32_t> > ALPHA_0 (
    {{69, 8}} )
```

Todo read all pages of subframe 4

11.180.2.2 T_OA()

```
const std::vector<std::pair<int32_t, int32_t> > T_OA (
    {{69, 8}} )
```

Todo read all pages of subframe 5

11.180.3 Variable Documentation

11.180.3.1 GPS_CA_TELEMETRY_RATE_BITS_SECOND

```
constexpr int32_t GPS_CA_TELEMETRY_RATE_BITS_SECOND = 50
```

NAV message bit rate [bits/s].

Definition at line 61 of file GPS_L1_CA.h.

11.180.3.2 GPS_CA_TELEMETRY_RATE_SYMBOLS_SECOND

```
constexpr int32_t GPS_CA_TELEMETRY_RATE_SYMBOLS_SECOND = GPS_CA_TELEMETRY_RATE_BITS_SECOND *
GPS_CA_TELEMETRY_SYMBOLS_PER_BIT
```

NAV message bit rate [symbols/s].

Definition at line 63 of file GPS_L1_CA.h.

11.180.3.3 GPS_L1_CA_BIT_PERIOD_MS

```
constexpr uint32_t GPS_L1_CA_BIT_PERIOD_MS = 20U
```

GPS L1 C/A bit period [ms].

Definition at line 38 of file GPS_L1_CA.h.

11.180.3.4 GPS_L1_CA_CHIP_PERIOD_S

```
constexpr double GPS_L1_CA_CHIP_PERIOD_S = 9.7752e-07
```

GPS L1 C/A chip period [seconds].

Definition at line 36 of file GPS_L1_CA.h.

11.180.3.5 GPS_L1_CA_CODE_LENGTH_CHIPS

```
constexpr double GPS_L1_CA_CODE_LENGTH_CHIPS = 1023.0
```

GPS L1 C/A code length [chips].

Definition at line 34 of file GPS_L1_CA.h.

11.180.3.6 GPS_L1_CA_CODE_PERIOD_MS

```
constexpr uint32_t GPS_L1_CA_CODE_PERIOD_MS = 1U
```

GPS L1 C/A code period [ms].

Definition at line 37 of file GPS_L1_CA.h.

11.180.3.7 GPS_L1_CA_CODE_PERIOD_S

```
constexpr double GPS_L1_CA_CODE_PERIOD_S = 0.001
```

GPS L1 C/A code period [seconds].

Definition at line 35 of file GPS_L1_CA.h.

11.180.3.8 GPS_L1_CA_CODE_RATE_CPS

```
constexpr double GPS_L1_CA_CODE_RATE_CPS = 1.023e6
```

GPS L1 C/A code rate [chips/s].

Definition at line 33 of file GPS_L1_CA.h.

11.180.3.9 GPS_L1_CA_OPT_ACQ_FS_SPS

```
constexpr uint32_t GPS_L1_CA_OPT_ACQ_FS_SPS = 2000000
```

Sampling frequency that maximizes the acquisition SNR while using a non-multiple of chip rate.

Definition at line 51 of file GPS_L1_CA.h.

11.180.3.10 GPS_L1_FREQ_HZ

```
constexpr double GPS_L1_FREQ_HZ = FREQ1
```

L1 [Hz].

Definition at line 32 of file GPS_L1_CA.h.

11.180.3.11 GPS_SUBFRAME_BITS

```
constexpr int32_t GPS_SUBFRAME_BITS = 300
```

Number of bits per subframe in the NAV message [bits].

Definition at line 66 of file GPS_L1_CA.h.

11.180.3.12 GPS_SUBFRAME_LENGTH

```
constexpr int32_t GPS_SUBFRAME_LENGTH = 40
```

GPS_WORD_LENGTH x 10 = 40 bytes.

Definition at line 65 of file GPS_L1_CA.h.

11.180.3.13 GPS_SUBFRAME_MS

```
constexpr int32_t GPS_SUBFRAME_MS = 6000
```

Subframe duration [seconds].

Definition at line 68 of file GPS_L1_CA.h.

11.180.3.14 GPS_SUBFRAME_SECONDS

```
constexpr int32_t GPS_SUBFRAME_SECONDS = 6
```

Subframe duration [seconds].

Definition at line 67 of file GPS_L1_CA.h.

11.180.3.15 GPS_WORD_BITS

```
constexpr int32_t GPS_WORD_BITS = 30
```

Number of bits per word in the NAV message [bits].

Definition at line 69 of file GPS_L1_CA.h.

11.180.3.16 GPS_WORD_LENGTH

```
constexpr int32_t GPS_WORD_LENGTH = 4
```

CRC + GPS WORD (-2 -1 0 ... 29) Bits = 4 bytes.

Definition at line 64 of file GPS_L1_CA.h.

11.180.3.17 MAX_TOA_DELAY_MS

```
constexpr double MAX_TOA_DELAY_MS = 20.0
```

Maximum Time-Of-Arrival (TOA) difference between satellites for a receiver operated on Earth surface is 20 ms.

According to the GPS orbit model described in [1] Pag. 32. It should be taken into account to set the buffer size for the PRN start timestamp in the pseudoranges block. [1] J. Bao-Yen Tsui, Fundamentals of Global Positioning System Receivers. A Software Approach, John Wiley & Sons, Inc., Hoboken, NJ, 2nd edition, 2005.

Definition at line 48 of file GPS_L1_CA.h.

11.181 `gps_l1_ca_dll_pll_tracking.h` File Reference

Interface of an adapter of a DLL+PLL tracking loop block for GPS L1 C/A to a [TrackingInterface](#).

```
#include "dll_pll_veml_tracking.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GpsL1CaDllPllTracking](#)
This class implements a code DLL + carrier PLL tracking loop.

11.181.1 Detailed Description

Interface of an adapter of a DLL+PLL tracking loop block for GPS L1 C/A to a [TrackingInterface](#).

Author

Carlos Aviles, 2010. carlos.avilesr(at)gmail.com Javier Arribas, 2011. jarribas(at)cttc.es

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.182 `gps_l1_ca_dll_pll_tracking_fpga.h` File Reference

Interface of an adapter of a DLL+PLL tracking loop block for GPS L1 C/A to a [TrackingInterface](#) for the FPGA.

```
#include "dll_pll_veml_tracking_fpga.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GpsL1CaDllPllTrackingFpga](#)
This class implements a code DLL + carrier PLL tracking loop.

11.182.1 Detailed Description

Interface of an adapter of a DLL+PLL tracking loop block for GPS L1 C/A to a [TrackingInterface](#) for the FPGA.

Author

Marc Majoral, 2019, mmajoral(at)cttc.es

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.183 gps_l1_ca_dll_pll_tracking_gpu.h File Reference

Implementation of an adapter of a DLL+PLL tracking loop block using GPU accelerated functions for GPS L1 C/A to a [TrackingInterface](#).

```
#include "gps_l1_ca_dll_pll_tracking_gpu_cc.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GpsL1CaDllPllTrackingGPU](#)

This class implements a code DLL + carrier PLL tracking loop using GPU accelerated functions.

11.183.1 Detailed Description

Implementation of an adapter of a DLL+PLL tracking loop block using GPU accelerated functions for GPS L1 C/A to a [TrackingInterface](#).

Author

Javier Arribas, 2015. jarribas(at)cttc.es

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.184 gps_l1_ca_dll_pll_tracking_gpu_cc.h File Reference

Implementation of a code DLL + carrier PLL tracking block, GPU ACCELERATED.

```
#include "cuda_multicorrelator.h"
#include "gnss_synchro.h"
#include "tracking_2nd_DLL_filter.h"
#include "tracking_FLL_PLL_filter.h"
#include <gnuradio/block.h>
#include <fstream>
#include <map>
#include <string>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [Gps_L1_Ca_DLL_Pll_Tracking_GPU_cc](#)
This class implements a DLL + PLL tracking loop block.

Typedefs

- typedef boost::shared_ptr< [Gps_L1_Ca_DLL_Pll_Tracking_GPU_cc](#) > [gps_l1_ca_dll_pll_tracking_gpu_cc_sptr](#)

Functions

- [gps_l1_ca_dll_pll_tracking_gpu_cc_sptr](#) [gps_l1_ca_dll_pll_make_tracking_gpu_cc](#) (int64_t fs, in, uint32_t vector_length, bool dump, std::string dump_filename, float pll_bw_hz, float dll_bw_hz, float early_late_space_chips)

11.184.1 Detailed Description

Implementation of a code DLL + carrier PLL tracking block, GPU ACCELERATED.

Author

Javier Arribas, 2015. jarribas(at)cttc.es

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.185 gps_l1_ca_kf_tracking.h File Reference

Interface of an adapter of a DLL + Kalman carrier tracking loop block for GPS L1 C/A signals.

```
#include "gps_l1_ca_kf_tracking_cc.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GpsL1CaKfTracking](#)
This class implements a code DLL + carrier PLL tracking loop.

11.185.1 Detailed Description

Interface of an adapter of a DLL + Kalman carrier tracking loop block for GPS L1 C/A signals.

Author

Javier Arribas, 2018. jarribas(at)cttc.es
Jordi Vila-Valls 2018. jvila(at)cttc.es
Carles Fernandez-Prades 2018. cfernandez(at)cttc.es

Reference: J. Vila-Valls, P. Closas, M. Navarro and C. Fernandez-Prades, "Are PLLs Dead? A Tutorial on Kalman Filter-based Techniques for Digital Carrier Synchronization", IEEE Aerospace and Electronic Systems Magazine, Vol. 32, No. 7, pp. 28–45, July 2017. DOI: 10.1109/MAES.2017.150260

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.186 gps_l1_ca_kf_tracking_cc.h File Reference

Interface of a processing block of a DLL + Kalman carrier tracking loop for GPS L1 C/A signals.

```
#include "bayesian_estimation.h"
#include "cpu_multicorrelator_real_codes.h"
#include "gnss_synchro.h"
#include "tracking_2nd_DLL_filter.h"
#include "tracking_2nd_PLL_filter.h"
#include <armadillo>
#include <gnuradio/block.h>
#include <volk_gnssdr/volk_gnssdr_alloc.h>
#include <fstream>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [Gps_L1_Ca_Kf_Tracking_cc](#)
This class implements a DLL + PLL tracking loop block.

Typedefs

- using **gps_l1_ca_kf_tracking_cc_sptr** = boost::shared_ptr< [Gps_L1_Ca_Kf_Tracking_cc](#) >

Functions

- `gps_l1_ca_kf_tracking_cc_sptr gps_l1_ca_kf_make_tracking_cc (uint32_t order, int64_t if_freq, int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float dll_bw_hz, float early_late_space_chips, bool bce_run, uint32_t bce_ptrans, uint32_t bce_strans, int32_t bce_nu, int32_t bce_kappa)`

11.186.1 Detailed Description

Interface of a processing block of a DLL + Kalman carrier tracking loop for GPS L1 C/A signals.

Author

Javier Arribas, 2018. jarribas(at)cttc.es
Jordi Vila-Valls 2018. jvila(at)cttc.es
Carles Fernandez-Prades 2018. cfernandez(at)cttc.es

Reference: J. Vila-Valls, P. Closas, M. Navarro and C. Fernandez-Prades, "Are PLLs Dead? A Tutorial on Kalman Filter-based Techniques for Digital Carrier Synchronization", IEEE Aerospace and Electronic Systems Magazine, Vol. 32, No. 7, pp. 28–45, July 2017. DOI: 10.1109/MAES.2017.150260

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.187 `gps_l1_ca_pcps_acquisition.h` File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

```
#include "acq_conf.h"
#include "channel_fsm.h"
#include "complex_byte_to_float_x2.h"
#include "gnss_synchro.h"
#include "pcps_acquisition.h"
#include <gnuradio/blocks/float_to_complex.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GpsL1CaPcpsAcquisition](#)

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

11.187.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Authors

- Javier Arribas, 2011. jarribas(at)cttc.es
- Luis Esteve, 2012. luis(at)epsilon-formacion.com
- Marc Molina, 2013. marc.molina.pena(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.188 gps_l1_ca_pcps_acquisition_fine_doppler.h File Reference

Adapts a PCPS acquisition block with fine Doppler estimation to an [AcquisitionInterface](#) for GPS L1 C/A signals.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include "pcps_acquisition_fine_doppler_cc.h"
#include <memory>
#include <string>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [GpsL1CaPcpsAcquisitionFineDoppler](#)

This class Adapts a PCPS acquisition block with fine Doppler estimation to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Typedefs

- using `pcps_acquisition_fine_doppler_cc_sptr` = boost::shared_ptr< [pcps_acquisition_fine_doppler_cc](#) >

11.188.1 Detailed Description

Adapts a PCPS acquisition block with fine Doppler estimation to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Authors

- Javier Arribas, 2013. jarribas(at)cttc.es

*

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.189 gps_l1_ca_pcps_acquisition_fpga.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals for the FPGA.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include "pcps_acquisition_fpga.h"
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GpsL1CaPcpsAcquisitionFpga](#)

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for GPS L1 C/A signals.

11.189.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals for the FPGA.

Authors

- Marc Majoral, 2019. mmajoral(at)cttc.es
- Javier Arribas, 2019. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.190 gps_l1_ca_pcps_assisted_acquisition.h File Reference

Adapts a PCPS Assisted acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include "pcps_assisted_acquisition_cc.h"
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GpsL1CaPcpsAssistedAcquisition](#)

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

11.190.1 Detailed Description

Adapts a PCPS Assisted acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Authors

- Javier Arribas, 2011. jarribas(at)cttc.es

*

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.191 gps_l1_ca_pcps_openc1_acquisition.h File Reference

Adapts an OpenCL PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include "pcps_openc1_acquisition_cc.h"
#include <gnuradio/blocks/stream_to_vector.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GpsL1CaPcpsOpenCIAcquisition](#)

This class adapts an OpenCL PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

11.191.1 Detailed Description

Adapts an OpenCL PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Author

Marc Molina, 2013. marc.molina.pena(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.192 gps_l1_ca_pcps_quicksync_acquisition.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals implementing the QuickSync Algorithm.

```
#include "channel_fsm.h"
#include "configuration_interface.h"
#include "gnss_synchro.h"
#include "pcps_quicksync_acquisition_cc.h"
#include <gnuradio/blocks/stream_to_vector.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GpsL1CaPcpsQuickSyncAcquisition](#)

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

11.192.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals implementing the QuickSync Algorithm.

Date

June, 2014

Author

Damian Miralles Sanchez. dmiralles2009@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.193 gps_l1_ca_pcps_tong_acquisition.h File Reference

Adapts a PCPS Tong acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

```
#include "channel_fsm.h"
#include "configuration_interface.h"
#include "gnss_synchro.h"
#include "pcps_tong_acquisition_cc.h"
#include <gnuradio/blocks/stream_to_vector.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GpsL1CaPcpsTongAcquisition](#)

This class adapts a PCPS Tong acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

11.193.1 Detailed Description

Adapts a PCPS Tong acquisition block to an [AcquisitionInterface](#) for GPS L1 C/A signals.

Author

Marc Molina, 2013. marc.molina.pena(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.194 gps_l1_ca_tcp_connector_tracking.h File Reference

Interface of an adapter of a TCP connector block based on code DLL + carrier PLL for GPS L1 C/A to a [TrackingInterface](#).

```
#include "gps_l1_ca_tcp_connector_tracking_cc.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GpsL1CaTcpConnectorTracking](#)

This class implements a code DLL + carrier PLL tracking loop.

11.194.1 Detailed Description

Interface of an adapter of a TCP connector block based on code DLL + carrier PLL for GPS L1 C/A to a [TrackingInterface](#).

Author

David Pubill, 2012. dpubill(at)cttc.es Javier Arribas, 2011. jarribas(at)cttc.es

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.195 `gps_l1_ca_tcp_connector_tracking_cc.h` File Reference

Interface of a TCP connector block based on code DLL + carrier PLL.

```
#include "cpu_multicorrelator.h"
#include "gnss_synchro.h"
#include "tcp_communication.h"
#include <gnuradio/block.h>
#include <volk_gnssssdr/volk_gnssssdr_alloc.h>
#include <fstream>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [Gps_L1_Ca_Tcp_Connector_Tracking_cc](#)
This class implements a DLL + PLL tracking loop block.

Typedefs

- using `gps_l1_ca_tcp_connector_tracking_cc_sptr` = boost::shared_ptr< [Gps_L1_Ca_Tcp_Connector_Tracking_cc](#) >

Functions

- `gps_l1_ca_tcp_connector_tracking_cc_sptr` **gps_l1_ca_tcp_connector_make_tracking_cc** (int64_t fs_in, uint32_t vector_length, bool dump, const std::string &dump_filename, float early_late_space_chips, size_t port_ch0)

11.195.1 Detailed Description

Interface of a TCP connector block based on code DLL + carrier PLL.

Author

David Pubill, 2012. dpubill(at)cttc.es Javier Arribas, 2011. jarribas(at)cttc.es

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.196 gps_l1_ca_telemetry_decoder.h File Reference

Interface of an adapter of a GPS L1 C/A NAV data decoder block to a [TelemetryDecoderInterface](#).

```
#include "gnss_satellite.h"
#include "gnss_synchro.h"
#include "gps_l1_ca_telemetry_decoder_gs.h"
#include "telemetry_decoder_interface.h"
#include <gnuradio/runtime_types.h>
#include <cstdio>
#include <string>
```

Classes

- class [GpsL1CaTelemetryDecoder](#)

This class implements a NAV data decoder for GPS L1 C/A.

11.196.1 Detailed Description

Interface of an adapter of a GPS L1 C/A NAV data decoder block to a [TelemetryDecoderInterface](#).

Author

Javier Arribas, 2011. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.197 `gps_l1_ca_telemetry_decoder_gs.h` File Reference

Interface of a NAV message demodulator block based on Kay Borre book MATLAB-based GPS receiver.

```
#include "GPS_L1_CA.h"
#include "gnss_satellite.h"
#include "gnss_synchro.h"
#include "gps_navigation_message.h"
#include <boost/circular_buffer.hpp>
#include <gnuradio/block.h>
#include <gnuradio/types.h>
#include <array>
#include <cstdint>
#include <fstream>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [gps_l1_ca_telemetry_decoder_gs](#)

This class implements a block that decodes the NAV data defined in IS-GPS-200K.

Typedefs

- using `gps_l1_ca_telemetry_decoder_gs_sptr` = `boost::shared_ptr< gps_l1_ca_telemetry_decoder_gs >`

Functions

- `gps_l1_ca_telemetry_decoder_gs_sptr gps_l1_ca_make_telemetry_decoder_gs (const Gnss_Satellite &satellite, bool dump)`

11.197.1 Detailed Description

Interface of a NAV message demodulator block based on Kay Borre book MATLAB-based GPS receiver.

Javier Arribas, 2011. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.198 gps_l2_m_dll_pll_tracking.h File Reference

Interface of an adapter of a DLL+PLL tracking loop block for GPS L1 C/A to a [TrackingInterface](#).

```
#include "dll_pll_veml_tracking.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GpsL2MDIIPITracking](#)
This class implements a code DLL + carrier PLL tracking loop.

11.198.1 Detailed Description

Interface of an adapter of a DLL+PLL tracking loop block for GPS L1 C/A to a [TrackingInterface](#).

Author

Carlos Aviles, 2010. carlos.avilesr(at)gmail.com Javier Arribas, 2011. jarribas(at)cttc.es

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.199 gps_l2_m_dll_pll_tracking_fpga.h File Reference

Interface of an adapter of a DLL+PLL tracking loop block for GPS L2C to a [TrackingInterface](#) for the FPGA.

```
#include "dll_pll_veml_tracking_fpga.h"
#include "tracking_interface.h"
#include <gnuradio/runtime_types.h>
#include <cstdint>
#include <string>
```

Classes

- class [GpsL2MDIIPITrackingFpga](#)
This class implements a code DLL + carrier PLL tracking loop.

11.199.1 Detailed Description

Interface of an adapter of a DLL+PLL tracking loop block for GPS L2C to a [TrackingInterface](#) for the FPGA.

Author

Marc Majoral, 2019, mmajoral(at)cttc.es

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.200 gps_l2_m_pcps_acquisition.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L2 M signals.

```
#include "channel_fsm.h"
#include "complex_byte_to_float_x2.h"
#include "gnss_synchro.h"
#include "pcps_acquisition.h"
#include <gnuradio/blocks/float_to_complex.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GpsL2MPcpsAcquisition](#)

This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L2 M signals.

11.200.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L2 M signals.

Authors

- Javier Arribas, 2015. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.201 gps_l2_m_pcps_acquisition_fpga.h File Reference

Adapts an FPGA-offloaded PCPS acquisition block to an [AcquisitionInterface](#) for GPS L2 M signals.

```
#include "channel_fsm.h"
#include "pcps_acquisition_fpga.h"
#include <gnuradio/runtime_types.h>
#include <cstdint>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GpsL2MPcpsAcquisitionFpga](#)

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for GPS L2 M signals.

11.201.1 Detailed Description

Adapts an FPGA-offloaded PCPS acquisition block to an [AcquisitionInterface](#) for GPS L2 M signals.

Authors

- Javier Arribas, 2019. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.202 GPS_L2C.h File Reference

Defines system parameters for GPS L2C signal.

```
#include "GPS_CNAV.h"
#include "gnss_frequencies.h"
#include <cstdint>
#include <utility>
#include <vector>
```

Variables

- constexpr double `GPS_L2_FREQ_HZ` = `FREQ2`
L2 [Hz].
- constexpr double `GPS_L2_L_PERIOD_S` = 1.5
GPS L2 L code period [seconds].
- constexpr double `GPS_L2_M_CODE_RATE_CPS` = 0.5115e6
GPS L2 M code rate [chips/s].
- constexpr double `GPS_L2_M_PERIOD_S` = 0.02
GPS L2 M code period [seconds].
- constexpr double `GPS_L2_L_CODE_RATE_CPS` = 0.5115e6
GPS L2 L code rate [chips/s].
- constexpr int32_t `GPS_L2_M_CODE_LENGTH_CHIPS` = 10230
GPS L2 M code length [chips].
- constexpr int32_t `GPS_L2_L_CODE_LENGTH_CHIPS` = 767250
GPS L2 L code length [chips].
- constexpr int32_t `GPS_L2_CNAV_DATA_PAGE_BITS` = 300
GPS L2 CNAV page length, including preamble and CRC [bits].
- constexpr int32_t `GPS_L2_SYMBOLS_PER_BIT` = 2
- constexpr int32_t `GPS_L2_SAMPLES_PER_SYMBOL` = 1
- constexpr int32_t `GPS_L2_CNAV_DATA_PAGE_SYMBOLS` = 600
- constexpr int32_t `GPS_L2_CNAV_DATA_PAGE_DURATION_S` = 12
- constexpr int32_t `GPS_L2C_HISTORY_DEEP` = 5
- constexpr uint32_t `GPS_L2C_OPT_ACQ_FS_SPS` = 2000000
Sampling frequency that maximizes the acquisition SNR while using a non-multiple of chip rate.
- constexpr int32_t `GPS_L2C_M_INIT_REG` [115]

11.202.1 Detailed Description

Defines system parameters for GPS L2C signal.

Author

Javier Arribas, 2015. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.202.2 Variable Documentation

11.202.2.1 GPS_L2_CNAV_DATA_PAGE_BITS

```
constexpr int32_t GPS_L2_CNAV_DATA_PAGE_BITS = 300
```

GPS L2 CNAV page length, including preamble and CRC [bits].

Definition at line 40 of file GPS_L2C.h.

11.202.2.2 GPS_L2_FREQ_HZ

```
constexpr double GPS_L2_FREQ_HZ = FREQ2
```

L2 [Hz].

Definition at line 33 of file GPS_L2C.h.

11.202.2.3 GPS_L2_L_CODE_LENGTH_CHIPS

```
constexpr int32_t GPS_L2_L_CODE_LENGTH_CHIPS = 767250
```

GPS L2 L code length [chips].

Definition at line 39 of file GPS_L2C.h.

11.202.2.4 GPS_L2_L_CODE_RATE_CPS

```
constexpr double GPS_L2_L_CODE_RATE_CPS = 0.5115e6
```

GPS L2 L code rate [chips/s].

Definition at line 37 of file GPS_L2C.h.

11.202.2.5 GPS_L2_L_PERIOD_S

```
constexpr double GPS_L2_L_PERIOD_S = 1.5
```

GPS L2 L code period [seconds].

Definition at line 34 of file GPS_L2C.h.

11.202.2.6 GPS_L2_M_CODE_LENGTH_CHIPS

```
constexpr int32_t GPS_L2_M_CODE_LENGTH_CHIPS = 10230
```

GPS L2 M code length [chips].

Definition at line 38 of file GPS_L2C.h.

11.202.2.7 GPS_L2_M_CODE_RATE_CPS

```
constexpr double GPS_L2_M_CODE_RATE_CPS = 0.5115e6
```

GPS L2 M code rate [chips/s].

Definition at line 35 of file GPS_L2C.h.

11.202.2.8 GPS_L2_M_PERIOD_S

```
constexpr double GPS_L2_M_PERIOD_S = 0.02
```

GPS L2 M code period [seconds].

Definition at line 36 of file GPS_L2C.h.

11.202.2.9 GPS_L2C_M_INIT_REG

```
constexpr int32_t GPS_L2C_M_INIT_REG[115]
```

Initial value:

```
=
{0742417664, 0756014035, 0002747144, 0066265724,
 0601403471, 0703232733, 0124510070, 0617316361,
 0047541621, 0733031046, 0713512145, 0024437606,
 0021264003, 0230655351, 0001314400, 0222021506,
 0540264026, 0205521705, 0064022144, 0120161274,
 0044023533, 0724744327, 0045743577, 0741201660,
 0700274134, 0010247261, 0713433445, 0737324162,
 0311627434, 0710452007, 0722462133, 0050172213,
 0500653703, 0755077436, 0136717361, 0756675453,
 0435506112, 0771353753, 0226107701, 0022025110,
 0402466344, 0752566114, 0702011164, 0041216771,
 0047457275, 0266333164, 0713167356, 0060546335,
 0355173035, 0617201036, 0157465571, 0767360553,
 0023127030, 0431343777, 0747317317, 0045706125,
 0002744276, 0060036467, 0217744147, 0603340174,
 0326616775, 0063240065, 0111460621,
 0604055104, 0157065232, 0013305707, 0603552017,
 0230461355, 0603653437, 0652346475, 0743107103,
 0401521277, 0167335110, 0014013575, 0362051132,
 0617753265, 0216363634, 0755561123, 0365304033,
 0625025543, 0054420334, 0415473671, 0662364360,
 0373446602, 0417564100, 0000526452, 0226631300,
 0113752074, 0706134401, 0041352546, 0664630154,
 0276524255, 0714720530, 0714051771, 0044526647,
 0207164322, 0262120161, 0204244652, 0202133131,
 0714351204, 0657127260, 0130567507, 0670517677,
 0607275514, 0045413633, 0212645405, 0613700455,
 0706202440, 0705056276, 0020373522, 0746013617,
 0132720621, 0434015513, 0566721727, 0140633660}
```

Definition at line 51 of file GPS_L2C.h.

11.202.2.10 GPS_L2C_OPT_ACQ_FS_SPS

```
constexpr uint32_t GPS_L2C_OPT_ACQ_FS_SPS = 2000000
```

Sampling frequency that maximizes the acquisition SNR while using a non-multiple of chip rate.

Definition at line 49 of file GPS_L2C.h.

11.203 gps_l2c_signal.h File Reference

This class implements signal generators for the GPS L2C signals.

```
#include <complex>
#include <cstdint>
#include <gsl/gsl>
```

Functions

- void [gps_l2c_m_code_gen_complex](#) (own::span< std::complex< float >> _dest, uint32_t _prn)
Generates complex GPS L2C M code for the desired SV ID.
- void [gps_l2c_m_code_gen_float](#) (own::span< float > _dest, uint32_t _prn)
- void [gps_l2c_m_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, uint32_t _prn, int32_t _fs)
Generates complex GPS L2C M code for the desired SV ID, and sampled to specific sampling frequency.

11.203.1 Detailed Description

This class implements signal generators for the GPS L2C signals.

Author

Javier Arribas, 2015. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.203.2 Function Documentation

11.203.2.1 `gps_l2c_m_code_gen_complex()`

```
void gps_l2c_m_code_gen_complex (
    own::span< std::complex< float >> _dest,
    uint32_t _prn )
```

Generates complex GPS L2C M code for the desired SV ID.

11.203.2.2 `gps_l2c_m_code_gen_complex_sampled()`

```
void gps_l2c_m_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    uint32_t _prn,
    int32_t _fs )
```

Generates complex GPS L2C M code for the desired SV ID, and sampled to specific sampling frequency.

11.204 `gps_l2c_telemetry_decoder.h` File Reference

Interface of an adapter of a GPS L2C (CNAV) data decoder block to a [TelemetryDecoderInterface](#).

```
#include "gnss_satellite.h"
#include "gnss_synchro.h"
#include "gps_l2c_telemetry_decoder_gs.h"
#include "telemetry_decoder_interface.h"
#include <gnuradio/runtime_types.h>
#include <cstdint>
#include <string>
```

Classes

- class [GpsL2CTelemetryDecoder](#)
This class implements a NAV data decoder for GPS L2 M.

11.204.1 Detailed Description

Interface of an adapter of a GPS L2C (CNAV) data decoder block to a [TelemetryDecoderInterface](#).

Author

Javier Arribas, 2015. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.205 `gps_l2c_telemetry_decoder_gs.h` File Reference

Interface of a CNAV message demodulator block.

```
#include "gnss_satellite.h"
#include "gps_cnav_navigation_message.h"
#include <gnuradio/block.h>
#include <gnuradio/types.h>
#include <stdint>
#include <fstream>
#include <string>
#include <boost/shared_ptr.hpp>
#include "cnav_msg.h"
```

Classes

- class `gps_l2c_telemetry_decoder_gs`
This class implements a block that decodes CNAV data defined in IS-GPS-200K.

Typedefs

- using `gps_l2c_telemetry_decoder_gs_sptr` = `boost::shared_ptr< gps_l2c_telemetry_decoder_gs >`

Functions

- `gps_l2c_telemetry_decoder_gs_sptr gps_l2c_make_telemetry_decoder_gs` (const `Gnss_Satellite` &satellite, bool dump)

11.205.1 Detailed Description

Interface of a CNAV message demodulator block.

Javier Arribas, 2015. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.206 `GPS_L5.h` File Reference

Defines system parameters for GPS L5 signal.

```
#include "GPS_CNAV.h"
#include "gnss_frequencies.h"
#include <stdint>
```

Variables

- constexpr double `GPS_L5_FREQ_HZ` = `FREQ5`
L5 [Hz].
- constexpr double `GPS_L5I_CODE_RATE_CPS` = `10.23e6`
GPS L5I code rate [chips/s].
- constexpr double `GPS_L5I_PERIOD_S` = `0.001`
GPS L5I code period [seconds].
- constexpr double `GPS_L5I_SYMBOL_PERIOD_S` = `0.01`
GPS L5I symbol period [seconds].
- constexpr double `GPS_L5Q_CODE_RATE_CPS` = `10.23e6`
GPS L5Q code rate [chips/s].
- constexpr double `GPS_L5Q_PERIOD_S` = `0.001`
GPS L5Q code period [seconds].
- constexpr int32_t `GPS_L5Q_CODE_LENGTH_CHIPS` = `10230`
GPS L5Q code length [chips].
- constexpr int32_t `GPS_L5I_CODE_LENGTH_CHIPS` = `10230`
GPS L5I code length [chips].
- constexpr int32_t `GPS_L5I_PERIOD_MS` = `1`
GPS L5I code period [ms].
- constexpr int32_t `GPS_L5I_SYMBOL_PERIOD_MS` = `10`
GPS L5I symbol period [ms].
- constexpr int32_t `GPS_L5_HISTORY_DEEP` = `5`
- constexpr uint32_t `GPS_L5_OPT_ACQ_FS_SPS` = `10000000`
Sampling frequency that maximizes the acquisition SNR while using a non-multiple of chip rate.
- constexpr int32_t `GPS_L5I_INIT_REG` [210]
- constexpr int32_t `GPS_L5Q_INIT_REG` [210]
- constexpr int32_t `GPS_L5_CNAV_DATA_PAGE_BITS` = `300`
GPS L5 CNAV page length, including preamble and CRC [bits].
- constexpr int32_t `GPS_L5_SYMBOLS_PER_BIT` = `2`
- constexpr int32_t `GPS_L5_SAMPLES_PER_SYMBOL` = `10`
- constexpr int32_t `GPS_L5_CNAV_DATA_PAGE_SYMBOLS` = `600`
- constexpr int32_t `GPS_L5_CNAV_DATA_PAGE_DURATION_S` = `6`
- constexpr int32_t `GPS_L5I_NH_CODE_LENGTH` = `10`
- constexpr int32_t `GPS_L5I_NH_CODE` [10] = {0, 0, 0, 0, 1, 1, 0, 1, 0, 1}
- constexpr int32_t `GPS_L5Q_NH_CODE_LENGTH` = `20`
- constexpr int32_t `GPS_L5Q_NH_CODE` [20] = {0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0}
- constexpr char `GPS_L5I_NH_CODE_STR` [11] = "0000110101"
- constexpr char `GPS_L5Q_NH_CODE_STR` [21] = "00000100110101001110"

11.206.1 Detailed Description

Defines system parameters for GPS L5 signal.

Author

Javier Arribas, 2017. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.206.2 Variable Documentation

11.206.2.1 GPS_L5_CNAV_DATA_PAGE_BITS

```
constexpr int32_t GPS_L5_CNAV_DATA_PAGE_BITS = 300
```

GPS L5 CNAV page length, including preamble and CRC [bits].

Definition at line 160 of file GPS_L5.h.

11.206.2.2 GPS_L5_FREQ_HZ

```
constexpr double GPS_L5_FREQ_HZ = FREQ5
```

L5 [Hz].

Definition at line 30 of file GPS_L5.h.

11.206.2.3 GPS_L5_OPT_ACQ_FS_SPS

```
constexpr uint32_t GPS_L5_OPT_ACQ_FS_SPS = 10000000
```

Sampling frequency that maximizes the acquisition SNR while using a non-multiple of chip rate.

Definition at line 44 of file GPS_L5.h.

11.206.2.4 GPS_L5I_CODE_LENGTH_CHIPS

```
constexpr int32_t GPS_L5I_CODE_LENGTH_CHIPS = 10230
```

GPS L5I code length [chips].

Definition at line 37 of file GPS_L5.h.

11.206.2.5 GPS_L5I_CODE_RATE_CPS

```
constexpr double GPS_L5I_CODE_RATE_CPS = 10.23e6
```

GPS L5I code rate [chips/s].

Definition at line 31 of file GPS_L5.h.

11.206.2.6 GPS_L5I_PERIOD_MS

```
constexpr int32_t GPS_L5I_PERIOD_MS = 1
```

GPS L5I code period [ms].

Definition at line 38 of file GPS_L5.h.

11.206.2.7 GPS_L5I_PERIOD_S

```
constexpr double GPS_L5I_PERIOD_S = 0.001
```

GPS L5I code period [seconds].

Definition at line 32 of file GPS_L5.h.

11.206.2.8 GPS_L5I_SYMBOL_PERIOD_MS

```
constexpr int32_t GPS_L5I_SYMBOL_PERIOD_MS = 10
```

GPS L5I symbol period [ms].

Definition at line 39 of file GPS_L5.h.

11.206.2.9 GPS_L5I_SYMBOL_PERIOD_S

```
constexpr double GPS_L5I_SYMBOL_PERIOD_S = 0.01
```

GPS L5I symbol period [seconds].

Definition at line 33 of file GPS_L5.h.

11.206.2.10 GPS_L5Q_CODE_LENGTH_CHIPS

```
constexpr int32_t GPS_L5Q_CODE_LENGTH_CHIPS = 10230
```

GPS L5Q code length [chips].

Definition at line 36 of file GPS_L5.h.

11.206.2.11 GPS_L5Q_CODE_RATE_CPS

```
constexpr double GPS_L5Q_CODE_RATE_CPS = 10.23e6
```

GPS L5Q code rate [chips/s].

Definition at line 34 of file GPS_L5.h.

11.206.2.12 GPS_L5Q_PERIOD_S

```
constexpr double GPS_L5Q_PERIOD_S = 0.001
```

GPS L5Q code period [seconds].

Definition at line 35 of file GPS_L5.h.

11.207 gps_l5_dll_pll_tracking.h File Reference

Interface of an adapter of a DLL+PLL tracking loop block for GPS L5 to a [TrackingInterface](#).

```
#include "dll_pll_veml_tracking.h"  
#include "tracking_interface.h"  
#include <string>
```

Classes

- class [GpsL5DllPllTracking](#)
This class implements a code DLL + carrier PLL tracking loop.

11.207.1 Detailed Description

Interface of an adapter of a DLL+PLL tracking loop block for GPS L5 to a [TrackingInterface](#).

Author

Javier Arribas, 2017. jarribas(at)cttc.es

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.208 gps_l5_dll_pll_tracking_fpga.h File Reference

Interface of an adapter of a DLL+PLL tracking loop block for GPS L5 to a [TrackingInterface](#) for the FPGA.

```
#include "dll_pll_veml_tracking_fpga.h"
#include "tracking_interface.h"
#include <string>
```

Classes

- class [GpsL5DllPllTrackingFpga](#)

This class implements a code DLL + carrier PLL tracking loop.

11.208.1 Detailed Description

Interface of an adapter of a DLL+PLL tracking loop block for GPS L5 to a [TrackingInterface](#) for the FPGA.

Author

Marc Majoral, 2019. mmajoral(at)cttc.cat Javier Arribas, 2019. jarribas(at)cttc.es

Code DLL + carrier PLL according to the algorithms described in: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.209 gps_l5_signal.h File Reference

This class implements signal generators for the GPS L5 signals.

```
#include <complex>
#include <cstdint>
#include <gsl/gsl>
```

Functions

- void [gps_l5i_code_gen_complex](#) (own::span< std::complex< float >> _dest, uint32_t _prn)
Generates complex GPS L5I code for the desired SV ID.
- void [gps_l5i_code_gen_float](#) (own::span< float > _dest, uint32_t _prn)
Generates real GPS L5I code for the desired SV ID.
- void [gps_l5q_code_gen_complex](#) (own::span< std::complex< float >> _dest, uint32_t _prn)
Generates complex GPS L5Q code for the desired SV ID.
- void [gps_l5q_code_gen_float](#) (own::span< float > _dest, uint32_t _prn)
Generates real GPS L5Q code for the desired SV ID.
- void [gps_l5i_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, uint32_t _prn, int32_t _fs)
Generates complex GPS L5I code for the desired SV ID, and sampled to specific sampling frequency.
- void [gps_l5q_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, uint32_t _prn, int32_t _fs)
Generates complex GPS L5Q code for the desired SV ID, and sampled to specific sampling frequency.

11.209.1 Detailed Description

This class implements signal generators for the GPS L5 signals.

Author

Javier Arribas, 2017. jarribas(at)cttc.es

Copyright (C) 2010-2019 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.209.2 Function Documentation

11.209.2.1 [gps_l5i_code_gen_complex\(\)](#)

```
void gps_l5i_code_gen_complex (
    own::span< std::complex< float >> _dest,
    uint32_t _prn )
```

Generates complex GPS L5I code for the desired SV ID.

11.209.2.2 gps_l5i_code_gen_complex_sampled()

```
void gps_l5i_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    uint32_t _prn,
    int32_t _fs )
```

Generates complex GPS L5I code for the desired SV ID, and sampled to specific sampling frequency.

11.209.2.3 gps_l5i_code_gen_float()

```
void gps_l5i_code_gen_float (
    own::span< float > _dest,
    uint32_t _prn )
```

Generates real GPS L5I code for the desired SV ID.

11.209.2.4 gps_l5q_code_gen_complex()

```
void gps_l5q_code_gen_complex (
    own::span< std::complex< float >> _dest,
    uint32_t _prn )
```

Generates complex GPS L5Q code for the desired SV ID.

11.209.2.5 gps_l5q_code_gen_complex_sampled()

```
void gps_l5q_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    uint32_t _prn,
    int32_t _fs )
```

Generates complex GPS L5Q code for the desired SV ID, and sampled to specific sampling frequency.

11.209.2.6 gps_l5q_code_gen_float()

```
void gps_l5q_code_gen_float (
    own::span< float > _dest,
    uint32_t _prn )
```

Generates real GPS L5Q code for the desired SV ID.

11.210 gps_l5_telemetry_decoder.h File Reference

Interface of an adapter of a GPS L5 (CNAV) data decoder block to a [TelemetryDecoderInterface](#).

```
#include "gnss_satellite.h"
#include "gnss_synchro.h"
#include "gps_l5_telemetry_decoder_gs.h"
#include "telemetry_decoder_interface.h"
#include <gnuradio/runtime_types.h>
#include <cstdio>
#include <string>
```

Classes

- class [GpsL5TelemetryDecoder](#)

This class implements a NAV data decoder for GPS L5.

11.210.1 Detailed Description

Interface of an adapter of a GPS L5 (CNAV) data decoder block to a [TelemetryDecoderInterface](#).

Author

Antonio Ramos, 2017. antonio.ramos(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.211 gps_l5_telemetry_decoder_gs.h File Reference

Interface of a CNAV message demodulator block.

```
#include "GPS_L5.h"
#include "gnss_satellite.h"
#include "gps_cnav_navigation_message.h"
#include <boost/circular_buffer.hpp>
#include <gnuradio/block.h>
#include <gnuradio/types.h>
#include <stdint>
#include <fstream>
#include <string>
#include <boost/shared_ptr.hpp>
#include "cnav_msg.h"
```

Classes

- class [gps_l5_telemetry_decoder_gs](#)
This class implements a GPS L5 Telemetry decoder.

Typedefs

- using **gps_l5_telemetry_decoder_gs_sptr** = boost::shared_ptr< [gps_l5_telemetry_decoder_gs](#) >

Functions

- gps_l5_telemetry_decoder_gs_sptr **gps_l5_make_telemetry_decoder_gs** (const [Gnss_Satellite](#) &satellite, bool dump)

11.211.1 Detailed Description

Interface of a CNAV message demodulator block.

Antonio Ramos, 2017. antonio.ramos@cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.212 [gps_l5i_pcps_acquisition.h](#) File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L5i signals.

```
#include "channel_fsm.h"
#include "complex_byte_to_float_x2.h"
#include "gnss_synchro.h"
#include "pcps_acquisition.h"
#include <gnuradio/blocks/float_to_complex.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GpsL5iPcpsAcquisition](#)
This class adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L5i signals.

11.212.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L5i signals.

Authors

- Javier Arribas, 2017. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.213 gps_l5i_pcps_acquisition_fpga.h File Reference

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L5i signals for the FPGA.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include "pcps_acquisition_fpga.h"
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [GpsL5iPcpsAcquisitionFpga](#)

This class adapts a PCPS acquisition block off-loaded on an FPGA to an [AcquisitionInterface](#) for GPS L5i signals.

11.213.1 Detailed Description

Adapts a PCPS acquisition block to an [AcquisitionInterface](#) for GPS L5i signals for the FPGA.

Authors

- Marc Majoral, 2019. mmajoral(at)cttc.es
- Javier Arribas, 2019. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.214 gps_navigation_message.h File Reference

Interface of a GPS NAV Data message decoder.

```
#include "GPS_L1_CA.h"
#include "gps_ephemeris.h"
#include "gps_iono.h"
#include "gps_utc_model.h"
#include <bitset>
#include <stdint>
#include <map>
#include <string>
#include <utility>
#include <vector>
```

Classes

- class [Gps_Navigation_Message](#)

This class decodes a GPS NAV Data message as described in IS-GPS-200K.

11.214.1 Detailed Description

Interface of a GPS NAV Data message decoder.

Author

Javier Arribas, 2011. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.215 gps_sdr_signal_processing.h File Reference

This class implements various functions for GPS L1 CA signals.

```
#include <complex>
#include <stdint>
#include <gsl/gsl>
```

Functions

- void [gps_l1_ca_code_gen_int](#) (own::span< int32_t > _dest, int32_t _prn, uint32_t _chip_shift)
Generates int GPS L1 C/A code for the desired SV ID and code shift.
- void [gps_l1_ca_code_gen_float](#) (own::span< float > _dest, int32_t _prn, uint32_t _chip_shift)
Generates float GPS L1 C/A code for the desired SV ID and code shift.
- void [gps_l1_ca_code_gen_complex](#) (own::span< std::complex< float >> _dest, int32_t _prn, uint32_t _chip_shift)
Generates complex GPS L1 C/A code for the desired SV ID and code shift, and sampled to specific sampling frequency.
- void [gps_l1_ca_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, uint32_t _prn, int32_t _fs, uint32_t _chip_shift, uint32_t _ncodes)
Generates N complex GPS L1 C/A codes for the desired SV ID and code shift.
- void [gps_l1_ca_code_gen_complex_sampled](#) (own::span< std::complex< float >> _dest, uint32_t _prn, int32_t _fs, uint32_t _chip_shift)
Generates complex GPS L1 C/A code for the desired SV ID and code shift.

11.215.1 Detailed Description

This class implements various functions for GPS L1 CA signals.

Author

Javier Arribas, 2011. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.215.2 Function Documentation

11.215.2.1 [gps_l1_ca_code_gen_complex\(\)](#)

```
void gps_l1_ca_code_gen_complex (
    own::span< std::complex< float >> _dest,
    int32_t _prn,
    uint32_t _chip_shift )
```

Generates complex GPS L1 C/A code for the desired SV ID and code shift, and sampled to specific sampling frequency.

11.215.2.2 `gps_l1_ca_code_gen_complex_sampled()` [1/2]

```
void gps_l1_ca_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    uint32_t _prn,
    int32_t _fs,
    uint32_t _chip_shift,
    uint32_t _ncodes )
```

Generates N complex GPS L1 C/A codes for the desired SV ID and code shift.

11.215.2.3 `gps_l1_ca_code_gen_complex_sampled()` [2/2]

```
void gps_l1_ca_code_gen_complex_sampled (
    own::span< std::complex< float >> _dest,
    uint32_t _prn,
    int32_t _fs,
    uint32_t _chip_shift )
```

Generates complex GPS L1 C/A code for the desired SV ID and code shift.

11.215.2.4 `gps_l1_ca_code_gen_float()`

```
void gps_l1_ca_code_gen_float (
    own::span< float > _dest,
    int32_t _prn,
    uint32_t _chip_shift )
```

Generates float GPS L1 C/A code for the desired SV ID and code shift.

11.215.2.5 `gps_l1_ca_code_gen_int()`

```
void gps_l1_ca_code_gen_int (
    own::span< int32_t > _dest,
    int32_t _prn,
    uint32_t _chip_shift )
```

Generates int GPS L1 C/A code for the desired SV ID and code shift.

11.216 `gps_utc_model.h` File Reference

Interface of a GPS UTC MODEL storage.

```
#include <boost/serialization/nvp.hpp>
#include <cstdint>
```

Classes

- class [Gps_Utc_Model](#)

This class is a storage for the GPS UTC MODEL data as described in IS-GPS-200K.

11.216.1 Detailed Description

Interface of a GPS UTC MODEL storage.

Author

Javier Arribas, 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.217 gpx_printer.h File Reference

Interface of a class that prints PVT information to a gpx file.

```
#include <fstream>
#include <string>
```

Classes

- class [Gpx_Printer](#)

Prints PVT information to GPX format file.

11.217.1 Detailed Description

Interface of a class that prints PVT information to a gpx file.

Author

Álvaro Cebrián Juan, 2018. acebrianjuan(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.218 gr_complex_ip_packet_source.h File Reference

Receives ip frames containing samples in UDP frame encapsulation using a high performance packet capture library (libpcap)

```
#include <boost/thread.hpp>
#include <gnuradio/sync_block.h>
#include <arpa/inet.h>
#include <net/ethernet.h>
#include <net/if.h>
#include <netinet/if_ether.h>
#include <pcap.h>
#include <string>
#include <sys/ioctl.h>
#include <boost/shared_ptr.hpp>
```

Classes

- class [Gr_Complex_Ip_Packet_Source](#)

11.218.1 Detailed Description

Receives ip frames containing samples in UDP frame encapsulation using a high performance packet capture library (libpcap)

Javier Arribas jarribas (at) cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.219 hybrid_observables.h File Reference

Implementation of an adapter of an observables block accepting all kind of signals to a [ObservablesInterface](#).

```
#include "gnss_synchro.h"
#include "hybrid_observables_gs.h"
#include "observables_interface.h"
#include <gnuradio/gr_complex.h>
#include <gnuradio/runtime_types.h>
#include <cstdint>
#include <string>
```

Classes

- class [HybridObservables](#)

This class implements an [ObservablesInterface](#) for observables of all kind of GNSS signals.

11.219.1 Detailed Description

Implementation of an adapter of an observables block accepting all kind of signals to a [ObservablesInterface](#).

Author

Mara Branzanti 2013. mara.branzanti(at)gmail.com

Javier Arribas 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.220 hybrid_observables_gs.h File Reference

Interface of the observables computation block.

```
#include "obs_conf.h"
#include <boost/circular_buffer.hpp>
#include <gnuradio/block.h>
#include <gnuradio/types.h>
#include <cstddef>
#include <stdint>
#include <fstream>
#include <map>
#include <memory>
#include <string>
#include <typeinfo>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [Gnss_circular_deque< T >](#)
- class [hybrid_observables_gs](#)

This class implements a block that computes observables.

Typedefs

- using [hybrid_observables_gs_sptr](#) = boost::shared_ptr< [hybrid_observables_gs](#) >

Functions

- `hybrid_observables_gs_sptr` **hybrid_observables_gs_make** (const [Obs_Conf](#) &conf_)

11.220.1 Detailed Description

Interface of the observables computation block.

Author

Mara Branzanti 2013. [mara.branzanti\(at\)gmail.com](mailto:mara.branzanti(at)gmail.com)
Javier Arribas 2013. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)
Antonio Ramos 2018. [antonio.ramos\(at\)cttc.es](mailto:antonio.ramos(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.221 lbyte_to_cbyte.h File Reference

Adapts an I/Q interleaved byte (unsigned char) sample stream into a `std::complex<unsigned char>` stream.

```
#include "conjugate_ic.h"
#include "gnss_block_interface.h"
#include "interleaved_byte_to_complex_byte.h"
#include <gnuradio/blocks/file_sink.h>
#include <cstdint>
#include <string>
```

Classes

- class [lbyteToCbyte](#)

11.221.1 Detailed Description

Adapts an I/Q interleaved byte (unsigned char) sample stream into a `std::complex<unsigned char>` stream.

Author

Carles Fernandez Prades, [cfernandez\(at\)cttc.es](mailto:cfernandez(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.222 ibyte_to_complex.h File Reference

Adapts an I/Q interleaved byte integer sample stream to a `gr_complex` (float) stream.

```
#include "conjugate_cc.h"
#include "gnss_block_interface.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/interleaved_char_to_complex.h>
#include <stdint>
#include <string>
```

Classes

- class [lbyteToComplex](#)

Adapts an I/Q interleaved byte integer sample stream to a `gr_complex` (float) stream.

11.222.1 Detailed Description

Adapts an I/Q interleaved byte integer sample stream to a `gr_complex` (float) stream.

Author

Javier Arribas, jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.223 ibyte_to_cshort.h File Reference

Adapts a short interleaved sample stream into a `std::complex<short>` stream.

```
#include "conjugate_sc.h"
#include "gnss_block_interface.h"
#include "interleaved_byte_to_complex_short.h"
#include <gnuradio/blocks/file_sink.h>
#include <stdint>
#include <string>
```

Classes

- class [lbyteToCshort](#)

Adapts a short integer (16 bits) interleaved sample stream into a `std::complex<short>` stream.

11.223.1 Detailed Description

Adapts a short interleaved sample stream into a `std::complex<short>` stream.

Author

Carles Fernandez-Prades, [cfernandez\(at\)cttc.es](mailto:cfernandez(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.224 in_memory_configuration.h File Reference

A [ConfigurationInterface](#) for testing purposes.

```
#include "configuration_interface.h"
#include "string_converter.h"
#include <stdint>
#include <map>
#include <memory>
#include <string>
```

Classes

- class [InMemoryConfiguration](#)

This class is an implementation of the interface [ConfigurationInterface](#).

11.224.1 Detailed Description

A [ConfigurationInterface](#) for testing purposes.

Author

Carlos Aviles, 2010. [carlos.avilesr\(at\)gmail.com](mailto:carlos.avilesr(at)gmail.com)

This implementation accepts configuration parameters upon instantiation and it is intended to be used in unit testing.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.225 ini.h File Reference

This function parses an INI file into easy-to-access name/value pairs.

Macros

- `#define INI_ALLOW_MULTILINE 1`

Functions

- `int ini_parse (const char *filename, int(*handler)(void *user, const char *section, const char *name, const char *value), void *user)`

11.225.1 Detailed Description

This function parses an INI file into easy-to-access name/value pairs.

Author

Brush Technologies, 2009.

inih (INI Not Invented Here) is a simple .INI file parser written in C++. It's only a couple of pages of code, and it was designed to be small and simple, so it's good for embedded systems. To use it, just give `ini_parse()` an INI file, and it will call a callback for every name=value pair parsed, giving you strings for the section, name, and value. It's done this way because it works well on low-memory embedded systems, but also because it makes for a KISS implementation. Parse given INI-style file. May have [section]s, name=value pairs (whitespace stripped), and comments starting with ';' (semicolon). Section is "" if name=value pair parsed before any section heading. For each name=value pair parsed, call handler function with given user pointer as well as section, name, and value (data only valid for duration of handler call). Handler should return nonzero on success, zero on error. Returns 0 on success, line number of first error on parse error, on -1 on file open error

inih and INIReaden are released under the New BSD license:

Copyright (c) 2009, Brush Technology All rights reserved.

SPDX-License-Identifier: BSD-3-Clause

Go to the project home page for more info:

<https://github.com/benhoyt/inih>

11.226 INIReader.h File Reference

This class reads an INI file into easy-to-access name/value pairs.

```
#include <cstdint>
#include <map>
#include <string>
```

Classes

- class [INIRedReader](#)

Read an INI file into easy-to-access name/value pairs. (Note that I've gone for simplicity here rather than speed, but it should be pretty decent.)

11.226.1 Detailed Description

This class reads an INI file into easy-to-access name/value pairs.

Author

Brush Technologies, 2009.

inih (INI Not Invented Here) is a simple .INI file parser written in C++. It's only a couple of pages of code, and it was designed to be small and simple, so it's good for embedded systems. To use it, just give `ini_parse()` an INI file, and it will call a callback for every name=value pair parsed, giving you strings for the section, name, and value. It's done this way because it works well on low-memory embedded systems, but also because it makes for a KISS implementation.

inih and [INIRedReader](#) are released under the New BSD license:

Copyright (c) 2009, Brush Technology All rights reserved.

SPDX-License-Identifier: BSD-3-Clause

Go to the project home page for more info:

<https://github.com/benhoyt/inih>

11.227 interleaved_byte_to_complex_byte.h File Reference

Adapts an 8-bits interleaved sample stream into a 16-bits complex stream.

```
#include <gnuradio/sync_decimator.h>
#include <boost/shared_ptr.hpp>
```

Classes

- class [interleaved_byte_to_complex_byte](#)

This class adapts an 8-bits interleaved sample stream into a 16-bits complex stream (`std::complex<unsigned char>`)

Typedefs

- using [interleaved_byte_to_complex_byte_sptr](#) = `boost::shared_ptr< interleaved_byte_to_complex_byte >`

Functions

- interleaved_byte_to_complex_byte_sptr **make_interleaved_byte_to_complex_byte** ()

11.227.1 Detailed Description

Adapts an 8-bits interleaved sample stream into a 16-bits complex stream.

Author

Carles Fernandez Prades, cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.228 interleaved_byte_to_complex_short.h File Reference

Adapts a byte (8-bits) interleaved sample stream into a `std::complex<short>` stream.

```
#include <gnuradio/sync_decimator.h>
#include <boost/shared_ptr.hpp>
```

Classes

- class [interleaved_byte_to_complex_short](#)
This class adapts a short (16-bits) interleaved sample stream into a `std::complex<short>` stream.

Typedefs

- using **interleaved_byte_to_complex_short_sptr** = boost::shared_ptr< [interleaved_byte_to_complex_short](#) >

Functions

- interleaved_byte_to_complex_short_sptr **make_interleaved_byte_to_complex_short** ()

11.228.1 Detailed Description

Adapts a byte (8-bits) interleaved sample stream into a `std::complex<short>` stream.

Author

Javier Arribas (jarribas(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.229 interleaved_short_to_complex_short.h File Reference

Adapts a short (16-bits) interleaved sample stream into a `std::complex<short>` stream.

```
#include <gnuradio/sync_decimator.h>
#include <boost/shared_ptr.hpp>
```

Classes

- class [interleaved_short_to_complex_short](#)
This class adapts a short (16-bits) interleaved sample stream into a `std::complex<short>` stream.

Typedefs

- using `interleaved_short_to_complex_short_sptr` = `boost::shared_ptr<interleaved_short_to_complex_short>`

Functions

- `interleaved_short_to_complex_short_sptr` **make_interleaved_short_to_complex_short** ()

11.229.1 Detailed Description

Adapts a short (16-bits) interleaved sample stream into a `std::complex<short>` stream.

Author

Carles Fernandez Prades, cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.230 ishort_to_complex.h File Reference

Adapts an I/Q interleaved short integer sample stream to a `gr_complex` (float) stream.

```
#include "conjugate_cc.h"
#include "gnss_block_interface.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/interleaved_short_to_complex.h>
#include <stdint>
#include <string>
```

Classes

- class [lshortToComplex](#)

Adapts an I/Q interleaved short integer sample stream to a `gr_complex` (float) stream.

11.230.1 Detailed Description

Adapts an I/Q interleaved short integer sample stream to a `gr_complex` (float) stream.

Author

Javier Arribas, jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.231 ishort_to_cshort.h File Reference

Adapts a short interleaved sample stream into a `std::complex<short>` stream.

```
#include "conjugate_sc.h"
#include "gnss_block_interface.h"
#include "interleaved_short_to_complex_short.h"
#include <gnuradio/blocks/file_sink.h>
#include <stdint>
#include <string>
```

Classes

- class [lshortToCshort](#)

Adapts a short integer (16 bits) interleaved sample stream into a `std::complex<short>` stream.

11.231.1 Detailed Description

Adapts a short interleaved sample stream into a `std::complex<short>` stream.

Author

Carles Fernandez-Prades, cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.232 item_type_helpers.h File Reference

Utility functions for converting between item types.

```
#include <cstdint>
#include <functional>
#include <string>
```

Typedefs

- using `item_type_converter_t` = `std::function< void(void *, const void *, uint32_t)>`

Functions

- bool `item_type_valid` (const `std::string` &`item_type`)
Check if a string is a valid item type.
- `size_t` `item_type_size` (const `std::string` &`item_type`)
Return the size of the given item type, or zero if unknown.
- bool `item_type_is_complex` (const `std::string` &`item_type`)
Determine if an item_type is complex.
- `item_type_converter_t` `make_vector_converter` (const `std::string` &`input_type`, const `std::string` &`output_type`)
Create a function to convert an array of input_type to an array of output_type.

11.232.1 Detailed Description

Utility functions for converting between item types.

Authors

- Cillian O'Driscoll, 2019. cillian.odriscoll(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.232.2 Function Documentation

11.232.2.1 item_type_is_complex()

```
bool item_type_is_complex (
    const std::string & item_type )
```

Determine if an item_type is complex.

11.232.2.2 item_type_size()

```
size_t item_type_size (
    const std::string & item_type )
```

Return the size of the given item type, or zero if unknown.

11.232.2.3 item_type_valid()

```
bool item_type_valid (
    const std::string & item_type )
```

Check if a string is a valid item type.

Valid item types include: "byte", "short", "float", "ibyte", "ishort", "cbyte", "cshort", "gr_complex"

11.232.2.4 make_vector_converter()

```
item_type_converter_t make_vector_converter (
    const std::string & input_type,
    const std::string & output_type )
```

Create a function to convert an array of input_type to an array of output_type.

Provides a generic interface to generate conversion functions for mapping arrays of items.

Parameters

<i>input_type</i>	- String representation of the input item type
<i>output_type</i>	- String representation of the output item type

The item types accepted are:

1. "byte" for 8 bit integers
 2. "cbyte" for complex (interleaved) 8 bit integers
 3. "ibyte" for complex (interleaved) 8 bit integers
-
1. "short" for 16 bit integers
 2. "cshort" for complex (interleaved) 16 bit integers
 3. "ishort" for complex (interleaved) 16 bit integers
 4. "float" for 32 bit floating point values
 5. "gr_complex" for complex (interleaved) 32 bit floating point values

Returns

A function object with the following prototype: `void convert_fun(void *dest, void *src, int num_items);`

11.233 kml_printer.h File Reference

Interface of a class that prints PVT information to a kml file.

```
#include <fstream>
#include <string>
```

Classes

- class [Kml_Printer](#)
Prints PVT information to OGC KML format file (can be viewed with Google Earth)

11.233.1 Detailed Description

Interface of a class that prints PVT information to a kml file.

Author

Javier Arribas, 2011. jarribas(at)cttc.es Álvaro Cebrián Juan, 2018. acebrianjuan(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.234 labsat23_source.h File Reference

Unpacks the Labsat 2 (ls2) and (ls3) capture files.

```
#include "concurrent_queue.h"
#include <gnuradio/block.h>
#include <pmt/pmt.h>
#include <cstdint>
#include <fstream>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [labsat23_source](#)

This class implements conversion between Labsat2 and 3 format byte packet samples to gr_complex.

Typedefs

- using **labsat23_source_sptr** = boost::shared_ptr< [labsat23_source](#) >

Functions

- labsat23_source_sptr **labsat23_make_source_sptr** (const char *signal_file_basename, int channel_selector, [Concurrent_Queue](#)< pmt::pmt_t > *queue)

11.234.1 Detailed Description

Unpacks the Labsat 2 (ls2) and (ls3) capture files.

Javier Arribas jarribas (at) cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.235 labsat_signal_source.h File Reference

Labsat 2 and 3 front-end signal sampler driver.

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/hier_block2.h>
#include <pmt/pmt.h>
#include <memory>
#include <string>
```

Classes

- class [LabsatSignalSource](#)

This class reads samples stored by a LabSat 2 or LabSat 3 device.

11.235.1 Detailed Description

Labsat 2 and 3 front-end signal sampler driver.

Author

Javier Arribas, jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.236 lock_detectors.h File Reference

Interface of a library with a set of code and carrier phase lock detectors.

```
#include <gnuradio/gr_complex.h>
```

Functions

- float [cn0_svn_estimator](#) (const gr_complex *Prompt_buffer, int length, float coh_integration_time_s)
cn0_svn_estimator is a Carrier-to-Noise (CN0) estimator based on the Signal-to-Noise Variance (SNV) estimator
- float [cn0_m2m4_estimator](#) (const gr_complex *Prompt_buffer, int length, float coh_integration_time_s)
cn0_m2m4_estimator is a Carrier-to-Noise (CN0) estimator based on the Second- and Fourth-Order Moments Method (M2M4)
- float [carrier_lock_detector](#) (gr_complex *Prompt_buffer, int length)
A carrier lock detector.

11.236.1 Detailed Description

Interface of a library with a set of code and carrier phase lock detectors.

SNV_CN0 is a Carrier-to-Noise (CN0) estimator based on the Signal-to-Noise Variance (SNV) estimator [1]. Carrier lock detector using normalised estimate of the cosine of twice the carrier phase error [2].

[1] Marco Pini, Emanuela Falletti and Maurizio Fantino, "Performance Evaluation of C/N0 Estimators using a Real Time GNSS Software Receiver," IEEE 10th International Symposium on Spread Spectrum Techniques and Applications, pp.28-30, August 2008.

[2] Van Dierendonck, A.J. (1996), Global Positioning System: Theory and Applications, Volume I, Chapter 8: GPS Receivers, AJ Systems, Los Altos, CA 94024. Inc.: 329-407.

Authors

- Javier Arribas, 2011. jarribas(at)cttc.es
- Luis Esteve, 2012. luis(at)epsilon-formacion.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)
 GNSS-SDR is a software defined Global Navigation Satellite Systems receiver
 This file is part of GNSS-SDR.
 SPDX-License-Identifier: GPL-3.0-or-later

11.236.2 Function Documentation

11.236.2.1 carrier_lock_detector()

```
float carrier_lock_detector (
    gr_complex * Prompt_buffer,
    int length )
```

A carrier lock detector.

The Carrier Phase Lock Detector block uses the estimate of the cosine of twice the carrier phase error is given by

$$C2\phi = \frac{NBD}{NBP}, \quad (11.1)$$

where $NBD = (\sum_{i=0}^{N-1} |Im(Pc(i))|)^2 + (\sum_{i=0}^{N-1} |Re(Pc(i))|)^2$, $NBP = \sum_{i=0}^{N-1} Im(Pc(i))^2 - \sum_{i=0}^{N-1} Re(Pc(i))^2$, and $Pc(i)$ is the prompt correlator output for the sample index i . Ref: Van Dierendonck, A.J. (1996), Global Positioning System: Theory and Applications, Volume I, Chapter 8: GPS Receivers, AJ Systems, Los Altos, CA 94024. Inc.: 329-407.

11.236.2.2 cn0_m2m4_estimator()

```
float cn0_m2m4_estimator (
    const gr_complex * Prompt_buffer,
    int length,
    float coh_integration_time_s )
```

cn0_m2m4_estimator is a Carrier-to-Noise (CN0) estimator based on the Second- and Fourth-Order Moments Method (M2M4)

Signal-to-Noise (SNR) (ρ) estimator using the Moments Method:

$$\hat{\rho} = \frac{\sqrt{2\hat{M}_2^2 - \hat{M}_4}}{\hat{M}_2 - \sqrt{2\hat{M}_2^2 - \hat{M}_4}}, \quad (11.2)$$

where $\hat{M}_2 = \frac{1}{N} \sum_{k=0}^{K-1} |P[k]|^2$, $\hat{M}_4 = \frac{1}{K} \sum_{k=0}^{K-1} |P[k]|^4$, $|\cdot|$ is the absolute value, and $P[k]$ is the prompt correlator output for the sample index k .

The SNR value is converted to CN0 [dB-Hz] taking into account the coherent integration time, using the following formula:

$$CN0_{dB} = 10 * \log(\hat{\rho}) - 10 * \log(T_{int}), \quad (11.3)$$

where T_{int} is the coherent integration time, in seconds.

Ref: D. R. Pauluzzi, N. C. Beaulieu, "A comparison of SNR estimation techniques for the AWGN channel," IEEE Trans. on Comm., vol. 48, no. 10, pp. 1681–1691, Oct. 2000.

11.236.2.3 `cn0_svn_estimator()`

```
float cn0_svn_estimator (
    const gr_complex * Prompt_buffer,
    int length,
    float coh_integration_time_s )
```

`cn0_svn_estimator` is a Carrier-to-Noise (CN0) estimator based on the Signal-to-Noise Variance (SNV) estimator

Signal-to-Noise (SNR) (ρ) estimator using the Signal-to-Noise Variance (SNV) estimator:

$$\hat{\rho} = \frac{\hat{P}_s}{\hat{P}_n} = \frac{\hat{P}_s}{\hat{P}_{tot} - \hat{P}_s}, \quad (11.4)$$

where $\hat{P}_s = \left(\frac{1}{N} \sum_{i=0}^{N-1} |Re(Pc(i))| \right)^2$ is the estimation of the signal power, $\hat{P}_{tot} = \frac{1}{N} \sum_{i=0}^{N-1} |Pc(i)|^2$ is the estimator of the total power, $|\cdot|$ is the absolute value, $Re(\cdot)$ stands for the real part of the value, and $Pc(i)$ is the prompt correlator output for the sample index i .

The SNR value is converted to CN0 [dB-Hz], taking into account the coherent integration time, using the following formula:

$$CN0_{dB} = 10 * \log(\hat{\rho}) - 10 * \log(T_{int}), \quad (11.5)$$

where T_{int} is the coherent integration time, in seconds.

Ref: Marco Pini, Emanuela Falletti and Maurizio Fantino, "Performance Evaluation of C/N0 Estimators using a Real Time GNSS Software Receiver," IEEE 10th International Symposium on Spread Spectrum Techniques and Applications, pp.28-30, August 2008.

11.237 MATH_CONSTANTS.h File Reference

Defines useful mathematical constants and their scaled versions.

Variables

- `constexpr double GNSS_OMEGA_EARTH_DOT` = 7.2921151467e-5
Default Earth rotation rate, [rad/s].
- `constexpr double SPEED_OF_LIGHT_M_S` = 299792458.0
Speed of light in vacuum [m/s].
- `constexpr double SPEED_OF_LIGHT_M_MS` = 299792.4580
Speed of light in vacuum [m/ms].
- `constexpr double GPS_GM` = 3.986005e14
Universal gravitational constant times the mass of the Earth, [m^3/s^2] IS-GPS-200K, pag 92.
- `constexpr double GPS_F` = -4.442807633e-10
Constant, [$s/(m)^{1/2}$], IS-GPS-200K, pag. 92.
- `constexpr double GALILEO_GM` = 3.986004418e14
Geocentric gravitational constant [m^3/s^2], OS SIS ICD v1.3, pag. 44.
- `constexpr double GALILEO_F` = -4.442807309e-10
Constant, [$s/(m)^{1/2}$], OS SIS ICD v1.3, pag. 47.
- `constexpr double GLONASS_OMEGA_EARTH_DOT` = 7.292115e-5
Earth rotation rate, [rad/s] ICD L1, L2 GLONASS Edition 5.1 2008 pag. 55.
- `constexpr double GLONASS_GM` = 398600.44e9

- Universal gravitational constant times the mass of the Earth, $[m^3/s^2]$.*
- constexpr double [BEIDOU_OMEGA_EARTH_DOT](#) = 7.2921150e-5
- Earth rotation rate, $[rad/s]$ as defined in BDS-SIS-ICD-B1I-3.0 2019-02, pag. 3.*
- constexpr double [BEIDOU_GM](#) = 3.986004418e14
- Universal gravitational constant times the mass of the Earth, $[m^3/s^2]$ as defined in CGCS2000.*
- constexpr double [BEIDOU_F](#) = -4.442807309e-10
- Constant, $[s/(m)^{1/2}] F=-2(GM)^{.5}/C^2$.*
- constexpr double [GNSS_PI](#) = 3.1415926535898
- pi constant as defined for GNSS*
- constexpr double [HALF_PI](#) = [GNSS_PI](#) / 2.0
- pi/2*
- constexpr double [TWO_PI](#) = 2.0 * [GNSS_PI](#)
- 2 * pi*
- constexpr double [TWO_P3](#) = 8.0
- 2^3*
- constexpr double [TWO_P4](#) = 16.0
- 2^4*
- constexpr double [TWO_P11](#) = 2048.0
- 2^11*
- constexpr double [TWO_P12](#) = 4096.0
- 2^12*
- constexpr double [TWO_P14](#) = 16384.0
- 2^14*
- constexpr double [TWO_P16](#) = 65536.0
- 2^16*
- constexpr double [TWO_P19](#) = 524288.0
- 2^19*
- constexpr double [TWO_P31](#) = 2147483648.0
- 2^31*
- constexpr double [TWO_P32](#) = 4294967296.0
- 2^32*
- constexpr double [TWO_P56](#) = 7.205759403792794e+016
- 2^56*
- constexpr double [TWO_P57](#) = 1.441151880758559e+017
- 2^57*
- constexpr double [TWO_N2](#) = 0.25
- 2^-2*
- constexpr double [TWO_N5](#) = 0.03125
- 2^-5*
- constexpr double [TWO_N6](#) = 0.015625
- 2^-6*
- constexpr double [TWO_N8](#) = 0.00390625
- 2^-8*
- constexpr double [TWO_N9](#) = 0.001953125
- 2^-9*
- constexpr double [TWO_N10](#) = 0.0009765625
- 2^-10*
- constexpr double [TWO_N11](#) = 4.882812500000000e-004
- 2^-11*
- constexpr double [TWO_N14](#) = 0.00006103515625
- 2^-14*

- constexpr double [TWO_N15](#) = 0.00003051757813
 2^{-15}
- constexpr double [TWO_N16](#) = 0.0000152587890625
 2^{-16}
- constexpr double [TWO_N17](#) = 7.629394531250000e-006
 2^{-17}
- constexpr double [TWO_N18](#) = 3.814697265625000e-006
 2^{-18}
- constexpr double [TWO_N19](#) = 1.907348632812500e-006
 2^{-19}
- constexpr double [TWO_N20](#) = 9.536743164062500e-007
 2^{-20}
- constexpr double [TWO_N21](#) = 4.768371582031250e-007
 2^{-21}
- constexpr double [TWO_N23](#) = 1.192092895507810e-007
 2^{-23}
- constexpr double [TWO_N24](#) = 5.960464477539063e-008
 2^{-24}
- constexpr double [TWO_N25](#) = 2.980232238769531e-008
 2^{-25}
- constexpr double [TWO_N27](#) = 7.450580596923828e-009
 2^{-27}
- constexpr double [TWO_N29](#) = 1.862645149230957e-009
 2^{-29}
- constexpr double [TWO_N30](#) = 9.313225746154785e-010
 2^{-30}
- constexpr double [TWO_N31](#) = 4.656612873077393e-010
 2^{-31}
- constexpr double [TWO_N32](#) = 2.328306436538696e-010
 2^{-32}
- constexpr double [TWO_N33](#) = 1.164153218269348e-010
 2^{-33}
- constexpr double [TWO_N34](#) = 5.82076609134674e-011
 2^{-34}
- constexpr double [TWO_N35](#) = 2.91038304567337e-011
 2^{-35}
- constexpr double [TWO_N38](#) = 3.637978807091713e-012
 2^{-38}
- constexpr double [TWO_N39](#) = 1.818989403545856e-012
 2^{-39}
- constexpr double [TWO_N40](#) = 9.094947017729280e-013
 2^{-40}
- constexpr double [TWO_N43](#) = 1.136868377216160e-013
 2^{-43}
- constexpr double [TWO_N44](#) = 5.684341886080802e-14
 2^{-44}
- constexpr double [TWO_N46](#) = 1.4210854715202e-014
 2^{-46}
- constexpr double [TWO_N48](#) = 3.552713678800501e-15
 2^{-46}
- constexpr double [TWO_N50](#) = 8.881784197001252e-016

- 2^{-50}
- constexpr double [TWO_N51](#) = 4.44089209850063e-016
- 2^{-51}
- constexpr double [TWO_N55](#) = 2.775557561562891e-017
- 2^{-55}
- constexpr double [TWO_N57](#) = 6.938893903907228e-18
- 2^{-57}
- constexpr double [TWO_N59](#) = 1.73472347597681e-018
- 2^{-59}
- constexpr double [TWO_N60](#) = 8.673617379884036e-19
- 2^{-60}
- constexpr double [TWO_N66](#) = 1.3552527156068805425093160010874271392822265625e-20
- 2^{-66}
- constexpr double [TWO_N68](#) = 3.388131789017201e-21
- 2^{-68}
- constexpr double [PI_TWO_N19](#) = 5.992112452678286e-006
- $\pi \cdot 2^{-19}$.
- constexpr double [PI_TWO_N43](#) = 3.571577341960839e-013
- $\pi \cdot 2^{-43}$.
- constexpr double [PI_TWO_N31](#) = 1.462918079267160e-009
- $\pi \cdot 2^{-31}$.
- constexpr double [PI_TWO_N38](#) = 1.142904749427469e-011
- $\pi \cdot 2^{-38}$.
- constexpr double [PI_TWO_N23](#) = 3.745070282923929e-007
- $\pi \cdot 2^{-23}$.
- constexpr double [D2R](#) = [GNSS_PI](#) / 180.0
- deg to rad*
- constexpr double [R2D](#) = 180.0 / [GNSS_PI](#)
- rad to deg*
- constexpr double [SC2RAD](#) = [GNSS_PI](#)
- semi-circle to radian (IS-GPS)*
- constexpr double [AS2R](#) = [D2R](#) / 3600.0
- arc sec to radian*
- constexpr double [AU](#) = 149597870691.0
- 1 Astronomical Unit AU (m) distance from Earth to the Sun.*

11.237.1 Detailed Description

Defines useful mathematical constants and their scaled versions.

Author

Javier Arribas, 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.237.2 Variable Documentation

11.237.2.1 AS2R

```
constexpr double AS2R = D2R / 3600.0
```

arc sec to radian

Definition at line 120 of file MATH_CONSTANTS.h.

11.237.2.2 AU

```
constexpr double AU = 149597870691.0
```

1 Astronomical Unit AU (m) distance from Earth to the Sun.

Definition at line 122 of file MATH_CONSTANTS.h.

11.237.2.3 BEIDOU_F

```
constexpr double BEIDOU_F = -4.442807309e-10
```

Constant, $[s/(m)^{(1/2)}] F = -2(GM)^{.5}/C^2$.

Definition at line 42 of file MATH_CONSTANTS.h.

11.237.2.4 BEIDOU_GM

```
constexpr double BEIDOU_GM = 3.986004418e14
```

Universal gravitational constant times the mass of the Earth, $[m^3/s^2]$ as defined in CGCS2000.

Definition at line 41 of file MATH_CONSTANTS.h.

11.237.2.5 BEIDOU_OMEGA_EARTH_DOT

```
constexpr double BEIDOU_OMEGA_EARTH_DOT = 7.2921150e-5
```

Earth rotation rate, $[rad/s]$ as defined in BDS-SIS-ICD-B1I-3.0 2019-02, pag. 3.

Definition at line 40 of file MATH_CONSTANTS.h.

11.237.2.6 D2R

```
constexpr double D2R = GNSS_PI / 180.0
```

deg to rad

Definition at line 117 of file MATH_CONSTANTS.h.

11.237.2.7 GALILEO_F

```
constexpr double GALILEO_F = -4.442807309e-10
```

Constant, $[s/(m)^{(1/2)}]$. OS SIS ICD v1.3, pag. 47.

Definition at line 33 of file MATH_CONSTANTS.h.

11.237.2.8 GALILEO_GM

```
constexpr double GALILEO_GM = 3.986004418e14
```

Geocentric gravitational constant $[m^3/s^2]$, OS SIS ICD v1.3, pag. 44.

Definition at line 32 of file MATH_CONSTANTS.h.

11.237.2.9 GLONASS_GM

```
constexpr double GLONASS_GM = 398600.44e9
```

Universal gravitational constant times the mass of the Earth, $[m^3/s^2]$.

Definition at line 37 of file MATH_CONSTANTS.h.

11.237.2.10 GLONASS_OMEGA_EARTH_DOT

```
constexpr double GLONASS_OMEGA_EARTH_DOT = 7.292115e-5
```

Earth rotation rate, $[rad/s]$ ICD L1, L2 GLONASS Edition 5.1 2008 pag. 55.

Definition at line 36 of file MATH_CONSTANTS.h.

11.237.2.11 GNSS_OMEGA_EARTH_DOT

```
constexpr double GNSS_OMEGA_EARTH_DOT = 7.2921151467e-5
```

Default Earth rotation rate, [rad/s].

Definition at line 23 of file MATH_CONSTANTS.h.

11.237.2.12 GNSS_PI

```
constexpr double GNSS_PI = 3.1415926535898
```

pi constant as defined for GNSS

Definition at line 44 of file MATH_CONSTANTS.h.

11.237.2.13 GPS_F

```
constexpr double GPS_F = -4.442807633e-10
```

Constant, $[s/(m)^{(1/2)}]$, IS-GPS-200K, pag. 92.

Definition at line 29 of file MATH_CONSTANTS.h.

11.237.2.14 GPS_GM

```
constexpr double GPS_GM = 3.986005e14
```

Universal gravitational constant times the mass of the Earth, $[m^3/s^2]$ IS-GPS-200K, pag 92.

Definition at line 28 of file MATH_CONSTANTS.h.

11.237.2.15 HALF_PI

```
constexpr double HALF_PI = GNSS_PI / 2.0
```

pi/2

Definition at line 45 of file MATH_CONSTANTS.h.

11.237.2.16 PI_TWO_N19

```
constexpr double PI_TWO_N19 = 5.992112452678286e-006
```

$\text{Pi} \cdot 2^{-19}$.

Definition at line 111 of file MATH_CONSTANTS.h.

11.237.2.17 PI_TWO_N23

```
constexpr double PI_TWO_N23 = 3.745070282923929e-007
```

$\text{Pi} \cdot 2^{-23}$.

Definition at line 115 of file MATH_CONSTANTS.h.

11.237.2.18 PI_TWO_N31

```
constexpr double PI_TWO_N31 = 1.462918079267160e-009
```

$\text{Pi} \cdot 2^{-31}$.

Definition at line 113 of file MATH_CONSTANTS.h.

11.237.2.19 PI_TWO_N38

```
constexpr double PI_TWO_N38 = 1.142904749427469e-011
```

$\text{Pi} \cdot 2^{-38}$.

Definition at line 114 of file MATH_CONSTANTS.h.

11.237.2.20 PI_TWO_N43

```
constexpr double PI_TWO_N43 = 3.571577341960839e-013
```

$\text{Pi} \cdot 2^{-43}$.

Definition at line 112 of file MATH_CONSTANTS.h.

11.237.2.21 R2D

```
constexpr double R2D = 180.0 / GNSS_PI
```

rad to deg

Definition at line 118 of file MATH_CONSTANTS.h.

11.237.2.22 SC2RAD

```
constexpr double SC2RAD = GNSS_PI
```

semi-circle to radian (IS-GPS)

Definition at line 119 of file MATH_CONSTANTS.h.

11.237.2.23 SPEED_OF_LIGHT_M_MS

```
constexpr double SPEED_OF_LIGHT_M_MS = 299792.4580
```

Speed of light in vacuum [m/ms].

Definition at line 25 of file MATH_CONSTANTS.h.

11.237.2.24 SPEED_OF_LIGHT_M_S

```
constexpr double SPEED_OF_LIGHT_M_S = 299792458.0
```

Speed of light in vacuum [m/s].

Definition at line 24 of file MATH_CONSTANTS.h.

11.237.2.25 TWO_N10

```
constexpr double TWO_N10 = 0.0009765625
```

2⁻¹⁰

Definition at line 73 of file MATH_CONSTANTS.h.

11.237.2.26 TWO_N11

```
constexpr double TWO_N11 = 4.882812500000000e-004
```

 2^{-11}

Definition at line 74 of file MATH_CONSTANTS.h.

11.237.2.27 TWO_N14

```
constexpr double TWO_N14 = 0.00006103515625
```

 2^{-14}

Definition at line 75 of file MATH_CONSTANTS.h.

11.237.2.28 TWO_N15

```
constexpr double TWO_N15 = 0.00003051757813
```

 2^{-15}

Definition at line 76 of file MATH_CONSTANTS.h.

11.237.2.29 TWO_N16

```
constexpr double TWO_N16 = 0.0000152587890625
```

 2^{-16}

Definition at line 77 of file MATH_CONSTANTS.h.

11.237.2.30 TWO_N17

```
constexpr double TWO_N17 = 7.629394531250000e-006
```

 2^{-17}

Definition at line 78 of file MATH_CONSTANTS.h.

11.237.2.31 TWO_N18

```
constexpr double TWO_N18 = 3.814697265625000e-006
```

 2^{-18}

Definition at line 79 of file MATH_CONSTANTS.h.

11.237.2.32 TWO_N19

```
constexpr double TWO_N19 = 1.907348632812500e-006
```

 2^{-19}

Definition at line 80 of file MATH_CONSTANTS.h.

11.237.2.33 TWO_N2

```
constexpr double TWO_N2 = 0.25
```

 2^{-2}

Definition at line 68 of file MATH_CONSTANTS.h.

11.237.2.34 TWO_N20

```
constexpr double TWO_N20 = 9.536743164062500e-007
```

 2^{-20}

Definition at line 81 of file MATH_CONSTANTS.h.

11.237.2.35 TWO_N21

```
constexpr double TWO_N21 = 4.768371582031250e-007
```

 2^{-21}

Definition at line 82 of file MATH_CONSTANTS.h.

11.237.2.36 TWO_N23

```
constexpr double TWO_N23 = 1.192092895507810e-007
```

 2^{-23}

Definition at line 83 of file MATH_CONSTANTS.h.

11.237.2.37 TWO_N24

```
constexpr double TWO_N24 = 5.960464477539063e-008
```

 2^{-24}

Definition at line 84 of file MATH_CONSTANTS.h.

11.237.2.38 TWO_N25

```
constexpr double TWO_N25 = 2.980232238769531e-008
```

 2^{-25}

Definition at line 85 of file MATH_CONSTANTS.h.

11.237.2.39 TWO_N27

```
constexpr double TWO_N27 = 7.450580596923828e-009
```

 2^{-27}

Definition at line 86 of file MATH_CONSTANTS.h.

11.237.2.40 TWO_N29

```
constexpr double TWO_N29 = 1.862645149230957e-009
```

 2^{-29}

Definition at line 87 of file MATH_CONSTANTS.h.

11.237.2.41 TWO_N30

```
constexpr double TWO_N30 = 9.313225746154785e-010
```

 2^{-30}

Definition at line 88 of file MATH_CONSTANTS.h.

11.237.2.42 TWO_N31

```
constexpr double TWO_N31 = 4.656612873077393e-010
```

 2^{-31}

Definition at line 89 of file MATH_CONSTANTS.h.

11.237.2.43 TWO_N32

```
constexpr double TWO_N32 = 2.328306436538696e-010
```

 2^{-32}

Definition at line 90 of file MATH_CONSTANTS.h.

11.237.2.44 TWO_N33

```
constexpr double TWO_N33 = 1.164153218269348e-010
```

 2^{-33}

Definition at line 91 of file MATH_CONSTANTS.h.

11.237.2.45 TWO_N34

```
constexpr double TWO_N34 = 5.82076609134674e-011
```

 2^{-34}

Definition at line 92 of file MATH_CONSTANTS.h.

11.237.2.46 TWO_N35

```
constexpr double TWO_N35 = 2.91038304567337e-011
```

 2^{-35}

Definition at line 93 of file MATH_CONSTANTS.h.

11.237.2.47 TWO_N38

```
constexpr double TWO_N38 = 3.637978807091713e-012
```

 2^{-38}

Definition at line 94 of file MATH_CONSTANTS.h.

11.237.2.48 TWO_N39

```
constexpr double TWO_N39 = 1.818989403545856e-012
```

 2^{-39}

Definition at line 95 of file MATH_CONSTANTS.h.

11.237.2.49 TWO_N40

```
constexpr double TWO_N40 = 9.094947017729280e-013
```

 2^{-40}

Definition at line 96 of file MATH_CONSTANTS.h.

11.237.2.50 TWO_N43

```
constexpr double TWO_N43 = 1.136868377216160e-013
```

 2^{-43}

Definition at line 97 of file MATH_CONSTANTS.h.

11.237.2.51 TWO_N44

```
constexpr double TWO_N44 = 5.684341886080802e-14
```

 2^{-44}

Definition at line 98 of file MATH_CONSTANTS.h.

11.237.2.52 TWO_N46

```
constexpr double TWO_N46 = 1.4210854715202e-014
```

 2^{-46}

Definition at line 99 of file MATH_CONSTANTS.h.

11.237.2.53 TWO_N48

```
constexpr double TWO_N48 = 3.552713678800501e-15
```

 2^{-46}

Definition at line 100 of file MATH_CONSTANTS.h.

11.237.2.54 TWO_N5

```
constexpr double TWO_N5 = 0.03125
```

 2^{-5}

Definition at line 69 of file MATH_CONSTANTS.h.

11.237.2.55 TWO_N50

```
constexpr double TWO_N50 = 8.881784197001252e-016
```

 2^{-50}

Definition at line 102 of file MATH_CONSTANTS.h.

11.237.2.56 TWO_N51

```
constexpr double TWO_N51 = 4.44089209850063e-016
```

 2^{-51}

Definition at line 103 of file MATH_CONSTANTS.h.

11.237.2.57 TWO_N55

```
constexpr double TWO_N55 = 2.775557561562891e-017
```

 2^{-55}

Definition at line 104 of file MATH_CONSTANTS.h.

11.237.2.58 TWO_N57

```
constexpr double TWO_N57 = 6.938893903907228e-18
```

 2^{-57}

Definition at line 105 of file MATH_CONSTANTS.h.

11.237.2.59 TWO_N59

```
constexpr double TWO_N59 = 1.73472347597681e-018
```

 2^{-59}

Definition at line 106 of file MATH_CONSTANTS.h.

11.237.2.60 TWO_N6

```
constexpr double TWO_N6 = 0.015625
```

 2^{-6}

Definition at line 70 of file MATH_CONSTANTS.h.

11.237.2.61 TWO_N60

```
constexpr double TWO_N60 = 8.673617379884036e-19
```

2^{-60}

Definition at line 107 of file MATH_CONSTANTS.h.

11.237.2.62 TWO_N66

```
constexpr double TWO_N66 = 1.3552527156068805425093160010874271392822265625e-20
```

2^{-66}

Definition at line 108 of file MATH_CONSTANTS.h.

11.237.2.63 TWO_N68

```
constexpr double TWO_N68 = 3.388131789017201e-21
```

2^{-68}

Definition at line 109 of file MATH_CONSTANTS.h.

11.237.2.64 TWO_N8

```
constexpr double TWO_N8 = 0.00390625
```

2^{-8}

Definition at line 71 of file MATH_CONSTANTS.h.

11.237.2.65 TWO_N9

```
constexpr double TWO_N9 = 0.001953125
```

2^{-9}

Definition at line 72 of file MATH_CONSTANTS.h.

11.237.2.66 TWO_P11

```
constexpr double TWO_P11 = 2048.0
```

 2^{11}

Definition at line 58 of file MATH_CONSTANTS.h.

11.237.2.67 TWO_P12

```
constexpr double TWO_P12 = 4096.0
```

 2^{12}

Definition at line 59 of file MATH_CONSTANTS.h.

11.237.2.68 TWO_P14

```
constexpr double TWO_P14 = 16384.0
```

 2^{14}

Definition at line 60 of file MATH_CONSTANTS.h.

11.237.2.69 TWO_P16

```
constexpr double TWO_P16 = 65536.0
```

 2^{16}

Definition at line 61 of file MATH_CONSTANTS.h.

11.237.2.70 TWO_P19

```
constexpr double TWO_P19 = 524288.0
```

 2^{19}

Definition at line 62 of file MATH_CONSTANTS.h.

11.237.2.71 TWO_P3

```
constexpr double TWO_P3 = 8.0
```

 2^3

Definition at line 56 of file MATH_CONSTANTS.h.

11.237.2.72 TWO_P31

```
constexpr double TWO_P31 = 2147483648.0
```

 2^{31}

Definition at line 63 of file MATH_CONSTANTS.h.

11.237.2.73 TWO_P32

```
constexpr double TWO_P32 = 4294967296.0
```

 2^{32}

Definition at line 64 of file MATH_CONSTANTS.h.

11.237.2.74 TWO_P4

```
constexpr double TWO_P4 = 16.0
```

 2^4

Definition at line 57 of file MATH_CONSTANTS.h.

11.237.2.75 TWO_P56

```
constexpr double TWO_P56 = 7.205759403792794e+016
```

 2^{56}

Definition at line 65 of file MATH_CONSTANTS.h.

11.237.2.76 TWO_P57

```
constexpr double TWO_P57 = 1.441151880758559e+017
```

2^{57}

Definition at line 66 of file MATH_CONSTANTS.h.

11.237.2.77 TWO_PI

```
constexpr double TWO_PI = 2.0 * GNSS_PI
```

$2 * \pi$

Definition at line 46 of file MATH_CONSTANTS.h.

11.238 mmse_resampler_conditioner.h File Reference

Interface of an adapter of a mmse resampler conditioner block to a `SignalConditionerInterface`.

```
#include "gnss_block_interface.h"
#include <gnuradio/filter/fir_filter_ccf.h>
#include <gnuradio/filter/fractional_resampler_cc.h>
#include <gnuradio/filter/firdes.h>
#include <string>
```

Classes

- class [MmseResamplerConditioner](#)
Interface of a MMSE resampler block adapter to a `SignalConditionerInterface`.

11.238.1 Detailed Description

Interface of an adapter of a mmse resampler conditioner block to a `SignalConditionerInterface`.

Author

Antonio Ramos, 2018. antonio.ramos(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.239 monitor_pvt.h File Reference

Interface of the [Monitor_Pvt](#) class.

```
#include <boost/serialization/nvp.hpp>
#include <stdint>
```

Classes

- class [Monitor_Pvt](#)
This class contains parameters and outputs of the PVT block.

11.239.1 Detailed Description

Interface of the [Monitor_Pvt](#) class.

Author

Álvaro Cebrián Juan, 2019. [acebrianjuan\(at\)gmail.com](mailto:acebrianjuan(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.240 monitor_pvt_udp_sink.h File Reference

Interface of a class that sends serialized [Monitor_Pvt](#) objects over udp to one or multiple endpoints.

```
#include "monitor_pvt.h"
#include "serdes_monitor_pvt.h"
#include <boost/asio.hpp>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [Monitor_Pvt_Udp_Sink](#)

Typedefs

- using **b_io_context** = boost::asio::io_service

11.240.1 Detailed Description

Interface of a class that sends serialized [Monitor_Pvt](#) objects over udp to one or multiple endpoints.

Author

Álvaro Cebrián Juan, 2019. [acebrianjuan\(at\)gmail.com](mailto:acebrianjuan(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.241 multichannel_file_signal_source.h File Reference

Implementation of a class that reads signals samples from files at different frequency band and adapts it to a [SignalSourceInterface](#).

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/file_source.h>
#include <gnuradio/blocks/throttle.h>
#include <gnuradio/hier_block2.h>
#include <pmt/pmt.h>
#include <cstdint>
#include <memory>
#include <string>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [MultichannelFileSignalSource](#)

Class that reads signals samples from files at different frequency bands and adapts it to a [SignalSourceInterface](#).

11.241.1 Detailed Description

Implementation of a class that reads signals samples from files at different frequency band and adapts it to a [SignalSourceInterface](#).

Author

Javier Arribas, 2019 [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)

This class represents a file signal source. Internally it uses a GNU Radio's `gr_file_source` as a connector to the data.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.242 nmea_printer.h File Reference

Interface of a NMEA 2.1 printer for GNSS-SDR This class provides a implementation of a subset of the NMEA-0183 standard for interfacing marine electronic devices as defined by the National Marine Electronics Association (NMEA). See <https://www.nmea.org/> for the NMEA 183 standard.

```
#include <boost/date_time/posix_time/ptime.hpp>
#include <fstream>
#include <memory>
#include <string>
```

Classes

- class [Nmea_Printer](#)

This class provides a implementation of a subset of the NMEA-0183 standard for interfacing marine electronic devices as defined by the National Marine Electronics Association (NMEA).

11.242.1 Detailed Description

Interface of a NMEA 2.1 printer for GNSS-SDR This class provides a implementation of a subset of the NMEA-0183 standard for interfacing marine electronic devices as defined by the National Marine Electronics Association (NMEA). See <https://www.nmea.org/> for the NMEA 183 standard.

Author

Javier Arribas, 2012. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.243 nonlinear_tracking.h File Reference

Interface of a library for nonlinear tracking algorithms.

```
#include <armadillo>
#include <gnuradio/gr_complex.h>
```

Classes

- class [ModelFunction](#)
- class [CubatureFilter](#)
- class [UnscentedFilter](#)

11.243.1 Detailed Description

Interface of a library for nonlinear tracking algorithms.

[CubatureFilter](#) implements the functionality of the Cubature Kalman Filter, which uses multidimensional cubature rules to estimate the time evolution of a nonlinear system. [UnscentedFilter](#) implements an Unscented Kalman Filter which uses Unscented Transform rules to perform a similar estimation.

[1] I Arasaratnam and S Haykin. Cubature kalman filters. IEEE Transactions on Automatic Control, 54(6):1254–1269, 2009.

Authors

- Gerald LaMountain, 2019. gerald(at)ece.neu.edu
- Jordi Vila-Valls 2019. jvila(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.244 notch_cc.h File Reference

Implements a notch filter algorithm.

```
#include <boost/shared_ptr.hpp>
#include <gnuradio/block.h>
#include <gnuradio/fft/fft.h>
#include <volk_gnssdr/volk_gnssdr_alloc.h>
#include <cstdint>
#include <memory>
```

Classes

- class [Notch](#)

This class implements a real-time software-defined multi state notch filter.

Typedefs

- using **notch_sptr** = boost::shared_ptr< [Notch](#) >

Functions

- notch_sptr **make_notch_filter** (float pfa, float p_c_factor, int32_t length_, int32_t n_segments_est, int32_t n_segments_reset)

11.244.1 Detailed Description

Implements a notch filter algorithm.

Author

Antonio Ramos (antonio.ramosdet(at)gmail.com)

Copyright (C) 2010-2019 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.245 notch_filter.h File Reference

Adapter of a multistate [Notch](#) filter.

```
#include "gnss_block_interface.h"
#include "notch_cc.h"
#include <gnuradio/blocks/file_sink.h>
#include <string>
#include <vector>
```

Classes

- class [NotchFilter](#)

11.245.1 Detailed Description

Adapter of a multistate [Notch](#) filter.

Author

Antonio Ramos, 2017. antonio.ramosdet(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.246 notch_filter_lite.h File Reference

Adapts a light version of a multistate notch filter.

```
#include "gnss_block_interface.h"
#include "notch_lite_cc.h"
#include <gnuradio/blocks/file_sink.h>
#include <string>
#include <vector>
```

Classes

- class [NotchFilterLite](#)

11.246.1 Detailed Description

Adapts a light version of a multistate notch filter.

Author

Antonio Ramos, 2017. antonio.ramosdet(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.247 notch_lite_cc.h File Reference

Implements a notch filter light algorithm.

```
#include <boost/shared_ptr.hpp>
#include <gnuradio/block.h>
#include <gnuradio/fft/fft.h>
#include <volk_gnssssdr/volk_gnssssdr_alloc.h>
#include <stdint>
#include <memory>
```

Classes

- class [NotchLite](#)

This class implements a real-time software-defined multi state notch filter light version.

Typedefs

- using **notch_lite_sptr** = boost::shared_ptr< [NotchLite](#) >

Functions

- notch_lite_sptr **make_notch_filter_lite** (float p_c_factor, float pfa, int32_t length_, int32_t n_segments_est, int32_t n_segments_reset, int32_t n_segments_coeff)

11.247.1 Detailed Description

Implements a notch filter light algorithm.

Author

Antonio Ramos (antonio.ramosdet(at)gmail.com)

Copyright (C) 2010-2019 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.248 nsr_file_signal_source.h File Reference

Implementation of a class that reads signals samples from a NSR 2 bits sampler front-end file and adapts it to a `SignalSourceInterface`. More information about the front-end here <http://www.ifen.com/products/sx-scientific-gnss-solutions/nsr-software-receiver.html>.

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include "unpack_byte_2bit_samples.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/file_source.h>
#include <gnuradio/blocks/throttle.h>
#include <gnuradio/hier_block2.h>
#include <pmt/pmt.h>
#include <memory>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [NsrFileSignalSource](#)

Class that reads signals samples from a file and adapts it to a `SignalSourceInterface`.

11.248.1 Detailed Description

Implementation of a class that reads signals samples from a NSR 2 bits sampler front-end file and adapts it to a SignalSourceInterface. More information about the front-end here <http://www.ifen.fr/com/products/sx-scientific-gnss-solutions/nsr-software-receiver.html>.

Author

Javier Arribas, 2013 jarribas(at)cttc.es

This class represents a file signal source.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.249 obs_conf.h File Reference

Class that contains all the configuration parameters for generic observables block.

```
#include <cstdint>
#include <string>
```

Classes

- class [Obs_Conf](#)

11.249.1 Detailed Description

Class that contains all the configuration parameters for generic observables block.

Author

Javier Arribas, 2020. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.250 obsdiff_flags.h File Reference

Helper file for unit testing.

```
#include <gflags/gflags.h>
```

Functions

- **DEFINE_double** (skip_obs_transitory_s, 30.0, "Skip the initial observable outputs to avoid transitory results [s]")
- **DEFINE_double** (skip_obs_ends_s, 5.0, "Skip the lasts observable outputs to avoid transitory results [s]")
- **DEFINE_bool** (single_diffs, false, "Compute also the single difference errors for Accumulated Carrier Phase and Carrier Doppler (requires LO synchronization between receivers)")
- **DEFINE_bool** (compare_with_5X, false, "Compare the E5a Doppler and Carrier Phases with the E5 full bw in RINEX (expect discrepancy due to the center frequencies difference)")
- **DEFINE_bool** (dupli_sat, false, "Enable special observable test mode where the scenario contains duplicated satellite orbits")
- **DEFINE_bool** (single_diff, false, "Enable special observable test mode using only rover observables")
- **DEFINE_string** (dupli_sat_prns, "1,2,3,4", "List of duplicated satellites PRN pairs (i.e. 1,2,3,4 indicates that the PRNs 1,2 share the same orbit. The same applies for PRNs 3,4)")
- **DEFINE_string** (base_rinex_obs, "base.obs", "Filename of reference RINEX observation file")
- **DEFINE_string** (rinex_nav, "base.nav", "Filename of reference RINEX navigation file")
- **DEFINE_string** (rover_rinex_obs, "base.obs", "Filename of test RINEX observation file")
- **DEFINE_string** (system, "G", "GNSS satellite system: G for GPS, E for Galileo")
- **DEFINE_string** (signal, "1C", "GNSS signal: 1C for GPS L1 CA, 1B for Galileo E1")
- **DEFINE_bool** (remove_rx_clock_error, false, "Compute and remove the receivers clock error prior to compute observable differences (requires a valid RINEX nav file for both receivers)")

11.250.1 Detailed Description

Helper file for unit testing.

Author

Javier Arribas, 2020. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.251 observable_tests_flags.h File Reference

Helper file for unit testing.

```
#include <gflags/gflags.h>
#include <limits>
```

Functions

- **DEFINE_double** (skip_obs_transitory_s, 30.0, "Skip the initial observable outputs to avoid transitory results [s]")
- **DEFINE_bool** (compute_single_diffs, false, "Compute also the single difference errors for Accumulated Carrier Phase and Carrier Doppler (requires LO synchronization between receivers)")
- **DEFINE_bool** (compare_with_5X, false, "Compare the E5a Doppler and Carrier Phases with the E5 full bw in RINEX (expect discrepancy due to the center frequencies differences)")
- **DEFINE_bool** (duplicated_satellites_test, false, "Enable special observable test mode where the scenario contains duplicated satellite orbits")
- **DEFINE_string** (duplicated_satellites_prns, "1,2,3,4", "List of duplicated satellites PRN pairs (i.e. 1,2,3,4 indicates that the PRNs 1,2 share the same orbit. The same applies for PRNs 3,4)")

11.251.1 Detailed Description

Helper file for unit testing.

Author

Javier Arribas, 2018. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.252 observables_dump_reader.h File Reference

Helper file for unit testing.

```
#include <cstdint>
#include <fstream>
#include <string>
#include <vector>
```

Classes

- class [Observables_Dump_Reader](#)

11.252.1 Detailed Description

Helper file for unit testing.

Author

Javier Arribas, 2017. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.253 observables_interface.h File Reference

This class represents an interface to an Observables block.

```
#include "gnss_block_interface.h"
```

Classes

- class [ObservablesInterface](#)

This abstract class represents an interface to an observables block.

11.253.1 Detailed Description

This class represents an interface to an Observables block.

Author

Javier Arribas, 2011. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)

Abstract class for Observables modules. Since all its methods are virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.254 osmosdr_signal_source.h File Reference

Signal source wrapper for OsmoSDR-compatible front-ends, such as HackRF or Realtek's RTL2832U-based USB dongle DVB-T receivers (see <https://osmocom.org/projects/rtl-sdr/wiki> for more information)

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include <gnuradio/blocks/file_sink.h>
#include <pmt/pmt.h>
#include <cstdint>
#include <memory>
#include <osmosdr/source.h>
#include <stdexcept>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [OsmosdrSignalSource](#)

This class reads samples OmoSDR-compatible front-ends, such as HackRF or Realtek's RTL2832U-based USB dongle DVB-T receivers (see <https://osmocom.org/projects/rtl-sdr/wiki>)

11.254.1 Detailed Description

Signal source wrapper for OmoSDR-compatible front-ends, such as HackRF or Realtek's RTL2832U-based USB dongle DVB-T receivers (see <https://osmocom.org/projects/rtl-sdr/wiki> for more information)

Author

Javier Arribas, 2012. [jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.255 pass_through.h File Reference

Interface of a block that just puts its input in its output.

```
#include "conjugate_cc.h"
#include "conjugate_ic.h"
#include "conjugate_sc.h"
#include "gnss_block_interface.h"
#include <gnuradio/blocks/copy.h>
#include <gnuradio/runtime_types.h>
#include <cstdio>
#include <string>
```

Classes

- class [Pass_Through](#)

This class implements a block that connects input and output (does nothing)

11.255.1 Detailed Description

Interface of a block that just puts its input in its output.

Author

Carlos Aviles, 2010. [carlos.avilesr\(at\)gmail.com](mailto:carlos.avilesr(at)gmail.com)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.256 pcps_acquisition.h File Reference

This class implements a Parallel Code Phase Search Acquisition.

```
#include "acq_conf.h"
#include "channel_fsm.h"
#include <armadillo>
#include <glog/logging.h>
#include <gnuradio/block.h>
#include <gnuradio/fft/fft.h>
#include <gnuradio/gr_complex.h>
#include <gnuradio/thread/thread.h>
#include <gnuradio/types.h>
#include <volk/volk_complex.h>
#include <volk_gnssssdr/volk_gnssssdr_alloc.h>
#include <complex>
#include <cstdint>
#include <memory>
#include <string>
#include <utility>
#include <gsl/gsl>
#include <boost/shared_ptr.hpp>
```

Classes

- class [pcps_acquisition](#)
This class implements a Parallel Code Phase Search Acquisition.

Typedefs

- using **pcps_acquisition_sptr** = boost::shared_ptr< [pcps_acquisition](#) >

Functions

- pcps_acquisition_sptr **pcps_make_acquisition** (const [Acq_Conf](#) &conf_)

11.256.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition.

Acquisition strategy (Kay Borre book + CFAR threshold).

1. Compute the input signal power estimation
2. Doppler serial search loop
3. Perform the FFT-based circular convolution (parallel time search)
4. Record the maximum peak and the associated synchronization parameters
5. Compute the test statistics and compare to the threshold
6. Declare positive or negative acquisition using a message queue

Kay Borre book: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, "A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach", Birkhauser, 2007. pp 81-84

Authors

- Javier Arribas, 2011. jarribas(at)cttc.es
- Luis Esteve, 2012. luis(at)epsilon-formacion.com
- Marc Molina, 2013. marc.molina.pena@gmail.com
- Cillian O'Driscoll, 2017. cillian(at)ieee.org
- Antonio Ramos, 2017. antonio.ramos@cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.257 pcps_acquisition_fine_doppler_cc.h File Reference

This class implements a Parallel Code Phase Search Acquisition with multi-dwells and fine Doppler estimation for GPS L1 C/A signal.

```
#include "acq_conf.h"
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include <armadillo>
#include <gnuradio/block.h>
#include <gnuradio/fft/fft.h>
#include <gnuradio/gr_complex.h>
#include <volk_gnssssdr/volk_gnssssdr_alloc.h>
#include <stdint>
#include <fstream>
#include <memory>
#include <string>
#include <utility>
#include <boost/shared_ptr.hpp>
```

Classes

- class [pcps_acquisition_fine_doppler_cc](#)
This class implements a Parallel Code Phase Search Acquisition.

Typedefs

- using **pcps_acquisition_fine_doppler_cc_sptr** = boost::shared_ptr< [pcps_acquisition_fine_doppler_cc](#) >

Functions

- pcps_acquisition_fine_doppler_cc_sptr **pcps_make_acquisition_fine_doppler_cc** (const [Acq_Conf](#) &conf_)

11.257.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition with multi-dwells and fine Doppler estimation for GPS L1 C/A signal.

Acquisition strategy (Kay Borre book).

1. Compute the input signal power estimation
2. Doppler serial search loop
3. Perform the FFT-based circular convolution (parallel time search)
4. Record the maximum peak and the associated synchronization parameters
5. Compute the test statistics and compare to the threshold
6. Declare positive or negative acquisition using a message port

Kay Borre book: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, "A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach", Birkhauser, 2007. pp 81-84

Authors

- Javier Arribas, 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.258 pcps_acquisition_fpga.h File Reference

This class implements a Parallel Code Phase Search Acquisition for the FPGA.

```
#include "channel_fsm.h"
#include "fpga_acquisition.h"
#include <glog/logging.h>
#include <cstdint>
#include <memory>
#include <string>
```

Classes

- struct [pcpsconf_fpga_t](#)
- class [pcps_acquisition_fpga](#)

This class implements a Parallel Code Phase Search Acquisition that uses the FPGA.

Typedefs

- using **pcps_acquisition_fpga_sptr** = std::shared_ptr< [pcps_acquisition_fpga](#) >

Functions

- pcps_acquisition_fpga_sptr **pcps_make_acquisition_fpga** ([pcpsconf_fpga_t](#) conf_)

11.258.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition for the FPGA.

Kay Borre book: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, "A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach", Birkhauser, 2007. pp 81-84

Authors

- Marc Majoral, 2019. mmajoral(at)cttc.es
- Javier Arribas, 2019. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.259 pcps_assisted_acquisition_cc.h File Reference

This class implements a Parallel Code Phase Search Acquisition with assistance and multi-dwells.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include <gnuradio/block.h>
#include <gnuradio/fft/fft.h>
#include <gnuradio/gr_complex.h>
#include <fstream>
#include <memory>
#include <string>
#include <utility>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [pcps_assisted_acquisition_cc](#)
This class implements a Parallel Code Phase Search Acquisition.

Typedefs

- using **pcps_assisted_acquisition_cc_sptr** = boost::shared_ptr< [pcps_assisted_acquisition_cc](#) >

Functions

- `pcps_assisted_acquisition_cc_sptr pcps_make_assisted_acquisition_cc (int32_t max_dwells, uint32_t sampled_ms, int32_t doppler_max, int32_t doppler_min, int64_t fs_in, int32_t samples_per_ms, bool dump, const std::string &dump_filename)`

11.259.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition with assistance and multi-dwells.

Acquisition strategy (Kay Borre book + CFAR threshold).

1. Compute the input signal power estimation
2. Doppler serial search loop
3. Perform the FFT-based circular convolution (parallel time search)
4. Record the maximum peak and the associated synchronization parameters
5. Compute the test statistics and compare to the threshold
6. Declare positive or negative acquisition using a message queue

Kay Borre book: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, "A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach", Birkhauser, 2007. pp 81-84

Authors

- Javier Arribas, 2013. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.260 `pcps_cccwsr_acquisition_cc.h` File Reference

This class implements a Parallel Code Phase Search acquisition with Coherent [Channel](#) Combining With Sign Recovery scheme.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include <gnuradio/block.h>
#include <gnuradio/fft/fft.h>
#include <gnuradio/gr_complex.h>
#include <fstream>
#include <memory>
#include <string>
#include <utility>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [pcps_cccwsr_acquisition_cc](#)

This class implements a Parallel Code Phase Search Acquisition with Coherent [Channel](#) Combining With Sign Recovery scheme.

Typedefs

- using **pcps_cccwsr_acquisition_cc_sptr** = boost::shared_ptr< [pcps_cccwsr_acquisition_cc](#) >

Functions

- pcps_cccwsr_acquisition_cc_sptr **pcps_cccwsr_make_acquisition_cc** (uint32_t sampled_ms, uint32_t max_dwells, uint32_t doppler_max, int64_t fs_in, int32_t samples_per_ms, int32_t samples_per_code, bool dump, const std::string &dump_filename)

11.260.1 Detailed Description

This class implements a Parallel Code Phase Search acquisition with Coherent [Channel](#) Combining With Sign Recovery scheme.

Author

Marc Molina, 2013. marc.molina.pena(at)gmail.com

D.Borio, C.O'Driscoll, G.Lachapelle, "Coherent, Noncoherent and Differentially Coherent Combining Techniques for Acquisition of New Composite GNSS Signals", IEEE Transactions On Aerospace and Electronic Systems vol. 45 no. 3, July 2009, section IV

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.261 pcps_openc1_acquisition_cc.h File Reference

This class implements a Parallel Code Phase Search Acquisition using OpenCL to offload some functions to the GPU.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include "openc1/fft_internal.h"
#include <gnuradio/block.h>
#include <gnuradio/fft/fft.h>
#include <gnuradio/gr_complex.h>
#include "openc1/cl.hpp"
#include <cstdint>
#include <fstream>
#include <memory>
#include <string>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class `pcps_openc1_acquisition_cc`
This class implements a Parallel Code Phase Search Acquisition.

Typedefs

- typedef boost::shared_ptr< `pcps_openc1_acquisition_cc` > `pcps_openc1_acquisition_cc_sptr`

Functions

- `pcps_openc1_acquisition_cc_sptr pcps_make_openc1_acquisition_cc` (uint32_t sampled_ms, uint32_t max_dwells, uint32_t doppler_max, int64_t fs_in, int samples_per_ms, int samples_per_code, bool bit_transition_flag, bool dump, const std::string &dump_filename)

11.261.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition using OpenCL to offload some functions to the GPU.

Acquisition strategy (Kay Borre book + CFAR threshold).

1. Compute the input signal power estimation
2. Doppler serial search loop
3. Perform the FFT-based circular convolution (parallel time search)
4. Record the maximum peak and the associated synchronization parameters
5. Compute the test statistics and compare to the threshold
6. Declare positive or negative acquisition using a message port

Kay Borre book: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, "A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach", Birkha user, 2007. pp 81-84

Authors

- Javier Arribas, 2011. jarribas(at)cttc.es
- Luis Esteve, 2012. luis(at)epsilon-formacion.com
- Marc Molina, 2013. marc.molina.pena@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.262 pcps_quicksync_acquisition_cc.h File Reference

This class implements a Parallel Code Phase Search Acquisition with the QuickSync Algorithm.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include <gnuradio/block.h>
#include <gnuradio/fft/fft.h>
#include <gnuradio/gr_complex.h>
#include <algorithm>
#include <cassert>
#include <fstream>
#include <functional>
#include <memory>
#include <string>
#include <utility>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [pcps_quicksync_acquisition_cc](#)

This class implements a Parallel Code Phase Search Acquisition with the implementation of the Sparse QuickSync Algorithm.

Typedefs

- using **pcps_quicksync_acquisition_cc_sptr** = boost::shared_ptr< [pcps_quicksync_acquisition_cc](#) >

Functions

- `pcps_quicksync_acquisition_cc_sptr pcps_quicksync_make_acquisition_cc (uint32_t folding_factor, uint32_t sampled_ms, uint32_t max_dwells, uint32_t doppler_max, int64_t fs_in, int32_t samples_per_ms, int32_t samples_per_code, bool bit_transition_flag, bool dump, const std::string &dump_filename)`

11.262.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition with the QuickSync Algorithm.

Acquisition strategy (Kay Borre book CFAR + threshold).

1. Compute the input signal power estimation
2. Doppler serial search loop
3. Perform folding of the incoming signal and local generated code
4. Perform the FFT-based circular convolution (parallel time search)
5. Record the maximum peak and the associated synchronization parameters
6. Compute the test statistics and compare to the threshold

7. Declare positive or negative acquisition using a message port
8. Obtain the adequate acquisition parameters by correlating the incoming signal shifted by the possible folded delays

Kay Borre book: K.Borre, D.M.Akos, N.Bertelsen, P.Rinder, and S.H.Jensen, "A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach", Birkha user, 2007. pp 81-84

Date

Jun2 2014

Author

Damian Miralles Sanchez, dmiralles2009@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.263 pcps_tong_acquisition_cc.h File Reference

This class implements a Parallel Code Phase Search Acquisition with Tong algorithm.

```
#include "channel_fsm.h"
#include "gnss_synchro.h"
#include <gnuradio/block.h>
#include <gnuradio/fft/fft.h>
#include <gnuradio/gr_complex.h>
#include <fstream>
#include <memory>
#include <string>
#include <utility>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [pcps_tong_acquisition_cc](#)

This class implements a Parallel Code Phase Search Acquisition with Tong algorithm.

Typedefs

- using [pcps_tong_acquisition_cc_sptr](#) = boost::shared_ptr< [pcps_tong_acquisition_cc](#) >

Functions

- `pcps_tong_acquisition_cc_sptr pcps_tong_make_acquisition_cc (uint32_t sampled_ms, uint32_t doppler_max, int64_t fs_in, int32_t samples_per_ms, int32_t samples_per_code, uint32_t tong_init_val, uint32_t tong_max_val, uint32_t tong_max_dwells, bool dump, const std::string &dump_filename)`

11.263.1 Detailed Description

This class implements a Parallel Code Phase Search Acquisition with Tong algorithm.

Author

Marc Molina, 2013. marc.molina.pena(at)gmail.com

Acquisition strategy (Kaplan book + CFAR threshold).

1. Compute the input signal power estimation.
2. Doppler serial search loop.
3. Perform the FFT-based circular convolution (parallel time search).
4. Compute the tests statistics for all the cells.
5. Accumulate the grid of tests statistics with the previous grids.
6. Record the maximum peak and the associated synchronization parameters.
7. Compare the maximum averaged test statistics with a threshold.
8. If the test statistics exceeds the threshold, increment the Tong counter.
9. Otherwise, decrement the Tong counter.
10. If the Tong counter is equal to a given maximum value, declare positive
11. acquisition. If the Tong counter is equal to zero, declare negative
12. acquisition. Otherwise, process the next block.

Kaplan book: D.Kaplan, J.Hegarty, "Understanding GPS. Principles and Applications", Artech House, 2006, pp 223-227

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.264 plutosdr_signal_source.h File Reference

Signal source for PlutoSDR.

```
#include "gnss_block_interface.h"
#include <gnuradio/blocks/file_sink.h>
#include <iio/pluto_source.h>
#include "concurrent_queue.h"
#include <pmt/pmt.h>
#include <cstdint>
#include <memory>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [PlutosdrSignalSource](#)

11.264.1 Detailed Description

Signal source for PlutoSDR.

Author

Rodrigo Muñoz, 2017, [rmunozl\(at\)inacap.cl](mailto:rmunozl@inacap.cl), [rodrigo.munoz\(at\)proteinlab.cl](mailto:rodrigo.munoz@proteinlab.cl)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.265 position_test_flags.h File Reference

Helper file for unit testing.

```
#include <gflags/gflags.h>
#include <limits>
#include <string>
```


Functions

- **DEFINE_string** (config_file_ptest, std::string(""), "File containing the configuration parameters for the position test.")
- **DEFINE_bool** (plot_position_test, false, "Plots results of with gnuplot")
- **DEFINE_bool** (static_scenario, true, "Compute figures of merit for static user position (DRMS, CEP, etc..)")
- **DEFINE_bool** (use_ref_motion_file, false, "Enable or disable the use of a reference file containing the true receiver position, velocity and acceleration.")
- **DEFINE_int32** (ref_motion_file_type, 1, "Type of reference motion file: 1- Spirent CSV motion file")
- **DEFINE_string** (ref_motion_filename, std::string("motion.csv"), "Path and filename for the reference motion file")
- **DEFINE_string** (pvt_solver_dump_filename, std::string("PVT.dat"), "Path and filename for the PVT solver binary dump file")
- **DEFINE_double** (static_2D_error_m, 2.0, "Static scenario 2D (East, North) positioning error threshold [meters]")
- **DEFINE_double** (static_3D_error_m, 5.0, "Static scenario 3D (East, North, Up) positioning error threshold [meters]")
- **DEFINE_double** (accuracy_CEP, 2.0, "Static scenario 2D (East, North) accuracy Circular Error Position (CEP) threshold [meters]")
- **DEFINE_double** (precision_SEP, 10.0, "Static scenario 3D (East, North, Up) precision Spherical Error Position (SEP) threshold [meters]")
- **DEFINE_double** (dynamic_3D_position_RMSE, 10.0, "Dynamic scenario 3D (ECEF) accuracy RMS↵E threshold [meters]")
- **DEFINE_double** (dynamic_3D_velocity_RMSE, 5.0, "Dynamic scenario 3D (ECEF) velocity accuracy RM↵SE threshold [meters/second]")
- **DEFINE_bool** (enable_carrier_smoothing, false, "Activates carrier smoothing of pseudoranges")

11.265.1 Detailed Description

Helper file for unit testing.

Author

Javier Arribas, 2018. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.266 pulse_blanking_cc.h File Reference

Implements a pulse blanking algorithm.

```
#include <boost/shared_ptr.hpp>
#include <gnuradio/block.h>
#include <volk_gnssssdr/volk_gnssssdr_alloc.h>
#include <cstdint>
```

Classes

- class [pulse_blanking_cc](#)

Typedefs

- using **pulse_blanking_cc_sptr** = boost::shared_ptr< [pulse_blanking_cc](#) >

Functions

- [pulse_blanking_cc_sptr](#) **make_pulse_blanking_cc** (float pfa, int32_t length_, int32_t n_segments_est, int32_t n_segments_reset)

11.266.1 Detailed Description

Implements a pulse blanking algorithm.

Author

Javier Arribas ([jarribas\(at\)cttc.es](mailto:jarribas(at)cttc.es))

Antonio Ramos ([antonio.ramosdet\(at\)gmail.com](mailto:antonio.ramosdet(at)gmail.com))

Copyright (C) 2010-2019 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.267 pulse_blanking_filter.h File Reference

Instantiates the GNSS-SDR pulse blanking filter.

```
#include "gnss_block_interface.h"
#include "pulse_blanking_cc.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/filter/freq_xlating_fir_filter_ccf.h>
#include <string>
```

Classes

- class [PulseBlankingFilter](#)

11.267.1 Detailed Description

Instantiates the GNSS-SDR pulse blanking filter.

Author

Javier Arribas 2017 Antonio Ramos 2017

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.268 pvt_conf.h File Reference

Class that contains all the configuration parameters for the PVT block.

```
#include <cstdint>
#include <map>
#include <string>
```

Classes

- class [Pvt_Conf](#)

11.268.1 Detailed Description

Class that contains all the configuration parameters for the PVT block.

Author

Carles Fernandez, 2018. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.269 pvt_interface.h File Reference

This class represents an interface to a PVT block.

```
#include "galileo_almanac.h"
#include "galileo_ephemeris.h"
#include "gnss_block_interface.h"
#include "gps_almanac.h"
#include "gps_ephemeris.h"
#include <map>
```

Classes

- class [PvtInterface](#)

This class represents an interface to a PVT block.

11.269.1 Detailed Description

This class represents an interface to a PVT block.

Author

Javier Arribas, 2011. jarribas(at)cttc.es

Abstract class for PVT solvers. Since all its methods are virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.270 pvt_solution.h File Reference

Interface of a base class for a PVT solution.

```
#include <boost/date_time/posix_time/posix_time.hpp>
#include <array>
#include <deque>
```

Classes

- class [Pvt_Solution](#)

Base class for a PVT solution.

11.270.1 Detailed Description

Interface of a base class for a PVT solution.

Author

Carles Fernandez-Prades, 2015. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.271 raw_array_signal_source.h File Reference

CTTC Experimental GNSS 8 channels array signal source.

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/hier_block2.h>
#include <pmt/pmt.h>
#include <stdint>
#include <memory>
#include <string>
```

Classes

- class [RawArraySignalSource](#)

This class reads samples from a GN3S USB dongle, a RF front-end signal sampler.

11.271.1 Detailed Description

CTTC Experimental GNSS 8 channels array signal source.

Author

Javier Arribas, jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.272 rinex_printer.h File Reference

Interface of a RINEX 2.11 / 3.01 printer See <ftp://igs.org/pub/data/format/rinex301.pdf>.

```
#include <boost/date_time/posix_time/posix_time.hpp>
#include <stdint>
#include <stdlib>
#include <fstream>
#include <iomanip>
#include <map>
#include <sstream>
#include <string>
```

Classes

- class [Rinex_Printer](#)

Class that handles the generation of Receiver INdependent EXchange format (RINEX) files.

Functions

- `std::string asString` (long double x, `std::string::size_type` precision)
- `int64_t asInt` (const `std::string` &s)

11.272.1 Detailed Description

Interface of a RINEX 2.11 / 3.01 printer See <ftp://igs.org/pub/data/format/rinex301.pdf>.

Receiver Independent EXchange Format (RINEX): The first proposal for the Receiver Independent Exchange Format RINEX was developed by the Astronomical Institute of the University of Berne for the easy exchange of the GPS data to be collected during the large European GPS campaign EUREF 89, which involved more than 60 GPS receivers of 4 different manufacturers. The governing aspect during the development was the fact that most geodetic processing software for GPS data use a well-defined set of observables: 1) The carrier-phase measurement at one or both carriers (actually being a measurement on the beat frequency between the received carrier of the satellite signal and a receiver-generated reference frequency). 2) The pseudorange (code) measurement, equivalent to the difference of the time of reception (expressed in the time frame of the receiver) and the time of transmission (expressed in the time frame of the satellite) of a distinct satellite signal. 3) The observation time being the reading of the receiver clock at the instant of validity of the carrier-phase and/or the code measurements. Note: A collection of the formats currently used by the IGS can be found here: <https://kb.igs.org/hc/en-us/articles/201096516-IGS-Formats>

Carles Fernandez Prades, 2011. [cfernandez\(at\)cttc.es](mailto:cfernandez@cttc.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.273 rtcm.h File Reference

Interface for the RTCM 3.2 Standard.

```
#include "concurrent_queue.h"
#include "galileo_ephemeris.h"
#include "glonass_gnav_ephemeris.h"
#include "glonass_gnav_utc_model.h"
#include "gnss_synchro.h"
#include "gps_cnav_ephemeris.h"
#include "gps_ephemeris.h"
#include <boost/asio.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <glog/logging.h>
#include <algorithm>
#include <array>
#include <bitset>
#include <cstdint>
#include <cstdint>
#include <cstring>
#include <deque>
#include <list>
#include <map>
#include <memory>
#include <set>
#include <string>
#include <thread>
#include <utility>
#include <vector>
```

Classes

- class [Rtcm](#)

This class implements the generation and reading of some Message Types defined in the RTCM 3.2 Standard, plus some utilities to handle messages.

Typedefs

- using **b_io_context** = boost::asio::io_service

11.273.1 Detailed Description

Interface for the RTCM 3.2 Standard.

Author

Carles Fernandez-Prades, 2015. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.274 rtcm_printer.h File Reference

Interface of a RTCM 3.2 printer for GNSS-SDR This class provides a implementation of a subset of the RTCM Standard 10403.2 for Differential GNSS Services.

```
#include <stdint>
#include <fstream>
#include <map>
#include <memory>
#include <string>
```

Classes

- class [Rtcm_Printer](#)

This class provides a implementation of a subset of the RTCM Standard 10403.2 messages.

11.274.1 Detailed Description

Interface of a RTCM 3.2 printer for GNSS-SDR This class provides a implementation of a subset of the RTCM Standard 10403.2 for Differential GNSS Services.

Author

Carles Fernandez-Prades, 2014. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.275 rtklib.h File Reference

main header file for the rtklib library

```
#include "MATH_CONSTANTS.h"
#include "gnss_frequencies.h"
#include "gnss_obs_codes.h"
#include <cctype>
#include <cmath>
#include <cstdarg>
#include <stdint>
#include <stdlib>
#include <netinet/in.h>
#include <pthread.h>
#include <string>
```


Classes

- struct [gtime_t](#)
- struct [obsd_t](#)
- struct [obs_t](#)
- struct [erpd_t](#)
- struct [erp_t](#)
- struct [pcv_t](#)
- struct [pcvs_t](#)
- struct [alm_t](#)
- struct [eph_t](#)
- struct [geph_t](#)
- struct [peph_t](#)
- struct [pclk_t](#)
- struct [seph_t](#)
- struct [tled_t](#)
- struct [tle_t](#)
- struct [tec_t](#)
- struct [fcbd_t](#)
- struct [sbsmsg_t](#)
- struct [sbs_t](#)
- struct [sbsfcrr_t](#)
- struct [sbslcorr_t](#)
- struct [sbssatp_t](#)
- struct [sbssat_t](#)
- struct [sbsigp_t](#)
- struct [sbsigpband_t](#)
- struct [sbsion_t](#)
- struct [dgps_t](#)
- struct [ssr_t](#)
- struct [lexmsg_t](#)
- struct [lex_t](#)
- struct [lexeph_t](#)
- struct [lexion_t](#)
- struct [stec_t](#)
- struct [trop_t](#)
- struct [pppcorr_t](#)
- struct [nav_t](#)
- struct [sta_t](#)
- struct [sol_t](#)
- struct [solbuf_t](#)
- struct [solstat_t](#)
- struct [solstatbuf_t](#)
- struct [rtcm_t](#)
- struct [url_t](#)
- struct [opt_t](#)
- struct [exterr_t](#)
- struct [snrmask_t](#)
- struct [prcopt_t](#)
- struct [solopt_t](#)
- struct [ssat_t](#)
- struct [ambc_t](#)
- struct [rtk_t](#)
- struct [half_cyc_tag](#)
- struct [stream_t](#)

- struct [serial_t](#)
- struct [file_t](#)
- struct [tcp_t](#)
- struct [tcpsvr_t](#)
- struct [tcpcli_t](#)
- struct [ntrip_t](#)
- struct [ftp_t](#)
- struct [raw_t](#)
- struct [rtksvr_t](#)
- struct [msm_h_t](#)

Macros

- `#define dev_t int`
- `#define socket_t int`
- `#define closesocket close`
- `#define lock_t pthread_mutex_t`
- `#define initlock(f) pthread_mutex_init(f, NULL)`
- `#define rtk_lock(f) pthread_mutex_lock(f)`
- `#define rtk_unlock(f) pthread_mutex_unlock(f)`
- `#define VER_RTKLIB "2.4.2"`
- `#define NTRIP_AGENT "RTKLIB/" VER_RTKLIB`
- `#define NTRIP_CLI_PORT 2101 /* default ntrip-client connection port */`
- `#define NTRIP_SVR_PORT 80 /* default ntrip-server connection port */`
- `#define NTRIP_MAXRSP 32768 /* max size of ntrip response */`
- `#define NTRIP_MAXSTR 256 /* max length of mountpoint string */`
- `#define NTRIP_RSP_OK_CLI "ICY 200 OK\r\n" /* ntrip response: client */`
- `#define NTRIP_RSP_OK_SVR "OK\r\n" /* ntrip response: server */`
- `#define NTRIP_RSP_SRCTBL "SOURCETABLE 200 OK\r\n" /* ntrip response: source table */`
- `#define NTRIP_RSP_TBLEND "ENDSOURCETABLE"`
- `#define NTRIP_RSP_HTTP "HTTP/" /* ntrip response: http */`
- `#define NTRIP_RSP_ERROR "ERROR" /* ntrip response: error */`
- `#define FTP_CMD "wget" /* ftp/http command */`
- `#define ENAGLO`
- `#define ENABDS`
- `#define STR_MODE_R 0x1 /* stream mode: read */`
- `#define STR_MODE_W 0x2 /* stream mode: write */`
- `#define STR_MODE_RW 0x3 /* stream mode: read/write */`
- `#define STR_NONE 0 /* stream type: none */`
- `#define STR_SERIAL 1 /* stream type: serial */`
- `#define STR_FILE 2 /* stream type: file */`
- `#define STR_TCPSVR 3 /* stream type: TCP server */`
- `#define STR_TCPCLI 4 /* stream type: TCP client */`
- `#define STR_UDP 5 /* stream type: UDP stream */`
- `#define STR_NTRIPSVR 6 /* stream type: NTRIP server */`
- `#define STR_NTRIPCLI 7 /* stream type: NTRIP client */`
- `#define STR_FTP 8 /* stream type: ftp */`
- `#define STR_HTTP 9 /* stream type: http */`
- `#define NP_PPP(opt) ((opt)->dynamics ? 9 : 3) /* number of pos solution */`
- `#define IC_PPP(s, opt) (NP_PPP(opt) + (s)) /* state index of clocks (s=0:gps,1:glo) */`
- `#define IT_PPP(opt) (IC_PPP(0, opt) + NSYS) /* state index of tropos */`
- `#define NR_PPP(opt) (IT_PPP(opt) + ((opt)->tropopt < TROPOPT_EST ? 0 : ((opt)->tropopt == TROPOPT_EST ? 1 : 3))) /* number of solutions */`
- `#define IB_PPP(s, opt) (NR_PPP(opt) + (s)-1) /* state index of phase bias */`

- `#define NX_PPP(opt) (IB_PPP(MAXSAT, opt) + 1) /* number of estimated states */`
- `#define NF_RTK(opt) ((opt)->ionoopt == IONOOPT_IFLC ? 1 : (opt)->nf)`
- `#define NP_RTK(opt) ((opt)->dynamics == 0 ? 3 : 9)`
- `#define NI_RTK(opt) ((opt)->ionoopt != IONOOPT_EST ? 0 : MAXSAT)`
- `#define NT_RTK(opt) ((opt)->tropopt < TROPOPT_EST ? 0 : ((opt)->tropopt < TROPOPT_ESTG ? 2 : 6))`
- `#define NL_RTK(opt) ((opt)->glomodear != 2 ? 0 : NFREQGLO)`
- `#define NB_RTK(opt) ((opt)->mode <= PMODE_DGPS ? 0 : MAXSAT * NF_RTK(opt))`
- `#define NR_RTK(opt) (NP_RTK(opt) + NI_RTK(opt) + NT_RTK(opt) + NL_RTK(opt))`
- `#define NX_RTK(opt) (NR_RTK(opt) + NB_RTK(opt))`

Typedefs

- using `fatalfunc_t` = `void(const char *)`
fatal callback function type
- typedef struct `half_cyc_tag` `half_cyc_t`

Variables

- const int `TINTACT` = 200
period for stream active (ms)
- const int `SERIBUFFSIZE` = 4096
serial buffer size (bytes)
- const int `TIMETAGH_LEN` = 64
time tag file header length
- const int `MAXCLI` = 32
max client connection for tcp svr
- const int `MAXSTATMSG` = 32
max length of status message
- const int `FTP_TIMEOUT` = 30
ftp/http timeout (s)
- const int `MAXRAWLEN` = 4096
max length of receiver raw message
- const int `MAXSOLBUF` = 256
max number of solution buffer
- const int `MAXSBSMSG` = 32
max number of SBAS msg in RTK server
- const int `MAXOBSBUF` = 128
max number of observation data buffer
- const int `FILEPATHSEP` = '/'
- const double `RE_WGS84` = 6378137.0
earth semimajor axis (WGS84) (m)
- const double `FE_WGS84` = (1.0 / 298.257223563)
earth flattening (WGS84)
- const double `HION` = 350000.0
ionosphere height (m)
- const double `PRN_HWBIAIS` = 1e-6
process noise of h/w bias (m/MHz/sqrt(s))
- const double `INT_SWAP_STAT` = 86400.0
swap interval of solution status file (s)

- const double `INT_SWAP_TRAC` = 86400.0
swap interval of trace file (s)
- const unsigned int `POLYCRC32` = 0xEDB88320u
CRC32 polynomial.
- const unsigned int `POLYCRC24Q` = 0x1864CFBu
CRC24Q polynomial.
- const int `PMODE_SINGLE` = 0
positioning mode: single
- const int `PMODE_DGPS` = 1
positioning mode: DGPS/DGNSS
- const int `PMODE_KINEMA` = 2
positioning mode: kinematic
- const int `PMODE_STATIC` = 3
positioning mode: static
- const int `PMODE_MOVEB` = 4
positioning mode: moving-base
- const int `PMODE_FIXED` = 5
positioning mode: fixed
- const int `PMODE_PPP_KINEMA` = 6
positioning mode: PPP-kinematic
- const int `PMODE_PPP_STATIC` = 7
positioning mode: PPP-static
- const int `PMODE_PPP_FIXED` = 8
positioning mode: PPP-fixed
- const int `SOLF_LLH` = 0
solution format: lat/lon/height
- const int `SOLF_XYZ` = 1
solution format: x/y/z-ecef
- const int `SOLF_ENU` = 2
solution format: e/n/u-baseline
- const int `SOLF_NMEA` = 3
solution format: NMEA-183
- const int `SOLF_STAT` = 4
solution format: solution status
- const int `SOLF_GSIF` = 5
solution format: GSI F1/F2
- const int `SOLQ_NONE` = 0
solution status: no solution
- const int `SOLQ_FIX` = 1
solution status: fix
- const int `SOLQ_FLOAT` = 2
solution status: float
- const int `SOLQ_SBAS` = 3
solution status: SBAS
- const int `SOLQ_DGPS` = 4
solution status: DGPS/DGNSS
- const int `SOLQ_SINGLE` = 5
solution status: single
- const int `SOLQ_PPP` = 6
solution status: PPP
- const int `SOLQ_DR` = 7

- solution status: dead reckoning*
- const int **MAXSOLQ** = 7
max number of solution status
- const int **TIMES_GPST** = 0
time system: gps time
- const int **TIMES_UTC** = 1
time system: utc
- const int **TIMES_JST** = 2
time system: jst
- const double **ERR_SAAS** = 0.3
saastamoinen model error std (m)
- const double **ERR_BRDCI** = 0.5
broadcast iono model error factor
- const double **ERR_CBIAS** = 0.3
code bias error std (m)
- const double **REL_HUMI** = 0.7
relative humidity for saastamoinen model
- const double **GAP_RESION** = 120
default gap to reset ionos parameters (ep)
- const int **MAXFREQ** = 7
max NFREQ
- const int **MAXLEAPS** = 64
max number of leap seconds table
- const double **DTTOL** = 0.005
tolerance of time difference (s)
- const int **NFREQ** = 3
number of carrier frequencies
- const int **NFREQGLO** = 2
number of carrier frequencies of GLONASS
- const int **NEXOBS** = 0
number of extended obs codes
- const int **MAXANT** = 64
max length of station name/antenna type
- const int **MINPRNGPS** = 1
min satellite PRN number of GPS
- const int **MAXPRNGPS** = 32
max satellite PRN number of GPS
- const int **NSATGPS** = (**MAXPRNGPS** - **MINPRNGPS** + 1)
number of GPS satellites
- const int **NSYSGPS** = 1
- const int **SYS_NONE** = 0x00
navigation system: none
- const int **SYS_GPS** = 0x01
navigation system: GPS
- const int **SYS_SBS** = 0x02
navigation system: SBAS
- const int **SYS_GLO** = 0x04
navigation system: GLONASS
- const int **SYS_GAL** = 0x08
navigation system: Galileo
- const int **SYS_QZS** = 0x10

```

    navigation system: QZSS
• const int SYS_BDS = 0x20
    navigation system: BeiDou
• const int SYS_IRN = 0x40
    navigation system: IRNS
• const int SYS_LEO = 0x80
    navigation system: LEO
• const int SYS_ALL = 0xFF
    navigation system: all
• const int MINPRNGLO = 1
    min satellite slot number of GLONASS
• const int MAXPRNGLO = 27
    max satellite slot number of GLONASS
• const int NSATGLO = (MAXPRNGLO - MINPRNGLO + 1)
    number of GLONASS satellites
• const int NSYSGLO = 1
• const int MINPRNGAL = 1
    min satellite PRN number of Galileo
• const int MAXPRNGAL = 36
    max satellite PRN number of Galileo
• const int NSATGAL = (MAXPRNGAL - MINPRNGAL + 1)
    number of Galileo satellites
• const int NSYSGAL = 1
• const int MINPRNQZS = 0
• const int MAXPRNQZS = 0
• const int MINPRNQZS_S = 0
• const int MAXPRNQZS_S = 0
• const int NSATQZS = 0
• const int NSYSQZS = 0
• const int MINPRNBDS = 1
    min satellite sat number of BeiDou
• const int MAXPRNBDS = 37
    max satellite sat number of BeiDou
• const int NSATBDS = (MAXPRNBDS - MINPRNBDS + 1)
    number of BeiDou satellites
• const int NSYSBDS = 1
• const int MINPRNIRN = 0
• const int MAXPRNIRN = 0
• const int NSATIRN = 0
• const int NSYSIRN = 0
• const int MINPRNLEO = 0
• const int MAXPRNLEO = 0
• const int NSATLEO = 0
• const int NSYSLEO = 0
• const int NSYS = (NSYSGPS + NSYSGLO + NSYSGAL + NSYSQZS + NSYSBDS + NSYSIRN + NSYSLEO)
    number of systems
• const int MINPRNSBS = 120
    min satellite PRN number of SBAS
• const int MAXPRNSBS = 142
    max satellite PRN number of SBAS
• const int NSATSBS = (MAXPRNSBS - MINPRNSBS + 1)
    number of SBAS satellites

```

- const int **MAXSAT** = (NSATGPS + NSATGLO + NSATGAL + NSATQZS + NSATBDS + NSATIRN + NSATSBS + NSATLEO)
- const int **MAXSTA** = 255
- const int **MAXOBS** = 64
max number of obs in an epoch
- const int **MAXRCV** = 64
max receiver number (1 to MAXRCV)
- const int **MAXOBS** = 64
max number of obs type in RINEX
- const double **MAXD** = 7200.0
max time difference to GPS Toe (s)
- const double **MAXD_QZS** = 7200.0
max time difference to QZSS Toe (s)
- const double **MAXD_GAL** = 10800.0
max time difference to Galileo Toe (s)
- const double **MAXD_BDS** = 21600.0
max time difference to BeiDou Toe (s)
- const double **MAXD_GLO** = 1800.0
max time difference to GLONASS Toe (s)
- const double **MAXD_SBS** = 360.0
max time difference to SBAS Toe (s)
- const double **MAXD_S** = 86400.0
max time difference to ephemeris toe (s) for other
- const double **MAXGDOP** = 300.0
max GDOP
- const int **MAXSBSURA** = 8
max URA of SBAS satellite
- const int **MAXBAND** = 10
max SBAS band of IGP
- const int **MAXNIGP** = 201
max number of IGP in SBAS band
- const int **MAXNGEO** = 4
max number of GEO satellites
- const int **MAXSOLMSG** = 8191
max length of solution message
- const int **MAXERRMSG** = 4096
max length of error/warning message
- const int **IONOOPT_OFF** = 0
ionosphere option: correction off
- const int **IONOOPT_BRDC** = 1
ionosphere option: broadcast model
- const int **IONOOPT_SBAS** = 2
ionosphere option: SBAS model
- const int **IONOOPT_IFLC** = 3
ionosphere option: L1/L2 or L1/L5 iono-free LC
- const int **IONOOPT_EST** = 4
ionosphere option: estimation
- const int **IONOOPT_TEC** = 5
ionosphere option: IONEX TEC model
- const int **IONOOPT_QZS** = 6
ionosphere option: QZSS broadcast model

- const int [IONOOPT_LEX](#) = 7
ionosphere option: QZSS LEX ionosphere
- const int [IONOOPT_STEC](#) = 8
ionosphere option: SLANT TEC model
- const int [TROPOPT_OFF](#) = 0
troposphere option: correction off
- const int [TROPOPT_SAAS](#) = 1
troposphere option: Saastamoinen model
- const int [TROPOPT_SBAS](#) = 2
troposphere option: SBAS model
- const int [TROPOPT_EST](#) = 3
troposphere option: ZTD estimation
- const int [TROPOPT_ESTG](#) = 4
troposphere option: ZTD+grad estimation
- const int [TROPOPT_COR](#) = 5
troposphere option: ZTD correction
- const int [TROPOPT_CORG](#) = 6
troposphere option: ZTD+grad correction
- const int [EPHOPT_BRDC](#) = 0
ephemeris option: broadcast ephemeris
- const int [EPHOPT_PREC](#) = 1
ephemeris option: precise ephemeris
- const int [EPHOPT_SBAS](#) = 2
ephemeris option: broadcast + SBAS
- const int [EPHOPT_SSRAPC](#) = 3
ephemeris option: broadcast + SSR_APC
- const int [EPHOPT_SSRCOM](#) = 4
ephemeris option: broadcast + SSR_COM
- const int [EPHOPT_LEX](#) = 5
ephemeris option: QZSS LEX ephemeris
- const double [EFACT_GPS](#) = 1.0
error factor: GPS
- const double [EFACT_GLO](#) = 1.5
error factor: GLONASS
- const double [EFACT_GAL](#) = 1.0
error factor: Galileo
- const double [EFACT_QZS](#) = 1.0
error factor: QZSS
- const double [EFACT_BDS](#) = 1.0
error factor: BeiDou
- const double [EFACT_IRN](#) = 1.5
error factor: IRNSS
- const double [EFACT_SBS](#) = 3.0
error factor: SBAS
- const int [MAXEXFILE](#) = 1024
max number of expanded files
- const double [MAXSBSAGEF](#) = 30.0
max age of SBAS fast correction (s)
- const double [MAXSBSAGEL](#) = 1800.0
max age of SBAS long term corr (s)
- const int [ARMODE_OFF](#) = 0

- AR mode: off.*
- const int **ARMODE_CONT** = 1
- AR mode: continuous.*
- const int **ARMODE_INST** = 2
- AR mode: instantaneous.*
- const int **ARMODE_FIXHOLD** = 3
- AR mode: fix and hold.*
- const int **ARMODE_PPPAR** = 4
- AR mode: PPP-AR.*
- const int **ARMODE_PPPAR_ILS** = 5
- AR mode: AR mode: PPP-AR ILS.*
- const int **ARMODE_WLNL** = 6
- const int **ARMODE_TCAR** = 7
- const int **POSOPT_RINEX** = 3
- pos option: rinex header pos*
- const int **MAXSTRPATH** = 1024
- max length of stream path*
- const int **MAXSTRMSG** = 1024
- max length of stream message*
- const double **CHISQR** [100]
- const double **LAM_CARR** [**MAXFREQ**]
- const int **STRFMT_RTCM2** = 0
- const int **STRFMT_RTCM3** = 1
- const int **STRFMT_SP3** = 16
- const int **STRFMT_RNXCLK** = 17
- const int **STRFMT_SBAS** = 18
- const int **STRFMT_NMEA** = 19
- const int **MAXSTRRTK** = 8

11.275.1 Detailed Description

main header file for the rtklib library

Authors

- 2007-2013, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

11.275.2 Typedef Documentation

11.275.2.1 fatalfunc_t

```
using fatalfunc_t = void(const char *)
```

fatal callback function type

Definition at line 321 of file rtklib.h.

11.275.3 Variable Documentation

11.275.3.1 ARMODE_CONT

```
const int ARMODE_CONT = 1
```

AR mode: continuous.

Definition at line 308 of file rtklib.h.

11.275.3.2 ARMODE_FIXHOLD

```
const int ARMODE_FIXHOLD = 3
```

AR mode: fix and hold.

Definition at line 310 of file rtklib.h.

11.275.3.3 ARMODE_INST

```
const int ARMODE_INST = 2
```

AR mode: instantaneous.

Definition at line 309 of file rtklib.h.

11.275.3.4 ARMODE_OFF

```
const int ARMODE_OFF = 0
```

AR mode: off.

Definition at line 307 of file rtklib.h.

11.275.3.5 ARMODE_PPPAR

```
const int ARMODE_PPPAR = 4
```

AR mode: PPP-AR.

Definition at line 311 of file rtklib.h.

11.275.3.6 ARMODE_PPPAR_ILS

```
const int ARMODE_PPPAR_ILS = 5
```

AR mode: AR mode: PPP-AR ILS.

Definition at line 312 of file rtklib.h.

11.275.3.7 CHISQR

```
const double CHISQR[100]
```

Initial value:

```
= {  
    10.8, 13.8, 16.3, 18.5, 20.5, 22.5, 24.3, 26.1, 27.9, 29.6,  
    31.3, 32.9, 34.5, 36.1, 37.7, 39.3, 40.8, 42.3, 43.8, 45.3,  
    46.8, 48.3, 49.7, 51.2, 52.6, 54.1, 55.5, 56.9, 58.3, 59.7,  
    61.1, 62.5, 63.9, 65.2, 66.6, 68.0, 69.3, 70.7, 72.1, 73.4,  
    74.7, 76.0, 77.3, 78.6, 80.0, 81.3, 82.6, 84.0, 85.4, 86.7,  
    88.0, 89.3, 90.6, 91.9, 93.3, 94.7, 96.0, 97.4, 98.7, 100,  
    101, 102, 103, 104, 105, 107, 108, 109, 110, 112,  
    113, 114, 115, 116, 118, 119, 120, 122, 123, 125,  
    126, 127, 128, 129, 131, 132, 133, 134, 135, 137,  
    138, 139, 140, 142, 143, 144, 145, 147, 148, 149}
```

Definition at line 1284 of file rtklib.h.

11.275.3.8 DTTOL

```
const double DTTOL = 0.005
```

tolerance of time difference (s)

Definition at line 140 of file rtklib.h.

11.275.3.9 EFACT_BDS

```
const double EFACT_BDS = 1.0
```

error factor: BeiDou

Definition at line 299 of file rtklib.h.

11.275.3.10 EFACT_GAL

```
const double EFACT_GAL = 1.0
```

error factor: Galileo

Definition at line 297 of file rtklib.h.

11.275.3.11 EFACT_GLO

```
const double EFACT_GLO = 1.5
```

error factor: GLONASS

Definition at line 296 of file rtklib.h.

11.275.3.12 EFACT_GPS

```
const double EFACT_GPS = 1.0
```

error factor: GPS

Definition at line 295 of file rtklib.h.

11.275.3.13 EFACT_IRN

```
const double EFACT_IRN = 1.5
```

error factor: IRNSS

Definition at line 300 of file rtklib.h.

11.275.3.14 EFACT_QZS

```
const double EFACT_QZS = 1.0
```

error factor: QZSS

Definition at line 298 of file rtklib.h.

11.275.3.15 EFACT_SBS

```
const double EFACT_SBS = 3.0
```

error factor: SBAS

Definition at line 301 of file rtklib.h.

11.275.3.16 EPHOPT_BRDC

```
const int EPHOPT_BRDC = 0
```

ephemeris option: broadcast ephemeris

Definition at line 288 of file rtklib.h.

11.275.3.17 EPHOPT_LEX

```
const int EPHOPT_LEX = 5
```

ephemeris option: QZSS LEX ephemeris

Definition at line 293 of file rtklib.h.

11.275.3.18 EPHOPT_PREC

```
const int EPHOPT_PREC = 1
```

ephemeris option: precise ephemeris

Definition at line 289 of file rtklib.h.

11.275.3.19 EPHOPT_SBAS

```
const int EPHOPT_SBAS = 2
```

ephemeris option: broadcast + SBAS

Definition at line 290 of file rtklib.h.

11.275.3.20 EPHOPT_SSRAPC

```
const int EPHOPT_SSRAPC = 3
```

ephemeris option: broadcast + SSR_APC

Definition at line 291 of file rtklib.h.

11.275.3.21 EPHOPT_SSRCOM

```
const int EPHOPT_SSRCOM = 4
```

ephemeris option: broadcast + SSR_COM

Definition at line 292 of file rtklib.h.

11.275.3.22 ERR_BRDCI

```
const double ERR_BRDCI = 0.5
```

broadcast iono model error factor

Definition at line 132 of file rtklib.h.

11.275.3.23 ERR_CBIAS

```
const double ERR_CBIAS = 0.3
```

code bias error std (m)

Definition at line 133 of file rtklib.h.

11.275.3.24 ERR_SAAS

```
const double ERR_SAAS = 0.3
```

saastamoinen model error std (m)

Definition at line 131 of file rtklib.h.

11.275.3.25 FE_WGS84

```
const double FE_WGS84 = (1.0 / 298.257223563)
```

earth flattening (WGS84)

Definition at line 88 of file rtklib.h.

11.275.3.26 FTP_TIMEOUT

```
const int FTP_TIMEOUT = 30
```

ftp/http timeout (s)

Definition at line 80 of file rtklib.h.

11.275.3.27 GAP_RESION

```
const double GAP_RESION = 120
```

default gap to reset ionos parameters (ep)

Definition at line 135 of file rtklib.h.

11.275.3.28 HION

```
const double HION = 350000.0
```

ionosphere height (m)

Definition at line 90 of file rtklib.h.

11.275.3.29 INT_SWAP_STAT

```
const double INT_SWAP_STAT = 86400.0
```

swap interval of solution status file (s)

Definition at line 93 of file rtklib.h.

11.275.3.30 INT_SWAP_TRAC

```
const double INT_SWAP_TRAC = 86400.0
```

swap interval of trace file (s)

Definition at line 94 of file rtklib.h.

11.275.3.31 IONOOPT_BRDC

```
const int IONOOPT_BRDC = 1
```

ionosphere option: broadcast model

Definition at line 270 of file rtklib.h.

11.275.3.32 IONOOPT_EST

```
const int IONOOPT_EST = 4
```

ionosphere option: estimation

Definition at line 273 of file rtklib.h.

11.275.3.33 IONOOPT_IFLC

```
const int IONOOPT_IFLC = 3
```

ionosphere option: L1/L2 or L1/L5 iono-free LC

Definition at line 272 of file rtklib.h.

11.275.3.34 IONOOPT_LEX

```
const int IONOOPT_LEX = 7
```

ionosphere option: QZSS LEX ionosphere

Definition at line 276 of file rtklib.h.

11.275.3.35 IONOOPT_OFF

```
const int IONOOPT_OFF = 0
```

ionosphere option: correction off

Definition at line 269 of file rtklib.h.

11.275.3.36 IONOOPT_QZS

```
const int IONOOPT_QZS = 6
```

ionosphere option: QZSS broadcast model

Definition at line 275 of file rtklib.h.

11.275.3.37 IONOOPT_SBAS

```
const int IONOOPT_SBAS = 2
```

ionosphere option: SBAS model

Definition at line 271 of file rtklib.h.

11.275.3.38 IONOOPT_STEC

```
const int IONOOPT_STEC = 8
```

ionosphere option: SLANT TEC model

Definition at line 277 of file rtklib.h.

11.275.3.39 IONOOPT_TEC

```
const int IONOOPT_TEC = 5
```

ionosphere option: IONEX TEC model

Definition at line 274 of file rtklib.h.

11.275.3.40 LAM_CARR

```
const double LAM_CARR[MAXFREQ]
```

Initial value:

```
= {  
    SPEED_OF_LIGHT_M_S / FREQ1, SPEED_OF_LIGHT_M_S /  
    FREQ2, SPEED_OF_LIGHT_M_S / FREQ5,  
    SPEED_OF_LIGHT_M_S / FREQ6, SPEED_OF_LIGHT_M_S /  
    FREQ7,  
    SPEED_OF_LIGHT_M_S / FREQ8, SPEED_OF_LIGHT_M_S /  
    FREQ9}
```

Definition at line 1297 of file rtklib.h.

11.275.3.41 MAXANT

```
const int MAXANT = 64
```

max length of station name/antenna type

Definition at line 145 of file rtklib.h.

11.275.3.42 MAXBAND

```
const int MAXBAND = 10
```

max SBAS band of IGP

Definition at line 262 of file rtklib.h.

11.275.3.43 MAXCLI

```
const int MAXCLI = 32
```

max client connection for tcp svr

Definition at line 77 of file rtklib.h.

11.275.3.44 MAXDTOE

```
const double MAXDTOE = 7200.0
```

max time difference to GPS Toe (s)

Definition at line 252 of file rtklib.h.

11.275.3.45 MAXDTOE_BDS

```
const double MAXDTOE_BDS = 21600.0
```

max time difference to BeiDou Toe (s)

Definition at line 255 of file rtklib.h.

11.275.3.46 MAXDTOE_GAL

```
const double MAXDTOE_GAL = 10800.0
```

max time difference to Galileo Toe (s)

Definition at line 254 of file rtklib.h.

11.275.3.47 MAXDToe_GLO

```
const double MAXDToe_GLO = 1800.0
```

max time difference to GLONASS Toe (s)

Definition at line 256 of file rtklib.h.

11.275.3.48 MAXDToe_QZS

```
const double MAXDToe_QZS = 7200.0
```

max time difference to QZSS Toe (s)

Definition at line 253 of file rtklib.h.

11.275.3.49 MAXDToe_S

```
const double MAXDToe_S = 86400.0
```

max time difference to ephemeris toe (s) for other

Definition at line 258 of file rtklib.h.

11.275.3.50 MAXDToe_SBS

```
const double MAXDToe_SBS = 360.0
```

max time difference to SBAS Toe (s)

Definition at line 257 of file rtklib.h.

11.275.3.51 MAXERRMSG

```
const int MAXERRMSG = 4096
```

max length of error/warning message

Definition at line 267 of file rtklib.h.

11.275.3.52 MAXEXFILE

```
const int MAXEXFILE = 1024
```

max number of expanded files

Definition at line 303 of file rtklib.h.

11.275.3.53 MAXFREQ

```
const int MAXFREQ = 7
```

max NFREQ

Definition at line 137 of file rtklib.h.

11.275.3.54 MAXGDOP

```
const double MAXGDOP = 300.0
```

max GDOP

Definition at line 259 of file rtklib.h.

11.275.3.55 MAXLEAPS

```
const int MAXLEAPS = 64
```

max number of leap seconds table

Definition at line 139 of file rtklib.h.

11.275.3.56 MAXNGEO

```
const int MAXNGEO = 4
```

max number of GEO satellites

Definition at line 264 of file rtklib.h.

11.275.3.57 MAXNIGP

```
const int MAXNIGP = 201
```

max number of IGP in SBAS band

Definition at line 263 of file rtklib.h.

11.275.3.58 MAXOBS

```
const int MAXOBS = 64
```

max number of obs in an epoch

Definition at line 247 of file rtklib.h.

11.275.3.59 MAXOBSBUF

```
const int MAXOBSBUF = 128
```

max number of observation data buffer

Definition at line 84 of file rtklib.h.

11.275.3.60 MAXOBSTYPE

```
const int MAXOBSTYPE = 64
```

max number of obs type in RINEX

Definition at line 251 of file rtklib.h.

11.275.3.61 MAXPRNBDS

```
const int MAXPRNBDS = 37
```

max satellite sat number of BeiDou

Definition at line 202 of file rtklib.h.

11.275.3.62 MAXPRNGAL

```
const int MAXPRNGAL = 36
```

max satellite PRN number of Galileo

Definition at line 179 of file rtklib.h.

11.275.3.63 MAXPRNGLO

```
const int MAXPRNGLO = 27
```

max satellite slot number of GLONASS

Definition at line 167 of file rtklib.h.

11.275.3.64 MAXPRNGPS

```
const int MAXPRNGPS = 32
```

max satellite PRN number of GPS

Definition at line 148 of file rtklib.h.

11.275.3.65 MAXPRNSBS

```
const int MAXPRNSBS = 142
```

max satellite PRN number of SBAS

Definition at line 239 of file rtklib.h.

11.275.3.66 MAXRAWLEN

```
const int MAXRAWLEN = 4096
```

max length of receiver raw message

Definition at line 81 of file rtklib.h.

11.275.3.67 MAXRCV

```
const int MAXRCV = 64
```

max receiver number (1 to MAXRCV)

Definition at line 250 of file rtklib.h.

11.275.3.68 MAXSBSAGEF

```
const double MAXSBSAGEF = 30.0
```

max age of SBAS fast correction (s)

Definition at line 304 of file rtklib.h.

11.275.3.69 MAXSBSAGEL

```
const double MAXSBSAGEL = 1800.0
```

max age of SBAS long term corr (s)

Definition at line 305 of file rtklib.h.

11.275.3.70 MAXSBSMSG

```
const int MAXSBSMSG = 32
```

max number of SBAS msg in RTK server

Definition at line 83 of file rtklib.h.

11.275.3.71 MAXSBSURA

```
const int MAXSBSURA = 8
```

max URA of SBAS satellite

Definition at line 261 of file rtklib.h.

11.275.3.72 MAXSOLBUF

```
const int MAXSOLBUF = 256
```

max number of solution buffer

Definition at line 82 of file rtklib.h.

11.275.3.73 MAXSOLMSG

```
const int MAXSOLMSG = 8191
```

max length of solution message

Definition at line 266 of file rtklib.h.

11.275.3.74 MAXSOLQ

```
const int MAXSOLQ = 7
```

max number of solution status

Definition at line 124 of file rtklib.h.

11.275.3.75 MAXSTATMSG

```
const int MAXSTATMSG = 32
```

max length of status message

Definition at line 78 of file rtklib.h.

11.275.3.76 MAXSTRMSG

```
const int MAXSTRMSG = 1024
```

max length of stream message

Definition at line 319 of file rtklib.h.

11.275.3.77 MAXSTRPATH

```
const int MAXSTRPATH = 1024
```

max length of stream path

Definition at line 318 of file rtklib.h.

11.275.3.78 MINPRNBDS

```
const int MINPRNBDS = 1
```

min satellite sat number of BeiDou

Definition at line 201 of file rtklib.h.

11.275.3.79 MINPRNGAL

```
const int MINPRNGAL = 1
```

min satellite PRN number of Galileo

Definition at line 178 of file rtklib.h.

11.275.3.80 MINPRNGLO

```
const int MINPRNGLO = 1
```

min satellite slot number of GLONASS

Definition at line 166 of file rtklib.h.

11.275.3.81 MINPRNGPS

```
const int MINPRNGPS = 1
```

min satellite PRN number of GPS

Definition at line 147 of file rtklib.h.

11.275.3.82 MINPRNSBS

```
const int MINPRNSBS = 120
```

min satellite PRN number of SBAS

Definition at line 238 of file rtklib.h.

11.275.3.83 NEXOBS

```
const int NEXOBS = 0
```

number of extended obs codes

Definition at line 144 of file rtklib.h.

11.275.3.84 NFREQ

```
const int NFREQ = 3
```

number of carrier frequencies

Definition at line 142 of file rtklib.h.

11.275.3.85 NFREQGLO

```
const int NFREQGLO = 2
```

number of carrier frequencies of GLONASS

Definition at line 143 of file rtklib.h.

11.275.3.86 NSATBDS

```
const int NSATBDS = (MAXPRNBDS - MINPRNBDS + 1)
```

number of BeiDou satellites

Definition at line 203 of file rtklib.h.

11.275.3.87 NSATGAL

```
const int NSATGAL = (MAXPRNGAL - MINPRNGAL + 1)
```

number of Galileo satellites

Definition at line 180 of file rtklib.h.

11.275.3.88 NSATGLO

```
const int NSATGLO = (MAXPRNGLO - MINPRNGLO + 1)
```

number of GLONASS satellites

Definition at line 168 of file rtklib.h.

11.275.3.89 NSATGPS

```
const int NSATGPS = (MAXPRNGPS - MINPRNGPS + 1)
```

number of GPS satellites

Definition at line 149 of file rtklib.h.

11.275.3.90 NSATSBS

```
const int NSATSBS = (MAXPRNSBS - MINPRNSBS + 1)
```

number of SBAS satellites

Definition at line 240 of file rtklib.h.

11.275.3.91 NSYS

```
const int NSYS = (NSYSGPS + NSYSGLO + NSYSGAL + NSYSQZS + NSYSBDS + NSYSIRN + NSYSLEO)
```

number of systems

Definition at line 236 of file rtklib.h.

11.275.3.92 PMODE_DGPS

```
const int PMODE_DGPS = 1
```

positioning mode: DGPS/DGNSS

Definition at line 100 of file rtklib.h.

11.275.3.93 PMODE_FIXED

```
const int PMODE_FIXED = 5
```

positioning mode: fixed

Definition at line 104 of file rtklib.h.

11.275.3.94 PMODE_KINEMA

```
const int PMODE_KINEMA = 2
```

positioning mode: kinematic

Definition at line 101 of file rtklib.h.

11.275.3.95 PMODE_MOVEB

```
const int PMODE_MOVEB = 4
```

positioning mode: moving-base

Definition at line 103 of file rtklib.h.

11.275.3.96 PMODE_PPP_FIXED

```
const int PMODE_PPP_FIXED = 8
```

positioning mode: PPP-fixed

Definition at line 107 of file rtklib.h.

11.275.3.97 PMODE_PPP_KINEMA

```
const int PMODE_PPP_KINEMA = 6
```

positioning mode: PPP-kinematic

Definition at line 105 of file rtklib.h.

11.275.3.98 PMODE_PPP_STATIC

```
const int PMODE_PPP_STATIC = 7
```

positioning mode: PPP-static

Definition at line 106 of file rtklib.h.

11.275.3.99 PMODE_SINGLE

```
const int PMODE_SINGLE = 0
```

positioning mode: single

Definition at line 99 of file rtklib.h.

11.275.3.100 PMODE_STATIC

```
const int PMODE_STATIC = 3
```

positioning mode: static

Definition at line 102 of file rtklib.h.

11.275.3.101 POLYCRC24Q

```
const unsigned int POLYCRC24Q = 0x1864CFBu
```

CRC24Q polynomial.

Definition at line 97 of file rtklib.h.

11.275.3.102 POLYCRC32

```
const unsigned int POLYCRC32 = 0xEDB88320u
```

CRC32 polynomial.

Definition at line 96 of file rtklib.h.

11.275.3.103 POSOPT_RINEX

```
const int POSOPT_RINEX = 3
```

pos option: rinex header pos

Definition at line 317 of file rtklib.h.

11.275.3.104 PRN_HWBIAS

```
const double PRN_HWBIAS = 1e-6
```

process noise of h/w bias (m/MHz/sqrt(s))

Definition at line 91 of file rtklib.h.

11.275.3.105 RE_WGS84

```
const double RE_WGS84 = 6378137.0
```

earth semimajor axis (WGS84) (m)

Definition at line 87 of file rtklib.h.

11.275.3.106 REL_HUMI

```
const double REL_HUMI = 0.7
```

relative humidity for saastamoinen model

Definition at line 134 of file rtklib.h.

11.275.3.107 SERIBUFFSIZE

```
const int SERIBUFFSIZE = 4096
```

serial buffer size (bytes)

Definition at line 75 of file rtklib.h.

11.275.3.108 SOLF_ENU

```
const int SOLF_ENU = 2
```

solution format: e/n/u-baseline

Definition at line 111 of file rtklib.h.

11.275.3.109 SOLF_GSIF

```
const int SOLF_GSIF = 5
```

solution format: GSI F1/F2

Definition at line 114 of file rtklib.h.

11.275.3.110 SOLF_LLH

```
const int SOLF_LLH = 0
```

solution format: lat/lon/height

Definition at line 109 of file rtklib.h.

11.275.3.111 SOLF_NMEA

```
const int SOLF_NMEA = 3
```

solution format: NMEA-183

Definition at line 112 of file rtklib.h.

11.275.3.112 SOLF_STAT

```
const int SOLF_STAT = 4
```

solution format: solution status

Definition at line 113 of file rtklib.h.

11.275.3.113 SOLF_XYZ

```
const int SOLF_XYZ = 1
```

solution format: x/y/z-ecef

Definition at line 110 of file rtklib.h.

11.275.3.114 SOLQ_DGPS

```
const int SOLQ_DGPS = 4
```

solution status: DGPS/DGNSS

Definition at line 120 of file rtklib.h.

11.275.3.115 SOLQ_DR

```
const int SOLQ_DR = 7
```

solution status: dead reckoning

Definition at line 123 of file rtklib.h.

11.275.3.116 SOLQ_FIX

```
const int SOLQ_FIX = 1
```

solution status: fix

Definition at line 117 of file rtklib.h.

11.275.3.117 SOLQ_FLOAT

```
const int SOLQ_FLOAT = 2
```

solution status: float

Definition at line 118 of file rtklib.h.

11.275.3.118 SOLQ_NONE

```
const int SOLQ_NONE = 0
```

solution status: no solution

Definition at line 116 of file rtklib.h.

11.275.3.119 SOLQ_PPP

```
const int SOLQ_PPP = 6
```

solution status: PPP

Definition at line 122 of file rtklib.h.

11.275.3.120 SOLQ_SBAS

```
const int SOLQ_SBAS = 3
```

solution status: SBAS

Definition at line 119 of file rtklib.h.

11.275.3.121 SOLQ_SINGLE

```
const int SOLQ_SINGLE = 5
```

solution status: single

Definition at line 121 of file rtklib.h.

11.275.3.122 SYS_ALL

```
const int SYS_ALL = 0xFF
```

navigation system: all

Definition at line 161 of file rtklib.h.

11.275.3.123 SYS_BDS

```
const int SYS_BDS = 0x20
```

navigation system: BeiDou

Definition at line 158 of file rtklib.h.

11.275.3.124 SYS_GAL

```
const int SYS_GAL = 0x08
```

navigation system: Galileo

Definition at line 156 of file rtklib.h.

11.275.3.125 SYS_GLO

```
const int SYS_GLO = 0x04
```

navigation system: GLONASS

Definition at line 155 of file rtklib.h.

11.275.3.126 SYS_GPS

```
const int SYS_GPS = 0x01
```

navigation system: GPS

Definition at line 153 of file rtklib.h.

11.275.3.127 SYS_IRN

```
const int SYS_IRN = 0x40
```

navigation system: IRNS

Definition at line 159 of file rtklib.h.

11.275.3.128 SYS_LEO

```
const int SYS_LEO = 0x80
```

navigation system: LEO

Definition at line 160 of file rtklib.h.

11.275.3.129 SYS_NONE

```
const int SYS_NONE = 0x00
```

navigation system: none

Definition at line 152 of file rtklib.h.

11.275.3.130 SYS_QZS

```
const int SYS_QZS = 0x10
```

navigation system: QZSS

Definition at line 157 of file rtklib.h.

11.275.3.131 SYS_SBS

```
const int SYS_SBS = 0x02
```

navigation system: SBAS

Definition at line 154 of file rtklib.h.

11.275.3.132 TIMES_GPST

```
const int TIMES_GPST = 0
```

time system: gps time

Definition at line 126 of file rtklib.h.

11.275.3.133 TIMES_JST

```
const int TIMES_JST = 2
```

time system: jst

Definition at line 128 of file rtklib.h.

11.275.3.134 TIMES_UTC

```
const int TIMES_UTC = 1
```

time system: utc

Definition at line 127 of file rtklib.h.

11.275.3.135 TIMETAGH_LEN

```
const int TIMETAGH_LEN = 64
```

time tag file header length

Definition at line 76 of file rtklib.h.

11.275.3.136 TINTACT

```
const int TINTACT = 200
```

period for stream active (ms)

Definition at line 74 of file rtklib.h.

11.275.3.137 TROPOPT_COR

```
const int TROPOPT_COR = 5
```

troposphere option: ZTD correction

Definition at line 284 of file rtklib.h.

11.275.3.138 TROPOPT_CORG

```
const int TROPOPT_CORG = 6
```

troposphere option: ZTD+grad correction

Definition at line 285 of file rtklib.h.

11.275.3.139 TROPOPT_EST

```
const int TROPOPT_EST = 3
```

troposphere option: ZTD estimation

Definition at line 282 of file rtklib.h.

11.275.3.140 TROPOPT_ESTG

```
const int TROPOPT_ESTG = 4
```

troposphere option: ZTD+grad estimation

Definition at line 283 of file rtklib.h.

11.275.3.141 TROPOPT_OFF

```
const int TROPOPT_OFF = 0
```

troposphere option: correction off

Definition at line 279 of file rtklib.h.

11.275.3.142 TROPOPT_SAAS

```
const int TROPOPT_SAAS = 1
```

troposphere option: Saastamoinen model

Definition at line 280 of file rtklib.h.

11.275.3.143 TROPOPT_SBAS

```
const int TROPOPT_SBAS = 2
```

troposphere option: SBAS model

Definition at line 281 of file rtklib.h.

11.276 rtklib_conversions.h File Reference

GNSS-SDR to RTKLIB data structures conversion functions.

```
#include "rtklib.h"
```

Functions

- [eph_t eph_to_rtklib](#) (const [Galileo_Ephemeris](#) &gal_eph)
- [eph_t eph_to_rtklib](#) (const [Gps_Ephemeris](#) &gps_eph, bool pre_2009_file)
- [eph_t eph_to_rtklib](#) (const [Gps_CNAV_Ephemeris](#) &gps_cnav_eph)
- [eph_t eph_to_rtklib](#) (const [Beidou_Dnav_Ephemeris](#) &bei_eph)
- [alm_t alm_to_rtklib](#) (const [Gps_Almanac](#) &gps_alm)
- [alm_t alm_to_rtklib](#) (const [Galileo_Almanac](#) &gal_alm)
- [geph_t eph_to_rtklib](#) (const [Glonass_Gnav_Ephemeris](#) &glonass_gnav_eph, const [Glonass_Gnav_Utc_Model](#) &gnav_clock_model)
Transforms a [Glonass_Gnav_Ephemeris](#) to its RTKLIB counterpart.
- [obsd_t insert_obs_to_rtklib](#) ([obsd_t](#) &rtklib_obs, const [Gnss_Synchro](#) &gnss_synchro, int week, int band, bool pre_2009_file=false)

11.276.1 Detailed Description

GNSS-SDR to RTKLIB data structures conversion functions.

Author

2017, Javier Arribas

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.276.2 Function Documentation

11.276.2.1 eph_to_rtklib()

```
geph_t eph_to_rtklib (
    const Glonass_Gnav_Ephemeris & glonass_gnav_eph,
    const Glonass_Gnav_Utc_Model & gnav_clock_model )
```

Transforms a [Glonass_Gnav_Ephemeris](#) to its RTKLIB counterpart.

Parameters

<i>glonass_gnav_eph</i>	GLONASS GNAV Ephemeris structure
-------------------------	----------------------------------

Returns

Ephemeris structure for RTKLIB parsing

11.277 rtklib_ephemeris.h File Reference

satellite ephemeris and clock functions

```
#include "rtklib.h"
```

Functions

- double **var_uraeph** (int ura)
- double **var_urassr** (int ura)
- void **alm2pos** (gtime_t time, const alm_t *alm, double *rs, double *dts)
- double **eph2clk** (gtime_t time, const eph_t *eph)
- void **eph2pos** (gtime_t time, const eph_t *eph, double *rs, double *dts, double *var)
- void **deq** (const double *x, double *xdot, const double *acc)
- void **glorbit** (double t, double *x, const double *acc)
- double **geph2clk** (gtime_t time, const geph_t *geph)
- void **geph2pos** (gtime_t time, const geph_t *geph, double *rs, double *dts, double *var)
- double **seph2clk** (gtime_t time, const seph_t *seph)
- void **seph2pos** (gtime_t time, const seph_t *seph, double *rs, double *dts, double *var)
- eph_t * **seleph** (gtime_t time, int sat, int iode, const nav_t *nav)
- geph_t * **selgeph** (gtime_t time, int sat, int iode, const nav_t *nav)
- seph_t * **selseph** (gtime_t time, int sat, const nav_t *nav)
- int **ephclk** (gtime_t time, gtime_t teph, int sat, const nav_t *nav, double *dts)
- int **ephpos** (gtime_t time, gtime_t teph, int sat, const nav_t *nav, int iode, double *rs, double *dts, double *var, int *svh)
- int **satpos_sbass** (gtime_t time, gtime_t teph, int sat, const nav_t *nav, double *rs, double *dts, double *var, int *svh)
- int **satpos_ssrr** (gtime_t time, gtime_t teph, int sat, const nav_t *nav, int opt, double *rs, double *dts, double *var, int *svh)
- int **satpos** (gtime_t time, gtime_t teph, int sat, int ephopt, const nav_t *nav, double *rs, double *dts, double *var, int *svh)
- void **satposs** (gtime_t teph, const obsd_t *obs, int n, const nav_t *nav, int ephopt, double *rs, double *dts, double *var, int *svh)

11.277.1 Detailed Description

satellite ephemeris and clock functions

Authors

- 2007-2013, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

11.278 rtklib_ionex.h File Reference

ionex functions

```
#include "rtklib.h"
```

Functions

- int **getindex** (double value, const double *range)
- int **nitem** (const double *range)
- int **dataindex** (int i, int j, int k, const int *ndata)
- [tec_t](#) * **addtec** (const double *lats, const double *lons, const double *hgts, double rb, [nav_t](#) *nav)
- void **readionexdcb** (FILE *fp, double *dcb, double *rms)
- double **readionexh** (FILE *fp, double *lats, double *lons, double *hgts, double *rb, double *nexp, double *dcb, double *rms)
- int **readionexb** (FILE *fp, const double *lats, const double *lons, const double *hgts, double rb, double nexp, [nav_t](#) *nav)
- void **combtec** ([nav_t](#) *nav)
- void **readtec** (const char *file, [nav_t](#) *nav, int opt)
- int **interptec** (const [tec_t](#) *tec, int k, const double *posp, double *value, double *rms)
- int **iondelay** ([gtime_t](#) time, const [tec_t](#) *tec, const double *pos, const double *azel, int opt, double *delay, double *var)
- int **iontec** ([gtime_t](#) time, const [nav_t](#) *nav, const double *pos, const double *azel, int opt, double *delay, double *var)

Variables

- const double **VAR_NOTEC** = 30.0 * 30.0
- const double **MIN_EL** = 0.0
- const double **MIN_HGT** = -1000.0

11.278.1 Detailed Description

ionex functions

Authors

- 2007-2013, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

References: [1] S.Schear, W.Gurtner and J.Feltens, IONEX: The IONosphere Map EXchange Format Version 1, February 25, 1998 [2] S.Schaer, R.Markus, B.Gerhard and A.S.Timon, Daily Global Ionosphere Maps based on GPS Carrier Phase Data Routinely produced by CODE Analysis Center, Proceeding of the IGS Analysis Center Workshop, 1996

11.279 rtklib_lambda.h File Reference

Integer ambiguity resolution.

```
#include "rtklib.h"
```

Macros

- #define **SGN_LAMBDA**(x) ((x) <= 0.0 ? -1.0 : 1.0)
- #define **ROUND_LAMBDA**(x) (floor((x) + 0.5))
- #define **SWAP_LAMBDA**(x, y)

Functions

- int **LD** (int n, const double *Q, double *L, double *D)
- void **gauss** (int n, double *L, double *Z, int i, int j)
- void **perm** (int n, double *L, double *D, int j, double del, double *Z)
- void **reduction** (int n, double *L, double *D, double *Z)
- int **search** (int n, int m, const double *L, const double *D, const double *zs, double *zn, double *s)
- int **lambda** (int n, int m, const double *a, const double *Q, double *F, double *s)
- int **lambda_reduction** (int n, const double *Q, double *Z)
- int **lambda_search** (int n, int m, const double *a, const double *Q, double *F, double *s)

Variables

- const int **LOOPMAX** = 10000

11.279.1 Detailed Description

Integer ambiguity resolution.

Authors

- 2007-2008, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2008, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

References: [1] P.J.G.Teunissen, The least-square ambiguity decorrelation adjustment: a method for fast GPS ambiguity estimation, J.Geodesy, Vol.70, 65-82, 1995 [2] X.-W.Chang, X.Yang, T.Zhou, MLAMBDA: A modified LAMBDA method for integer least-squares estimation, J.Geodesy, Vol.79, 552-565, 2005

11.279.2 Macro Definition Documentation

11.279.2.1 SWAP_LAMBDA

```
#define SWAP_LAMBDA(
    x,
    y )
```

Value:

```
do
    {
        double tmp_;
        tmp_ = x;
        x = y;
        y = tmp_;
    }
while (0)
```

Definition at line 50 of file rtklib_lambda.h.

11.280 rtklib_pntpos.h File Reference

standard code-based positioning

```
#include "rtklib.h"
#include "rtklib_rtkcmn.h"
```

Functions

- double **varerr** (const [prcopt_t](#) *opt, double el, int sys)
- double **gettgd** (int sat, const [nav_t](#) *nav)
- double **getiscl1** (int sat, const [nav_t](#) *nav)
- double **getiscl2** (int sat, const [nav_t](#) *nav)
- double **getiscl5i** (int sat, const [nav_t](#) *nav)
- double **getiscl5q** (int sat, const [nav_t](#) *nav)
- double **prange** (const [obsd_t](#) *obs, const [nav_t](#) *nav, const double *azel, int iter, const [prcopt_t](#) *opt, double *var)
- int **ionocorr** ([gtime_t](#) time, const [nav_t](#) *nav, int sat, const double *pos, const double *azel, int ionoopt, double *ion, double *var)
- int **tropcorr** ([gtime_t](#) time, const [nav_t](#) *nav, const double *pos, const double *azel, int tropopt, double *trp, double *var)
- int **rescode** (int iter, const [obsd_t](#) *obs, int n, const double *rs, const double *dts, const double *vare, const int *svh, const [nav_t](#) *nav, const double *x, const [prcopt_t](#) *opt, double *v, double *H, double *var, double *azel, int *vsat, double *resp, int *ns)
- int **valsol** (const double *azel, const int *vsat, int n, const [prcopt_t](#) *opt, const double *v, int nv, int nx, char *msg)
- int **estpos** (const [obsd_t](#) *obs, int n, const double *rs, const double *dts, const double *vare, const int *svh, const [nav_t](#) *nav, const [prcopt_t](#) *opt, [sol_t](#) *sol, double *azel, int *vsat, double *resp, char *msg)
- int **raim_fde** (const [obsd_t](#) *obs, int n, const double *rs, const double *dts, const double *vare, const int *svh, const [nav_t](#) *nav, const [prcopt_t](#) *opt, [sol_t](#) *sol, double *azel, int *vsat, double *resp, char *msg)
- int **resdop** (const [obsd_t](#) *obs, int n, const double *rs, const double *dts, const [nav_t](#) *nav, const double *rr, const double *x, const double *azel, const int *vsat, double *v, double *H)
- void **estvel** (const [obsd_t](#) *obs, int n, const double *rs, const double *dts, const [nav_t](#) *nav, const [prcopt_t](#) *opt, [sol_t](#) *sol, const double *azel, const int *vsat)

- int `pntpos` (const `obsd_t` *obs, int n, const `nav_t` *nav, const `prcopt_t` *opt, `sol_t` *sol, double *azel, `ssat_t` *ssat, char *msg)

single-point positioning compute receiver position, velocity, clock bias by single-point positioning with pseudorange and doppler observables args : `obsd_t` *obs *I observation data* int n *I number of observation data* `nav_t` *nav *I navigation data* `prcopt_t` *opt *I processing options* `sol_t` *sol *IO solution* double *azel *IO azimuth/elevation angle (rad) (NULL: no output)* `ssat_t` *ssat *IO satellite status (NULL: no output)* char *msg *O error message for error exit* return : status(1:ok,0:error) notes : assuming sbas-gps, galileo-gps, qzss-gps, compass-gps time offset and receiver bias are negligible (only involving glonass-gps time offset and receiver bias)

Variables

- const int `NX` = 4 + 3

of estimated parameters

- const int `MAXITR` = 10
max number of iteration for point pos
- const double `ERR_ION` = 5.0
ionospheric delay std (m)
- const double `ERR_TROP` = 3.0
tropspheric delay std (m)

11.280.1 Detailed Description

standard code-based positioning

Authors

- 2007-2013, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

11.280.2 Function Documentation

11.280.2.1 pntpos()

```
int pntpos (
    const obsd_t * obs,
    int n,
    const nav_t * nav,
    const prcopt_t * opt,
    sol_t * sol,
    double * azel,
    ssat_t * ssat,
    char * msg )
```

single-point positioning compute receiver position, velocity, clock bias by single-point positioning with pseudorange and doppler observables args : `obsd_t` *obs I observation data `int n` I number of observation data `nav_t` *nav I navigation data `prcopt_t` *opt I processing options `sol_t` *sol IO solution double *azel IO azimuth/elevation angle (rad) (NULL: no output) `ssat_t` *ssat IO satellite status (NULL: no output) char *msg O error message for error exit return : status(1:ok,0:error) notes : assuming sbas-gps, galileo-gps, qzss-gps, compass-gps time offset and receiver bias are negligible (only involving glonass-gps time offset and receiver bias)

11.280.3 Variable Documentation

11.280.3.1 ERR_ION

```
const double ERR_ION = 5.0
```

ionospheric delay std (m)

Definition at line 42 of file `rtklib_pntpos.h`.

11.280.3.2 ERR_TROP

```
const double ERR_TROP = 3.0
```

tropspheric delay std (m)

Definition at line 43 of file `rtklib_pntpos.h`.

11.280.3.3 MAXITR

```
const int MAXITR = 10
```

max number of iteration for point pos

Definition at line 41 of file `rtklib_pntpos.h`.

11.280.3.4 NX

```
const int NX = 4 + 3
```

of estimated parameters

Definition at line 40 of file rtklib_pntpos.h.

11.281 rtklib_ppp.h File Reference

Precise Point Positioning.

```
#include "rtklib.h"
```

Macros

- #define **MIN_PPP**(x, y) ((x) <= (y) ? (x) : (y))
- #define **ROUND_PPP**(x) static_cast<int>(floor((x) + 0.5))
- #define **SWAP_I**(x, y)
- #define **SWAP_D**(x, y)

Functions

- double **lam_LC** (int i, int j, int k)
- double **L_LC** (int i, int j, int k, const double *L)
- double **P_LC** (int i, int j, int k, const double *P)
- double **var_LC** (int i, int j, int k, double sig)
- double **q_gamma** (double a, double x, double log_gamma_a)
- double **p_gamma** (double a, double x, double log_gamma_a)
- double **f_erfc** (double x)
- double **conffunc** (int N, double B, double sig)
- void **average_LC** (rtk_t *rtk, const obsd_t *obs, int n, const nav_t *nav, const double *azel)
- int **fix_amb_WL** (rtk_t *rtk, const nav_t *nav, int sat1, int sat2, int *NW)
- int **is_depend** (int sat1, int sat2, int *flgs, int *max_flg)
- int **sel_amb** (int *sat1, int *sat2, double *N, double *var, int n)
- int **fix_sol** (rtk_t *rtk, const int *sat1, const int *sat2, const double *NC, int n)
- int **fix_amb_ROUND** (rtk_t *rtk, int *sat1, int *sat2, const int *NW, int n)
- int **fix_amb_ILS** (rtk_t *rtk, int *sat1, int *sat2, int *NW, int n)
- int **pppamb** (rtk_t *rtk, const obsd_t *obs, int n, const nav_t *nav, const double *azel)
- void **pppoutsolstat** (rtk_t *rtk, int level, FILE *fp)
- void **testclipse** (const obsd_t *obs, int n, const nav_t *nav, double *rs)
- double **varerr** (int sat, int sys, double el, int type, const prcopt_t *opt)
- void **initx** (rtk_t *rtk, double xi, double var, int i)
- int **ifmeas** (const obsd_t *obs, const nav_t *nav, const double *azel, const prcopt_t *opt, const double *dantr, const double *dants, double phw, double *meas, double *var)
- double **gettgd_ppp** (int sat, const nav_t *nav)
- int **corr_ion** (itime_t time, const nav_t *nav, int sat, const double *pos, const double *azel, int ionoopt, double *ion, double *var, int *brk)

- int **corrmeas** (const [obsd_t](#) *obs, const [nav_t](#) *nav, const double *pos, const double *azel, const [prcopt_t](#) *opt, const double *dantr, const double *dants, double phw, double *meas, double *var, int *brk)
- double **gfmeas** (const [obsd_t](#) *obs, const [nav_t](#) *nav)
- void **udpos_ppp** ([rtk_t](#) *rtk)
- void **udclk_ppp** ([rtk_t](#) *rtk)
- void **udtrop_ppp** ([rtk_t](#) *rtk)
- void **detslp_ll** ([rtk_t](#) *rtk, const [obsd_t](#) *obs, int n)
- void **detslp_gf** ([rtk_t](#) *rtk, const [obsd_t](#) *obs, int n, const [nav_t](#) *nav)
- void **udbias_ppp** ([rtk_t](#) *rtk, const [obsd_t](#) *obs, int n, const [nav_t](#) *nav)
- void **udstate_ppp** ([rtk_t](#) *rtk, const [obsd_t](#) *obs, int n, const [nav_t](#) *nav)
- void **satantpcv** (const double *rs, const double *rr, const [pcv_t](#) *pcv, double *dant)
- double **prectrop** ([gtime_t](#) time, const double *pos, const double *azel, const [prcopt_t](#) *opt, const double *x, double *dtdx, double *var)
- int **res_ppp** (int iter, const [obsd_t](#) *obs, int n, const double *rs, const double *dts, const double *vare, const int *svh, const [nav_t](#) *nav, const double *x, [rtk_t](#) *rtk, double *v, double *H, double *R, double *azel)
- int **pppnx** (const [prcopt_t](#) *opt)
- void **pppos** ([rtk_t](#) *rtk, const [obsd_t](#) *obs, int n, const [nav_t](#) *nav)

Variables

- const double **MIN_ARC_GAP** = 300.0
- const double **CONST_AMB** = 0.001
- const double **THRES_RES** = 0.3
- const double **LOG_PI** = 1.14472988584940017
- const double **SQRT2** = 1.41421356237309510
- const double **VAR_POS_PPP** = std::pow(100.0, 2.0)
- const double **VAR_CLK** = std::pow(100.0, 2.0)
- const double **VAR_ZTD** = std::pow(0.3, 2.0)
- const double **VAR_GRA_PPP** = std::pow(0.001, 2.0)
- const double **VAR_BIAS** = std::pow(100.0, 2.0)
- const double **VAR_IONO_OFF** = std::pow(10.0, 2.0)

11.281.1 Detailed Description

Precise Point Positioning.

Authors

- 2007-2008, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2008, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

11.281.2 Macro Definition Documentation

11.281.2.1 SWAP_D

```
#define SWAP_D(  
    x,  
    y )
```

Value:

```
do  
    {  
        double _z = x;  
        x = y;  
        y = _z;  
    }  
while (0)
```

Definition at line 51 of file rtklib_ppp.h.

11.281.2.2 SWAP_I

```
#define SWAP_I(  
    x,  
    y )
```

Value:

```
do  
    {  
        int _z = x;  
        x = y;  
        y = _z;  
    }  
while (0)
```

Definition at line 43 of file rtklib_ppp.h.

11.282 rtklib_preceph.h File Reference

precise ephemeris and clock functions

```
#include "rtklib.h"
```

Functions

- int **code2sys** (char code)
- int **readsp3h** (FILE *fp, [gtime_t](#) *time, char *type, int *sats, double *bfact, char *tsys)
- int **addpeph** ([nav_t](#) *nav, [peph_t](#) *peph)
- void **readsp3b** (FILE *fp, char type, int *sats, int ns, const double *bfact, char *tsys, int index, int opt, [nav_t](#) *nav)
- int **cmppeph** (const void *p1, const void *p2)
- void **combpeph** ([nav_t](#) *nav, int opt)
- void **readsp3** (const char *file, [nav_t](#) *nav, int opt)
- int **readsap** (const char *file, [gtime_t](#) time, [nav_t](#) *nav)
- int **readdcbf** (const char *file, [nav_t](#) *nav, const [sta_t](#) *sta)
- int **readdcb** (const char *file, [nav_t](#) *nav, const [sta_t](#) *sta)
- double **interppl** (const double *x, double *y, int n)
- int **pephpos** ([gtime_t](#) time, int sat, const [nav_t](#) *nav, double *rs, double *dts, double *vare, double *varc)
- int **pephclk** ([gtime_t](#) time, int sat, const [nav_t](#) *nav, double *dts, double *varc)
- void **satantoff** ([gtime_t](#) time, const double *rs, int sat, const [nav_t](#) *nav, double *dant)
- int **peph2pos** ([gtime_t](#) time, int sat, const [nav_t](#) *nav, int opt, double *rs, double *dts, double *var)

Variables

- const int **NMAX** = 10
- const double **MAXDTE** = 900.0
- const double **EXTERR_CLK** = 1e-3
- const double **EXTERR_EPH** = 5e-7

11.282.1 Detailed Description

precise ephemeris and clock functions

Authors

- 2007-2013, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

References : [1] S.Hilla, The Extended Standard Product 3 Orbit Format (SP3-c), 12 February, 2007 [2] J.Ray, W. Gurtner, RINEX Extensions to Handle Clock Information, 27 August, 1998 [3] D.D.McCarthy, IERS Technical Note 21, IERS Conventions 1996, July 1996 [4] D.A.Vallado, Fundamentals of Astrodynamics and Applications 2nd ed, Space Technology Library, 2004

11.283 rtklib_pvt.h File Reference

Interface of a Position Velocity and Time computation block.

```
#include "gnss_synchro.h"
#include "pvt_interface.h"
#include "rtklib.h"
#include "rtklib_pvt_gs.h"
#include <gnuradio/gr_complex.h>
#include <gnuradio/runtime_types.h>
#include <cstdio>
#include <ctime>
#include <map>
#include <string>
```

Classes

- class [Rtklib_Pvt](#)
This class implements a [PvtInterface](#) for the RTKLIB PVT block.

11.283.1 Detailed Description

Interface of a Position Velocity and Time computation block.

Author

Javier Arribas, 2017. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.284 rtklib_pvt_gs.h File Reference

Interface of a Position Velocity and Time computation block.

```
#include "gnss_synchro.h"
#include "rtklib.h"
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <gnuradio/sync_block.h>
#include <gnuradio/types.h>
#include <pmt/pmt.h>
#include <chrono>
#include <cstdio>
#include <stdint>
#include <ctime>
#include <map>
#include <memory>
#include <string>
#include <sys/types.h>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [rtklib_pvt_gs](#)

This class implements a block that computes the PVT solution using the RTKLIB integrated library.

Typedefs

- using [rtklib_pvt_gs_sptr](#) = boost::shared_ptr< [rtklib_pvt_gs](#) >

Functions

- [rtklib_pvt_gs_sptr](#) [rtklib_make_pvt_gs](#) (uint32_t nchannels, const [Pvt_Conf](#) &conf_, const [rtk_t](#) &rtk)

11.284.1 Detailed Description

Interface of a Position Velocity and Time computation block.

Author

Javier Arribas, 2017. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.285 rtklib_rtcn.h File Reference

RTCM functions headers.

```
#include "rtklib.h"
#include "rtklib_rtcn2.h"
#include "rtklib_rtcn3.h"
```

Macros

- **#define RTCM2PREAMB** 0x66 /* rtcn ver.2 frame preamble */
- **#define RTCM3PREAMB** 0xD3 /* rtcn ver.3 frame preamble */

Functions

- int **init_rtc2** ([rtc2_t](#) *rtc2)
- void **free_rtc2** ([rtc2_t](#) *rtc2)
- int **input_rtc2** ([rtc2_t](#) *rtc2, unsigned char data)
- int **input_rtc23** ([rtc2_t](#) *rtc2, unsigned char data)
- int **input_rtc2f** ([rtc2_t](#) *rtc2, FILE *fp)
- int **input_rtc23f** ([rtc2_t](#) *rtc2, FILE *fp)
- int **gen_rtc2** ([rtc2_t](#) *rtc2, int type, int sync)

11.285.1 Detailed Description

RTC2 functions headers.

Authors

- 2007-2013, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

11.286 rtklib_rtc2.h File Reference

RTC2 v2 functions headers.

```
#include "rtklib.h"
```

Functions

- void **adjhour** ([rtcm_t](#) *rtcm, double zcnt)
- int **obsindex** ([obs_t](#) *obs, [gtime_t](#) time, int sat)
- int **decode_type1** ([rtcm_t](#) *rtcm)
- int **decode_type3** ([rtcm_t](#) *rtcm)
- int **decode_type14** ([rtcm_t](#) *rtcm, bool pre_2009_file=false)
- int **decode_type16** ([rtcm_t](#) *rtcm)
- int **decode_type17** ([rtcm_t](#) *rtcm, bool pre_2009_file=false)
- int **decode_type18** ([rtcm_t](#) *rtcm)
- int **decode_type19** ([rtcm_t](#) *rtcm)
- int **decode_type22** ([rtcm_t](#) *rtcm)
- int **decode_type23** ([rtcm_t](#) *rtcm)
- int **decode_type24** ([rtcm_t](#) *rtcm)
- int **decode_type31** ([rtcm_t](#) *rtcm)
- int **decode_type32** ([rtcm_t](#) *rtcm)
- int **decode_type34** ([rtcm_t](#) *rtcm)
- int **decode_type36** ([rtcm_t](#) *rtcm)
- int **decode_type37** ([rtcm_t](#) *rtcm)
- int **decode_type59** ([rtcm_t](#) *rtcm)
- int **decode_rtc2** ([rtcm_t](#) *rtcm)

11.286.1 Detailed Description

RTCM v2 functions headers.

Authors

- 2007-2013, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

11.287 rtklib_rtc3.h File Reference

RTCM v3 functions headers.

```
#include "rtklib.h"
```

Functions

- double **getbitg** (const unsigned char *buff, int pos, int len)
- void **adjweek** (rtcm_t *rtcm, double tow)
- int **adjbdtweek** (int week)
- void **adjday_glot** (rtcm_t *rtcm, double tod)
- double **adjcp** (rtcm_t *rtcm, int sat, int freq, double cp)
- int **lossoflock** (rtcm_t *rtcm, int sat, int freq, int lock)
- unsigned char **snratio** (double snr)
- int **obsindex3** (obs_t *obs, gtime_t time, int sat)
- int **test_staid** (rtcm_t *rtcm, int staid)
- int **decode_head1001** (rtcm_t *rtcm, int *sync)
- int **decode_type1001** (rtcm_t *rtcm)
- int **decode_type1002** (rtcm_t *rtcm)
- int **decode_type1003** (rtcm_t *rtcm)
- int **decode_type1004** (rtcm_t *rtcm)
- double **getbits_38** (const unsigned char *buff, int pos)
- int **decode_type1005** (rtcm_t *rtcm)
- int **decode_type1006** (rtcm_t *rtcm)
- int **decode_type1007** (rtcm_t *rtcm)
- int **decode_type1008** (rtcm_t *rtcm)
- int **decode_head1009** (rtcm_t *rtcm, int *sync)
- int **decode_type1009** (rtcm_t *rtcm)
- int **decode_type1010** (rtcm_t *rtcm)
- int **decode_type1011** (rtcm_t *rtcm)
- int **decode_type1012** (rtcm_t *rtcm)
- int **decode_type1013** (rtcm_t *rtcm)
- int **decode_type1019** (rtcm_t *rtcm, bool pre_2009_file=false)
- int **decode_type1020** (rtcm_t *rtcm)
- int **decode_type1021** (rtcm_t *rtcm)
- int **decode_type1022** (rtcm_t *rtcm)
- int **decode_type1023** (rtcm_t *rtcm)
- int **decode_type1024** (rtcm_t *rtcm)
- int **decode_type1025** (rtcm_t *rtcm)
- int **decode_type1026** (rtcm_t *rtcm)
- int **decode_type1027** (rtcm_t *rtcm)
- int **decode_type1029** (rtcm_t *rtcm)
- int **decode_type1030** (rtcm_t *rtcm)
- int **decode_type1031** (rtcm_t *rtcm)
- int **decode_type1032** (rtcm_t *rtcm)
- int **decode_type1033** (rtcm_t *rtcm)
- int **decode_type1034** (rtcm_t *rtcm)
- int **decode_type1035** (rtcm_t *rtcm)
- int **decode_type1037** (rtcm_t *rtcm)
- int **decode_type1038** (rtcm_t *rtcm)
- int **decode_type1039** (rtcm_t *rtcm)
- int **decode_type1044** (rtcm_t *rtcm, bool pre_2009_file=false)
- int **decode_type1045** (rtcm_t *rtcm)
- int **decode_type1046** (rtcm_t *rtcm)
- int **decode_type1047** (rtcm_t *rtcm)
- int **decode_type1063** (rtcm_t *rtcm)
- int **decode_ssr1_head** (rtcm_t *rtcm, int sys, int *sync, int *iod, double *udint, int *refd, int *hsize)
- int **decode_ssr2_head** (rtcm_t *rtcm, int sys, int *sync, int *iod, double *udint, int *hsize)
- int **decode_ssr7_head** (rtcm_t *rtcm, int sys, int *sync, int *iod, double *udint, int *dispe, int *mw, int *hsize)

- int **decode_ssr1** ([rtcm_t](#) *rtcm, int sys)
- int **decode_ssr2** ([rtcm_t](#) *rtcm, int sys)
- int **decode_ssr3** ([rtcm_t](#) *rtcm, int sys)
- int **decode_ssr4** ([rtcm_t](#) *rtcm, int sys)
- int **decode_ssr5** ([rtcm_t](#) *rtcm, int sys)
- int **decode_ssr6** ([rtcm_t](#) *rtcm, int sys)
- int **decode_ssr7** ([rtcm_t](#) *rtcm, int sys)
- void **sigindex** (int sys, const unsigned char *code, const int *freq, int n, const char *opt, int *ind)
- void **save_msm_obs** ([rtcm_t](#) *rtcm, int sys, [msm_h_t](#) *h, const double *r, const double *pr, const double *cp, const double *rr, const double *rrf, const double *cni, const int *lock, const int *ex, const int *half)
- int **decode_msm_head** ([rtcm_t](#) *rtcm, int sys, int *sync, int *iod, [msm_h_t](#) *h, int *hsize)
- int **decode_msm0** ([rtcm_t](#) *rtcm, int sys)
- int **decode_msm4** ([rtcm_t](#) *rtcm, int sys)
- int **decode_msm5** ([rtcm_t](#) *rtcm, int sys)
- int **decode_msm6** ([rtcm_t](#) *rtcm, int sys)
- int **decode_msm7** ([rtcm_t](#) *rtcm, int sys)
- int **decode_type1230** ([rtcm_t](#) *rtcm)
- int **decode_rtc3** ([rtcm_t](#) *rtcm)

Variables

- const double **PRUNIT_GPS** = 299792.458
- const double **PRUNIT_GLO** = 599584.916
- const double **RANGE_MS** = [SPEED_OF_LIGHT_M_S](#) * 0.001
- const double **SSRUDINT** [16]
- const int **CODES_GPS** []
- const int **CODES_GLO** []
- const int **CODES_GAL** []
- const int **CODES_QZS** []
- const int **CODES_BDS** []
- const int **CODES_SBS** []

11.287.1 Detailed Description

RTCM v3 functions headers.

Authors

- 2007-2013, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

11.287.2 Variable Documentation

11.287.2.1 CODES_BDS

```
const int CODES_BDS[ ]
```

Initial value:

```
= {  
    CODE_L1I, CODE_L1Q, CODE_L1X, CODE_L7I,  
    CODE_L7Q, CODE_L7X, CODE_L6I, CODE_L6Q,  
    CODE_L6X}
```

Definition at line 73 of file rtklib_rtc3.h.

11.287.2.2 CODES_GAL

```
const int CODES_GAL[ ]
```

Initial value:

```
= {  
    CODE_L1A, CODE_L1B, CODE_L1C, CODE_L1X,  
    CODE_L1Z, CODE_L5I, CODE_L5Q, CODE_L5X,  
    CODE_L7I, CODE_L7Q, CODE_L7X, CODE_L8I,  
    CODE_L8Q, CODE_L8X, CODE_L6A, CODE_L6B,  
    CODE_L6C, CODE_L6X, CODE_L6Z}
```

Definition at line 62 of file rtklib_rtc3.h.

11.287.2.3 CODES_GLO

```
const int CODES_GLO[ ]
```

Initial value:

```
= {  
    CODE_L1C, CODE_L1P, CODE_L2C, CODE_L2P}
```

Definition at line 58 of file rtklib_rtc3.h.

11.287.2.4 CODES_GPS

```
const int CODES_GPS[]
```

Initial value:

```
= {  
    CODE_L1C, CODE_L1P, CODE_L1W, CODE_L1Y,  
    CODE_L1M, CODE_L2C, CODE_L2D, CODE_L2S,  
    CODE_L2L, CODE_L2X, CODE_L2P, CODE_L2W,  
    CODE_L2Y, CODE_L2M, CODE_L5I, CODE_L5Q,  
    CODE_L5X}
```

Definition at line 52 of file rtklib_rtc3.h.

11.287.2.5 CODES_QZS

```
const int CODES_QZS[]
```

Initial value:

```
= {  
    CODE_L1C, CODE_L1S, CODE_L1L, CODE_L2S,  
    CODE_L2L, CODE_L2X, CODE_L5I, CODE_L5Q,  
    CODE_L5X, CODE_L6S, CODE_L6L, CODE_L6X,  
    CODE_L1X}
```

Definition at line 68 of file rtklib_rtc3.h.

11.287.2.6 CODES_SBS

```
const int CODES_SBS[]
```

Initial value:

```
= {  
    CODE_L1C, CODE_L5I, CODE_L5Q, CODE_L5X}
```

Definition at line 78 of file rtklib_rtc3.h.

11.287.2.7 SSRUDINT

```
const double SSRUDINT[16]
```

Initial value:

```
= {  
    1, 2, 5, 10, 15, 30, 60, 120, 240, 300, 600, 900, 1800, 3600, 7200, 10800}
```

Definition at line 47 of file rtklib_rtc3.h.

11.288 rtklib_rtkcmn.h File Reference

rtklib common functions

```
#include "rtklib.h"
#include <string>
```

Macros

- **#define Rx**(t, X)
- **#define Ry**(t, X)
- **#define Rz**(t, X)

Functions

- void **fatalerr** (const char *format,...)
- int **satno** (int sys, int prn)
- int **satsys** (int sat, int *prn)
- int **satid2no** (const char *id)
- void **satno2id** (int sat, char *id)
- int **satexclude** (int sat, int svh, const [prcopt_t](#) *opt)
- int **testsnr** (int base, int freq, double el, double snr, const [snrmask_t](#) *mask)
- unsigned char **obs2code** (const char *obs, int *freq)
- char * **code2obs** (unsigned char code, int *freq)
- void **setcodepri** (int sys, int freq, const char *pri)
- int **getcodepri** (int sys, unsigned char code, const char *opt)
- unsigned int **getbitu** (const unsigned char *buff, int pos, int len)
- int **getbits** (const unsigned char *buff, int pos, int len)
- void **setbitu** (unsigned char *buff, int pos, int len, unsigned int data)
- void **setbits** (unsigned char *buff, int pos, int len, int data)
- unsigned int **rtk_crc32** (const unsigned char *buff, int len)
- unsigned int **rtk_crc24q** (const unsigned char *buff, int len)
- unsigned short **rtk_crc16** (const unsigned char *buff, int len)
- int **decode_word** (unsigned int word, unsigned char *data)
- double * **mat** (int n, int m)
- int * **imat** (int n, int m)
- double * **zeros** (int n, int m)
- double * **eye** (int n)
- double **dot** (const double *a, const double *b, int n)
- double **norm_rtk** (const double *a, int n)
- void **cross3** (const double *a, const double *b, double *c)
- int **normv3** (const double *a, double *b)
- void **matcpy** (double *A, const double *B, int n, int m)
- void **matmul** (const char *tr, int n, int k, int m, double alpha, const double *A, const double *B, double beta, double *C)
- int **matinv** (double *A, int n)
- int **solve** (const char *tr, const double *A, const double *Y, int n, int m, double *X)
- int **lsq** (const double *A, const double *y, int n, int m, double *x, double *Q)
- int **filter_** (const double *x, const double *P, const double *H, const double *v, const double *R, int n, int m, double *xp, double *Pp)
- int **filter** (double *x, double *P, const double *H, const double *v, const double *R, int n, int m)

- int **smoother** (const double *xf, const double *Qf, const double *xb, const double *Qb, int n, double *xs, double *Qs)
- void **matfprint** (const double A[], int n, int m, int p, int q, FILE *fp)
- void **matsprint** (const double A[], int n, int m, int p, int q, std::string &buffer)
- void **matprint** (const double A[], int n, int m, int p, int q)
- double **str2num** (const char *s, int i, int n)
- int **str2time** (const char *s, int i, int n, [gtime_t](#) *t)
- [gtime_t](#) **epoch2time** (const double *ep)
- void **time2epoch** ([gtime_t](#) t, double *ep)
- [gtime_t](#) **gpst2time** (int week, double sec)
- double **time2gpst** ([gtime_t](#) t, int *week)
- [gtime_t](#) **gst2time** (int week, double sec)
- double **time2gst** ([gtime_t](#) t, int *week)
- [gtime_t](#) **bd2time** (int week, double sec)
- double **time2bd2** ([gtime_t](#) t, int *week)
- [gtime_t](#) **timeadd** ([gtime_t](#) t, double sec)
- double **timediff** ([gtime_t](#) t1, [gtime_t](#) t2)
- double **timediffweekcrossover** ([gtime_t](#) t1, [gtime_t](#) t2)
- [gtime_t](#) **timeget** ()
- void **timeset** ([gtime_t](#) t)
- int **read_leaps_text** (FILE *fp)
- int **read_leaps_usno** (FILE *fp)
- int **read_leaps** (const char *file)
- [gtime_t](#) **gpst2utc** ([gtime_t](#) t)
- [gtime_t](#) **utc2gpst** ([gtime_t](#) t)
- [gtime_t](#) **gpst2bd2** ([gtime_t](#) t)
- [gtime_t](#) **bd2gpst** ([gtime_t](#) t)
- double **time2sec** ([gtime_t](#) time, [gtime_t](#) *day)
- double **utc2gmst** ([gtime_t](#) t, double ut1_utc)
- void **time2str** ([gtime_t](#) t, char *s, int n)
- char * **time_str** ([gtime_t](#) t, int n)
- double **time2doy** ([gtime_t](#) t)
- int **adjgpsweek** (int week, bool pre_2009_file=false)
- unsigned int **tickget** ()
- void **sleepms** (int ms)
- void **deg2dms** (double deg, double *dms, int ndec)
- void **deg2dms** (double deg, double *dms)
- double **dms2deg** (const double *dms)
- void **ecef2pos** (const double *r, double *pos)
- void **pos2ecef** (const double *pos, double *r)
- void **xyz2enu** (const double *pos, double *E)
- void **ecef2enu** (const double *pos, const double *r, double *e)
- void **enu2ecef** (const double *pos, const double *e, double *r)
- void **covenu** (const double *pos, const double *P, double *Q)
- void **covecef** (const double *pos, const double *Q, double *P)
- void **ast_args** (double t, double *f)
- void **nut_iau1980** (double t, const double *f, double *dpsi, double *deps)
- void **eci2ecef** ([gtime_t](#) tutc, const double *erpv, double *U, double *gmst)
- int **decodef** (char *p, int n, double *v)
- void **addpcv** (const [pcv_t](#) *pcv, [pcvs_t](#) *pcvs)
- int **readngspcv** (const char *file, [pcvs_t](#) *pcvs)
- int **readantex** (const char *file, [pcvs_t](#) *pcvs)
- int **readpcv** (const char *file, [pcvs_t](#) *pcvs)
- [pcv_t](#) * **searchpcv** (int sat, const char *type, [gtime_t](#) time, const [pcvs_t](#) *pcvs)
- void **readpos** (const char *file, const char *rcv, double *pos)

- int **readblqrecord** (FILE *fp, double *odisp)
- int **readblq** (const char *file, const char *sta, double *odisp)
- int **readerp** (const char *file, [erp_t](#) *erp)
- int **geterp** (const [erp_t](#) *erp, [gtime_t](#) time, double *erpv)
- int **cmpeph** (const void *p1, const void *p2)
- void **unigeph** ([nav_t](#) *nav)
- int **cmpgeph** (const void *p1, const void *p2)
- void **uniggeph** ([nav_t](#) *nav)
- int **cmpseph** (const void *p1, const void *p2)
- void **uniqseph** ([nav_t](#) *nav)
- void **uniquav** ([nav_t](#) *nav)
- int **cmpobs** (const void *p1, const void *p2)
- int **sortobs** ([obs_t](#) *obs)
- int **screent** ([gtime_t](#) time, [gtime_t](#) ts, [gtime_t](#) te, double tint)
- int **readnav** (const char *file, [nav_t](#) *nav)
- int **savenav** (const char *file, const [nav_t](#) *nav)
- void **freeobs** ([obs_t](#) *obs)
- void **freenav** ([nav_t](#) *nav, int opt)
- void **traceopen** (const char *file)
- void **traceclose** ()
- void **tracelevel** (int level)
- void **traceswap** ()
- void **trace** (int level, const char *format,...)
- void **tracet** (int level, const char *format,...)
- void **tracemat** (int level, const double *A, int n, int m, int p, int q)
- void **traceobs** (int level, const [obsd_t](#) *obs, int n)
- int **execcmd** (const char *cmd)
- void **createdir** (const char *path)
- int **repstr** (char *str, const char *pat, const char *rep)
- int **reppath** (const char *path, char *rpath, [gtime_t](#) time, const char *rov, const char *base)
- int **reppaths** (const char *path, char *rpath[], int nmax, [gtime_t](#) ts, [gtime_t](#) te, const char *rov, const char *base)
- double **satwavelen** (int sat, int frq, const [nav_t](#) *nav)
- double **geodist** (const double *rs, const double *rr, double *e)
- double **satazel** (const double *pos, const double *e, double *azel)
- void **dops** (int ns, const double *azel, double elmin, double *dop)
- double **ionmodel** ([gtime_t](#) t, const double *ion, const double *pos, const double *azel)
- double **ionmapf** (const double *pos, const double *azel)
- double **ionppp** (const double *pos, const double *azel, double re, double hion, double *posp)
- double **tropmodel** ([gtime_t](#) time, const double *pos, const double *azel, double humi)
- double **interp** (const double coef[], double lat)
- double **mapf** (double el, double a, double b, double c)
- double **nmf** ([gtime_t](#) time, const double pos[], const double azel[], double *mapfw)
- double **tropmapf** ([gtime_t](#) time, const double pos[], const double azel[], double *mapfw)
- double **interpvar** (double ang, const double *var)
- void **antmodel** (const [pcv_t](#) *pcv, const double *del, const double *azel, int opt, double *dant)
- void **antmodel_s** (const [pcv_t](#) *pcv, double nadir, double *dant)
- void **sunmoonpos_eci** ([gtime_t](#) tut, double *rsun, double *rmoon)
- void **sunmoonpos** ([gtime_t](#) tutc, const double *erpv, double *rsun, double *rmoon, double *gmst)
- void **csmooth** ([obs_t](#) *obs, int ns)
- int **rtk_uncompress** (const char *file, char *uncfile)
- int **expath** (const char *path, char *paths[], int nmax)
- void **windupcorr** ([gtime_t](#) time, const double *rs, const double *rr, double *phw)

11.288.1 Detailed Description

rtklib common functions

Authors

- 2007-2013, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

References : [1] IS-GPS-200K, Navstar GPS Space Segment/Navigation User Interfaces, 7 March, 2006 [2] RTCA/DO-229C, Minimum operational performance standards for global positioning system/wide area augmentation system airborne equipment, RTCA inc, November 28, 2001 [3] M.Rothacher, R.Schmid, ANTEX: The Antenna Exchange Format Version 1.4, 15 September, 2010 [4] A.Gelb ed., Applied Optimal Estimation, The M.I.T Press, 1974 [5] A.E.Niell, Global mapping functions for the atmosphere delay at radio wavelengths, Journal of geophysical research, 1996 [6] W.Gurtner and L.Estey, RINEX The Receiver Independent Exchange Format Version 3.00, November 28, 2007 [7] J.Kouba, A Guide to using International GNSS Service (IGS) products, May 2009 [8] China Satellite Navigation Office, BeiDou navigation satellite system signal in space interface control document, open service signal B1I (version 1.0), Dec 2012 [9] J.Boehm, A.Niell, P.Tregoning and H.Shuh, Global Mapping Function (GMF): A new empirical mapping function base on numerical weather model data, Geophysical Research Letters, 33, L07304, 2006 [10] GLONASS/GPS/Galileo/Compass/SBAS NV08C receiver series BINR interface protocol specification ver.1.3, August, 2012

11.288.2 Macro Definition Documentation

11.288.2.1 Rx

```
#define Rx(  
    t,  
    X )
```

Value:

```
do  
{  
    (X)[0] = 1.0;  
    (X)[1] = (X)[2] = (X)[3] = (X)[6] = 0.0;  
    (X)[4] = (X)[8] = cos(t);  
    (X)[7] = sin(t);  
    (X)[5] = -(X)[7];  
}  
while (0)
```

Definition at line 66 of file rtklib_rtkcmn.h.

11.288.2.2 Ry

```
#define Ry(
    t,
    X )
```

Value:

```
do
{
    (X)[4] = 1.0;
    (X)[1] = (X)[3] = (X)[5] = (X)[7] = 0.0;
    (X)[0] = (X)[8] = cos(t);
    (X)[2] = sin(t);
    (X)[6] = -(X)[2];
}
while (0)
```

Definition at line 77 of file rtklib_rtkcmn.h.

11.288.2.3 Rz

```
#define Rz(
    t,
    X )
```

Value:

```
do
{
    (X)[8] = 1.0;
    (X)[2] = (X)[5] = (X)[6] = (X)[7] = 0.0;
    (X)[0] = (X)[4] = cos(t);
    (X)[3] = sin(t);
    (X)[1] = -(X)[3];
}
while (0)
```

Definition at line 88 of file rtklib_rtkcmn.h.

11.289 rtklib_rtkpos.h File Reference

rtklib ppp-related functions

```
#include "rtklib.h"
#include "rtklib_rtkcmn.h"
```

Macros

- **#define II_RTK**(s, opt) (NP_RTK(opt) + (s)-1) /* ionos (s:satellite no) */
- **#define IT_RTK**(r, opt) (NP_RTK(opt) + NI_RTK(opt) + NT_RTK(opt) / 2 * (r)) /* tropos (r:0=rov,1:ref) */
- **#define IL_RTK**(f, opt) (NP_RTK(opt) + NI_RTK(opt) + NT_RTK(opt) + (f)) /* receiver h/w bias */
- **#define IB_RTK**(s, f, opt) (NR_RTK(opt) + MAXSAT * (f) + (s)-1) /* phase bias (s:satno,f:freq) */

Functions

- int **rtkopenstat** (const char *file, int level)
- void **rtkclosestat** ()
- void **rtkoutstat** (rtk_t *rtk)
- void **swapsolstat** ()
- void **outsolstat** (rtk_t *rtk)
- void **errmsg** (rtk_t *rtk, const char *format,...)
- double **sdobs** (const obsd_t *obs, int i, int j, int f)
- double **gfobs_L1L2** (const obsd_t *obs, int i, int j, const double *lam)
- double **gfobs_L1L5** (const obsd_t *obs, int i, int j, const double *lam)
- double **varerr** (int sat, int sys, double el, double bl, double dt, int f, const prcopt_t *opt)
- double **baseline** (const double *ru, const double *rb, double *dr)
- void **initx_rtk** (rtk_t *rtk, double xi, double var, int i)
- int **selsat** (const obsd_t *obs, const double *azel, int nu, int nr, const prcopt_t *opt, int *sat, int *iu, int *ir)
- void **udpos** (rtk_t *rtk, double tt)
- void **udion** (rtk_t *rtk, double tt, double bl, const int *sat, int ns)
- void **udtrop** (rtk_t *rtk, double tt, double bl)
- void **udrcvbias** (rtk_t *rtk, double tt)
- void **detslp_ll** (rtk_t *rtk, const obsd_t *obs, int i, int rcv)
- void **detslp_gf_L1L2** (rtk_t *rtk, const obsd_t *obs, int i, int j, const nav_t *nav)
- void **detslp_gf_L1L5** (rtk_t *rtk, const obsd_t *obs, int i, int j, const nav_t *nav)
- void **detslp_dop** (rtk_t *rtk, const obsd_t *obs, int i, int rcv, const nav_t *nav)
- void **udbias** (rtk_t *rtk, double tt, const obsd_t *obs, const int *sat, const int *iu, const int *ir, int ns, const nav_t *nav)
- void **udstate** (rtk_t *rtk, const obsd_t *obs, const int *sat, const int *iu, const int *ir, int ns, const nav_t *nav)
- void **zdres_sat** (int base, double r, const obsd_t *obs, const nav_t *nav, const double *azel, const double *dant, const prcopt_t *opt, double *y)
- int **zdres** (int base, const obsd_t *obs, int n, const double *rs, const double *dts, const int *svh, const nav_t *nav, const double *rr, const prcopt_t *opt, int index, double *y, double *e, double *azel)
- int **validobs** (int i, int j, int f, int nf, const double *y)
- void **ddcov** (const int *nb, int n, const double *Ri, const double *Rj, int nv, double *R)
- int **constbl** (rtk_t *rtk, const double *x, const double *P, double *v, double *H, double *Ri, double *Rj, int index)
- double **prectrop** (gtime_t time, const double *pos, int r, const double *azel, const prcopt_t *opt, const double *x, double *dtdx)
- double **gloicbcorr** (int sat1, int sat2, const prcopt_t *opt, double lam1, double lam2, int f)
- int **test_sys** (int sys, int m)
- int **ddres** (rtk_t *rtk, const nav_t *nav, double dt, const double *x, const double *P, const int *sat, double *y, const double *e, double *azel, const int *iu, const int *ir, int ns, double *v, double *H, double *R, int *vflg)
- double **intpres** (gtime_t time, const obsd_t *obs, int n, const nav_t *nav, rtk_t *rtk, double *y)
- int **ddmat** (rtk_t *rtk, double *D)
- void **restamb** (rtk_t *rtk, const double *bias, int nb, double *xa)
- void **holdamb** (rtk_t *rtk, const double *xa)
- int **resamb_LAMBDA** (rtk_t *rtk, double *bias, double *xa)
- int **valpos** (rtk_t *rtk, const double *v, const double *R, const int *vflg, int nv, double thres)
- int **relpos** (rtk_t *rtk, const obsd_t *obs, int nu, int nr, const nav_t *nav)
- void **rtkinit** (rtk_t *rtk, const prcopt_t *opt)
- void **rtkfree** (rtk_t *rtk)
- int **rtkpos** (rtk_t *rtk, const obsd_t *obs, int n, const nav_t *nav)

Variables

- const double **VAR_POS** = std::pow(30.0, 2.0)
- const double **VAR_VEL** = std::pow(10.0, 2.0)
- const double **VAR_ACC** = std::pow(10.0, 2.0)
- const double **VAR_HWBIAS** = std::pow(1.0, 2.0)
- const double **VAR_GRA** = std::pow(0.001, 2.0)
- const double **INIT_ZWD** = 0.15
- const double **PRN_HWBIA** = 1E-6
- const double **MAXAC** = 30.0
- const double **VAR_HOLDAMB** = 0.001
- const double **TTOL_MOVEB** = (1.0 + 2 * [DTTOL](#))

11.289.1 Detailed Description

rtklib ppp-related functions

Authors

- 2007-2013, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

11.290 rtklib_rtksvr.h File Reference

rtk server functions

```
#include "rtklib.h"
```

Functions

- void **writesolhead** ([stream_t](#) *stream, const [solopt_t](#) *solopt)
- void **saveoutbuf** ([rtksvr_t](#) *svr, unsigned char *buff, int n, int index)
- void **writesol** ([rtksvr_t](#) *svr, int index)
- void **updatenav** ([nav_t](#) *nav)
- void **updatefcn** ([rtksvr_t](#) *svr)
- void **updatesvr** ([rtksvr_t](#) *svr, int ret, [obs_t](#) *obs, [nav_t](#) *nav, int sat, [sbsmsg_t](#) *sbsmsg, int index, int iobs)
- int **decoderaw** ([rtksvr_t](#) *svr, int index)
- void **decodefile** ([rtksvr_t](#) *svr, int index)
- void * **rtksvrthread** (void *arg)
- int **rtksvrinit** ([rtksvr_t](#) *svr)
- void **rtksvrfree** ([rtksvr_t](#) *svr)
- void **rtksvrlock** ([rtksvr_t](#) *svr)
- void **rtksvrunlock** ([rtksvr_t](#) *svr)
- int **rtksvrstart** ([rtksvr_t](#) *svr, int cycle, int bufsize, int *strs, char **paths, const int *formats, int navsel, char **cmds, char **rcvopts, int nmeacycle, int nmeareq, const double *nmeapos, [prcopt_t](#) *prcopt, [solopt_t](#) *solopt, [stream_t](#) *moni)
- void **rtksvrstop** ([rtksvr_t](#) *svr, char **cmds)
- int **rtksvropenstr** ([rtksvr_t](#) *svr, int index, int str, const char *path, const [solopt_t](#) *solopt)
- void **rtksvrclosestr** ([rtksvr_t](#) *svr, int index)
- int **rtksvrostat** ([rtksvr_t](#) *svr, int rcv, [gtime_t](#) *time, int *sat, double *az, double *el, int **snr, int *vsat)
- void **rtksvrsstat** ([rtksvr_t](#) *svr, int *sstat, char *msg)

Variables

- const [solopt_t](#) **SOLOPT_DEFAULT**
- const [prcopt_t](#) **PRCOPT_DEFAULT**

11.290.1 Detailed Description

rtk server functions

Authors

- 2007-2013, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

11.290.2 Variable Documentation

11.290.2.1 PRCOPT_DEFAULT

```
const prcopt_t PRCOPT_DEFAULT
```

Initial value:

```
= {
    PMODE_SINGLE, 0, 2, SYS_GPS,
    15.0 * D2R, {}, {}, {},
    0, 1, 1, 1,
    5, 0, 10, 1,
    0, 0, 0, 0,
    1, 0, 0, 0, 0,
    0, 0,
    {100.0, 100.0, 100.0},
    {100.0, 0.003, 0.003, 0.0, 1.0},
    {30.0, 0.03, 0.3},
    {1e-4, 1e-3, 1e-4, 1e-1, 1e-2, 0.0},
    5E-12,
    {3.0, 0.9999, 0.25, 0.1, 0.05, 0, 0, 0},
    0.0, 0.0, 0.05,
    30.0, 30.0, 30.0,
    {}, {}, {},
    {"", ""},
    {}, {}, {},
    0, 0, 0, {"", ""}, {}, 0, {}, {}, {}, {}, {}, {}, {}, {}, {}, 0, {}
}
```

Definition at line 49 of file rtklib_rtksvr.h.

11.290.2.2 SOLOPT_DEFAULT

```
const solopt_t SOLOPT_DEFAULT
```

Initial value:

```
= {
    SOLF_LLH, TIMES_GPST, 1, 3,
    0, 1, 0, 0, 0, 0,
    0, 0, 0,
    {0.0, 0.0},
    " ", "", 0
}
```

Definition at line 40 of file rtklib_rtksvr.h.

11.291 rtklib_sbas.h File Reference

sbas functions

```
#include "rtklib.h"
```

Functions

- char * **getfield** (char *p, int pos)
- double **varfcorr** (int udre)
- double **varicorr** (int give)
- double **degfcorr** (int ai)
- int **decode_sbstype1** (const [sbsmsg_t](#) *msg, [sbssat_t](#) *sbssat)
- int **decode_sbstype2** (const [sbsmsg_t](#) *msg, [sbssat_t](#) *sbssat)
- int **decode_sbstype6** (const [sbsmsg_t](#) *msg, [sbssat_t](#) *sbssat)
- int **decode_sbstype7** (const [sbsmsg_t](#) *msg, [sbssat_t](#) *sbssat)
- int **decode_sbstype9** (const [sbsmsg_t](#) *msg, [nav_t](#) *nav)
- int **decode_sbstype18** (const [sbsmsg_t](#) *msg, [sbsion_t](#) *sbsion)
- int **decode_longcorr0** (const [sbsmsg_t](#) *msg, int p, [sbssat_t](#) *sbssat)
- int **decode_longcorr1** (const [sbsmsg_t](#) *msg, int p, [sbssat_t](#) *sbssat)
- int **decode_longcorrh** (const [sbsmsg_t](#) *msg, int p, [sbssat_t](#) *sbssat)
- int **decode_sbstype24** (const [sbsmsg_t](#) *msg, [sbssat_t](#) *sbssat)
- int **decode_sbstype25** (const [sbsmsg_t](#) *msg, [sbssat_t](#) *sbssat)
- int **decode_sbstype26** (const [sbsmsg_t](#) *msg, [sbsion_t](#) *sbsion)
- int **sbsupdatecorr** (const [sbsmsg_t](#) *msg, [nav_t](#) *nav)
- void **readmsgs** (const char *file, int sel, [gtime_t](#) ts, [gtime_t](#) te, [sbs_t](#) *sbs)
- int **cmpmsgs** (const void *p1, const void *p2)
- int **sbsreadmsgt** (const char *file, int sel, [gtime_t](#) ts, [gtime_t](#) te, [sbs_t](#) *sbs)
- int **sbsreadmsg** (const char *file, int sel, [sbs_t](#) *sbs)
- void **sbsoutmsg** (FILE *fp, [sbsmsg_t](#) *sbsmsg)
- void **searchigp** ([gtime_t](#) time, const double *pos, const [sbsion_t](#) *ion, const [sbsigp_t](#) **igp, double *x, double *y)
- int **sbsioncorr** ([gtime_t](#) time, const [nav_t](#) *nav, const double *pos, const double *azel, double *delay, double *var)
- void **getmet** (double lat, double *met)
- double **sbstropcorr** ([gtime_t](#) time, const double *pos, const double *azel, double *var)
- int **sbslongcorr** ([gtime_t](#) time, int sat, const [sbssat_t](#) *sbssat, double *drs, double *ddts)
- int **sbsfastcorr** ([gtime_t](#) time, int sat, const [sbssat_t](#) *sbssat, double *prc, double *var)
- int **sbssatcorr** ([gtime_t](#) time, int sat, const [nav_t](#) *nav, double *rs, double *dts, double *var)
- int **sbsdecodemsg** ([gtime_t](#) time, int prn, const unsigned int *words, [sbsmsg_t](#) *sbsmsg)

Variables

- const int **WEEKOFFSET** = 1024
- const [sbsigpband_t](#) **IGPBAND1** [9][8]
- const [sbsigpband_t](#) **IGPBAND2** [2][5]

11.291.1 Detailed Description

sbas functions

Authors

- 2007-2013, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

References : [1] RTCA/DO-229C, Minimum operational performance standards for global positioning system/wide area augmentation system airborne equipment, RTCA inc, November 28, 2001 [2] IS-QZSS v.1.1, Quasi-Zenith Satellite System Navigation Service Interface Specification for QZSS, Japan Aerospace Exploration Agency, July 31, 2009

11.291.2 Variable Documentation

11.291.2.1 IGPBAND1

```
const sbsigband_t IGPBAND1[9][8]
```

Initial value:

```
= {
  {{-180, X1, 1, 28}, {-175, X2, 29, 51}, {-170, X3, 52, 78}, {-165, X2, 79, 101},
   {-160, X3, 102, 128}, {-155, X2, 129, 151}, {-150, X3, 152, 178}, {-145, X2, 179, 201}},
  {{-140, X4, 1, 28}, {-135, X2, 29, 51}, {-130, X3, 52, 78}, {-125, X2, 79, 101},
   {-120, X3, 102, 128}, {-115, X2, 129, 151}, {-110, X3, 152, 178}, {-105, X2, 179, 201}},
  {{-100, X3, 1, 27}, {-95, X2, 28, 50}, {-90, X1, 51, 78}, {-85, X2, 79, 101},
   {-80, X3, 102, 128}, {-75, X2, 129, 151}, {-70, X3, 152, 178}, {-65, X2, 179, 201}},
  {{-60, X3, 1, 27}, {-55, X2, 28, 50}, {-50, X4, 51, 78}, {-45, X2, 79, 101},
   {-40, X3, 102, 128}, {-35, X2, 129, 151}, {-30, X3, 152, 178}, {-25, X2, 179, 201}},
  {{-20, X3, 1, 27}, {-15, X2, 28, 50}, {-10, X3, 51, 77}, {-5, X2, 78, 100},
   {0, X1, 101, 128}, {5, X2, 129, 151}, {10, X3, 152, 178}, {15, X2, 179, 201}},
  {{20, X3, 1, 27}, {25, X2, 28, 50}, {30, X3, 51, 77}, {35, X2, 78, 100},
   {40, X4, 101, 128}, {45, X2, 129, 151}, {50, X3, 152, 178}, {55, X2, 179, 201}},
  {{60, X3, 1, 27}, {65, X2, 28, 50}, {70, X3, 51, 77}, {75, X2, 78, 100},
   {80, X3, 101, 127}, {85, X2, 128, 150}, {90, X1, 151, 178}, {95, X2, 179, 201}},
  {{100, X3, 1, 27}, {105, X2, 28, 50}, {110, X3, 51, 77}, {115, X2, 78, 100},
   {120, X3, 101, 127}, {125, X2, 128, 150}, {130, X4, 151, 178}, {135, X2, 179, 201}},
  {{140, X3, 1, 27}, {145, X2, 28, 50}, {150, X3, 51, 77}, {155, X2, 78, 100},
   {160, X3, 101, 127}, {165, X2, 128, 150}, {170, X3, 151, 177}, {175, X2, 178, 200}}}
```

Definition at line 73 of file rtklib_sbass.h.

11.291.2.2 IGPBAND2

```
const sbsigpband_t IGPBAND2[2][5]
```

Initial value:

```
= {
    {{60, X5, 1, 72}, {65, X6, 73, 108}, {70, X6, 109, 144}, {75, X6, 145, 180},
     {85, X7, 181, 192}},
    {{-60, X5, 1, 72}, {-65, X6, 73, 108}, {-70, X6, 109, 144}, {-75, X6, 145, 180},
     {-85, X8, 181, 192}}
```

Definition at line 92 of file rtklib_sbas.h.

11.292 rtklib_solution.h File Reference

solution functions headers

```
#include "rtklib.h"
```

Macros

- `#define COMMENTH "%" /* comment line indicator for solution */`
- `#define MSG_DISCONNECT "$_DISCONNECT\r\n" /* disconnect message */`

Functions

- `const char * opt2sep (const solopt_t *opt)`
- `int tonum (char *buff, const char *sep, double *v)`
- `double sqvar (double covar)`
- `double dmm2deg (double dmm)`
- `void septime (double t, double *t1, double *t2, double *t3)`
- `void soltocov (const sol_t *sol, double *P)`
- `void covtosol (const double *P, sol_t *sol)`
- `int decode_nmearmc (char **val, int n, sol_t *sol)`
- `int decode_nmeagga (char **val, int n, sol_t *sol)`
- `int decode_nmea (char *buff, sol_t *sol)`
- `char * decode_soltime (char *buff, const solopt_t *opt, gtime_t *time)`
- `int decode_solxyz (char *buff, const solopt_t *opt, sol_t *sol)`
- `int decode_solllh (char *buff, const solopt_t *opt, sol_t *sol)`
- `int decode_solenu (char *buff, const solopt_t *opt, sol_t *sol)`
- `int decode_solgsi (char *buff, const solopt_t *opt, sol_t *sol)`
- `int decode_solpos (char *buff, const solopt_t *opt, sol_t *sol)`
- `void decode_refpos (char *buff, const solopt_t *opt, double *rb)`
- `int decode_sol (char *buff, const solopt_t *opt, sol_t *sol, double *rb)`
- `void decode_solopt (char *buff, solopt_t *opt)`
- `void readsolopt (FILE *fp, solopt_t *opt)`
- `int inputsol (unsigned char data, gtime_t ts, gtime_t te, double tint, int qflag, const solopt_t *opt, solbuf_t *solbuf)`
- `int readsoldata (FILE *fp, gtime_t ts, gtime_t te, double tint, int qflag, const solopt_t *opt, solbuf_t *solbuf)`

- int **cmpsol** (const void *p1, const void *p2)
- int **sort_solbuf** (solbuf_t *solbuf)
- int **readsolt** (char *files[], int nfile, gtime_t ts, gtime_t te, double tint, int qflag, solbuf_t *solbuf)
- int **readsol** (char *files[], int nfile, solbuf_t *sol)
- int **addsol** (solbuf_t *solbuf, const sol_t *sol)
- sol_t * **getsol** (solbuf_t *solbuf, int index)
- void **initsolbuf** (solbuf_t *solbuf, int cyclic, int nmax)
- void **freesolbuf** (solbuf_t *solbuf)
- void **freesolstatbuf** (solstatbuf_t *solstatbuf)
- int **cmpsolstat** (const void *p1, const void *p2)
- int **sort_solstat** (solstatbuf_t *statbuf)
- int **decode_solstat** (char *buff, solstat_t *stat)
- void **addsolstat** (solstatbuf_t *statbuf, const solstat_t *stat)
- int **readsolstatdata** (FILE *fp, gtime_t ts, gtime_t te, double tint, solstatbuf_t *statbuf)
- int **readsolstatt** (char *files[], int nfile, gtime_t ts, gtime_t te, double tint, solstatbuf_t *statbuf)
- int **readsolstat** (char *files[], int nfile, solstatbuf_t *statbuf)
- int **outecef** (unsigned char *buff, const char *s, const sol_t *sol, const solopt_t *opt)
- int **outpos** (unsigned char *buff, const char *s, const sol_t *sol, const solopt_t *opt)
- int **outenu** (unsigned char *buff, const char *s, const sol_t *sol, const double *rb, const solopt_t *opt)
- int **outnmea_rmc** (unsigned char *buff, const sol_t *sol)
- int **outnmea_gga** (unsigned char *buff, const sol_t *sol)
- int **outnmea_gsa** (unsigned char *buff, const sol_t *sol, const ssat_t *ssat)
- int **outnmea_gsv** (unsigned char *buff, const sol_t *sol, const ssat_t *ssat)
- int **outprcopts** (unsigned char *buff, const prcopt_t *opt)
- int **outsolheads** (unsigned char *buff, const solopt_t *opt)
- int **outsols** (unsigned char *buff, const sol_t *sol, const double *rb, const solopt_t *opt)
- int **outsolexs** (unsigned char *buff, const sol_t *sol, const ssat_t *ssat, const solopt_t *opt)
- void **outprcopt** (FILE *fp, const prcopt_t *opt)
- void **outsolhead** (FILE *fp, const solopt_t *opt)
- void **outsol** (FILE *fp, const sol_t *sol, const double *rb, const solopt_t *opt)
- void **outsolex** (FILE *fp, const sol_t *sol, const ssat_t *ssat, const solopt_t *opt)

11.292.1 Detailed Description

solution functions headers

Authors

- 2007-2013, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

11.293 rtklib_solver.h File Reference

PVT solver based on rtklib library functions adapted to the GNSS-SDR data flow and structures.

```
#include "beidou_dnav_almanac.h"
#include "beidou_dnav_ephemeris.h"
#include "beidou_dnav_iono.h"
#include "beidou_dnav_utc_model.h"
#include "galileo_almanac.h"
#include "galileo_ephemeris.h"
#include "galileo_iono.h"
#include "galileo_utc_model.h"
#include "glonass_gnav_almanac.h"
#include "glonass_gnav_ephemeris.h"
#include "glonass_gnav_utc_model.h"
#include "gnss_synchro.h"
#include "gps_almanac.h"
#include "gps_cnav_ephemeris.h"
#include "gps_cnav_iono.h"
#include "gps_cnav_utc_model.h"
#include "gps_ephemeris.h"
#include "gps_iono.h"
#include "gps_utc_model.h"
#include "monitor_pvt.h"
#include "pvt_solution.h"
#include "rtklib.h"
#include <array>
#include <fstream>
#include <map>
#include <string>
```

Classes

- class [Rtklib_Solver](#)

This class implements a PVT solution based on RTKLIB.

11.293.1 Detailed Description

PVT solver based on rtklib library functions adapted to the GNSS-SDR data flow and structures.

Authors

- 2017, Javier Arribas
- 2017, Carles Fernandez
- 2007-2013, T. Takasu

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017-2019, Javier Arribas Copyright (C) 2017-2019, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

11.294 rtklib_solver_dump_reader.h File Reference

Helper file for unit testing.

```
#include <cstdint>
#include <fstream>
#include <string>
#include <vector>
```

Classes

- class [Rtklib_Solver_Dump_Reader](#)

11.294.1 Detailed Description

Helper file for unit testing.

Author

Javier Arribas, 2017. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.295 rtklib_stream.h File Reference

streaming functions

```
#include "rtklib.h"
```

Macros

- #define **TINTACT** 200 /* period for stream active (ms) */
- #define **SERIBUFSIZE** 4096 /* serial buffer size (bytes) */
- #define **TIMETAGH_LEN** 64 /* time tag file header length */
- #define **MAXCLI** 32 /* max client connection for tcp svr */
- #define **MAXSTATMSG** 32 /* max length of status message */
- #define **VER_RTKLIB** "2.4.2"
- #define **NTRIP_AGENT** "RTKLIB/" VER_RTKLIB
- #define **NTRIP_CLI_PORT** 2101 /* default ntrip-client connection port */
- #define **NTRIP_SVR_PORT** 80 /* default ntrip-server connection port */
- #define **NTRIP_MAXRSP** 32768 /* max size of ntrip response */
- #define **NTRIP_MAXSTR** 256 /* max length of mountpoint string */
- #define **NTRIP_RSP_OK_CLI** "ICY 200 OK\r\n" /* ntrip response: client */
- #define **NTRIP_RSP_OK_SVR** "OK\r\n" /* ntrip response: server */
- #define **NTRIP_RSP_SRCTBL** "SOURCETABLE 200 OK\r\n" /* ntrip response: source table */
- #define **NTRIP_RSP_TBLEND** "ENDSOURCETABLE"
- #define **NTRIP_RSP_HTTP** "HTTP/" /* ntrip response: http */
- #define **NTRIP_RSP_ERROR** "ERROR" /* ntrip response: error */
- #define **FTP_CMD** "wget" /* ftp/http command */
- #define **FTP_TIMEOUT** 30 /* ftp/http timeout (s) */

Functions

- [serial_t](#) * **openserial** (const char *path, int mode, char *msg)
- void **closeserial** ([serial_t](#) *serial)
- int **readserial** ([serial_t](#) *serial, unsigned char *buff, int n, char *msg)
- int **writeserial** ([serial_t](#) *serial, unsigned char *buff, int n, char *msg)
- int **stateserial** ([serial_t](#) *serial)
- int **openfile_** ([file_t](#) *file, [gtime_t](#) time, char *msg)
- void **closefile_** ([file_t](#) *file)
- [file_t](#) * **openfile** (const char *path, int mode, char *msg)
- void **closefile** ([file_t](#) *file)
- void **swapfile** ([file_t](#) *file, [gtime_t](#) time, char *msg)
- void **swapclose** ([file_t](#) *file)
- int **statefile** ([file_t](#) *file)
- int **readfile** ([file_t](#) *file, unsigned char *buff, int nmax, char *msg)
- int **writefile** ([file_t](#) *file, unsigned char *buff, int n, char *msg)
- void **syncfile** ([file_t](#) *file1, [file_t](#) *file2)
- void **decodetcppath** (const char *path, char *addr, char *port, char *user, char *passwd, char *mntpt, char *str)
- int **errsock** ()
- int **setsock** (socket_t sock, char *msg)
- socket_t **accept_nb** (socket_t sock, struct sockaddr *addr, socklen_t *len)
- int **connect_nb** (socket_t sock, struct sockaddr *addr, socklen_t len)
- int **recv_nb** (socket_t sock, unsigned char *buff, int n)
- int **send_nb** (socket_t sock, unsigned char *buff, int n)
- int **gentcp** ([tcp_t](#) *tcp, int type, char *msg)
- void **discontcp** ([tcp_t](#) *tcp, int tcon)
- [tcpsvr_t](#) * **opentcpsvr** (const char *path, char *msg)
- void **closetcpsvr** ([tcpsvr_t](#) *tcpsvr)
- void **updatetcpsvr** ([tcpsvr_t](#) *tcpsvr, char *msg)
- int **accsock** ([tcpsvr_t](#) *tcpsvr, char *msg)
- int **waittcpsvr** ([tcpsvr_t](#) *tcpsvr, char *msg)
- int **readtcpsvr** ([tcpsvr_t](#) *tcpsvr, unsigned char *buff, int n, char *msg)
- int **writetcpsvr** ([tcpsvr_t](#) *tcpsvr, unsigned char *buff, int n, char *msg)
- int **statetcpsvr** ([tcpsvr_t](#) *tcpsvr)
- int **consock** ([tcpcli_t](#) *tcpcli, char *msg)
- [tcpcli_t](#) * **opentcpcli** (const char *path, char *msg)
- void **closetcpcli** ([tcpcli_t](#) *tcpcli)
- int **waittcpcli** ([tcpcli_t](#) *tcpcli, char *msg)
- int **readtcpcli** ([tcpcli_t](#) *tcpcli, unsigned char *buff, int n, char *msg)
- int **writetcpcli** ([tcpcli_t](#) *tcpcli, unsigned char *buff, int n, char *msg)
- int **statetcpcli** ([tcpcli_t](#) *tcpcli)
- int **enbase64** (char *str, const unsigned char *byte, int n)
- int **reqntrip_s** ([ntrip_t](#) *ntrip, char *msg)
- int **reqntrip_c** ([ntrip_t](#) *ntrip, char *msg)
- int **rspntrip_s** ([ntrip_t](#) *ntrip, char *msg)
- int **rspntrip_c** ([ntrip_t](#) *ntrip, char *msg)
- int **waitntrip** ([ntrip_t](#) *ntrip, char *msg)
- [ntrip_t](#) * **openntrip** (const char *path, int type, char *msg)
- void **closetrip** ([ntrip_t](#) *ntrip)
- int **readntrip** ([ntrip_t](#) *ntrip, unsigned char *buff, int n, char *msg)
- int **writenrip** ([ntrip_t](#) *ntrip, unsigned char *buff, int n, char *msg)
- int **statenrip** ([ntrip_t](#) *ntrip)
- void **decodeftppath** (const char *path, char *addr, char *file, char *user, char *passwd, int *topts)
- [gtime_t](#) **nextdltime** (const int *topts, int stat)

- void * **ftpthread** (void *arg)
- **ftp_t** * **openftp** (const char *path, int type, char *msg)
- void **closeftp** (**ftp_t** *ftp)
- int **readftp** (**ftp_t** *ftp, unsigned char *buff, int n, char *msg)
- int **stateftp** (**ftp_t** *ftp)
- void **strinitcom** ()
- void **strinit** (**stream_t** *stream)
- int **stropen** (**stream_t** *stream, int type, int mode, const char *path)
- void **strclose** (**stream_t** *stream)
- void **strsync** (**stream_t** *stream1, **stream_t** *stream2)
- void **strlock** (**stream_t** *stream)
- void **strunlock** (**stream_t** *stream)
- int **stread** (**stream_t** *stream, unsigned char *buff, int n)
- int **strwrite** (**stream_t** *stream, unsigned char *buff, int n)
- int **strstat** (**stream_t** *stream, char *msg)
- void **strsum** (**stream_t** *stream, int *inb, int *inr, int *outb, int *outr)
- void **strsetopt** (const int *opt)
- void **strsettimeout** (**stream_t** *stream, int inactive_timeout, int tirecon)
- void **strsetdir** (const char *dir)
- void **strsetproxy** (const char *addr)
- **gtime_t** **strgettime** (**stream_t** *stream)
- void **strsendnmea** (**stream_t** *stream, const double *pos)
- int **gen_hex** (const char *msg, unsigned char *buff)
- void **strsendcmd** (**stream_t** *str, const char *cmd)

11.295.1 Detailed Description

streaming functions

Authors

- 2007-2013, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2007-2013, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

11.296 rtklib_tides.h File Reference

Tidal displacement corrections.

```
#include "rtklib.h"
```

Functions

- void **tide_pl** (const double *eu, const double *rp, double GMp, const double *pos, double *dr)
- void **tide_solid** (const double *rsun, const double *rmoon, const double *pos, const double *E, double gmst, int opt, double *dr)
- void **tide_oload** ([gtime_t](#) tut, const double *odisp, double *denu)
- void **iers_mean_pole** ([gtime_t](#) tut, double *xp_bar, double *yp_bar)
- void **tide_pole** ([gtime_t](#) tut, const double *pos, const double *erpv, double *denu)
- void **tidedisp** ([gtime_t](#) tutc, const double *rr, int opt, const [erp_t](#) *erp, const double *odisp, double *dr)

Variables

- const double **GME** = 3.986004415E+14
- const double **GMS** = 1.327124E+20
- const double **GMM** = 4.902801E+12

11.296.1 Detailed Description

Tidal displacement corrections.

Authors

- 2015, T. Takasu
- 2017, Javier Arribas
- 2017, Carles Fernandez

This is a derived work from RTKLIB <http://www.rtklib.com/> The original source code at <https://github.com/tomojitakasu/RTKLIB> is released under the BSD 2-clause license with an additional exclusive clause that does not apply here. This additional clause is reproduced below:

" The software package includes some companion executive binaries or shared libraries necessary to execute APs on Windows. These licenses succeed to the original ones of these software. "

Neither the executive binaries nor the shared libraries are required by, used or included in GNSS-SDR.

Copyright (C) 2015, T. Takasu Copyright (C) 2017, Javier Arribas Copyright (C) 2017, Carles Fernandez All rights reserved.

SPDX-License-Identifier: BSD-2-Clause

References: [1] D.D.McCarthy, IERS Technical Note 21, IERS Conventions 1996, July 1996 [2] D.D.McCarthy and G.Petit, IERS Technical Note 32, IERS Conventions 2003, November 2003 [3] D.A.Vallado, Fundamentals of Astrodynamics and Applications 2nd ed, Space Technology Library, 2004 [4] J.Kouba, A Guide to using International GNSS Service (IGS) products, May 2009 [5] G.Petit and B.Luzum (eds), IERS Technical Note No. 36, IERS

Conventions (2010), 2010

11.297 rtl_tcp_commands.h File Reference

Defines structures and constants for communicating with rtl_tcp.

```
#include <boost/asio/ip/tcp.hpp>
#include <boost/system/error_code.hpp>
```

Enumerations

- enum [RTL_TCP_COMMAND](#) {
RTL_TCP_SET_FREQUENCY = 1, **RTL_TCP_SET_SAMPLE_RATE** = 2, **RTL_TCP_SET_GAIN_MODE** =
3, **RTL_TCP_SET_GAIN** = 4,
RTL_TCP_SET_IF_GAIN = 6, **RTL_TCP_SET_AGC_MODE** = 8 }
Command IDs for configuration rtl_tcp.

Functions

- boost::system::error_code [rtl_tcp_command](#) ([RTL_TCP_COMMAND](#) id, unsigned param, boost::asio::ip::tcp::socket &socket)
Send a command to rtl_tcp over the given socket.

11.297.1 Detailed Description

Defines structures and constants for communicating with rtl_tcp.

Author

Anthony Arnold, 2015. anthony.arnold@uqconnect.edu.au

This file contains information taken from librtlsdr:

<https://git.osmocom.org/rtl-sdr>

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.297.2 Enumeration Type Documentation

11.297.2.1 RTL_TCP_COMMAND

enum [RTL_TCP_COMMAND](#)

Command IDs for configuration rtl_tcp.

Definition at line 28 of file rtl_tcp_commands.h.

11.297.3 Function Documentation

11.297.3.1 rtl_tcp_command()

```
boost::system::error_code rtl_tcp_command (
    RTL\_TCP\_COMMAND id,
    unsigned param,
    boost::asio::ip::tcp::socket & socket )
```

Send a command to rtl_tcp over the given socket.

11.298 rtl_tcp_dongle_info.h File Reference

Interface for a structure sent by rtl_tcp defining the hardware.

```
#include <boost/asio/ip/tcp.hpp>
```

Classes

- class [Rtl_Tcp_Dongle_Info](#)

This class represents the dongle information which is sent by rtl_tcp.

11.298.1 Detailed Description

Interface for a structure sent by rtl_tcp defining the hardware.

Author

Anthony Arnold, 2015. anthony.arnold@uqconnect.edu.au

This file contains information taken from librtlsdr:

<https://git.osmocom.org/rtl-sdr>

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.299 rtl_tcp_signal_source.h File Reference

Signal source which reads from rtl_tcp. (see <https://osmocom.org/projects/rtl-sdr/wiki> for more information)

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include "rtl_tcp_signal_source_c.h"
#include <gnuradio/blocks/deinterleave.h>
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/float_to_complex.h>
#include <pmt/pmt.h>
#include <memory>
#include <stdexcept>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [RtlTcpSignalSource](#)

This class reads from rtl_tcp, which streams interleaved I/Q samples over TCP. (see <https://osmocom.org/projects/rtl-sdr/wiki>)

11.299.1 Detailed Description

Signal source which reads from rtl_tcp. (see <https://osmocom.org/projects/rtl-sdr/wiki> for more information)

Author

Anthony Arnold, 2015. anthony.arnold@uqconnect.edu.au

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.300 rtl_tcp_signal_source_c.h File Reference

Interface of an rtl_tcp signal source reader.

```
#include "rtl_tcp_dongle_info.h"
#include <boost/array.hpp>
#include <boost/asio.hpp>
#include <boost/circular_buffer.hpp>
#include <boost/thread/condition.hpp>
#include <boost/thread/mutex.hpp>
#include <gnuradio/sync_block.h>
#include <cstdint>
#include <string>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [rtl_tcp_signal_source_c](#)

This class reads interleaved I/Q samples from an rtl_tcp server and outputs complex types.

Typedefs

- using **rtl_tcp_signal_source_c_sptr** = boost::shared_ptr< [rtl_tcp_signal_source_c](#) >
- using **b_io_context** = boost::asio::io_service

Functions

- rtl_tcp_signal_source_c_sptr **rtl_tcp_make_signal_source_c** (const std::string &address, int16_t port, bool flip_iq=false)

11.300.1 Detailed Description

Interface of an rtl_tcp signal source reader.

Author

Anthony Arnold, 2015. anthony.arnold@uqconnect.edu.au

The implementation of this block is a combination of various helpful sources. The data format and command structure is taken from the original Osmocom rtl_tcp_source_f (<https://git.osmocom.org/gr-osmosdr>). The asynchronous reading code comes from the examples provides by Boost.Asio and the bounded buffer producer-consumer solution is taken from the Boost.CircularBuffer examples (<https://www.boost.org/>).

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.301 sbas_ephemeris.h File Reference

Interface of a SBAS REFERENCE LOCATION storage.

```
#include <ostream>
```

Classes

- class [Sbas_Ephemeris](#)

This class stores SBAS SV ephemeris data.

11.301.1 Detailed Description

Interface of a SBAS REFERENCE LOCATION storage.

Author

Daniel Fehr, 2013. [daniel.co\(at\)bluewin.ch](mailto:daniel.co(at)bluewin.ch)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.302 sbas_l1_telemetry_decoder.h File Reference

Interface of an adapter of a SBAS telemetry data decoder block to a [TelemetryDecoderInterface](#).

```
#include "gnss_satellite.h"
#include "gnss_synchro.h"
#include "sbas_l1_telemetry_decoder_gs.h"
#include "telemetry_decoder_interface.h"
#include <gnuradio/runtime_types.h>
#include <cstdint>
#include <string>
```

Classes

- class [SbasL1TelemetryDecoder](#)

This class implements a NAV data decoder for SBAS frames in L1 radio link.

11.302.1 Detailed Description

Interface of an adapter of a SBAS telemetry data decoder block to a [TelemetryDecoderInterface](#).

Author

Daniel Fehr 2013. daniel.co(at)bluewin.ch

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.303 sbas_l1_telemetry_decoder_gs.h File Reference

Interface of a SBAS telemetry data decoder block.

```
#include "gnss_satellite.h"
#include <boost/crc.hpp>
#include <gnuradio/block.h>
#include <gnuradio/types.h>
#include <cstdint>
#include <cstdint>
#include <deque>
#include <fstream>
#include <memory>
#include <string>
#include <utility>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [sbas_l1_telemetry_decoder_gs](#)

This class implements a block that decodes the SBAS integrity and corrections data defined in RTCA MOPS DO-229.

Typedefs

- using **sbas_l1_telemetry_decoder_gs_sptr** = boost::shared_ptr< [sbas_l1_telemetry_decoder_gs](#) >

Functions

- **sbas_l1_telemetry_decoder_gs_sptr sbas_l1_make_telemetry_decoder_gs** (const [Gnss_Satellite](#) &satellite, bool dump)

11.303.1 Detailed Description

Interface of a SBAS telemetry data decoder block.

Author

Daniel Fehr 2013. daniel.co(at)bluewin.ch

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.304 serdes_gnss_synchro.h File Reference

Serialization / Deserialization of [Gnss_Synchro](#) objects using Protocol Buffers.

```
#include "gnss_synchro.h"
#include "gnss_synchro.pb.h"
#include <array>
#include <string>
#include <utility>
#include <vector>
```

Classes

- class [Serdes_Gnss_Synchro](#)

This class implements serialization and deserialization of [Gnss_Synchro](#) objects using Protocol Buffers.

11.304.1 Detailed Description

Serialization / Deserialization of [Gnss_Synchro](#) objects using Protocol Buffers.

Author

Carles Fernandez-Prades, 2019. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.305 serdes_monitor_pvt.h File Reference

Serialization / Deserialization of [Monitor_Pvt](#) objects using Protocol Buffers.

```
#include "monitor_pvt.h"
#include "monitor_pvt.pb.h"
#include <memory>
#include <string>
#include <utility>
```

Classes

- class [Serdes_Monitor_Pvt](#)

This class implements serialization and deserialization of [Monitor_Pvt](#) objects using Protocol Buffers.

11.305.1 Detailed Description

Serialization / Deserialization of [Monitor_Pvt](#) objects using Protocol Buffers.

Author

Carles Fernandez-Prades, 2019. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.306 short_x2_to_cshort.h File Reference

Adapts two short streams into a `std::complex<short>` stream.

```
#include <boost/shared_ptr.hpp>
#include <gnuradio/sync_block.h>
#include <gnuradio/types.h>
```

Classes

- class [short_x2_to_cshort](#)

This class adapts two short streams into a `std::complex<short>` stream.

Typedefs

- using `short_x2_to_cshort_sptr` = `boost::shared_ptr< short_x2_to_cshort >`

Functions

- short_x2_to_cshort_sptr **make_short_x2_to_cshort** ()

11.306.1 Detailed Description

Adapts two short streams into a `std::complex<short>` stream.

Author

Carles Fernandez Prades, cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.307 signal_conditioner.h File Reference

It wraps blocks to change data type, filter and resample input data.

```
#include "gnss_block_interface.h"
#include <gnuradio/block.h>
#include <cstdint>
#include <memory>
#include <string>
```

Classes

- class [SignalConditioner](#)

This class wraps blocks to change data_type_adapter, input_filter and resampler to be applied to the input flow of sampled signal.

11.307.1 Detailed Description

It wraps blocks to change data type, filter and resample input data.

Author

Luis Esteve, 2012. luis(at)epsilon-formacion.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.308 signal_generator.h File Reference

Adapter of a class that generates synthesized GNSS signal.

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include "signal_generator_c.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/vector_to_stream.h>
#include <gnuradio/hier_block2.h>
#include <pmt/pmt.h>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class [SignalGenerator](#)

This class generates synthesized GNSS signal.

11.308.1 Detailed Description

Adapter of a class that generates synthesized GNSS signal.

Author

Marc Molina, 2013. marc.molina.pena@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.309 signal_generator_c.h File Reference

GNU Radio source block that generates synthesized GNSS signal.

```
#include "gnss_signal.h"
#include <gnuradio/block.h>
#include <random>
#include <string>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [signal_generator_c](#)
This class generates synthesized GNSS signal.

Typedefs

- using **signal_generator_c_sptr** = boost::shared_ptr< [signal_generator_c](#) >

Functions

- [signal_generator_c_sptr](#) [signal_make_generator_c](#) (const std::vector< std::string > &signal1, const std::vector< std::string > &system, const std::vector< unsigned int > &PRN, const std::vector< float > &C_{N0}_dB, const std::vector< float > &doppler_Hz, const std::vector< unsigned int > &delay_chips, const std::vector< unsigned int > &delay_sec, bool data_flag, bool noise_flag, unsigned int fs_in, unsigned int vector_length, float BW_BB)
Return a shared_ptr to a new instance of gen_source.

11.309.1 Detailed Description

GNU Radio source block that generates synthesized GNSS signal.

Author

Marc Molina, 2013. marc.molina.pena@gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.309.2 Function Documentation

11.309.2.1 signal_make_generator_c()

```
signal_generator_c_sptr signal_make_generator_c (
    const std::vector< std::string > & signal1,
    const std::vector< std::string > & system,
    const std::vector< unsigned int > & PRN,
    const std::vector< float > & CN0_dB,
    const std::vector< float > & doppler_Hz,
    const std::vector< unsigned int > & delay_chips,
    const std::vector< unsigned int > & delay_sec,
    bool data_flag,
    bool noise_flag,
    unsigned int fs_in,
    unsigned int vector_length,
    float BW_BB )
```

Return a shared_ptr to a new instance of gen_source.

To avoid accidental use of raw pointers, gen_source's constructor is private. signal_make_generator_c is the public interface for creating new instances.

11.310 signal_generator_flags.h File Reference

Helper file for unit testing.

```
#include <gflags/gflags.h>
#include <limits>
#include <string>
```

Functions

- **DEFINE_bool** (disable_generator, false, "Disable the signal generator (a external signal file must be available for the test)")
- **DEFINE_string** (generator_binary, std::string(SW_GENERATOR_BIN), "Path of software-defined signal generator binary")
- **DEFINE_string** (rinex_nav_file, std::string(DEFAULT_RINEX_NAV), "Input RINEX navigation file")
- **DEFINE_int32** (duration, 100, "Duration of the experiment [in seconds, max = 300]")
- **DEFINE_string** (static_position, "30.286502,120.032669,100", "Static receiver position [latitude,longitude,height]")
- **DEFINE_string** (dynamic_position, "", "Observer positions file, in .csv or .nmea format")
- **DEFINE_string** (filename_rinex_obs, "sim.16o", "Filename of output RINEX navigation file")
- **DEFINE_string** (filename_raw_data, "signal_out.bin", "Filename of output raw data file")
- **DEFINE_int32** (fs_gen_sps, 2600000, "Sampling frequency [sps]")
- **DEFINE_int32** (test_satellite_PRN, 1, "PRN of the satellite under test (must be visible during the observation time)")
- **DEFINE_int32** (test_satellite_PRN2, 2, "PRN of the satellite under test (must be visible during the observation time)")
- **DEFINE_string** (test_satellite_PRN_list, "1,2,3,6,9,10,12,17,20,23,28", "List of PRN of the satellites under test (must be visible during the observation time)")
- **DEFINE_double** (CN0_dBHz, std::numeric_limits< double >::infinity(), "Enable noise generator and set the CN0 [dB-Hz]")

11.310.1 Detailed Description

Helper file for unit testing.

Author

Carles Fernandez-Prades, 2017. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.311 spir_file_signal_source.h File Reference

Implementation of a class that reads signals samples from a SPIR file and adapts it to a `SignalSourceInterface`.

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include "unpack_intspir_1bit_samples.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/file_source.h>
#include <gnuradio/blocks/throttle.h>
#include <gnuradio/hier_block2.h>
#include <pmt/pmt.h>
#include <cstdint>
#include <memory>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [SpirFileSignalSource](#)

Class that reads signals samples from a file and adapts it to a `SignalSourceInterface`.

11.311.1 Detailed Description

Implementation of a class that reads signals samples from a SPIR file and adapts it to a `SignalSourceInterface`.

Author

Fran Fabra, 2014 [fabra\(at\)ice.csic.es](mailto:fabra(at)ice.csic.es)

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is not part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.312 spir_gss6450_file_signal_source.h File Reference

Implementation of a class that reads signals samples from a SPIR file and adapts it to a `SignalSourceInterface`.

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include "gnss_sdr_valve.h"
#include "unpack_spir_gss6450_samples.h"
#include <gnuradio/blocks/deinterleave.h>
#include <gnuradio/blocks/endian_swap.h>
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/file_source.h>
```

```
#include <gnuradio/blocks/null_sink.h>
#include <gnuradio/blocks/throttle.h>
#include <gnuradio/hier_block2.h>
#include <pmt/pmt.h>
#include <cstdint>
#include <memory>
#include <string>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [SpirGSS6450FileSignalSource](#)

Class that reads signals samples from a file and adapts it to a `SignalSourceInterface`.

11.312.1 Detailed Description

Implementation of a class that reads signals samples from a SPIR file and adapts it to a `SignalSourceInterface`.

Author

Antonio Ramos, 2017 antonio.ramos(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is not part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.313 spirent_motion_csv_dump_reader.h File Reference

Helper file for unit testing.

```
#include <cstdint>
#include <fstream>
#include <string>
#include <vector>
```

Classes

- class [Spirent_Motion_Csv_Dump_Reader](#)

11.313.1 Detailed Description

Helper file for unit testing.

Author

Javier Arribas, 2018. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.314 string_converter.h File Reference

Interface of a class that interprets the contents of a string and converts it into different types.

```
#include <stdint>
#include <string>
```

Classes

- class [StringConverter](#)

Class that interprets the contents of a string and converts it into different types.

11.314.1 Detailed Description

Interface of a class that interprets the contents of a string and converts it into different types.

Author

Carlos Aviles, 2010. carlos.avilesr(at)gmail.com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.315 swift_common.h File Reference

Common definitions used throughout the libswiftnav library.

```
#include <inttypes.h>
#include <stdbool.h>
#include <stdint.h>
```

Macros

- **#define ABS(x)** ((x) < 0 ? -(x) : (x))
- **#define MIN(x, y)** (((x) < (y)) ? (x) : (y))
- **#define MAX(x, y)** (((x) > (y)) ? (x) : (y))
- **#define CLAMP_DIFF(a, b)** (MAX((a), (b)) - (b))

11.315.1 Detailed Description

Common definitions used throughout the libswiftnav library.

Author

Henry Hallam henry@swift-nav.com Fergus Noble fergus@swift-nav.com

This file was originally borrowed from libswiftnav <https://github.com/swift-nav/libswiftnav>, a portable C library implementing GNSS related functions and algorithms, and then modified by J. Arribas and C. Fernandez

Copyright (C) 2012 Swift Navigation Inc. Contact: Henry Hallam henry@swift-nav.com Fergus Noble fergus@swift-nav.com

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: LGPL-3.0-only

11.316 tcp_cmd_interface.h File Reference

Class that implements a TCP/IP telecommand command line interface for GNSS-SDR.

```
#include "concurrent_queue.h"
#include <pmt/pmt.h>
#include <array>
#include <cstdint>
#include <ctime>
#include <functional>
#include <memory>
#include <string>
#include <unordered_map>
#include <vector>
```

Classes

- class [TcpCmdInterface](#)

11.316.1 Detailed Description

Class that implements a TCP/IP telecommand command line interface for GNSS-SDR.

Javier Arribas jarribas (at) cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.317 tcp_communication.h File Reference

Interface of the TCP communication class.

```
#include "tcp_packet_data.h"
#include <boost/array.hpp>
#include <boost/asio.hpp>
```

Classes

- class [Tcp_Communication](#)
TCP communication class.

Macros

- #define **NUM_TX_VARIABLES_GALILEO_E1** 13
- #define **NUM_TX_VARIABLES_GPS_L1_CA** 9
- #define **NUM_RX_VARIABLES** 4

Typedefs

- using **b_io_context** = boost::asio::io_service

11.317.1 Detailed Description

Interface of the TCP communication class.

Author

David Pubill, 2011. dpubill(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.318 tcp_packet_data.h File Reference

Interface of the TCP data packet class.

Classes

- class [Tcp_Packet_Data](#)
Class that implements a TCP data packet.

11.318.1 Detailed Description

Interface of the TCP data packet class.

Author

David Pubill, 2011. dpubill(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.319 telemetry_decoder_interface.h File Reference

This class represents an interface to a telemetry decoder block.

```
#include "gnss_block_interface.h"
#include "gnss_satellite.h"
```

Classes

- class [TelemetryDecoderInterface](#)

This abstract class represents an interface to a navigation GNSS block.

11.319.1 Detailed Description

This class represents an interface to a telemetry decoder block.

Author

Javier Arribas, 2011. jarribas(at)cttc.es

Abstract class for telemetry decoders. Since all its methods are virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.320 test_flags.h File Reference

Helper file for unit testing.

```
#include <gflags/gflags.h>
#include <string>
```

Functions

- **DEFINE_string** (gnuplot_executable, "", "Gnuplot binary path")
- **DEFINE_bool** (plot_acq_grid, false, "Plots acquisition grid with gnuplot")
- **DEFINE_int32** (plot_decimate, 1, "Decimate plots")

11.320.1 Detailed Description

Helper file for unit testing.

Author

Carles Fernandez-Prades, 2017. cfernandez(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.321 tlm_dump_reader.h File Reference

Helper file for unit testing.

```
#include <cstdint>
#include <fstream>
#include <string>
#include <vector>
```

Classes

- class [Tlm_Dump_Reader](#)

11.321.1 Detailed Description

Helper file for unit testing.

Author

Javier Arribas, 2017. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.322 tracking_2nd_DLL_filter.h File Reference

Interface of a 2nd order DLL filter for code tracking loop.

Classes

- class [Tracking_2nd_DLL_filter](#)

This class implements a 2nd order DLL filter for code tracking loop.

11.322.1 Detailed Description

Interface of a 2nd order DLL filter for code tracking loop.

Author

Javier Arribas, 2011. jarribas(at)cttc.es

Class that implements a 2nd order PLL filter for code tracking loop. The algorithm is described in: K.Borre, D.M.↔ Akos, N.Bertelsen, P.Rinder, and S. H. Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007, Applied and Numerical Harmonic Analysis.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.323 tracking_2nd_PLL_filter.h File Reference

Interface of a 2nd order PLL filter for carrier tracking loop.

Classes

- class [Tracking_2nd_PLL_filter](#)

This class implements a 2nd order PLL filter for carrier tracking loop.

11.323.1 Detailed Description

Interface of a 2nd order PLL filter for carrier tracking loop.

Author

Javier Arribas, 2011. jarribas(at)cttc.es

Class that implements 2 order PLL filter for tracking carrier loop. The algorithm is described in K.Borre, D.M.↔ Akos, N.Bertelsen, P.Rinder, and S.H. Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, Birkhauser, 2007, Applied and Numerical Harmonic Analysis.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.324 tracking_discriminators.h File Reference

Interface of a library with a set of code tracking and carrier tracking discriminators.

```
#include <gnuradio/gr_complex.h>
#include <cmath>
```

Functions

- double [fil_four_quadrant_atan](#) (gr_complex prompt_s1, gr_complex prompt_s2, double t1, double t2)
- double [fil_diff_atan](#) (gr_complex prompt_s1, gr_complex prompt_s2, double t1, double t2)
- double [phase_unwrap](#) (double phase_rad)
Phase unwrapping function, input is [rad].
- double [pll_four_quadrant_atan](#) (gr_complex prompt_s1)
PLL four quadrant arctan discriminator.
- double [pll_cloop_two_quadrant_atan](#) (gr_complex prompt_s1)
PLL Costas loop two quadrant arctan discriminator.
- double [dll_nc_e_minus_l_normalized](#) (gr_complex early_s1, gr_complex late_s1, float spc=0.5, float slope=1.0, float y_intercept=1.0)
DLL Noncoherent Early minus Late envelope normalized discriminator.
- double [dll_nc_vemlp_normalized](#) (gr_complex very_early_s1, gr_complex early_s1, gr_complex late_s1, gr_complex very_late_s1)
DLL Noncoherent Very Early Minus Late Power (VEMLP) normalized discriminator.
- template<typename Fun >
double **CalculateSlope** (Fun &&f, double x)
- template<typename Fun >
double **CalculateSlopeAbs** (Fun &&f, double x)
- template<typename Fun >
double **GetYIntercept** (Fun &&f, double x)
- template<typename Fun >
double **GetYInterceptAbs** (Fun &&f, double x)
- template<int M = 1, int N = M>
double **SinBocCorrelationFunction** (double offset_in_chips)
- template<int M = 1, int N = M>
double **CosBocCorrelationFunction** (double offset_in_chips)

11.324.1 Detailed Description

Interface of a library with a set of code tracking and carrier tracking discriminators.

Authors

- Javier Arribas, 2011. jarribas(at)cttc.es
- Luis Esteve, 2012. luis(at)epsilon-formacion.com

Library with a set of code tracking and carrier tracking discriminators that is used by the tracking algorithms.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.324.2 Function Documentation

11.324.2.1 dll_nc_e_minus_l_normalized()

```
double dll_nc_e_minus_l_normalized (
    gr_complex early_s1,
    gr_complex late_s1,
    float spc = 0.5,
    float slope = 1.0,
    float y_intercept = 1.0 )
```

DLL Noncoherent Early minus Late envelope normalized discriminator.

DLL Noncoherent Early minus Late envelope normalized discriminator:

$$error = \frac{y_{intercept} - slope * \epsilon \frac{E - L}{E + L}}{slope}, \quad (11.6)$$

where $E = \sqrt{I_{ES}^2 + Q_{ES}^2}$ is the Early correlator output absolute value and $L = \sqrt{I_{LS}^2 + Q_{LS}^2}$ is the Late correlator output absolute value. The output is in [chips].

11.324.2.2 dll_nc_vemlp_normalized()

```
double dll_nc_vemlp_normalized (
    gr_complex very_early_s1,
    gr_complex early_s1,
    gr_complex late_s1,
    gr_complex very_late_s1 )
```

DLL Noncoherent Very Early Minus Late Power (VEMLP) normalized discriminator.

DLL Noncoherent Very Early Minus Late Power (VEMLP) normalized discriminator, using the outputs of four correlators, Very Early (VE), Early (E), Late (L) and Very Late (VL):

$$error = \frac{E - L}{E + L}, \quad (11.7)$$

where $E = \sqrt{I_{VE}^2 + Q_{VE}^2 + I_E^2 + Q_E^2}$ and $L = \sqrt{I_{VL}^2 + Q_{VL}^2 + I_L^2 + Q_L^2}$. The output is in [chips].

11.324.2.3 fll_four_quadrant_atan()

```
double fll_four_quadrant_atan (
    gr_complex prompt_s1,
    gr_complex prompt_s2,
    double t1,
    double t2 )
```

brief FLL four quadrant arctan discriminator

FLL four quadrant arctan discriminator:

$$\frac{\phi_2 - \phi_1}{t_2 - t_1} = \frac{ATAN2(cross, dot)}{t_1 - t_2}, \quad (11.8)$$

where $cross = I_{PS1}Q_{PS2} - I_{PS2}Q_{PS1}$ and $dot = I_{PS1}I_{PS2} + Q_{PS1}Q_{PS2}$, I_{PS1}, Q_{PS1} are the inphase and quadrature prompt correlator outputs respectively at sample time t_1 , and I_{PS2}, Q_{PS2} are the inphase and quadrature prompt correlator outputs respectively at sample time t_2 . The output is in [radians/second].

11.324.2.4 phase_unwrap()

```
double phase_unwrap (
    double phase_rad )
```

Phase unwrapping function, input is [rad].

11.324.2.5 pll_cloop_two_quadrant_atan()

```
double pll_cloop_two_quadrant_atan (
    gr_complex prompt_sl )
```

PLL Costas loop two quadrant arctan discriminator.

PLL Costas loop two quadrant arctan discriminator:

$$\phi = ATAN\left(\frac{Q_{PS}}{I_{PS}}\right), \quad (11.9)$$

where I_{PS1} , Q_{PS1} are the inphase and quadrature prompt correlator outputs respectively. The output is in [radians].

11.324.2.6 pll_four_quadrant_atan()

```
double pll_four_quadrant_atan (
    gr_complex prompt_sl )
```

PLL four quadrant arctan discriminator.

PLL four quadrant arctan discriminator:

$$\phi = ATAN2(Q_{PS}, I_{PS}), \quad (11.10)$$

where I_{PS1} , Q_{PS1} are the inphase and quadrature prompt correlator outputs respectively. The output is in [radians].

11.325 tracking_dump_reader.h File Reference

Helper file for unit testing.

```
#include <cstdint>
#include <fstream>
#include <string>
#include <vector>
```

Classes

- class [Tracking_Dump_Reader](#)

11.325.1 Detailed Description

Helper file for unit testing.

Author

Javier Arribas, 2017. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.326 tracking_FLL_PLL_filter.h File Reference

Interface of a hybrid FLL and PLL filter for tracking carrier loop.

Classes

- class [Tracking_FLL_PLL_filter](#)

This class implements a hybrid FLL and PLL filter for tracking carrier loop.

11.326.1 Detailed Description

Interface of a hybrid FLL and PLL filter for tracking carrier loop.

Author

Javier Arribas, 2011. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.327 tracking_interface.h File Reference

This class represents an interface to a tracking block.

```
#include "gnss_block_interface.h"
#include "gnss_synchro.h"
```

Classes

- class [Concurrent_Queue< Data >](#)
This class implements a thread-safe std::queue.
- class [TrackingInterface](#)
This abstract class represents an interface to a tracking block.

11.327.1 Detailed Description

This class represents an interface to a tracking block.

Author

Carlos Aviles, 2010. carlos.avilesr(at)gmail.com

Abstract class for tracking interfaces. Since all its methods are virtual, this class cannot be instantiated directly, and a subclass can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.328 tracking_loop_filter.h File Reference

Generic 1st to 3rd order loop filter implementation.

```
#include <vector>
```

Classes

- class [Tracking_loop_filter](#)
This class implements a generic 1st, 2nd or 3rd order loop filter.

11.328.1 Detailed Description

Generic 1st to 3rd order loop filter implementation.

Author

Cillian O'Driscoll, 2015. cillian.odriscoll(at)gmail.com

Class implementing a generic 1st, 2nd or 3rd order loop filter. Based on the bilinear transform of the standard Wiener filter.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.329 tracking_tests_flags.h File Reference

Helper file for unit testing.

```
#include <gflags/gflags.h>
#include <limits>
#include <string>
```

Functions

- **DEFINE_string** (trk_test_implementation, std::string("GPS_L1_CA_DLL_PLL_Tracking"), "Tracking block implementation under test, defaults to GPS_L1_CA_DLL_PLL_Tracking")
- **DEFINE_bool** (enable_external_signal_file, false, "Use an external signal file capture instead of the software-defined signal generator")
- **DEFINE_double** (external_signal_acquisition_threshold, 2.5, "Threshold for satellite acquisition when external file is used")
- **DEFINE_int32** (external_signal_acquisition_dwells, 5, "Maximum dwells count for satellite acquisition when external file is used")
- **DEFINE_double** (external_signal_acquisition_doppler_max_hz, 5000.0, "Doppler max for satellite acquisition when external file is used")
- **DEFINE_double** (external_signal_acquisition_doppler_step_hz, 125.0, "Doppler step for satellite acquisition when external file is used")
- **DEFINE_bool** (use_acquisition_resampler, false, "Reduce the sampling rate of the input signal for the acquisition in order to optimize the SNR and decrease the processor load")
- **DEFINE_string** (signal_file, std::string("signal_out.bin"), "Path of the external signal capture file")
- **DEFINE_double** (CN0_dBHz_start, std::numeric_limits< double >::infinity(), "Enable noise generator and set the CN0 start sweep value [dB-Hz]")
- **DEFINE_double** (CN0_dBHz_stop, std::numeric_limits< double >::infinity(), "Enable noise generator and set the CN0 stop sweep value [dB-Hz]")
- **DEFINE_double** (CN0_dB_step, 3.0, "Noise generator CN0 sweep step value [dB]")
- **DEFINE_double** (PLL_bw_hz_start, 20.0, "PLL Wide configuration start sweep value [Hz]")
- **DEFINE_double** (PLL_bw_hz_stop, 20.0, "PLL Wide configuration stop sweep value [Hz]")
- **DEFINE_double** (PLL_bw_hz_step, 5.0, "PLL Wide configuration sweep step value [Hz]")
- **DEFINE_double** (DLL_bw_hz_start, 1.0, "DLL Wide configuration start sweep value [Hz]")
- **DEFINE_double** (DLL_bw_hz_stop, 1.0, "DLL Wide configuration stop sweep value [Hz]")
- **DEFINE_double** (DLL_bw_hz_step, 0.25, "DLL Wide configuration sweep step value [Hz]")
- **DEFINE_double** (fll_bw_hz, 4.0, "FLL filter bandwidth [Hz]")
- **DEFINE_bool** (enable_fll_pull_in, false, "Enable FLL in pull-in phase")
- **DEFINE_bool** (enable_fll_steady_state, false, "Enable FLL in steady-state phase")
- **DEFINE_double** (PLL_narrow_bw_hz, 5.0, "PLL Narrow configuration value [Hz]")
- **DEFINE_double** (DLL_narrow_bw_hz, 0.75, "DLL Narrow configuration value [Hz]")
- **DEFINE_double** (acq_Doppler_error_hz_start, 1000.0, "Acquisition Doppler error start sweep value [Hz]")
- **DEFINE_double** (acq_Doppler_error_hz_stop, -1000.0, "Acquisition Doppler error stop sweep value [Hz]")
- **DEFINE_double** (acq_Doppler_error_hz_step, -50.0, "Acquisition Doppler error sweep step value [Hz]")
- **DEFINE_double** (acq_Delay_error_chips_start, 2.0, "Acquisition Code Delay error start sweep value [Chips]")
- **DEFINE_double** (acq_Delay_error_chips_stop, -2.0, "Acquisition Code Delay error stop sweep value [Chips]")
- **DEFINE_double** (acq_Delay_error_chips_step, -0.1, "Acquisition Code Delay error sweep step value [Chips]")
- **DEFINE_double** (acq_to_trk_delay_s, 0.0, "Acquisition to Tracking delay value [s]")
- **DEFINE_int64** (skip_samples, 0, "Skip an initial transitory in the processed signal file capture [samples]")

- **DEFINE_int32** (plot_detail_level, 0, "Specify the desired plot detail (0,1,2): 0 - Minimum plots (default) 2 - Plot all tracking parameters")
- **DEFINE_double** (skip_trk_transitory_s, 1.0, "Skip the initial tracking output signal to avoid transitory results [s]")
- **DEFINE_int32** (extend_correlation_symbols, 1, "Set the tracking coherent correlation to N symbols (up to 20 for GPS L1 C/A)")
- **DEFINE_int32** (smoother_length, 10, "Set the moving average size for the carrier phase and code phase in case of high dynamics")
- **DEFINE_bool** (high_dyn, false, "Activates the code [resampler](#) and NCO generator for high dynamics")
- **DEFINE_bool** (plot_gps_l1_tracking_test, false, "Plots results of GpsL1CADIIPIITrackingTest with gnuplot")

11.329.1 Detailed Description

Helper file for unit testing.

Author

Javier Arribas, 2018. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.330 tracking_true_obs_reader.h File Reference

Helper file for unit testing.

```
#include <cstdint>
#include <fstream>
#include <string>
#include <vector>
```

Classes

- class [Tracking_True_Obs_Reader](#)

11.330.1 Detailed Description

Helper file for unit testing.

Author

Javier Arribas, 2017. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.331 true_observables_reader.h File Reference

Helper file for unit testing.

```
#include <stdint>
#include <fstream>
#include <string>
#include <vector>
```

Classes

- class [True_Observables_Reader](#)

11.331.1 Detailed Description

Helper file for unit testing.

Author

Javier Arribas, 2017. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.332 two_bit_cpx_file_signal_source.h File Reference

Interface of a class that reads signals samples from a 2 bit complex sampler front-end file and adapts it to a [SignalSourceInterface](#).

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include "unpack_byte_2bit_cpx_samples.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/file_source.h>
#include <gnuradio/blocks/interleaved_short_to_complex.h>
#include <gnuradio/blocks/throttle.h>
#include <gnuradio/hier_block2.h>
#include <pmt/pmt.h>
#include <stdint>
#include <memory>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [TwoBitCpxFileSignalSource](#)

Class that reads signals samples from a file and adapts it to a `SignalSourceInterface`.

11.332.1 Detailed Description

Interface of a class that reads signals samples from a 2 bit complex sampler front-end file and adapts it to a `SignalSourceInterface`.

Author

Javier Arribas, 2015 jarribas(at)cttc.es

This class represents a file signal source.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.333 `two_bit_packed_file_signal_source.h` File Reference

Interface of a class that reads signals samples from a file. Each sample is two bits, which are packed into bytes or shorts.

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include "unpack_2bit_samples.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/blocks/file_source.h>
#include <gnuradio/blocks/interleaved_char_to_complex.h>
#include <gnuradio/blocks/throttle.h>
#include <gnuradio/hier_block2.h>
#include <pmt/pmt.h>
#include <cstdint>
#include <memory>
#include <string>
#include <boost/shared_ptr.hpp>
```

Classes

- class [TwoBitPackedFileSignalSource](#)

Class that reads signals samples from a file and adapts it to a `SignalSourceInterface`.

11.333.1 Detailed Description

Interface of a class that reads signals samples from a file. Each sample is two bits, which are packed into bytes or shorts.

Author

Cillian O'Driscoll, 2015 cillian.odriscoll (at) gmail.com

This class represents a file signal source.

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.334 uhd_signal_source.h File Reference

Interface for the Universal Hardware Driver signal source.

```
#include "concurrent_queue.h"
#include "gnss_block_interface.h"
#include <gnuradio/blocks/file_sink.h>
#include <gnuradio/hier_block2.h>
#include <gnuradio/uhd/usrp_source.h>
#include <pmt/pmt.h>
#include <cstdint>
#include <memory>
#include <string>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [UhdSignalSource](#)

This class reads samples from a UHD device (see <http://code.ettus.com/redmine/ettus/projects/uhd/wiki>)

11.334.1 Detailed Description

Interface for the Universal Hardware Driver signal source.

Author

Javier Arribas, 2012. jarribas(at)cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.335 unpack_2bit_samples.h File Reference

Unpacks 2 bit samples samples may be packed in any of the following ways: 1) Into bytes [item == byte] 1a) Big endian ordering within the byte 1b) Little endian ordering within the byte 2) Into shorts [item == short] 2a) Big endian ordering of bytes, big endian within the byte 2b) Big endian ordering of bytes, little endian within the byte 2c) Little endian ordering of bytes, big endian within the byte 2d) Little endian ordering of bytes, little endian within the byte.

```
#include <gnuradio/sync_interpolator.h>
#include <stdint>
#include <vector>
#include <boost/shared_ptr.hpp>
```

Classes

- class [unpack_2bit_samples](#)

This class takes 2 bit samples that have been packed into bytes or shorts as input and generates a byte for each sample. It generates eight times as much data as is input (every two bits become 16 bits)

Typedefs

- using **unpack_2bit_samples_sptr** = boost::shared_ptr< [unpack_2bit_samples](#) >

Functions

- **unpack_2bit_samples_sptr make_unpack_2bit_samples** (bool big_endian_bytes, size_t item_size, bool big_endian_items, bool reverse_interleaving=false)

11.335.1 Detailed Description

Unpacks 2 bit samples samples may be packed in any of the following ways: 1) Into bytes [item == byte] 1a) Big endian ordering within the byte 1b) Little endian ordering within the byte 2) Into shorts [item == short] 2a) Big endian ordering of bytes, big endian within the byte 2b) Big endian ordering of bytes, little endian within the byte 2c) Little endian ordering of bytes, big endian within the byte 2d) Little endian ordering of bytes, little endian within the byte.

Within a byte the two possibilities look like this: 7 6 5 4 3 2 1 0 : Bit number $x_{n,1}$ $x_{n,0}$ $x_{n+1,1}$ $x_{n+1,0}$ $x_{n+2,1}$ $x_{n+2,0}$ $x_{n+3,1}$ $x_{n+3,0}$: Little endian $x_{n+3,1}$ $x_{n+3,0}$ $x_{n+2,1}$ $x_{n+2,0}$ $x_{n+1,1}$ $x_{n+1,0}$ $x_{n,1}$ $x_{n,0}$: Big Endian

For a short (uint16_t) the bytes are either transmitted as follows:

1 0 : Byte number Byte_n Byte_n+1 : Little endian Byte_n+1 Byte_n : Bit endian

The two bit values are assumed to have the following mapping:

x_1 x_0 Value 0 0 +1 0 1 +3 1 0 -3 1 1 -1

Letting x denote the two's complement interpretation of x_1 x_0 , then:

Value = $2*x + 1$

We want to output the data in the order:

Value_0, Value_1, Value_2, ..., Value_n, Value_n+1, Value_n+2, ...

Cillian O'Driscoll cillian.odriscoll (at) gmail . com

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.336 unpack_byte_2bit_cpx_samples.h File Reference

Unpacks byte samples to 2 bits complex samples. Packing Order Most Significant Nibble - Sample n Least Significant Nibble - Sample n+1 Packing order in Nibble Q1 Q0 I1 I0.

```
#include <gnuradio/sync_interpolator.h>
#include <boost/shared_ptr.hpp>
```

Classes

- class [unpack_byte_2bit_cpx_samples](#)

This class implements conversion between byte packet samples to 2bit_cpx samples 1 byte = 2 x complex 2bit I, + 2bit Q samples.

Typedefs

- using **unpack_byte_2bit_cpx_samples_sptr** = boost::shared_ptr< [unpack_byte_2bit_cpx_samples](#) >

Functions

- **unpack_byte_2bit_cpx_samples_sptr make_unpack_byte_2bit_cpx_samples ()**

11.336.1 Detailed Description

Unpacks byte samples to 2 bits complex samples. Packing Order Most Significant Nibble - Sample n Least Significant Nibble - Sample n+1 Packing order in Nibble Q1 Q0 I1 I0.

Javier Arribas jarribas (at) cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.337 unpack_byte_2bit_samples.h File Reference

Unpacks byte samples to NSR 2 bits samples.

```
#include <gnuradio/sync_interpolator.h>
#include <boost/shared_ptr.hpp>
```

Classes

- class [unpack_byte_2bit_samples](#)

This class implements conversion between byte packet samples to 2bit samples 1 byte = 4 2bit samples.

Typedefs

- using **unpack_byte_2bit_samples_sptr** = boost::shared_ptr< [unpack_byte_2bit_samples](#) >

Functions

- **unpack_byte_2bit_samples_sptr make_unpack_byte_2bit_samples** ()

11.337.1 Detailed Description

Unpacks byte samples to NSR 2 bits samples.

Javier Arribas jarribas (at) cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.338 unpack_byte_4bit_samples.h File Reference

Unpacks byte samples to 4 bits samples. Packing Order Packing order in Nibble I0 I1 I2 I3 I0 I1 I2 I3.

```
#include <gnuradio/sync_interpolator.h>
```

Classes

- class [unpack_byte_4bit_samples](#)

This class implements conversion between byte packet samples to 4bit_cpx samples 1 byte = 1 x complex 4bit I, + 4bit Q samples.

Typedefs

- using **unpack_byte_4bit_samples_sptr** = std::shared_ptr< [unpack_byte_4bit_samples](#) >

Functions

- unpack_byte_4bit_samples_sptr **make_unpack_byte_4bit_samples** ()

11.338.1 Detailed Description

Unpacks byte samples to 4 bits samples. Packing Order Packing order in Nibble I0 I1 I2 I3 I0 I1 I2 I3.

Javier Arribas jarribas (at) cttc.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.339 unpack_intspir_1bit_samples.h File Reference

Unpacks SPIR int samples to NSR 1 bit samples.

```
#include <gnuradio/sync_interpolator.h>
#include <boost/shared_ptr.hpp>
```

Classes

- class [unpack_intspir_1bit_samples](#)
This class implements conversion between byte packet samples to 2bit samples 1 byte = 4 2bit samples.

Typedefs

- using **unpack_intspir_1bit_samples_sptr** = boost::shared_ptr< [unpack_intspir_1bit_samples](#) >

Functions

- unpack_intspir_1bit_samples_sptr **make_unpack_intspir_1bit_samples** ()

11.339.1 Detailed Description

Unpacks SPIR int samples to NSR 1 bit samples.

Fran Fabra fabra (at) ice.csic.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is not part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.340 unpack_spir_gss6450_samples.h File Reference

Unpacks SPIR int samples.

```
#include <gnuradio/sync_interpolator.h>
#include <boost/shared_ptr.hpp>
```

Classes

- class [unpack_spir_gss6450_samples](#)

Typedefs

- using **unpack_spir_gss6450_samples_sptr** = boost::shared_ptr< [unpack_spir_gss6450_samples](#) >

Functions

- **unpack_spir_gss6450_samples_sptr make_unpack_spir_gss6450_samples** (int adc_nbit_)

11.340.1 Detailed Description

Unpacks SPIR int samples.

Author

Antonio Ramos, antonio.ramos(at)cttc.es

Javier Arribas jarribas (at) ctte.es

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is not part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

11.341 viterbi_decoder.h File Reference

Interface of a Viterbi decoder class based on the Iterative Solutions Coded Modulation Library by Matthew C. Valenti.

```
#include <cstdint>
#include <deque>
#include <vector>
```

Classes

- class [Viterbi_Decoder](#)
Class that implements a Viterbi decoder.

11.341.1 Detailed Description

Interface of a Viterbi decoder class based on the Iterative Solutions Coded Modulation Library by Matthew C. Valenti.

Author

Daniel Fehr 2013. daniel.co(at)bluewin.ch

Copyright (C) 2010-2020 (see AUTHORS file for a list of contributors)

GNSS-SDR is a software defined Global Navigation Satellite Systems receiver

This file is part of GNSS-SDR.

SPDX-License-Identifier: GPL-3.0-or-later

Index

- ~Acquisition_msg_rx
 - Acquisition_msg_rx, [74](#)
- ~ArraySignalConditioner
 - ArraySignalConditioner, [81](#)
- ~Channel
 - Channel, [135](#)
- ~ControlThread
 - ControlThread, [157](#)
- ~Exponential_Smoothing
 - Exponential_Smoothing, [174](#)
- ~FIRFilter
 - FIRFilter, [180](#)
- ~Fpga_Acquisition
 - Fpga_Acquisition, [184](#)
- ~Fpga_Multicorrelator_8sc
 - Fpga_Multicorrelator_8sc, [190](#)
- ~Fpga_Switch
 - Fpga_Switch, [194](#)
- ~Fpga_dynamic_bit_selection
 - Fpga_dynamic_bit_selection, [188](#)
- ~GNSSFlowgraph
 - GNSSFlowgraph, [410](#)
- ~GalileoE1DIIPIIVemlTrackingFpga
 - GalileoE1DIIPIIVemlTrackingFpga, [242](#)
- ~GalileoE1PcpsAmbiguousAcquisitionFpga
 - GalileoE1PcpsAmbiguousAcquisitionFpga, [255](#)
- ~GalileoE5aDIIPIITrackingFpga
 - GalileoE5aDIIPIITrackingFpga, [278](#)
- ~GalileoE5aPcpsAcquisitionFpga
 - GalileoE5aPcpsAcquisitionFpga, [291](#)
- ~GalileoE5bPcpsAcquisition
 - GalileoE5bPcpsAcquisition, [302](#)
- ~GalileoE5bPcpsAcquisitionFpga
 - GalileoE5bPcpsAcquisitionFpga, [308](#)
- ~GenSignalSource
 - GenSignalSource, [316](#)
- ~Gnss_Satellite
 - Gnss_Satellite, [380](#)
- ~Gnss_Synchro
 - Gnss_Synchro, [397](#)
- ~GpsL1CaDIIPIITrackingFpga
 - GpsL1CaDIIPIITrackingFpga, [482](#)
- ~GpsL1CaPcpsAcquisitionFpga
 - GpsL1CaPcpsAcquisitionFpga, [499](#)
- ~GpsL5DIIPIITrackingFpga
 - GpsL5DIIPIITrackingFpga, [540](#)
- ~GpsL5iPcpsAcquisitionFpga
 - GpsL5iPcpsAcquisitionFpga, [549](#)
- ~Nmea_Printer
 - Nmea_Printer, [580](#)
- ~Rinex_Printer
 - Rinex_Printer, [652](#)
- ~Rtcm_Printer
 - Rtcm_Printer, [685](#)
- ~SignalConditioner
 - SignalConditioner, [720](#)
- ~beidou_b1i_telemetry_decoder_gs
 - beidou_b1i_telemetry_decoder_gs, [85](#)
- ~beidou_b3i_telemetry_decoder_gs
 - beidou_b3i_telemetry_decoder_gs, [87](#)
- ~channel_msg_receiver_cc
 - channel_msg_receiver_cc, [138](#)
- ~channel_status_msg_receiver
 - channel_status_msg_receiver, [139](#)
- ~dll_pll_veml_tracking_fpga
 - dll_pll_veml_tracking_fpga, [170](#)
- ~galileo_e5a_noncoherentIQ_acquisition_caf_cc
 - galileo_e5a_noncoherentIQ_acquisition_caf_cc, [205](#)
- ~galileo_pcps_8ms_acquisition_cc
 - galileo_pcps_8ms_acquisition_cc, [229](#)
- ~glonass_l1_ca_telemetry_decoder_gs
 - glonass_l1_ca_telemetry_decoder_gs, [350](#)
- ~glonass_l2_ca_telemetry_decoder_gs
 - glonass_l2_ca_telemetry_decoder_gs, [354](#)
- ~gnss_synchro_monitor
 - gnss_synchro_monitor, [406](#)
- ~pcps_acquisition_fine_doppler_cc
 - pcps_acquisition_fine_doppler_cc, [596](#)
- ~pcps_acquisition_fpga
 - pcps_acquisition_fpga, [602](#)
- ~pcps_assisted_acquisition_cc
 - pcps_assisted_acquisition_cc, [608](#)
- ~pcps_cccwsr_acquisition_cc
 - pcps_cccwsr_acquisition_cc, [613](#)
- ~pcps_openc1_acquisition_cc
 - pcps_openc1_acquisition_cc, [617](#)
- ~pcps_quicksync_acquisition_cc
 - pcps_quicksync_acquisition_cc, [623](#)
- ~pcps_tong_acquisition_cc
 - pcps_tong_acquisition_cc, [628](#)
- ~rtklib_pvt_gs
 - rtklib_pvt_gs, [693](#)
- A_1
 - Galileo_Ephemeris, [213](#)
- ALPHA_0
 - GPS_L1_CA.h, [968](#)
- ARMODE_CONT

- rtklib.h, 1084
- ARMODE_FIXHOLD
 - rtklib.h, 1084
- ARMODE_INST
 - rtklib.h, 1084
- ARMODE_OFF
 - rtklib.h, 1084
- ARMODE_PPPAR_ILS
 - rtklib.h, 1085
- ARMODE_PPPAR
 - rtklib.h, 1085
- AS2R
 - MATH_CONSTANTS.h, 1028
- Acq_Conf, 71
- acq_conf.h, 757
- Acq_delay_samples
 - Gnss_Synchro, 399
- Acq_doppler_hz
 - Gnss_Synchro, 399
- Acq_doppler_step
 - Gnss_Synchro, 399
- Acq_samplestamp_samples
 - Gnss_Synchro, 399
- Acquisition_Dump_Reader, 72
 - Acquisition_Dump_Reader, 72, 73
 - operator=, 73
- acquisition_dump_reader.h, 757
- acquisition_interface.h, 758
- acquisition_manager
 - GNSSFlowgraph, 410
- Acquisition_msg_rx, 73
 - ~Acquisition_msg_rx, 74
- acquisition_msg_rx.h, 759
- AcquisitionInterface, 75
- ad9361_fpga_signal_source.h, 759
- ad9361_manager.h, 760
- Ad9361FpgaSignalSource, 76
 - implementation, 76
- af0_4
 - Galileo_Ephemeris, 213
- af1_4
 - Galileo_Ephemeris, 213
- af2_4
 - Galileo_Ephemeris, 213
- Agnss_Ref_Location, 77
 - Agnss_Ref_Location, 77
 - serialize, 78
- Agnss_Ref_Time, 78
 - Agnss_Ref_Time, 79
 - serialize, 79
- agnss_ref_location.h, 762
- agnss_ref_time.h, 762
- ai0_5
 - Galileo_Iono, 225
- ai1_5
 - Galileo_Iono, 226
- ai2_5
 - Galileo_Iono, 226
- alert
 - cnav_msg_t, 145
- alm_t, 79
- ambc_t, 80
- apply_action
 - GNSSFlowgraph, 410
- array_signal_conditioner.h, 763
- ArraySignalConditioner, 80
 - ~ArraySignalConditioner, 81
 - ArraySignalConditioner, 81
 - implementation, 82
- at
 - Gnss_circular_deque, 376
- AU
 - MATH_CONSTANTS.h, 1028
- Azimuth
 - Gps_Acq_Assist, 417
- b_L2_P_data_flag
 - Gps_Ephemeris, 452
- b_alert_flag
 - Beidou_Dnav_Ephemeris, 96
 - Gps_CNAV_Ephemeris, 428
 - Gps_Ephemeris, 452
- b_antispoofing_flag
 - Beidou_Dnav_Ephemeris, 96
 - Gps_CNAV_Ephemeris, 428
 - Gps_Ephemeris, 452
- b_fit_interval_flag
 - Beidou_Dnav_Ephemeris, 97
 - Gps_Ephemeris, 452
- b_integrity_status_flag
 - Beidou_Dnav_Ephemeris, 97
 - Gps_CNAV_Ephemeris, 429
 - Gps_Ephemeris, 452
- b_sv_do_not_use
 - Sbas_Ephemeris, 703
- BEIDOU_B1I_CODE_LENGTH_CHIPS
 - Beidou_B1I.h, 767
- BEIDOU_B1I_CODE_PERIOD_MS
 - Beidou_B1I.h, 767
- BEIDOU_B1I_CODE_PERIOD_S
 - Beidou_B1I.h, 767
- BEIDOU_B1I_CODE_RATE_CPS
 - Beidou_B1I.h, 767
- BEIDOU_B1I_FREQ_HZ
 - Beidou_B1I.h, 767
- BEIDOU_B3I_CODE_LENGTH_CHIPS
 - Beidou_B3I.h, 774
- BEIDOU_B3I_CODE_PERIOD_MS
 - Beidou_B3I.h, 774
- BEIDOU_B3I_CODE_PERIOD_S
 - Beidou_B3I.h, 774
- BEIDOU_B3I_CODE_RATE_CPS
 - Beidou_B3I.h, 774
- BEIDOU_B3I_FREQ_HZ
 - Beidou_B3I.h, 775
- BEIDOU_B3I_TELEMETRY_RATE_BITS_SECOND
 - Beidou_B3I.h, 775

- BEIDOU_DNAV_SUBFRAME_DATA_BITS
 - Beidou_DNAV.h, [785](#)
- BEIDOU_GM
 - MATH_CONSTANTS.h, [1028](#)
- BEIDOU_OMEGA_EARTH_DOT
 - MATH_CONSTANTS.h, [1028](#)
- BEIDOU_F
 - MATH_CONSTANTS.h, [1028](#)
- BGD_E1E5a_5
 - Galileo_Ephemeris, [213](#)
- BGD_E1E5b_5
 - Galileo_Ephemeris, [214](#)
- back
 - Gnss_circular_deque, [376](#)
- bayesian_estimation.h, [763](#)
- Bayesian_estimator, [82](#)
- beamformer, [83](#)
- beamformer.h, [764](#)
- beamformer_filter.h, [765](#)
- BeamformerFilter, [83](#)
 - implementation, [84](#)
- Beidou_B1I.h, [766](#)
 - BEIDOU_B1I_CODE_LENGTH_CHIPS, [767](#)
 - BEIDOU_B1I_CODE_PERIOD_MS, [767](#)
 - BEIDOU_B1I_CODE_PERIOD_S, [767](#)
 - BEIDOU_B1I_CODE_RATE_CPS, [767](#)
 - BEIDOU_B1I_FREQ_HZ, [767](#)
- Beidou_B3I.h, [773](#)
 - BEIDOU_B3I_CODE_LENGTH_CHIPS, [774](#)
 - BEIDOU_B3I_CODE_PERIOD_MS, [774](#)
 - BEIDOU_B3I_CODE_PERIOD_S, [774](#)
 - BEIDOU_B3I_CODE_RATE_CPS, [774](#)
 - BEIDOU_B3I_FREQ_HZ, [775](#)
 - BEIDOU_B3I_TELEMETRY_RATE_BITS_SEC↔
OND, [775](#)
- Beidou_DNAV.h, [781](#)
 - BEIDOU_DNAV_SUBFRAME_DATA_BITS, [785](#)
- Beidou_Dnav_Almanac, [88](#)
 - Beidou_Dnav_Almanac, [89](#)
 - d_A_f0, [89](#)
 - d_A_f1, [89](#)
 - d_M_0, [89](#)
 - d_OMEGA0, [90](#)
 - d_OMEGA_DOT, [90](#)
 - d_OMEGA, [90](#)
 - d_Toa, [90](#)
 - d_e_eccentricity, [89](#)
 - d_sqrt_A, [90](#)
 - i_SV_health, [91](#)
 - i_satellite_PRN, [91](#)
- Beidou_Dnav_Ephemeris, [91](#)
 - b_alert_flag, [96](#)
 - b_antispoofing_flag, [96](#)
 - b_fit_interval_flag, [97](#)
 - b_integrity_status_flag, [97](#)
 - Beidou_Dnav_Ephemeris, [94](#)
 - d_A_f0, [97](#)
 - d_A_f1, [97](#)
 - d_A_f2, [98](#)
 - d_AODC, [98](#)
 - d_AODE, [98](#)
 - d_Cic, [98](#)
 - d_Cis, [99](#)
 - d_Crc, [99](#)
 - d_Crs, [99](#)
 - d_Cuc, [99](#)
 - d_Cus, [100](#)
 - d_Delta_n, [100](#)
 - d_IDOT, [101](#)
 - d_M_0, [101](#)
 - d_OMEGA0, [102](#)
 - d_OMEGA_DOT, [102](#)
 - d_OMEGA, [101](#)
 - d_TGD1, [104](#)
 - d_TGD2, [104](#)
 - d_TOW, [105](#)
 - d_Toc, [104](#)
 - d_Toe, [104](#)
 - d_dtr, [100](#)
 - d_eccentricity, [100](#)
 - d_i_0, [101](#)
 - d_satClkDrift, [102](#)
 - d_satpos_X, [102](#)
 - d_satpos_Y, [103](#)
 - d_satpos_Z, [103](#)
 - d_satvel_X, [103](#)
 - d_satvel_Y, [103](#)
 - d_satvel_Z, [103](#)
 - d_sqrt_A, [104](#)
 - i_AODO, [105](#)
 - i_BEIDOU_week, [105](#)
 - i_SV_accuracy, [106](#)
 - i_nav_type, [105](#)
 - i_satellite_PRN, [106](#)
 - i_sig_type, [106](#)
 - satelliteBlock, [106](#)
 - satellitePosition, [94](#)
 - serialize, [94](#)
 - sv_clock_drift, [96](#)
 - sv_clock_relativistic_term, [96](#)
- Beidou_Dnav_Iono, [107](#)
 - Beidou_Dnav_Iono, [107](#)
 - d_alpha0, [108](#)
 - d_alpha1, [108](#)
 - d_alpha2, [108](#)
 - d_alpha3, [109](#)
 - d_beta0, [109](#)
 - d_beta1, [109](#)
 - d_beta2, [109](#)
 - d_beta3, [110](#)
 - serialize, [108](#)
 - valid, [110](#)
- Beidou_Dnav_Navigation_Message, [110](#)
 - Beidou_Dnav_Navigation_Message, [111](#)
 - d1_subframe_decoder, [112](#)
 - d2_subframe_decoder, [112](#)

- get_ephemeris, 112
- get_iono, 112
- get_utc_model, 112
- have_new_almanac, 112
- have_new_ephemeris, 113
- have_new_iono, 113
- have_new_utc_model, 113
- satellitePosition, 113
- set_satellite_PRN, 113
- sv_clock_correction, 113
- utc_time, 114
- Beidou_Dnav_Utc_Model, 114
 - d_A0_GAL, 115
 - d_A0_GLO, 115
 - d_A0_GPS, 115
 - d_A0_UTC, 115
 - d_A1_GAL, 115
 - d_A1_GLO, 116
 - d_A1_GPS, 116
 - d_A1_UTC, 116
 - d_DeltaT_LSF, 116
 - i_DeltaT_LS, 116
 - i_DN, 117
 - i_WN_LSF, 117
- beidou_b1i_code_gen_complex
 - beidou_b1i_signal_processing.h, 770
- beidou_b1i_code_gen_complex_sampled
 - beidou_b1i_signal_processing.h, 770
- beidou_b1i_code_gen_float
 - beidou_b1i_signal_processing.h, 771
- beidou_b1i_code_gen_int
 - beidou_b1i_signal_processing.h, 771
- beidou_b1i_dll_pll_tracking.h, 768
- beidou_b1i_pcps_acquisition.h, 768
- beidou_b1i_signal_processing.h, 769
 - beidou_b1i_code_gen_complex, 770
 - beidou_b1i_code_gen_complex_sampled, 770
 - beidou_b1i_code_gen_float, 771
 - beidou_b1i_code_gen_int, 771
- beidou_b1i_telemetry_decoder.h, 771
- beidou_b1i_telemetry_decoder_gs, 84
 - ~beidou_b1i_telemetry_decoder_gs, 85
 - general_work, 85
 - set_channel, 86
 - set_satellite, 86
- beidou_b1i_telemetry_decoder_gs.h, 772
- beidou_b3i_code_gen_complex
 - beidou_b3i_signal_processing.h, 777
- beidou_b3i_code_gen_complex_sampled
 - beidou_b3i_signal_processing.h, 778
- beidou_b3i_code_gen_float
 - beidou_b3i_signal_processing.h, 778
- beidou_b3i_code_gen_int
 - beidou_b3i_signal_processing.h, 778
- beidou_b3i_dll_pll_tracking.h, 775
- beidou_b3i_pcps_acquisition.h, 776
- beidou_b3i_signal_processing.h, 777
 - beidou_b3i_code_gen_complex, 777
 - beidou_b3i_code_gen_complex_sampled, 778
 - beidou_b3i_code_gen_float, 778
 - beidou_b3i_code_gen_int, 778
 - general_work, 87
 - set_channel, 87
 - set_satellite, 88
- beidou_b3i_telemetry_decoder.h, 779
- beidou_b3i_telemetry_decoder_gs, 86
 - ~beidou_b3i_telemetry_decoder_gs, 87
- beidou_b3i_telemetry_decoder_gs.h, 780
- beidou_dnav_almanac.h, 785
- beidou_dnav_ephemeris.h, 786
- beidou_dnav_ephemeris_map
 - Rtklib_Solver, 696
- beidou_dnav_iono.h, 787
- beidou_dnav_navigation_message.h, 787
- beidou_dnav_utc_model.h, 788
- BeidouB1iDIIPIITracking, 117
 - set_channel, 118
 - set_gnss_synchro, 118
 - stop_tracking, 118
- BeidouB1iPcpsAcquisition, 119
 - implementation, 120
 - init, 120
 - mag, 120
 - reset, 121
 - set_channel, 121
 - set_channel_fsm, 121
 - set_doppler_max, 121
 - set_doppler_step, 121
 - set_gnss_synchro, 122
 - set_local_code, 122
 - set_resampler_latency, 122
 - set_state, 122
 - set_threshold, 122
 - stop_acquisition, 123
- BeidouB1iTelemetryDecoder, 123
 - implementation, 124
- BeidouB3iDIIPIITracking, 124
 - set_channel, 125
 - set_gnss_synchro, 125
 - stop_tracking, 125
- BeidouB3iPcpsAcquisition, 126
 - implementation, 127
 - init, 127
 - mag, 127
 - reset, 128
 - set_channel, 128
 - set_channel_fsm, 128
 - set_doppler_max, 128
 - set_doppler_step, 129
 - set_gnss_synchro, 129
 - set_local_code, 129
 - set_resampler_latency, 129
 - set_state, 129
 - set_threshold, 130
 - stop_acquisition, 130
- BeidouB3iTelemetryDecoder, 130

- implementation, [131](#)
- bin_to_binary_data
 - Rtcm, [671](#)
- bin_to_double
 - Rtcm, [671](#)
- bin_to_hex
 - Rtcm, [672](#)
- bin_to_uint
 - Rtcm, [672](#)
- binary_data_to_bin
 - Rtcm, [672](#)
- bit_selection
 - Fpga_dynamic_bit_selection, [188](#)
- bits.h, [789](#)
- byte_to_short.h, [789](#)
- byte_x2_to_complex_byte, [131](#)
- byte_x2_to_complex_byte.h, [790](#)
- ByteToShort, [132](#)
 - implementation, [133](#)
- C_ic_4
 - Galileo_Ephemeris, [214](#)
- C_is_4
 - Galileo_Ephemeris, [214](#)
- C_rc_3
 - Galileo_Ephemeris, [214](#)
- C_rs_3
 - Galileo_Ephemeris, [215](#)
- C_uc_3
 - Galileo_Ephemeris, [215](#)
- C_us_3
 - Galileo_Ephemeris, [215](#)
- CHISQR
 - rtklib.h, [1085](#)
- CN0_dB_hz
 - Gnss_Synchro, [400](#)
- CODE_L1A
 - gnss_obs_codes.h, [928](#)
- CODE_L1B
 - gnss_obs_codes.h, [928](#)
- CODE_L1C
 - gnss_obs_codes.h, [929](#)
- CODE_L1E
 - gnss_obs_codes.h, [929](#)
- CODE_L1I
 - gnss_obs_codes.h, [929](#)
- CODE_L1L
 - gnss_obs_codes.h, [929](#)
- CODE_L1M
 - gnss_obs_codes.h, [929](#)
- CODE_L1N
 - gnss_obs_codes.h, [930](#)
- CODE_L1P
 - gnss_obs_codes.h, [930](#)
- CODE_L1Q
 - gnss_obs_codes.h, [930](#)
- CODE_L1S
 - gnss_obs_codes.h, [930](#)
- CODE_L1W
 - gnss_obs_codes.h, [930](#)
- CODE_L1X
 - gnss_obs_codes.h, [931](#)
- CODE_L1Y
 - gnss_obs_codes.h, [931](#)
- CODE_L1Z
 - gnss_obs_codes.h, [931](#)
- CODE_L2C
 - gnss_obs_codes.h, [931](#)
- CODE_L2D
 - gnss_obs_codes.h, [931](#)
- CODE_L2I
 - gnss_obs_codes.h, [932](#)
- CODE_L2L
 - gnss_obs_codes.h, [932](#)
- CODE_L2M
 - gnss_obs_codes.h, [932](#)
- CODE_L2N
 - gnss_obs_codes.h, [932](#)
- CODE_L2P
 - gnss_obs_codes.h, [932](#)
- CODE_L2Q
 - gnss_obs_codes.h, [933](#)
- CODE_L2S
 - gnss_obs_codes.h, [933](#)
- CODE_L2W
 - gnss_obs_codes.h, [933](#)
- CODE_L2X
 - gnss_obs_codes.h, [933](#)
- CODE_L2Y
 - gnss_obs_codes.h, [933](#)
- CODE_L3I
 - gnss_obs_codes.h, [934](#)
- CODE_L3Q
 - gnss_obs_codes.h, [934](#)
- CODE_L3X
 - gnss_obs_codes.h, [934](#)
- CODE_L5A
 - gnss_obs_codes.h, [934](#)
- CODE_L5B
 - gnss_obs_codes.h, [934](#)
- CODE_L5C
 - gnss_obs_codes.h, [935](#)
- CODE_L5I
 - gnss_obs_codes.h, [935](#)
- CODE_L5Q
 - gnss_obs_codes.h, [935](#)
- CODE_L5X
 - gnss_obs_codes.h, [935](#)
- CODE_L6A
 - gnss_obs_codes.h, [935](#)
- CODE_L6B
 - gnss_obs_codes.h, [936](#)
- CODE_L6C
 - gnss_obs_codes.h, [936](#)
- CODE_L6I
 - gnss_obs_codes.h, [936](#)
- CODE_L6L
 - gnss_obs_codes.h, [936](#)

- gnss_obs_codes.h, 936
- CODE_L6Q
 - gnss_obs_codes.h, 936
- CODE_L6S
 - gnss_obs_codes.h, 937
- CODE_L6X
 - gnss_obs_codes.h, 937
- CODE_L6Z
 - gnss_obs_codes.h, 937
- CODE_L7I
 - gnss_obs_codes.h, 937
- CODE_L7Q
 - gnss_obs_codes.h, 937
- CODE_L7X
 - gnss_obs_codes.h, 938
- CODE_L8I
 - gnss_obs_codes.h, 938
- CODE_L8Q
 - gnss_obs_codes.h, 938
- CODE_L8X
 - gnss_obs_codes.h, 938
- CODE_L9A
 - gnss_obs_codes.h, 938
- CODE_L9B
 - gnss_obs_codes.h, 939
- CODE_L9C
 - gnss_obs_codes.h, 939
- CODE_L9X
 - gnss_obs_codes.h, 939
- CODE_NONE
 - gnss_obs_codes.h, 939
- CODES_BDS
 - rtklib_rtc3.h, 1131
- CODES_GAL
 - rtklib_rtc3.h, 1131
- CODES_GLO
 - rtklib_rtc3.h, 1131
- CODES_GPS
 - rtklib_rtc3.h, 1131
- CODES_QZS
 - rtklib_rtc3.h, 1132
- CODES_SBS
 - rtklib_rtc3.h, 1132
- CRC_test
 - Glonass_Gnav_Navigation_Message, 341
- Carrier_Doppler_hz
 - Gnss_Synchro, 399
- carrier_lock_detector
 - lock_detectors.h, 1023
- Carrier_phase_rads
 - Gnss_Synchro, 400
- Carrier_wipeoff_multicorrelator_resampler
 - Fpga_Multicorrelator_8sc, 190
- cart2geo
 - geofunctions.h, 880
- cart2utm
 - geofunctions.h, 880
- Channel, 133
 - ~Channel, 135
 - Channel, 134
 - connect, 135
 - get_left_block, 135
 - get_left_block_acq, 135
 - get_left_block_trk, 136
 - implementation, 136
 - set_signal, 136
 - start_acquisition, 136
 - stop_channel, 136
- channel.h, 791
- Channel_Event, 137
- Channel_ID
 - Gnss_Synchro, 400
- channel_event.h, 792
- channel_fsm.h, 792
- channel_interface.h, 793
- channel_msg_receiver_cc, 137
 - ~channel_msg_receiver_cc, 138
- channel_msg_receiver_cc.h, 794
- channel_status_msg_receiver, 138
 - ~channel_status_msg_receiver, 139
 - get_current_status_map, 139
 - get_current_status_pvt, 139
- channel_status_msg_receiver.h, 794
- ChannelFsm, 140
- ChannelInterface, 141
- check_CRC
 - Rtcm, 672
- cl_fft_plan, 142
- cIFFT.h, 795
- cIFFT_Complex, 142
- cIFFT_Dim3, 143
- cIFFT_SplitComplex, 143
- clear
 - Gnss_circular_deque, 377
- clear_ephemeris
 - rtklib_pvt_gs, 693
- clksin
 - geofunctions.h, 880
- close_device
 - Fpga_Acquisition, 185
- clsin
 - geofunctions.h, 881
- cn0_m2m4_estimator
 - lock_detectors.h, 1023
- cn0_svn_estimator
 - lock_detectors.h, 1023
- cnav_msg.h, 796
- cnav_msg_decoder_t, 143
 - part1, 144
 - part2, 144
- cnav_msg_t, 144
 - alert, 145
 - msg_id, 145
 - prn, 145
 - raw_msg, 145
 - tow, 146

- cnav_v27_part_t, 146
 - crc_ok, 147
 - dec, 147
 - decisions, 147
 - decoded, 147
 - init, 147
 - invert, 147
 - message_lock, 148
 - n_crc_fail, 148
 - n_decoded, 148
 - n_symbols, 148
 - preamble_seen, 148
 - symbols, 149
- Code_Phase
 - Gps_Acq_Assist, 417
- Code_Phase_int
 - Gps_Acq_Assist, 417
- Code_Phase_window
 - Gps_Acq_Assist, 417
- Code_phase_samples
 - Gnss_Synchro, 400
- Command_Event, 149
- command_event.h, 797
- Common definitions, 67
- complex_byte_to_float_x2, 150
- complex_byte_to_float_x2.h, 798
- complex_exp_gen
 - gnss_signal_processing.h, 952
- complex_exp_gen_conj
 - gnss_signal_processing.h, 952
- complex_float_to_complex_byte, 150
- complex_float_to_complex_byte.h, 799
- compute_BDS_time
 - Rinex_Printer, 652
- compute_GLONASS_time
 - Glonnass_Gnav_Ephemeris, 327
- compute_GPS_time
 - Rinex_Printer, 653
- compute_Galileo_time
 - Rinex_Printer, 653
- compute_UTC_time
 - Rinex_Printer, 653
- Concurrent_Map< Data >, 151
- Concurrent_Queue< Data >, 152
- concurrent_map.h, 800
- concurrent_queue.h, 800
- configuration_interface.h, 801
- ConfigurationInterface, 152
- configure_acquisition
 - Fpga_Acquisition, 185
- conjugate_cc, 153
- conjugate_cc.h, 802
- conjugate_ic, 154
- conjugate_ic.h, 802
- conjugate_sc, 155
- conjugate_sc.h, 803
- connect
 - Channel, 135
 - GNSSFlowgraph, 411
 - GalileoE1DIIPIIVemlTrackingFpga, 242
 - GalileoE1PcpsAmbiguousAcquisitionFpga, 255
 - GalileoE5aDIIPIITrackingFpga, 278
 - GalileoE5aPcpsAcquisitionFpga, 291
 - GalileoE5bDIIPIITracking, 298
 - GalileoE5bPcpsAcquisition, 302
 - GalileoE5bPcpsAcquisitionFpga, 309
 - GalileoE5bTelemetryDecoder, 314
 - GpsL1CaDIIPIITrackingFpga, 482
 - GpsL1CaPcpsAcquisitionFpga, 499
 - GpsL5DIIPIITrackingFpga, 540
 - GpsL5iPcpsAcquisitionFpga, 549
- control_thread.h, 804
- ControlThread, 155
 - ~ControlThread, 157
 - ControlThread, 156
 - flowgraph, 157
 - run, 157
 - set_control_queue, 157
- convolutional.h, 805
 - Gamma, 806
 - nsc_enc_bit, 807
 - nsc_transit, 807
 - parity_counter, 807
 - Viterbi, 808
- correlation_length_ms
 - Gnss_Synchro, 401
- Cpu_Multicorrelator, 158
- Cpu_Multicorrelator_16sc, 158
- Cpu_Multicorrelator_Real_Codes, 159
- cpu_multicorrelator.h, 809
- cpu_multicorrelator_16sc.h, 809
- cpu_multicorrelator_real_codes.h, 810
- crc_ok
 - cnav_v27_part_t, 147
- createProtobuffer
 - Serdes_Gnss_Synchro, 712
 - Serdes_Monitor_Pvt, 715
- cshort_to_float_x2, 160
- cshort_to_float_x2.h, 811
- CubatureFilter, 160
- cuda_multicorrelator, 161
- cuda_multicorrelator.h, 811
- custom_udp_signal_source.h, 812
- CustomUDPSignalSource, 161
 - implementation, 162
- d1_subframe_decoder
 - Beidou_Dnav_Navigation_Message, 112
- d2_subframe_decoder
 - Beidou_Dnav_Navigation_Message, 112
- D2R
 - MATH_CONSTANTS.h, 1028
- d_A0
 - Gps_CNAV_Utc_Model, 445
 - Gps_Utc_Model, 477
- d_A0_GAL
 - Beidou_Dnav_Utc_Model, 115

- d_A0_GLO
 - Beidou_Dnav_Utc_Model, [115](#)
- d_A0_GPS
 - Beidou_Dnav_Utc_Model, [115](#)
- d_A0_UTC
 - Beidou_Dnav_Utc_Model, [115](#)
- d_A1
 - Gps_CNAV_Utc_Model, [445](#)
 - Gps_Utc_Model, [477](#)
- d_A1_GAL
 - Beidou_Dnav_Utc_Model, [115](#)
- d_A1_GLO
 - Beidou_Dnav_Utc_Model, [116](#)
- d_A1_GPS
 - Beidou_Dnav_Utc_Model, [116](#)
- d_A1_UTC
 - Beidou_Dnav_Utc_Model, [116](#)
- d_A2
 - Gps_CNAV_Utc_Model, [445](#)
 - Gps_Utc_Model, [477](#)
- d_A_DOT
 - Gps_CNAV_Ephemeris, [429](#)
- d_A_f0
 - Beidou_Dnav_Almanac, [89](#)
 - Beidou_Dnav_Ephemeris, [97](#)
 - Galileo_Almanac, [199](#)
 - Gps_Almanac, [421](#)
 - Gps_CNAV_Ephemeris, [429](#)
 - Gps_Ephemeris, [453](#)
- d_A_f1
 - Beidou_Dnav_Almanac, [89](#)
 - Beidou_Dnav_Ephemeris, [97](#)
 - Galileo_Almanac, [200](#)
 - Gps_Almanac, [421](#)
 - Gps_CNAV_Ephemeris, [429](#)
 - Gps_Ephemeris, [453](#)
- d_A_f2
 - Beidou_Dnav_Ephemeris, [98](#)
 - Gps_CNAV_Ephemeris, [430](#)
 - Gps_Ephemeris, [453](#)
- d_AODC
 - Beidou_Dnav_Ephemeris, [98](#)
- d_AODE
 - Beidou_Dnav_Ephemeris, [98](#)
- d_AXn
 - Glionass_Gnav_Ephemeris, [330](#)
- d_AYn
 - Glionass_Gnav_Ephemeris, [330](#)
- d_AZn
 - Glionass_Gnav_Ephemeris, [331](#)
- d_B1
 - Glionass_Gnav_Utc_Model, [345](#)
- d_B2
 - Glionass_Gnav_Utc_Model, [345](#)
- d_B_n
 - Glionass_Gnav_Ephemeris, [331](#)
- d_C_n
 - Glionass_Gnav_Almanac, [320](#)
- d_Cic
 - Beidou_Dnav_Ephemeris, [98](#)
 - Gps_CNAV_Ephemeris, [430](#)
 - Gps_Ephemeris, [453](#)
- d_Cis
 - Beidou_Dnav_Ephemeris, [99](#)
 - Gps_CNAV_Ephemeris, [430](#)
 - Gps_Ephemeris, [454](#)
- d_Crc
 - Beidou_Dnav_Ephemeris, [99](#)
 - Gps_CNAV_Ephemeris, [430](#)
 - Gps_Ephemeris, [454](#)
- d_Crs
 - Beidou_Dnav_Ephemeris, [99](#)
 - Gps_CNAV_Ephemeris, [431](#)
 - Gps_Ephemeris, [454](#)
- d_Cuc
 - Beidou_Dnav_Ephemeris, [99](#)
 - Gps_CNAV_Ephemeris, [431](#)
 - Gps_Ephemeris, [454](#)
- d_Cus
 - Beidou_Dnav_Ephemeris, [100](#)
 - Gps_CNAV_Ephemeris, [431](#)
 - Gps_Ephemeris, [455](#)
- d_DELTA_DOT_N
 - Gps_CNAV_Ephemeris, [432](#)
- d_DELTA_OMEGA_DOT
 - Gps_CNAV_Ephemeris, [432](#)
- d_DELTA_A
 - Gps_CNAV_Ephemeris, [431](#)
- d_Delta_T_n_A_dot
 - Glionass_Gnav_Almanac, [321](#)
- d_Delta_T_n_A
 - Glionass_Gnav_Almanac, [321](#)
- d_Delta_i
 - Galileo_Almanac, [200](#)
 - Gps_Almanac, [421](#)
- d_Delta_i_n_A
 - Glionass_Gnav_Almanac, [320](#)
- d_Delta_n
 - Beidou_Dnav_Ephemeris, [100](#)
 - Gps_CNAV_Ephemeris, [432](#)
 - Gps_Ephemeris, [455](#)
- d_Delta_sqrt_A
 - Galileo_Almanac, [200](#)
- d_Delta_tau_n
 - Glionass_Gnav_Ephemeris, [331](#)
- d_DeltaT_LSF
 - Beidou_Dnav_Utc_Model, [116](#)
 - Gps_CNAV_Utc_Model, [445](#)
 - Gps_Utc_Model, [478](#)
- d_DeltaT_LS
 - Gps_CNAV_Utc_Model, [445](#)
 - Gps_Utc_Model, [477](#)
- d_Doppler0
 - Gps_Acq_Assist, [418](#)
- d_Doppler1
 - Gps_Acq_Assist, [418](#)

- d_E_n
 - Glionass_Gnav_Ephemeris, 332
- d_F_T
 - Glionass_Gnav_Ephemeris, 332
- d_H_n_A
 - Glionass_Gnav_Almanac, 321
- d_IDOT
 - Beidou_Dnav_Ephemeris, 101
 - Gps_CNAV_Ephemeris, 433
 - Gps_Ephemeris, 456
- d_IODE_SF2
 - Gps_Ephemeris, 456
- d_IODE_SF3
 - Gps_Ephemeris, 457
- d_IODC
 - Gps_Ephemeris, 456
- d_KP
 - Glionass_Gnav_Almanac, 322
- d_M_0
 - Beidou_Dnav_Almanac, 89
 - Beidou_Dnav_Ephemeris, 101
 - Galileo_Almanac, 200
 - Gps_Almanac, 421
 - Gps_CNAV_Ephemeris, 433
 - Gps_Ephemeris, 457
- d_M_n_A
 - Glionass_Gnav_Almanac, 322
- d_N_4
 - Glionass_Gnav_Utc_Model, 345
- d_N_A
 - Glionass_Gnav_Utc_Model, 345
- d_N_T
 - Glionass_Gnav_Ephemeris, 334
- d_OMEGA0
 - Beidou_Dnav_Almanac, 90
 - Beidou_Dnav_Ephemeris, 102
 - Galileo_Almanac, 201
 - Gps_Almanac, 422
 - Gps_CNAV_Ephemeris, 434
 - Gps_Ephemeris, 457
- d_OMEGA_DOT
 - Beidou_Dnav_Almanac, 90
 - Beidou_Dnav_Ephemeris, 102
 - Galileo_Almanac, 201
 - Gps_Almanac, 422
 - Gps_Ephemeris, 458
- d_OMEGA
 - Beidou_Dnav_Almanac, 90
 - Beidou_Dnav_Ephemeris, 101
 - Galileo_Almanac, 201
 - Gps_Almanac, 421
 - Gps_CNAV_Ephemeris, 433
 - Gps_Ephemeris, 457
- d_P_1
 - Glionass_Gnav_Ephemeris, 334
- d_P_2
 - Glionass_Gnav_Ephemeris, 335
- d_P_3
 - Glionass_Gnav_Ephemeris, 335
- d_P_4
 - Glionass_Gnav_Ephemeris, 335
- d_TGD1
 - Beidou_Dnav_Ephemeris, 104
- d_TGD2
 - Beidou_Dnav_Ephemeris, 104
- d_TGD
 - Gps_CNAV_Ephemeris, 435
 - Gps_Ephemeris, 459
- d_TOW
 - Beidou_Dnav_Ephemeris, 105
 - Glionass_Gnav_Ephemeris, 337
 - Gps_Acq_Assist, 418
 - Gps_CNAV_Ephemeris, 436
 - Gps_Ephemeris, 460
- d_Toa
 - Beidou_Dnav_Almanac, 90
- d_Toc
 - Beidou_Dnav_Ephemeris, 104
 - Gps_CNAV_Ephemeris, 435
 - Gps_Ephemeris, 460
- d_Toe
 - Beidou_Dnav_Ephemeris, 104
 - Gps_Ephemeris, 460
- d_Toe1
 - Gps_CNAV_Ephemeris, 436
- d_Toe2
 - Gps_CNAV_Ephemeris, 436
- d_Top
 - Gps_CNAV_Ephemeris, 436
- d_URA0
 - Gps_CNAV_Ephemeris, 437
- d_URA1
 - Gps_CNAV_Ephemeris, 437
- d_URA2
 - Gps_CNAV_Ephemeris, 437
- d_VXn
 - Glionass_Gnav_Ephemeris, 337
- d_VYn
 - Glionass_Gnav_Ephemeris, 337
- d_VZn
 - Glionass_Gnav_Ephemeris, 337
- d_WN
 - Glionass_Gnav_Ephemeris, 337
- d_Xn
 - Glionass_Gnav_Ephemeris, 338
- d_Yn
 - Glionass_Gnav_Ephemeris, 338
- d_Zn
 - Glionass_Gnav_Ephemeris, 338
- d_acc
 - Sbas_Ephemeris, 703
- d_af0
 - Sbas_Ephemeris, 703
- d_af1
 - Sbas_Ephemeris, 703
- d_alpha0

- Beidou_Dnav_Iono, 108
- Gps_CNAV_Iono, 439
- Gps_Iono, 463
- d_alpha1
 - Beidou_Dnav_Iono, 108
 - Gps_CNAV_Iono, 439
 - Gps_Iono, 463
- d_alpha2
 - Beidou_Dnav_Iono, 108
 - Gps_CNAV_Iono, 440
 - Gps_Iono, 464
- d_alpha3
 - Beidou_Dnav_Iono, 109
 - Gps_CNAV_Iono, 440
 - Gps_Iono, 464
- d_beta0
 - Beidou_Dnav_Iono, 109
 - Gps_CNAV_Iono, 440
 - Gps_Iono, 464
- d_beta1
 - Beidou_Dnav_Iono, 109
 - Gps_CNAV_Iono, 440
 - Gps_Iono, 464
- d_beta2
 - Beidou_Dnav_Iono, 109
 - Gps_CNAV_Iono, 441
 - Gps_Iono, 465
- d_beta3
 - Beidou_Dnav_Iono, 110
 - Gps_CNAV_Iono, 441
 - Gps_Iono, 465
- d_dtr
 - Beidou_Dnav_Ephemeris, 100
 - Glionass_Gnav_Ephemeris, 331
 - Gps_CNAV_Ephemeris, 432
 - Gps_Ephemeris, 455
- d_e_eccentricity
 - Beidou_Dnav_Almanac, 89
 - Galileo_Almanac, 200
 - Gps_Almanac, 421
 - Gps_CNAV_Ephemeris, 432
 - Gps_Ephemeris, 455
- d_eccentricity
 - Beidou_Dnav_Ephemeris, 100
- d_epsilon_n_A
 - Glionass_Gnav_Almanac, 321
- d_gamma_n
 - Glionass_Gnav_Ephemeris, 332
- d_i_0
 - Beidou_Dnav_Ephemeris, 101
 - Gps_CNAV_Ephemeris, 433
 - Gps_Ephemeris, 456
- d_iode
 - Glionass_Gnav_Ephemeris, 332
- d_l3rd_n
 - Glionass_Gnav_Ephemeris, 333
- d_l5th_n
 - Glionass_Gnav_Ephemeris, 333
- d_l_n
 - Glionass_Gnav_Almanac, 322
- d_lambda_n_A
 - Glionass_Gnav_Almanac, 322
- d_M
 - Glionass_Gnav_Ephemeris, 333
- d_m
 - Glionass_Gnav_Ephemeris, 333
- d_n
 - Glionass_Gnav_Ephemeris, 334
- d_n_A
 - Glionass_Gnav_Almanac, 323
- d_omega_n_A
 - Glionass_Gnav_Almanac, 323
- d_P
 - Glionass_Gnav_Ephemeris, 334
- d_pos
 - Sbas_Ephemeris, 703
- d_satClkDrift
 - Beidou_Dnav_Ephemeris, 102
 - Glionass_Gnav_Ephemeris, 335
 - Gps_CNAV_Ephemeris, 434
 - Gps_Ephemeris, 458
- d_satpos_X
 - Beidou_Dnav_Ephemeris, 102
 - Galileo_Ephemeris, 215
 - Gps_CNAV_Ephemeris, 434
 - Gps_Ephemeris, 458
- d_satpos_Y
 - Beidou_Dnav_Ephemeris, 103
 - Galileo_Ephemeris, 216
 - Gps_CNAV_Ephemeris, 434
 - Gps_Ephemeris, 458
- d_satpos_Z
 - Beidou_Dnav_Ephemeris, 103
 - Galileo_Ephemeris, 216
 - Gps_CNAV_Ephemeris, 434
 - Gps_Ephemeris, 458
- d_satvel_X
 - Beidou_Dnav_Ephemeris, 103
 - Galileo_Ephemeris, 216
 - Gps_CNAV_Ephemeris, 435
 - Gps_Ephemeris, 459
- d_satvel_Y
 - Beidou_Dnav_Ephemeris, 103
 - Galileo_Ephemeris, 216
 - Gps_CNAV_Ephemeris, 435
 - Gps_Ephemeris, 459
- d_satvel_Z
 - Beidou_Dnav_Ephemeris, 103
 - Galileo_Ephemeris, 216
 - Gps_CNAV_Ephemeris, 435
 - Gps_Ephemeris, 459
- d_sqrt_A
 - Beidou_Dnav_Almanac, 90
 - Beidou_Dnav_Ephemeris, 104
 - Gps_Almanac, 422
 - Gps_Ephemeris, 459

- d_t_OT
 - Gps_CNAV_Utc_Model, [446](#)
 - Gps_Utc_Model, [478](#)
- d_t_b
 - Glonass_Gnav_Ephemeris, [336](#)
- d_t_k
 - Glonass_Gnav_Ephemeris, [336](#)
- d_t_lambda_n_A
 - Glonass_Gnav_Almanac, [323](#)
- d_tau_c
 - Glonass_Gnav_Ephemeris, [336](#)
 - Glonass_Gnav_Utc_Model, [346](#)
- d_tau_gps
 - Glonass_Gnav_Utc_Model, [346](#)
- d_tau_n
 - Glonass_Gnav_Ephemeris, [336](#)
- d_tau_n_A
 - Glonass_Gnav_Almanac, [323](#)
- d_tod
 - Glonass_Gnav_Ephemeris, [336](#)
- d_tof
 - Sbas_Ephemeris, [704](#)
- d_vel
 - Sbas_Ephemeris, [704](#)
- d_yr
 - Glonass_Gnav_Ephemeris, [338](#)
- DECLARE_bool
 - gnss_sdr_flags.h, [942](#)
- DECLARE_double
 - gnss_sdr_flags.h, [942](#), [943](#)
- DECLARE_int32
 - gnss_sdr_flags.h, [943](#), [944](#)
- DECLARE_string
 - gnss_sdr_flags.h, [944](#), [945](#)
- DFRQ1_GLO
 - gnss_frequencies.h, [923](#)
- DFRQ2_GLO
 - gnss_frequencies.h, [923](#)
- DTTOL
 - rtklib.h, [1085](#)
- dec
 - cnav_v27_part_t, [147](#)
- decisions
 - cnav_v27_part_t, [147](#)
- decode_block
 - Viterbi_Decoder, [755](#)
- decoded
 - cnav_v27_part_t, [147](#)
- delta_n_3
 - Galileo_Ephemeris, [217](#)
- dgps_t, [162](#)
- direct_resampler_conditioner.h, [813](#)
- direct_resampler_conditioner_cb, [163](#)
- direct_resampler_conditioner_cb.h, [814](#)
- direct_resampler_conditioner_cc, [164](#)
- direct_resampler_conditioner_cc.h, [814](#)
- direct_resampler_conditioner_cs, [164](#)
- direct_resampler_conditioner_cs.h, [815](#)
- DirectResamplerConditioner, [165](#)
 - implementation, [166](#)
- disable_secondary_codes
 - Fpga_Multicorrelator_8sc, [191](#)
- disconnect
 - GNSSFlowgraph, [411](#)
 - GalileoE1DIIPIIVemlTrackingFpga, [243](#)
 - GalileoE1PcpsAmbiguousAcquisitionFpga, [256](#)
 - GalileoE5aDIIPIITrackingFpga, [278](#)
 - GalileoE5aPcpsAcquisitionFpga, [291](#)
 - GalileoE5bDIIPIITracking, [298](#)
 - GalileoE5bPcpsAcquisition, [302](#)
 - GalileoE5bPcpsAcquisitionFpga, [309](#)
 - GalileoE5bTelemetryDecoder, [315](#)
 - GpsL1CaDIIPIITrackingFpga, [483](#)
 - GpsL1CaPcpsAcquisitionFpga, [499](#)
 - GpsL5DIIPIITrackingFpga, [540](#)
 - GpsL5IPcpsAcquisitionFpga, [549](#)
- display.h, [816](#)
- DII_PII_Conf, [166](#)
- DII_PII_Conf_Fpga, [167](#)
- dll_nc_e_minus_l_normalized
 - tracking_discriminators.h, [1172](#)
- dll_nc_vemlp_normalized
 - tracking_discriminators.h, [1173](#)
- dll_pll_conf.h, [817](#)
- dll_pll_conf_fpga.h, [818](#)
- dll_pll_veml_tracking, [169](#)
- dll_pll_veml_tracking.h, [818](#)
- dll_pll_veml_tracking_fpga, [170](#)
 - ~dll_pll_veml_tracking_fpga, [170](#)
- general_work, [171](#)
- reset, [171](#)
- set_channel, [171](#)
- set_gnss_synchro, [171](#)
- start_tracking, [171](#)
- stop_tracking, [172](#)
- dll_pll_veml_tracking_fpga.h, [819](#)
- dopplerUncertainty
 - Gps_Acq_Assist, [418](#)
- E1B_DVS_5
 - Galileo_Ephemeris, [217](#)
- E1B_HS_5
 - Galileo_Ephemeris, [217](#)
- E5a_DVS
 - Galileo_Ephemeris, [217](#)
- E5a_HS
 - Galileo_Ephemeris, [218](#)
- E5b_DVS_5
 - Galileo_Ephemeris, [218](#)
- E5b_HS_5
 - Galileo_Ephemeris, [218](#)
- e_1
 - Galileo_Ephemeris, [218](#)
- EFACT_BDS
 - rtklib.h, [1086](#)
- EFACT_GAL
 - rtklib.h, [1086](#)

- EFACT_GLO
 - rtklib.h, [1086](#)
- EFACT_GPS
 - rtklib.h, [1086](#)
- EFACT_IRN
 - rtklib.h, [1086](#)
- EFACT_QZS
 - rtklib.h, [1087](#)
- EFACT_SBS
 - rtklib.h, [1087](#)
- EPHOPT_BRDC
 - rtklib.h, [1087](#)
- EPHOPT_LEX
 - rtklib.h, [1087](#)
- EPHOPT_PREC
 - rtklib.h, [1087](#)
- EPHOPT_SBAS
 - rtklib.h, [1088](#)
- EPHOPT_SSRAPC
 - rtklib.h, [1088](#)
- EPHOPT_SSRCOM
 - rtklib.h, [1088](#)
- ERR_BRDCI
 - rtklib.h, [1088](#)
- ERR_CBIAS
 - rtklib.h, [1088](#)
- ERR_ION
 - rtklib_pntpos.h, [1120](#)
- ERR_SAAS
 - rtklib.h, [1089](#)
- ERR_TROP
 - rtklib_pntpos.h, [1120](#)
- edc.h, [820](#)
- Elevation
 - Gps_Acq_Assist, [418](#)
- enable_secondary_codes
 - Fpga_Multicorrelator_8sc, [191](#)
- eph_t, [172](#)
- eph_to_rtklib
 - rtklib_conversions.h, [1114](#)
- erp_t, [173](#)
- erpd_t, [173](#)
- estimate_doppler_from_eph
 - FrontEndCal, [196](#)
- Exponential_Smoother, [174](#)
 - ~Exponential_Smoother, [174](#)
 - Exponential_Smoother, [174](#), [175](#)
 - operator=, [175](#)
 - set_alpha, [175](#)
 - set_samples_for_initialization, [175](#)
- exponential_smoother.h, [821](#)
- exterr_t, [176](#)
- FE_WGS84
 - rtklib.h, [1089](#)
- FREQ1
 - gnss_frequencies.h, [923](#)
- FREQ1_BDS
 - gnss_frequencies.h, [923](#)
- FREQ1_GLO
 - gnss_frequencies.h, [923](#)
- FREQ2
 - gnss_frequencies.h, [923](#)
- FREQ2_BDS
 - gnss_frequencies.h, [924](#)
- FREQ2_GLO
 - gnss_frequencies.h, [924](#)
- FREQ3_BDS
 - gnss_frequencies.h, [924](#)
- FREQ3_GLO
 - gnss_frequencies.h, [924](#)
- FREQ5
 - gnss_frequencies.h, [924](#)
- FREQ6
 - gnss_frequencies.h, [925](#)
- FREQ7
 - gnss_frequencies.h, [925](#)
- FREQ8
 - gnss_frequencies.h, [925](#)
- FREQ9
 - gnss_frequencies.h, [925](#)
- FTP_TIMEOUT
 - rtklib.h, [1089](#)
- fatalfunc_t
 - rtklib.h, [1084](#)
- fcbd_t, [176](#)
- fec.h, [822](#)
- fft_base_kernels.h, [822](#)
- fft_internal.h, [823](#)
- file_configuration.h, [824](#)
- file_signal_source.h, [824](#)
- file_t, [177](#)
- FileConfiguration, [177](#)
- FileSignalSource, [178](#)
 - implementation, [179](#)
- findUtmZone
 - geofunctions.h, [881](#)
- fir_filter.h, [825](#)
- FirFilter, [179](#)
 - ~FirFilter, [180](#)
 - FirFilter, [180](#)
 - implementation, [181](#)
- Flag_valid_acquisition
 - Gnss_Synchro, [401](#)
- Flag_valid_pseudorange
 - Gnss_Synchro, [401](#)
- Flag_valid_symbol_output
 - Gnss_Synchro, [401](#)
- Flag_valid_word
 - Gnss_Synchro, [402](#)
- flexiband_signal_source.h, [826](#)
- FlexibandSignalSource, [181](#)
 - implementation, [182](#)
- fil_four_quadrant_atan
 - tracking_discriminators.h, [1173](#)
- flowgraph
 - ControlThread, [157](#)

- fmcomms2_signal_source.h, [827](#)
- Fmcomms2SignalSource, [182](#)
 - implementation, [183](#)
- Fpga_Acquisition, [183](#)
 - ~Fpga_Acquisition, [184](#)
 - close_device, [185](#)
 - configure_acquisition, [185](#)
 - Fpga_Acquisition, [184](#)
 - open_device, [185](#)
 - read_acquisition_results, [185](#)
 - read_fpga_total_scale_factor, [185](#)
 - reset_acquisition, [185](#)
 - run_acquisition, [186](#)
 - set_block_exp, [186](#)
 - set_doppler_max, [186](#)
 - set_doppler_step, [186](#)
 - set_doppler_sweep, [187](#)
 - set_local_code, [187](#)
 - write_local_code, [187](#)
- Fpga_Multicorrelator_8sc, [189](#)
 - ~Fpga_Multicorrelator_8sc, [190](#)
 - Carrier_wipeoff_multicorrelator_resampler, [190](#)
 - disable_secondary_codes, [191](#)
 - enable_secondary_codes, [191](#)
 - Fpga_Multicorrelator_8sc, [190](#)
 - free, [191](#)
 - initialize_secondary_code, [191](#)
 - lock_channel, [191](#)
 - read_sample_counter, [192](#)
 - set_channel, [192](#)
 - set_initial_sample, [192](#)
 - set_local_code_and_taps, [192](#)
 - set_output_vectors, [192](#)
 - set_secondary_code_lengths, [193](#)
 - unlock_channel, [193](#)
 - update_local_code, [193](#)
 - update_prn_code_length, [193](#)
- Fpga_Switch, [194](#)
 - ~Fpga_Switch, [194](#)
 - Fpga_Switch, [194](#)
 - set_switch_position, [194](#)
- fpga_acquisition.h, [827](#)
- Fpga_dynamic_bit_selection, [187](#)
 - ~Fpga_dynamic_bit_selection, [188](#)
 - bit_selection, [188](#)
 - Fpga_dynamic_bit_selection, [188](#)
- fpga_dynamic_bit_selection.h, [828](#)
- fpga_multicorrelator.h, [829](#)
- fpga_switch.h, [829](#)
- free
 - Fpga_Multicorrelator_8sc, [191](#)
- freq_xlating_fir_filter.h, [830](#)
- FreqXlatingFirFilter, [195](#)
 - implementation, [196](#)
- front
 - Gnss_circular_deque, [377](#)
- front_end_cal.h, [831](#)
- FrontEndCal, [196](#)
 - estimate_doppler_from_eph, [196](#)
 - GPS_L1_front_end_model_E4000, [197](#)
 - get_ephemeris, [197](#)
 - set_configuration, [197](#)
- fs
 - Gnss_Synchro, [402](#)
- ftp_t, [198](#)
- GALILEO_E1_B_CODE_LENGTH_CHIPS
 - Galileo_E1.h, [834](#)
- GALILEO_E1_B_SAMPLES_PER_SYMBOL
 - Galileo_E1.h, [834](#)
- GALILEO_E1_B_SYMBOL_RATE_BPS
 - Galileo_E1.h, [834](#)
- GALILEO_E1_C_SECONDARY_CODE_LENGTH
 - Galileo_E1.h, [834](#)
- GALILEO_E1_CODE_CHIP_RATE_CPS
 - Galileo_E1.h, [834](#)
- GALILEO_E1_CODE_PERIOD_MS
 - Galileo_E1.h, [835](#)
- GALILEO_E1_CODE_PERIOD_S
 - Galileo_E1.h, [835](#)
- GALILEO_E1_FREQ_HZ
 - Galileo_E1.h, [835](#)
- GALILEO_E1_HISTORY_DEEP
 - Galileo_E1.h, [835](#)
- GALILEO_E1_OPT_ACQ_FS_SPS
 - Galileo_E1.h, [835](#)
- GALILEO_E1_SUB_CARRIER_A_RATE_HZ
 - Galileo_E1.h, [836](#)
- GALILEO_E1_SUB_CARRIER_B_RATE_HZ
 - Galileo_E1.h, [836](#)
- GALILEO_E5A_CODE_CHIP_RATE_CPS
 - Galileo_E5a.h, [850](#)
- GALILEO_E5A_CODE_LENGTH_CHIPS
 - Galileo_E5a.h, [850](#)
- GALILEO_E5A_CODE_PERIOD_MS
 - Galileo_E5a.h, [850](#)
- GALILEO_E5A_CODE_PERIOD_S
 - Galileo_E5a.h, [850](#)
- GALILEO_E5A_FREQ_HZ
 - Galileo_E5a.h, [851](#)
- GALILEO_E5A_I_SECONDARY_CODE_LENGTH
 - Galileo_E5a.h, [851](#)
- GALILEO_E5A_I_TIERED_CODE_PERIOD_S
 - Galileo_E5a.h, [851](#)
- GALILEO_E5A_OPT_ACQ_FS_SPS
 - Galileo_E5a.h, [851](#)
- GALILEO_E5A_Q_SECONDARY_CODE_LENGTH
 - Galileo_E5a.h, [851](#)
- GALILEO_E5A_Q_TIERED_CODE_PERIOD_S
 - Galileo_E5a.h, [852](#)
- GALILEO_E5A_SYMBOL_RATE_BPS
 - Galileo_E5a.h, [852](#)
- GALILEO_E5B_CODE_CHIP_RATE_CPS
 - Galileo_E5b.h, [859](#)
- GALILEO_E5B_CODE_LENGTH_CHIPS
 - Galileo_E5b.h, [859](#)
- GALILEO_E5B_CODE_PERIOD_MS

- Galileo_E5b.h, [859](#)
- GALILEO_E5B_CODE_PERIOD_S
 - Galileo_E5b.h, [859](#)
- GALILEO_E5B_FREQ_HZ
 - Galileo_E5b.h, [859](#)
- GALILEO_E5B_I_SECONDARY_CODE_LENGTH
 - Galileo_E5b.h, [859](#)
- GALILEO_E5B_I_TIERED_CODE_PERIOD_S
 - Galileo_E5b.h, [860](#)
- GALILEO_E5B_OPT_ACQ_FS_SPS
 - Galileo_E5b.h, [860](#)
- GALILEO_E5B_Q_SECONDARY_CODE_LENGTH
 - Galileo_E5b.h, [860](#)
- GALILEO_E5B_Q_TIERED_CODE_PERIOD_S
 - Galileo_E5b.h, [860](#)
- GALILEO_E5B_SYMBOL_RATE_BPS
 - Galileo_E5b.h, [860](#)
- GALILEO_GM
 - MATH_CONSTANTS.h, [1029](#)
- GALILEO_INAV_PAGE_PART_SYMBOLS
 - Galileo_INAV.h, [872](#)
- GALILEO_INAV_PAGE_PART_WITH_PREABLE_S↔
 - ECONDS
 - Galileo_INAV.h, [873](#)
- GALILEO_INAV_PAGE_SYMBOLS
 - Galileo_INAV.h, [873](#)
- GALILEO_F
 - MATH_CONSTANTS.h, [1029](#)
- GAP_RESION
 - rtklib.h, [1089](#)
- GLONASS_C20
 - GLONASS_L1_L2_CA.h, [898](#)
- GLONASS_EARTH_INCLINATION
 - GLONASS_L1_L2_CA.h, [898](#)
- GLONASS_EARTH_RADIUS
 - GLONASS_L1_L2_CA.h, [898](#)
- GLONASS_F_M_A
 - GLONASS_L1_L2_CA.h, [899](#)
- GLONASS_FLATTENING
 - GLONASS_L1_L2_CA.h, [899](#)
- GLONASS_GNAV_HAMMING_CODE_BITS
 - GLONASS_L1_L2_CA.h, [899](#)
- GLONASS_GNAV_PREAMBLE
 - GLONASS_L1_L2_CA.h, [898](#)
- GLONASS_GNAV_STRING_BITS
 - GLONASS_L1_L2_CA.h, [899](#)
- GLONASS_GNAV_STRING_SYMBOLS
 - GLONASS_L1_L2_CA.h, [899](#)
- GLONASS_GNAV_TELEMETRY_RATE_BITS_SEC↔
 - OND
 - GLONASS_L1_L2_CA.h, [900](#)
- GLONASS_GNAV_TELEMETRY_RATE_SYMBOLS↔
 - _SECOND
 - GLONASS_L1_L2_CA.h, [900](#)
- GLONASS_GRAVITY_CORRECTION
 - GLONASS_L1_L2_CA.h, [900](#)
- GLONASS_GRAVITY
 - GLONASS_L1_L2_CA.h, [900](#)
- GLONASS_GM
 - MATH_CONSTANTS.h, [1029](#)
- GLONASS_J2
 - GLONASS_L1_L2_CA.h, [900](#)
- GLONASS_J4
 - GLONASS_L1_L2_CA.h, [901](#)
- GLONASS_J6
 - GLONASS_L1_L2_CA.h, [901](#)
- GLONASS_J8
 - GLONASS_L1_L2_CA.h, [901](#)
- GLONASS_L1_CA_CHIP_PERIOD_S
 - GLONASS_L1_L2_CA.h, [901](#)
- GLONASS_L1_CA_CODE_LENGTH_CHIPS
 - GLONASS_L1_L2_CA.h, [901](#)
- GLONASS_L1_CA_CODE_PERIOD_S
 - GLONASS_L1_L2_CA.h, [902](#)
- GLONASS_L1_CA_CODE_RATE_CPS
 - GLONASS_L1_L2_CA.h, [902](#)
- GLONASS_L1_CA_DFREQ_HZ
 - GLONASS_L1_L2_CA.h, [902](#)
- GLONASS_L1_CA_FREQ_HZ
 - GLONASS_L1_L2_CA.h, [902](#)
- GLONASS_L1_L2_CA.h, [894](#)
- GLONASS_C20, [898](#)
- GLONASS_EARTH_INCLINATION, [898](#)
- GLONASS_EARTH_RADIUS, [898](#)
- GLONASS_F_M_A, [899](#)
- GLONASS_FLATTENING, [899](#)
- GLONASS_GNAV_HAMMING_CODE_BITS, [899](#)
- GLONASS_GNAV_PREAMBLE, [898](#)
- GLONASS_GNAV_STRING_BITS, [899](#)
- GLONASS_GNAV_STRING_SYMBOLS, [899](#)
- GLONASS_GNAV_TELEMETRY_RATE_BITS↔
 - SECOND, [900](#)
- GLONASS_GNAV_TELEMETRY_RATE_SYMB↔
 - OLS_SECOND, [900](#)
- GLONASS_GRAVITY_CORRECTION, [900](#)
- GLONASS_GRAVITY, [900](#)
- GLONASS_J2, [900](#)
- GLONASS_J4, [901](#)
- GLONASS_J6, [901](#)
- GLONASS_J8, [901](#)
- GLONASS_L1_CA_CHIP_PERIOD_S, [901](#)
- GLONASS_L1_CA_CODE_LENGTH_CHIPS, [901](#)
- GLONASS_L1_CA_CODE_PERIOD_S, [902](#)
- GLONASS_L1_CA_CODE_RATE_CPS, [902](#)
- GLONASS_L1_CA_DFREQ_HZ, [902](#)
- GLONASS_L1_CA_FREQ_HZ, [902](#)
- GLONASS_L2_CA_CHIP_PERIOD_S, [902](#)
- GLONASS_L2_CA_CODE_LENGTH_CHIPS, [903](#)
- GLONASS_L2_CA_CODE_PERIOD_S, [903](#)
- GLONASS_L2_CA_CODE_RATE_CPS, [903](#)
- GLONASS_L2_CA_DFREQ_HZ, [903](#)
- GLONASS_L2_CA_FREQ_HZ, [903](#)
- GLONASS_LEAP_SECONDS, [904](#)
- GLONASS_MOON_ECCENTRICITY, [904](#)
- GLONASS_MOON_GM, [904](#)
- GLONASS_MOON_INCLINATION, [905](#)

- GLONASS_MOON_OMEGA_0, [905](#)
- GLONASS_MOON_OMEGA_1, [905](#)
- GLONASS_MOON_Q0, [905](#)
- GLONASS_MOON_Q1, [905](#)
- GLONASS_MOON_SEMI_MAJOR_AXIS, [906](#)
- GLONASS_SEMI_MAJOR_AXIS, [906](#)
- GLONASS_SUN_ECCENTRICITY, [906](#)
- GLONASS_SUN_GM, [906](#)
- GLONASS_SUN_OMEGA, [906](#)
- GLONASS_SUN_Q0, [907](#)
- GLONASS_SUN_Q1, [907](#)
- GLONASS_SUN_SEMI_MAJOR_AXIS, [907](#)
- GLONASS_TAU_0, [907](#)
- GLONASS_TAU_1, [907](#)
- GLONASS_U0, [908](#)
- GLONASS_L2_CA_CHIP_PERIOD_S
 - GLONASS_L1_L2_CA.h, [902](#)
- GLONASS_L2_CA_CODE_LENGTH_CHIPS
 - GLONASS_L1_L2_CA.h, [903](#)
- GLONASS_L2_CA_CODE_PERIOD_S
 - GLONASS_L1_L2_CA.h, [903](#)
- GLONASS_L2_CA_CODE_RATE_CPS
 - GLONASS_L1_L2_CA.h, [903](#)
- GLONASS_L2_CA_DFREQ_HZ
 - GLONASS_L1_L2_CA.h, [903](#)
- GLONASS_L2_CA_FREQ_HZ
 - GLONASS_L1_L2_CA.h, [903](#)
- GLONASS_LEAP_SECONDS
 - GLONASS_L1_L2_CA.h, [904](#)
- GLONASS_MOON_ECCENTRICITY
 - GLONASS_L1_L2_CA.h, [904](#)
- GLONASS_MOON_GM
 - GLONASS_L1_L2_CA.h, [904](#)
- GLONASS_MOON_INCLINATION
 - GLONASS_L1_L2_CA.h, [905](#)
- GLONASS_MOON_OMEGA_0
 - GLONASS_L1_L2_CA.h, [905](#)
- GLONASS_MOON_OMEGA_1
 - GLONASS_L1_L2_CA.h, [905](#)
- GLONASS_MOON_Q0
 - GLONASS_L1_L2_CA.h, [905](#)
- GLONASS_MOON_Q1
 - GLONASS_L1_L2_CA.h, [905](#)
- GLONASS_MOON_SEMI_MAJOR_AXIS
 - GLONASS_L1_L2_CA.h, [906](#)
- GLONASS_OMEGA_EARTH_DOT
 - MATH_CONSTANTS.h, [1029](#)
- GLONASS_SEMI_MAJOR_AXIS
 - GLONASS_L1_L2_CA.h, [906](#)
- GLONASS_SUN_ECCENTRICITY
 - GLONASS_L1_L2_CA.h, [906](#)
- GLONASS_SUN_GM
 - GLONASS_L1_L2_CA.h, [906](#)
- GLONASS_SUN_OMEGA
 - GLONASS_L1_L2_CA.h, [906](#)
- GLONASS_SUN_Q0
 - GLONASS_L1_L2_CA.h, [907](#)
- GLONASS_SUN_Q1
 - GLONASS_L1_L2_CA.h, [907](#)
- GLONASS_L1_L2_CA.h, [907](#)
- GLONASS_SUN_SEMI_MAJOR_AXIS, [907](#)
- GLONASS_TAU_0, [907](#)
- GLONASS_TAU_1, [907](#)
- GLONASS_U0, [908](#)
- GNSS_OMEGA_EARTH_DOT
 - MATH_CONSTANTS.h, [1029](#)
- GNSS_PI
 - MATH_CONSTANTS.h, [1030](#)
- GNSSBlockFactory, [406](#)
 - GetBlock, [407](#)
- GNSSBlockInterface, [408](#)
- GNSSFlowgraph, [409](#)
 - ~GNSSFlowgraph, [410](#)
 - acquisition_manager, [410](#)
 - apply_action, [410](#)
 - connect, [411](#)
 - disconnect, [411](#)
 - GNSSFlowgraph, [410](#)
 - get_pvt, [411](#)
 - priorize_satellites, [411](#)
 - send_telemetry_msg, [411](#)
 - set_configuration, [412](#)
 - start, [412](#)
 - stop, [412](#)
 - wait, [412](#)
- GPS_Bit_Number
 - Gps_Acq_Assist, [419](#)
- GPS_CA_TELEMETRY_RATE_BITS_SECOND
 - GPS_L1_CA.h, [968](#)
- GPS_CA_TELEMETRY_RATE_SYMBOLS_SECOND
 - GPS_L1_CA.h, [968](#)
- GPS_CNAV.h, [957](#)
- GPS_GM
 - MATH_CONSTANTS.h, [1030](#)
- GPS_L1_CA.h, [964](#)
 - ALPHA_0, [968](#)
 - GPS_CA_TELEMETRY_RATE_BITS_SECOND, [968](#)
 - GPS_CA_TELEMETRY_RATE_SYMBOLS_SECOND, [968](#)
 - GPS_L1_CA_BIT_PERIOD_MS, [968](#)
 - GPS_L1_CA_CHIP_PERIOD_S, [969](#)
 - GPS_L1_CA_CODE_LENGTH_CHIPS, [969](#)
 - GPS_L1_CA_CODE_PERIOD_MS, [969](#)
 - GPS_L1_CA_CODE_PERIOD_S, [969](#)
 - GPS_L1_CA_CODE_RATE_CPS, [969](#)
 - GPS_L1_CA_OPT_ACQ_FS_SPS, [970](#)
 - GPS_L1_FREQ_HZ, [970](#)
 - GPS_SUBFRAME_BITS, [970](#)
 - GPS_SUBFRAME_LENGTH, [970](#)
 - GPS_SUBFRAME_MS, [970](#)
 - GPS_SUBFRAME_SECONDS, [971](#)
 - GPS_WORD_BITS, [971](#)

- GPS_WORD_LENGTH, [971](#)
- MAX_TOA_DELAY_MS, [971](#)
- T_OA, [968](#)
- GPS_L1_CA_BIT_PERIOD_MS
 - GPS_L1_CA.h, [968](#)
- GPS_L1_CA_CHIP_PERIOD_S
 - GPS_L1_CA.h, [969](#)
- GPS_L1_CA_CODE_LENGTH_CHIPS
 - GPS_L1_CA.h, [969](#)
- GPS_L1_CA_CODE_PERIOD_MS
 - GPS_L1_CA.h, [969](#)
- GPS_L1_CA_CODE_PERIOD_S
 - GPS_L1_CA.h, [969](#)
- GPS_L1_CA_CODE_RATE_CPS
 - GPS_L1_CA.h, [969](#)
- GPS_L1_CA_OPT_ACQ_FS_SPS
 - GPS_L1_CA.h, [970](#)
- GPS_L1_FREQ_HZ
 - GPS_L1_CA.h, [970](#)
- GPS_L1_front_end_model_E4000
 - FrontEndCal, [197](#)
- GPS_L2, [68](#)
- GPS_L2_CNAV_DATA_PAGE_BITS
 - GPS_L2C.h, [988](#)
- GPS_L2_FREQ_HZ
 - GPS_L2C.h, [989](#)
- GPS_L2_L_CODE_LENGTH_CHIPS
 - GPS_L2C.h, [989](#)
- GPS_L2_L_CODE_RATE_CPS
 - GPS_L2C.h, [989](#)
- GPS_L2_L_PERIOD_S
 - GPS_L2C.h, [989](#)
- GPS_L2_M_CODE_LENGTH_CHIPS
 - GPS_L2C.h, [989](#)
- GPS_L2_M_CODE_RATE_CPS
 - GPS_L2C.h, [990](#)
- GPS_L2_M_PERIOD_S
 - GPS_L2C.h, [990](#)
- GPS_L2_V27_HISTORY_LENGTH_BITS
 - Gps_cnav_decoder, [69](#)
- GPS_L2C.h, [987](#)
 - GPS_L2_CNAV_DATA_PAGE_BITS, [988](#)
 - GPS_L2_FREQ_HZ, [989](#)
 - GPS_L2_L_CODE_LENGTH_CHIPS, [989](#)
 - GPS_L2_L_CODE_RATE_CPS, [989](#)
 - GPS_L2_L_PERIOD_S, [989](#)
 - GPS_L2_M_CODE_LENGTH_CHIPS, [989](#)
 - GPS_L2_M_CODE_RATE_CPS, [990](#)
 - GPS_L2_M_PERIOD_S, [990](#)
 - GPS_L2C_M_INIT_REG, [990](#)
 - GPS_L2C_OPT_ACQ_FS_SPS, [990](#)
- GPS_L2C_M_INIT_REG
 - GPS_L2C.h, [990](#)
- GPS_L2C_OPT_ACQ_FS_SPS
 - GPS_L2C.h, [990](#)
- GPS_L2C_V27_DECODE_BITS
 - Gps_cnav_decoder, [69](#)
- GPS_L2C_V27_DELAY_BITS
 - Gps_cnav_decoder, [69](#)
- GPS_L2C_V27_INIT_BITS
 - Gps_cnav_decoder, [70](#)
- GPS_L5.h, [993](#)
 - GPS_L5_CNAV_DATA_PAGE_BITS, [995](#)
 - GPS_L5_FREQ_HZ, [995](#)
 - GPS_L5_OPT_ACQ_FS_SPS, [995](#)
 - GPS_L5I_CODE_LENGTH_CHIPS, [995](#)
 - GPS_L5I_CODE_RATE_CPS, [995](#)
 - GPS_L5I_PERIOD_MS, [995](#)
 - GPS_L5I_PERIOD_S, [996](#)
 - GPS_L5I_SYMBOL_PERIOD_MS, [996](#)
 - GPS_L5I_SYMBOL_PERIOD_S, [996](#)
 - GPS_L5Q_CODE_LENGTH_CHIPS, [996](#)
 - GPS_L5Q_CODE_RATE_CPS, [996](#)
 - GPS_L5Q_PERIOD_S, [997](#)
- GPS_L5_CNAV_DATA_PAGE_BITS
 - GPS_L5.h, [995](#)
- GPS_L5_FREQ_HZ
 - GPS_L5.h, [995](#)
- GPS_L5_OPT_ACQ_FS_SPS
 - GPS_L5.h, [995](#)
- GPS_L5I_CODE_LENGTH_CHIPS
 - GPS_L5.h, [995](#)
- GPS_L5I_CODE_RATE_CPS
 - GPS_L5.h, [995](#)
- GPS_L5I_PERIOD_MS
 - GPS_L5.h, [995](#)
- GPS_L5I_PERIOD_S
 - GPS_L5.h, [996](#)
- GPS_L5I_SYMBOL_PERIOD_MS
 - GPS_L5.h, [996](#)
- GPS_L5I_SYMBOL_PERIOD_S
 - GPS_L5.h, [996](#)
- GPS_L5Q_CODE_LENGTH_CHIPS
 - GPS_L5.h, [996](#)
- GPS_L5Q_CODE_RATE_CPS
 - GPS_L5.h, [996](#)
- GPS_L5Q_PERIOD_S
 - GPS_L5.h, [997](#)
- GPS_SUBFRAME_BITS
 - GPS_L1_CA.h, [970](#)
- GPS_SUBFRAME_LENGTH
 - GPS_L1_CA.h, [970](#)
- GPS_SUBFRAME_MS
 - GPS_L1_CA.h, [970](#)
- GPS_SUBFRAME_SECONDS
 - GPS_L1_CA.h, [971](#)
- GPS_WORD_BITS
 - GPS_L1_CA.h, [971](#)
- GPS_WORD_LENGTH
 - GPS_L1_CA.h, [971](#)
- GPS_F
 - MATH_CONSTANTS.h, [1030](#)
- GPU_Complex, [555](#)
- GPU_Complex_Short, [555](#)
- GST_to_UTC_time
 - Galileo_Utc_Model, [237](#)

- Galileo_Almanac, [198](#)
 - d_A_f0, [199](#)
 - d_A_f1, [200](#)
 - d_Delta_i, [200](#)
 - d_Delta_sqrt_A, [200](#)
 - d_M_0, [200](#)
 - d_OMEGA0, [201](#)
 - d_OMEGA_DOT, [201](#)
 - d_OMEGA, [201](#)
 - d_e_eccentricity, [200](#)
 - Galileo_Almanac, [199](#)
 - i_satellite_PRN, [201](#)
- Galileo_Almanac_Helper, [202](#)
 - Galileo_Almanac_Helper, [203](#)
- Galileo_E1.h, [832](#)
 - GALILEO_E1_B_CODE_LENGTH_CHIPS, [834](#)
 - GALILEO_E1_B_SAMPLES_PER_SYMBOL, [834](#)
 - GALILEO_E1_B_SYMBOL_RATE_BPS, [834](#)
 - GALILEO_E1_C_SECONDARY_CODE_LENGTH, [834](#)
 - GALILEO_E1_CODE_CHIP_RATE_CPS, [834](#)
 - GALILEO_E1_CODE_PERIOD_MS, [835](#)
 - GALILEO_E1_CODE_PERIOD_S, [835](#)
 - GALILEO_E1_FREQ_HZ, [835](#)
 - GALILEO_E1_HISTORY_DEEP, [835](#)
 - GALILEO_E1_OPT_ACQ_FS_SPS, [835](#)
 - GALILEO_E1_SUB_CARRIER_A_RATE_HZ, [836](#)
 - GALILEO_E1_SUB_CARRIER_B_RATE_HZ, [836](#)
- Galileo_E1_Tcp_Connector_Tracking_cc, [203](#)
- Galileo_E5a.h, [848](#)
 - GALILEO_E5A_CODE_CHIP_RATE_CPS, [850](#)
 - GALILEO_E5A_CODE_LENGTH_CHIPS, [850](#)
 - GALILEO_E5A_CODE_PERIOD_MS, [850](#)
 - GALILEO_E5A_CODE_PERIOD_S, [850](#)
 - GALILEO_E5A_FREQ_HZ, [851](#)
 - GALILEO_E5A_I_SECONDARY_CODE_LENGTH, [851](#)
 - GALILEO_E5A_I_TIERED_CODE_PERIOD_S, [851](#)
 - GALILEO_E5A_OPT_ACQ_FS_SPS, [851](#)
 - GALILEO_E5A_Q_SECONDARY_CODE_LENGTH, [851](#)
 - GALILEO_E5A_Q_TIERED_CODE_PERIOD_S, [852](#)
 - GALILEO_E5A_SYMBOL_RATE_BPS, [852](#)
- Galileo_E5b.h, [857](#)
 - GALILEO_E5B_CODE_CHIP_RATE_CPS, [859](#)
 - GALILEO_E5B_CODE_LENGTH_CHIPS, [859](#)
 - GALILEO_E5B_CODE_PERIOD_MS, [859](#)
 - GALILEO_E5B_CODE_PERIOD_S, [859](#)
 - GALILEO_E5B_FREQ_HZ, [859](#)
 - GALILEO_E5B_I_SECONDARY_CODE_LENGTH, [859](#)
 - GALILEO_E5B_I_TIERED_CODE_PERIOD_S, [860](#)
 - GALILEO_E5B_OPT_ACQ_FS_SPS, [860](#)
 - GALILEO_E5B_Q_SECONDARY_CODE_LENGTH, [860](#)
- GALILEO_E5B_Q_TIERED_CODE_PERIOD_S, [860](#)
- GALILEO_E5B_SYMBOL_RATE_BPS, [860](#)
- Galileo_Ephemeris, [209](#)
 - A_1, [213](#)
 - af0_4, [213](#)
 - af1_4, [213](#)
 - af2_4, [213](#)
 - BGD_E1E5a_5, [213](#)
 - BGD_E1E5b_5, [214](#)
 - C_ic_4, [214](#)
 - C_is_4, [214](#)
 - C_rc_3, [214](#)
 - C_rs_3, [215](#)
 - C_uc_3, [215](#)
 - C_us_3, [215](#)
 - d_satpos_X, [215](#)
 - d_satpos_Y, [216](#)
 - d_satpos_Z, [216](#)
 - d_satvel_X, [216](#)
 - d_satvel_Y, [216](#)
 - d_satvel_Z, [216](#)
 - delta_n_3, [217](#)
 - E1B_DVS_5, [217](#)
 - E1B_HS_5, [217](#)
 - E5a_DVS, [217](#)
 - E5a_HS, [218](#)
 - E5b_DVS_5, [218](#)
 - E5b_HS_5, [218](#)
 - e_1, [218](#)
 - Galileo_System_Time, [211](#)
 - Galileo_dtr, [219](#)
 - i_0_2, [219](#)
 - i_satellite_PRN, [219](#)
 - iDot_2, [219](#)
 - M0_1, [220](#)
 - OMEGA_0_2, [220](#)
 - OMEGA_dot_3, [220](#)
 - omega_2, [220](#)
 - satellitePosition, [212](#)
 - serialize, [212](#)
 - sv_clock_drift, [212](#)
 - sv_clock_relativistic_term, [212](#)
 - t0c_4, [221](#)
 - t0e_1, [221](#)
 - TOW_5, [221](#)
 - WN_5, [221](#)
- Galileo_FNAV.h, [864](#)
- Galileo_Fnav_Message, [222](#)
- Galileo_INAV.h, [868](#)
 - GALILEO_INAV_PAGE_PART_SYMBOLS, [872](#)
 - GALILEO_INAV_PAGE_PART_WITH_PREABLE, [873](#)
 - GALILEO_INAV_PAGE_SYMBOLS, [873](#)
- Galileo_Inav_Message, [223](#)
- Galileo_Iono, [224](#)
 - ai0_5, [225](#)
 - ai1_5, [226](#)

- ai2_5, [226](#)
- Galileo_Iono, [225](#)
- Region1_flag_5, [226](#)
- Region2_flag_5, [226](#)
- Region3_flag_5, [227](#)
- Region4_flag_5, [227](#)
- Region5_flag_5, [227](#)
- serialize, [225](#)
- TOW_5, [227](#)
- WN_5, [228](#)
- Galileo_System_Time
 - Galileo_Ephemeris, [211](#)
- Galileo_Utc_Model, [236](#)
 - GST_to_UTC_time, [237](#)
 - Galileo_Utc_Model, [236](#)
 - serialize, [237](#)
 - t0t_6, [237](#)
 - WNot_6, [237](#)
- galileo_almanac.h, [831](#)
- galileo_almanac_helper.h, [832](#)
- Galileo_dtr
 - Galileo_Ephemeris, [219](#)
- galileo_e1_code_gen_complex_sampled
 - galileo_e1_signal_processing.h, [842](#), [843](#)
- galileo_e1_code_gen_float_sampled
 - galileo_e1_signal_processing.h, [843](#)
- galileo_e1_code_gen_sinboc11_float
 - galileo_e1_signal_processing.h, [843](#)
- galileo_e1_dll_pll_veml_tracking.h, [836](#)
- galileo_e1_dll_pll_veml_tracking_fpga.h, [837](#)
- galileo_e1_pcps_8ms_ambiguous_acquisition.h, [838](#)
- galileo_e1_pcps_ambiguous_acquisition.h, [838](#)
- galileo_e1_pcps_ambiguous_acquisition_fpga.h, [839](#)
- galileo_e1_pcps_cccwsr_ambiguous_acquisition.h, [840](#)
- galileo_e1_pcps_quicksync_ambiguous_acquisition.h, [840](#)
- galileo_e1_pcps_tong_ambiguous_acquisition.h, [841](#)
- galileo_e1_signal_processing.h, [842](#)
 - galileo_e1_code_gen_complex_sampled, [842](#), [843](#)
 - galileo_e1_code_gen_float_sampled, [843](#)
 - galileo_e1_code_gen_sinboc11_float, [843](#)
- galileo_e1_tcp_connector_tracking.h, [844](#)
- galileo_e1_tcp_connector_tracking_cc.h, [845](#)
- galileo_e1b_telemetry_decoder.h, [846](#)
- galileo_e5_a_code_gen_complex_primary
 - galileo_e5_signal_processing.h, [847](#)
- galileo_e5_a_code_gen_complex_sampled
 - galileo_e5_signal_processing.h, [847](#)
- galileo_e5_b_code_gen_complex_primary
 - galileo_e5_signal_processing.h, [848](#)
- galileo_e5_b_code_gen_complex_sampled
 - galileo_e5_signal_processing.h, [848](#)
- galileo_e5_signal_processing.h, [846](#)
 - galileo_e5_a_code_gen_complex_primary, [847](#)
 - galileo_e5_a_code_gen_complex_sampled, [847](#)
 - galileo_e5_b_code_gen_complex_primary, [848](#)
 - galileo_e5_b_code_gen_complex_sampled, [848](#)
- galileo_e5a_dll_pll_tracking.h, [852](#)
- galileo_e5a_dll_pll_tracking_fpga.h, [853](#)
- galileo_e5a_noncoherent_iq_acquisition_caf.h, [854](#)
- galileo_e5a_noncoherent_iq_acquisition_caf_cc.h, [854](#)
- galileo_e5a_noncoherentIQ_acquisition_caf_cc, [204](#)
 - ~galileo_e5a_noncoherentIQ_acquisition_caf_cc, [205](#)
- general_work, [206](#)
- init, [206](#)
- mag, [206](#)
- set_active, [206](#)
- set_channel, [207](#)
- set_channel_fsm, [207](#)
- set_doppler_max, [207](#)
- set_doppler_step, [207](#)
- set_gnss_synchro, [208](#)
- set_local_code, [208](#)
- set_state, [208](#)
- set_threshold, [209](#)
- galileo_e5a_pcps_acquisition.h, [855](#)
- galileo_e5a_pcps_acquisition_fpga.h, [856](#)
- galileo_e5a_telemetry_decoder.h, [857](#)
- galileo_e5b_dll_pll_tracking.h, [861](#)
- galileo_e5b_pcps_acquisition.h, [861](#)
- galileo_e5b_pcps_acquisition_fpga.h, [862](#)
- galileo_e5b_telemetry_decoder.h, [863](#)
- galileo_ephemeris.h, [864](#)
- galileo_ephemeris_map
 - Rtklib_Solver, [696](#)
- galileo_fnav_message.h, [868](#)
- galileo_inav_message.h, [873](#)
- galileo_iono.h, [874](#)
- galileo_pcps_8ms_acquisition_cc, [228](#)
 - ~galileo_pcps_8ms_acquisition_cc, [229](#)
- general_work, [230](#)
- init, [230](#)
- mag, [230](#)
- set_active, [230](#)
- set_channel, [231](#)
- set_channel_fsm, [231](#)
- set_doppler_max, [231](#)
- set_doppler_step, [231](#)
- set_gnss_synchro, [232](#)
- set_local_code, [232](#)
- set_state, [232](#)
- set_threshold, [234](#)
- galileo_pcps_8ms_acquisition_cc.h, [875](#)
- galileo_telemetry_decoder_gs, [234](#)
 - general_work, [235](#)
 - set_channel, [235](#)
 - set_satellite, [235](#)
- galileo_telemetry_decoder_gs.h, [876](#)
- galileo_utc_model.h, [877](#)
- GalileoE1BTelemetryDecoder, [238](#)
 - implementation, [239](#)
- GalileoE1DIIPIIVemlTracking, [239](#)
 - implementation, [240](#)
 - set_channel, [240](#)
 - set_gnss_synchro, [240](#)

- stop_tracking, 240
- GalileoE1DIIPIIVemITrackingFpga, 241
 - ~GalileoE1DIIPIIVemITrackingFpga, 242
 - connect, 242
 - disconnect, 243
 - GalileoE1DIIPIIVemITrackingFpga, 242
 - get_left_block, 243
 - get_right_block, 243
 - implementation, 243
 - item_size, 243
 - role, 244
 - set_channel, 244
 - set_gnss_synchro, 244
 - start_tracking, 244
 - stop_tracking, 245
- GalileoE1Pcps8msAmbiguousAcquisition, 245
 - implementation, 246
 - init, 247
 - mag, 247
 - reset, 247
 - set_channel, 247
 - set_channel_fsm, 247
 - set_doppler_max, 248
 - set_doppler_step, 248
 - set_gnss_synchro, 248
 - set_local_code, 248
 - set_threshold, 249
 - stop_acquisition, 249
- GalileoE1PcpsAmbiguousAcquisition, 249
 - implementation, 250
 - init, 251
 - mag, 251
 - reset, 251
 - set_channel, 251
 - set_channel_fsm, 251
 - set_doppler_center, 252
 - set_doppler_max, 252
 - set_doppler_step, 252
 - set_gnss_synchro, 252
 - set_local_code, 252
 - set_resampler_latency, 253
 - set_state, 253
 - set_threshold, 253
 - stop_acquisition, 253
- GalileoE1PcpsAmbiguousAcquisitionFpga, 254
 - ~GalileoE1PcpsAmbiguousAcquisitionFpga, 255
 - connect, 255
 - disconnect, 256
 - GalileoE1PcpsAmbiguousAcquisitionFpga, 255
 - get_left_block, 256
 - get_right_block, 256
 - implementation, 256
 - init, 256
 - item_size, 257
 - mag, 257
 - reset, 257
 - role, 257
 - set_channel, 257
 - set_channel_fsm, 258
 - set_doppler_center, 258
 - set_doppler_max, 258
 - set_doppler_step, 258
 - set_gnss_synchro, 258
 - set_local_code, 259
 - set_resampler_latency, 259
 - set_state, 259
 - set_threshold, 259
 - stop_acquisition, 259
- GalileoE1PcpsCccwsrAmbiguousAcquisition, 260
 - implementation, 261
 - init, 261
 - mag, 261
 - reset, 262
 - set_channel, 262
 - set_channel_fsm, 262
 - set_doppler_max, 262
 - set_doppler_step, 263
 - set_gnss_synchro, 263
 - set_state, 263
 - set_threshold, 263
 - stop_acquisition, 263
- GalileoE1PcpsQuickSyncAmbiguousAcquisition, 264
 - implementation, 265
 - init, 265
 - mag, 265
 - reset, 266
 - set_channel, 266
 - set_channel_fsm, 266
 - set_doppler_max, 266
 - set_doppler_step, 267
 - set_gnss_synchro, 267
 - set_local_code, 267
 - set_state, 267
 - set_threshold, 267
 - stop_acquisition, 268
- GalileoE1PcpsTongAmbiguousAcquisition, 268
 - implementation, 269
 - init, 270
 - mag, 270
 - reset, 270
 - set_channel, 270
 - set_channel_fsm, 270
 - set_doppler_max, 271
 - set_doppler_step, 271
 - set_gnss_synchro, 271
 - set_local_code, 271
 - set_state, 272
 - set_threshold, 272
 - stop_acquisition, 272
- GalileoE1TcpConnectorTracking, 273
 - implementation, 273
 - set_channel, 274
 - set_gnss_synchro, 274
 - stop_tracking, 274
- GalileoE5aDIIPIITracking, 275
 - implementation, 275

- set_channel, 276
 - set_gnss_synchro, 276
 - stop_tracking, 276
- GalileoE5aDIIPIITrackingFpga, 277
 - ~GalileoE5aDIIPIITrackingFpga, 278
 - connect, 278
 - disconnect, 278
 - GalileoE5aDIIPIITrackingFpga, 278
 - get_left_block, 278
 - get_right_block, 279
 - implementation, 279
 - item_size, 279
 - role, 279
 - set_channel, 279
 - set_gnss_synchro, 280
 - start_tracking, 280
 - stop_tracking, 280
- GalileoE5aNoncoherentIQAcquisitionCaf, 281
 - implementation, 282
 - init, 282
 - mag, 282
 - reset, 282
 - set_channel, 282
 - set_channel_fsm, 283
 - set_doppler_max, 283
 - set_doppler_step, 283
 - set_gnss_synchro, 283
 - set_local_code, 284
 - set_state, 284
 - set_threshold, 284
 - stop_acquisition, 284
- GalileoE5aPcpsAcquisition, 285
 - init, 286
 - mag, 286
 - reset, 286
 - set_channel, 287
 - set_channel_fsm, 287
 - set_doppler_center, 287
 - set_doppler_max, 287
 - set_doppler_step, 287
 - set_gnss_synchro, 288
 - set_local_code, 288
 - set_resampler_latency, 288
 - set_state, 288
 - set_threshold, 289
 - stop_acquisition, 289
- GalileoE5aPcpsAcquisitionFpga, 289
 - ~GalileoE5aPcpsAcquisitionFpga, 291
 - connect, 291
 - disconnect, 291
 - GalileoE5aPcpsAcquisitionFpga, 291
 - get_left_block, 292
 - get_right_block, 292
 - implementation, 292
 - init, 292
 - item_size, 292
 - mag, 293
 - reset, 293
 - role, 293
 - set_channel, 293
 - set_channel_fsm, 293
 - set_doppler_center, 294
 - set_doppler_max, 294
 - set_doppler_step, 294
 - set_gnss_synchro, 294
 - set_local_code, 294
 - set_resampler_latency, 295
 - set_single_doppler_flag, 295
 - set_state, 295
 - set_threshold, 295
 - stop_acquisition, 296
- GalileoE5aTelemetryDecoder, 296
 - implementation, 297
- GalileoE5bDIIPIITracking, 297
 - connect, 298
 - disconnect, 298
 - get_left_block, 299
 - get_right_block, 299
 - implementation, 299
 - set_channel, 299
 - set_gnss_synchro, 299
 - stop_tracking, 300
- GalileoE5bPcpsAcquisition, 300
 - ~GalileoE5bPcpsAcquisition, 302
 - connect, 302
 - disconnect, 302
 - GalileoE5bPcpsAcquisition, 301
 - get_left_block, 302
 - get_right_block, 302
 - implementation, 303
 - init, 303
 - item_size, 303
 - mag, 303
 - reset, 303
 - role, 304
 - set_channel, 304
 - set_channel_fsm, 304
 - set_doppler_center, 304
 - set_doppler_max, 305
 - set_doppler_step, 305
 - set_gnss_synchro, 305
 - set_local_code, 305
 - set_resampler_latency, 305
 - set_state, 306
 - set_threshold, 306
 - stop_acquisition, 306
- GalileoE5bPcpsAcquisitionFpga, 307
 - ~GalileoE5bPcpsAcquisitionFpga, 308
 - connect, 309
 - disconnect, 309
 - GalileoE5bPcpsAcquisitionFpga, 308
 - get_left_block, 309
 - get_right_block, 309
 - implementation, 309
 - init, 310
 - item_size, 310

- mag, [310](#)
- reset, [310](#)
- role, [310](#)
- set_channel, [311](#)
- set_channel_fsm, [311](#)
- set_doppler_center, [311](#)
- set_doppler_max, [311](#)
- set_doppler_step, [312](#)
- set_gnss_synchro, [312](#)
- set_local_code, [312](#)
- set_resampler_latency, [312](#)
- set_single_doppler_flag, [312](#)
- set_state, [313](#)
- set_threshold, [313](#)
- stop_acquisition, [313](#)
- GalileoE5bTelemetryDecoder, [314](#)
 - connect, [314](#)
 - disconnect, [315](#)
 - get_left_block, [315](#)
 - get_right_block, [315](#)
 - implementation, [315](#)
- Gamma
 - convolutional.h, [806](#)
- gen_signal_source.h, [878](#)
- GenSignalSource, [316](#)
 - ~GenSignalSource, [316](#)
 - GenSignalSource, [316](#)
 - implementation, [317](#)
- general_work
 - beidou_b1i_telemetry_decoder_gs, [85](#)
 - beidou_b3i_telemetry_decoder_gs, [87](#)
 - dll_pll_veml_tracking_fpga, [171](#)
 - galileo_e5a_noncoherentIQ_acquisition_caf_cc, [206](#)
 - galileo_pcps_8ms_acquisition_cc, [230](#)
 - galileo_telemetry_decoder_gs, [235](#)
 - glonass_l1_ca_telemetry_decoder_gs, [350](#)
 - glonass_l2_ca_telemetry_decoder_gs, [354](#)
 - gps_l1_ca_telemetry_decoder_gs, [469](#)
 - gps_l2c_telemetry_decoder_gs, [470](#)
 - pcps_acquisition, [592](#)
 - pcps_acquisition_fine_doppler_cc, [597](#)
 - pcps_assisted_acquisition_cc, [608](#)
 - pcps_cccwsr_acquisition_cc, [613](#)
 - pcps_openc1_acquisition_cc, [618](#)
 - pcps_quicksync_acquisition_cc, [624](#)
 - pcps_tong_acquisition_cc, [628](#)
 - sbas_l1_telemetry_decoder_gs, [706](#)
- Geo_to_ECEF
 - geofunctions.h, [881](#)
- GeoJSON_Printer, [317](#)
- geofunctions.h, [878](#)
 - cart2geo, [880](#)
 - cart2utm, [880](#)
 - clk2in, [880](#)
 - clk2in, [881](#)
 - findUtmZone, [881](#)
 - Geo_to_ECEF, [881](#)
 - Gravity_ECEF, [881](#)
 - great_circle_distance, [882](#)
 - pv_Geo_to_ECEF, [882](#)
 - Skew_symmetric, [882](#)
 - togeo, [882](#)
 - topocent, [883](#)
- geojson_printer.h, [883](#)
- geph_t, [318](#)
- Get
 - INIReader, [563](#)
- get
 - Gnss_circular_deque, [377](#)
- get_GPS_week
 - Gps_Navigation_Message, [474](#)
- get_LLH
 - TcpCmdInterface, [734](#)
- get_PRN
 - Gnss_Satellite, [381](#)
- get_TOW
 - Gps_Navigation_Message, [475](#)
- get_almanac
 - Glonass_Gnav_Navigation_Message, [341](#)
- get_avg_height
 - Pvt_Solution, [641](#)
- get_avg_latitude
 - Pvt_Solution, [641](#)
- get_avg_longitude
 - Pvt_Solution, [641](#)
- get_beidou_dnav_almanac_map
 - rtklib_pvt_gs, [693](#)
- get_beidou_dnav_ephemeris_map
 - rtklib_pvt_gs, [693](#)
- get_block
 - Gnss_Satellite, [381](#)
- get_clock_drift_ppm
 - Pvt_Solution, [641](#)
- get_code_nco
 - Tracking_2nd_DLL_filter, [738](#)
- get_course_over_ground
 - Pvt_Solution, [641](#)
- get_current_status_map
 - channel_status_msg_receiver, [139](#)
- get_current_status_pvt
 - channel_status_msg_receiver, [139](#)
- get_ephemeris
 - Beidou_Dnav_Navigation_Message, [112](#)
 - FrontEndCal, [197](#)
 - Glonass_Gnav_Navigation_Message, [341](#)
 - Gps_CNAV_Navigation_Message, [442](#)
 - Gps_Navigation_Message, [474](#)
- get_flag_iono_valid
 - Gps_Navigation_Message, [474](#)
- get_flag_utc_model_valid
 - Gps_Navigation_Message, [474](#)
- get_frame_number
 - Glonass_Gnav_Navigation_Message, [341](#)
- get_galileo_almanac_map
 - rtklib_pvt_gs, [693](#)

- get_galileo_ephemeris_map
 - rtklib_pvt_gs, [694](#)
- get_gps_almanac_map
 - rtklib_pvt_gs, [694](#)
- get_gps_ephemeris_map
 - rtklib_pvt_gs, [694](#)
- get_height
 - Pvt_Solution, [641](#)
- get_iono
 - Beidou_Dnav_Navigation_Message, [112](#)
 - Gps_CNAV_Navigation_Message, [443](#)
 - Gps_Navigation_Message, [474](#)
- get_latest_PVT
 - rtklib_pvt_gs, [694](#)
- get_latitude
 - Pvt_Solution, [642](#)
- get_leap_second
 - Rinex_Printer, [654](#)
- get_left_block
 - Channel, [135](#)
 - GalileoE1DIIPIIVemlTrackingFpga, [243](#)
 - GalileoE1PcpsAmbiguousAcquisitionFpga, [256](#)
 - GalileoE5aDIIPIITrackingFpga, [278](#)
 - GalileoE5aPcpsAcquisitionFpga, [292](#)
 - GalileoE5bDIIPIITracking, [299](#)
 - GalileoE5bPcpsAcquisition, [302](#)
 - GalileoE5bPcpsAcquisitionFpga, [309](#)
 - GalileoE5bTelemetryDecoder, [315](#)
 - GpsL1CaDIIPIITrackingFpga, [483](#)
 - GpsL1CaPcpsAcquisitionFpga, [500](#)
 - GpsL5DIIPIITrackingFpga, [540](#)
 - GpsL5iPcpsAcquisitionFpga, [550](#)
- get_left_block_acq
 - Channel, [135](#)
- get_left_block_trk
 - Channel, [136](#)
- get_longitude
 - Pvt_Solution, [642](#)
- get_num_valid_observations
 - Pvt_Solution, [642](#)
- get_pvt
 - GNSSFlowgraph, [411](#)
- get_rf_link
 - Gnss_Satellite, [381](#)
- get_right_block
 - GalileoE1DIIPIIVemlTrackingFpga, [243](#)
 - GalileoE1PcpsAmbiguousAcquisitionFpga, [256](#)
 - GalileoE5aDIIPIITrackingFpga, [279](#)
 - GalileoE5aPcpsAcquisitionFpga, [292](#)
 - GalileoE5bDIIPIITracking, [299](#)
 - GalileoE5bPcpsAcquisition, [302](#)
 - GalileoE5bPcpsAcquisitionFpga, [309](#)
 - GalileoE5bTelemetryDecoder, [315](#)
 - GpsL1CaDIIPIITrackingFpga, [483](#)
 - GpsL1CaPcpsAcquisitionFpga, [500](#)
 - GpsL5DIIPIITrackingFpga, [541](#)
 - GpsL5iPcpsAcquisitionFpga, [550](#)
- get_satellite
 - Gnss_Signal, [394](#)
- get_satellite_PRN
 - Gps_Navigation_Message, [474](#)
- get_signal_str
 - Gnss_Signal, [394](#)
- get_speed_over_ground
 - Pvt_Solution, [642](#)
- get_system
 - Gnss_Satellite, [381](#)
- get_system_short
 - Gnss_Satellite, [382](#)
- get_time_offset_s
 - Pvt_Solution, [642](#)
- get_utc_model
 - Beidou_Dnav_Navigation_Message, [112](#)
 - Glonass_Gnav_Navigation_Message, [342](#)
 - Gps_CNAV_Navigation_Message, [443](#)
 - Gps_Navigation_Message, [475](#)
- get_utc_time
 - TcpCmdInterface, [734](#)
- GetBlock
 - GNSSBlockFactory, [407](#)
- GetInteger
 - INIReader, [563](#)
- Glonass_Gnav_Almanac, [318](#)
 - d_C_n, [320](#)
 - d_Delta_T_n_A_dot, [321](#)
 - d_Delta_T_n_A, [321](#)
 - d_Delta_i_n_A, [320](#)
 - d_H_n_A, [321](#)
 - d_KP, [322](#)
 - d_M_n_A, [322](#)
 - d_epsilon_n_A, [321](#)
 - d_I_n, [322](#)
 - d_lambda_n_A, [322](#)
 - d_n_A, [323](#)
 - d_omega_n_A, [323](#)
 - d_t_lambda_n_A, [323](#)
 - d_tau_n_A, [323](#)
 - Glonass_Gnav_Almanac, [320](#)
 - i_satellite_PRN, [324](#)
 - i_satellite_freq_channel, [324](#)
 - i_satellite_slot_number, [324](#)
 - serialize, [320](#)
- Glonass_Gnav_Ephemeris, [325](#)
 - compute_GLONASS_time, [327](#)
 - d_AXn, [330](#)
 - d_AYn, [330](#)
 - d_AZn, [331](#)
 - d_B_n, [331](#)
 - d_Delta_tau_n, [331](#)
 - d_E_n, [332](#)
 - d_F_T, [332](#)
 - d_N_T, [334](#)
 - d_P_1, [334](#)
 - d_P_2, [335](#)
 - d_P_3, [335](#)
 - d_P_4, [335](#)

- d_TOW, 337
- d_VXn, 337
- d_VYn, 337
- d_VZn, 337
- d_WN, 337
- d_Xn, 338
- d_Yn, 338
- d_Zn, 338
- d_dtr, 331
- d_gamma_n, 332
- d_iode, 332
- d_l3rd_n, 333
- d_l5th_n, 333
- d_M, 333
- d_m, 333
- d_n, 334
- d_P, 334
- d_satClkDrift, 335
- d_t_b, 336
- d_t_k, 336
- d_tau_c, 336
- d_tau_n, 336
- d_tod, 336
- d_yr, 338
- Glonass_Gnav_Ephemeris, 327
- glot_to_gpst, 328
- glot_to_utc, 328
- i_satellite_PRN, 339
- i_satellite_freq_channel, 338
- i_satellite_slot_number, 339
- serialize, 329
- sv_clock_drift, 330
- Glonass_Gnav_Navigation_Message, 339
 - CRC_test, 341
 - get_almanac, 341
 - get_ephemeris, 341
 - get_frame_number, 341
 - get_utc_model, 342
 - Glonass_Gnav_Navigation_Message, 340
 - have_new_almanac, 342
 - have_new_ephemeris, 342
 - have_new_utc_model, 342
 - string_decoder, 342
- Glonass_Gnav_Utc_Model, 343
 - d_B1, 345
 - d_B2, 345
 - d_N_4, 345
 - d_N_A, 345
 - d_tau_c, 346
 - d_tau_gps, 346
 - Glonass_Gnav_Utc_Model, 344
 - serialize, 344
 - utc_time, 344
- Glonass_L1_Ca_Dll_Pll_Tracking_cc, 348
- Glonass_L2_Ca_Dll_Pll_Tracking_cc, 352
- glonass_gnav_almanac
 - Rtklib_Solver, 696
- glonass_gnav_almanac.h, 884
- glonass_gnav_ephemeris.h, 885
- glonass_gnav_ephemeris_map
 - Rtklib_Solver, 697
- glonass_gnav_navigation_message.h, 885
- glonass_gnav_utc_model
 - Rtklib_Solver, 697
- glonass_gnav_utc_model.h, 886
- glonass_l1_ca_code_gen_complex
 - glonass_l1_signal_processing.h, 909
- glonass_l1_ca_code_gen_complex_sampled
 - glonass_l1_signal_processing.h, 909
- glonass_l1_ca_dll_pll_c_aid_tracking.h, 887
- glonass_l1_ca_dll_pll_c_aid_tracking_cc, 346
- glonass_l1_ca_dll_pll_c_aid_tracking_cc.h, 888
- glonass_l1_ca_dll_pll_c_aid_tracking_sc, 347
- glonass_l1_ca_dll_pll_c_aid_tracking_sc.h, 889
- glonass_l1_ca_dll_pll_tracking.h, 890
- glonass_l1_ca_dll_pll_tracking_cc.h, 890
- glonass_l1_ca_pcps_acquisition.h, 891
- glonass_l1_ca_telemetry_decoder.h, 892
- glonass_l1_ca_telemetry_decoder_gs, 349
 - ~glonass_l1_ca_telemetry_decoder_gs, 350
- general_work, 350
- set_channel, 350
- set_satellite, 350
- glonass_l1_ca_telemetry_decoder_gs.h, 893
- glonass_l1_signal_processing.h, 908
 - glonass_l1_ca_code_gen_complex, 909
 - glonass_l1_ca_code_gen_complex_sampled, 909
- glonass_l2_ca_code_gen_complex
 - glonass_l2_signal_processing.h, 917
- glonass_l2_ca_code_gen_complex_sampled
 - glonass_l2_signal_processing.h, 917
- glonass_l2_ca_dll_pll_c_aid_tracking.h, 909
- glonass_l2_ca_dll_pll_c_aid_tracking_cc, 351
- glonass_l2_ca_dll_pll_c_aid_tracking_cc.h, 910
- glonass_l2_ca_dll_pll_c_aid_tracking_sc, 352
- glonass_l2_ca_dll_pll_c_aid_tracking_sc.h, 911
- glonass_l2_ca_dll_pll_tracking.h, 912
- glonass_l2_ca_dll_pll_tracking_cc.h, 913
- glonass_l2_ca_pcps_acquisition.h, 914
- glonass_l2_ca_telemetry_decoder.h, 914
- glonass_l2_ca_telemetry_decoder_gs, 353
 - ~glonass_l2_ca_telemetry_decoder_gs, 354
- general_work, 354
- set_channel, 354
- set_satellite, 354
- glonass_l2_ca_telemetry_decoder_gs.h, 915
- glonass_l2_signal_processing.h, 916
 - glonass_l2_ca_code_gen_complex, 917
 - glonass_l2_ca_code_gen_complex_sampled, 917
- GlonassL1CaDllPllCAidTracking, 355
 - implementation, 356
 - set_channel, 356
 - set_gnss_synchro, 356
 - stop_tracking, 356
- GlonassL1CaDllPllTracking, 357
 - implementation, 358

- set_channel, 358
 - set_gnss_synchro, 358
 - stop_tracking, 358
- GlonassL1CaPcpsAcquisition, 359
 - implementation, 360
 - init, 360
 - mag, 360
 - reset, 361
 - set_channel, 361
 - set_channel_fsm, 361
 - set_doppler_max, 361
 - set_doppler_step, 362
 - set_gnss_synchro, 362
 - set_local_code, 362
 - set_state, 362
 - set_threshold, 362
 - stop_acquisition, 363
- GlonassL1CaTelemetryDecoder, 363
 - implementation, 364
- GlonassL2CaDIIPICAidTracking, 364
 - implementation, 365
 - set_channel, 365
 - set_gnss_synchro, 365
 - stop_tracking, 366
- GlonassL2CaDIIPITracking, 366
 - implementation, 367
 - set_channel, 367
 - set_gnss_synchro, 367
 - stop_tracking, 367
- GlonassL2CaPcpsAcquisition, 368
 - implementation, 369
 - init, 369
 - mag, 369
 - reset, 370
 - set_channel, 370
 - set_channel_fsm, 370
 - set_doppler_max, 370
 - set_doppler_step, 371
 - set_gnss_synchro, 371
 - set_local_code, 371
 - set_state, 371
 - set_threshold, 371
 - stop_acquisition, 372
- GlonassL2CaTelemetryDecoder, 372
 - implementation, 373
- glot_to_gpst
 - Glonass_Gnav_Ephemeris, 328
- glot_to_utc
 - Glonass_Gnav_Ephemeris, 328
- gn3s_signal_source.h, 917
- Gn3sSignalSource, 373
 - implementation, 374
- GnMaxSignalSource, 374
 - implementation, 375
- gnmax_signal_source.h, 918
- Gnss_Satellite, 379
 - ~Gnss_Satellite, 380
 - get_PRN, 381
 - get_block, 381
 - get_rf_link, 381
 - get_system, 381
 - get_system_short, 382
 - Gnss_Satellite, 380, 381
 - operator<<, 383
 - operator=, 382
 - operator==, 383
 - update_PRN, 382
 - what_block, 382
- Gnss_Sdr_Supl_Client, 384
 - load_cnav_ephemeris_xml, 386
 - load_cnav_utc_xml, 387
 - load_ephemeris_xml, 387
 - load_gal_almanac_xml, 387
 - load_gal_ephemeris_xml, 387
 - load_gal_iono_xml, 387
 - load_gal_utc_xml, 387
 - load_glo_utc_xml, 388
 - load_gnav_ephemeris_xml, 388
 - load_gps_almanac_xml, 388
 - load_iono_xml, 388
 - load_ref_location_xml, 388
 - load_ref_time_xml, 388
 - load_utc_xml, 389
 - save_cnav_ephemeris_map_xml, 389
 - save_cnav_utc_xml, 389
 - save_ephemeris_map_xml, 389
 - save_gal_almanac_xml, 389
 - save_gal_ephemeris_map_xml, 390
 - save_gal_iono_xml, 390
 - save_gal_utc_xml, 390
 - save_glo_utc_xml, 390
 - save_gnav_ephemeris_map_xml, 390
 - save_gps_almanac_xml, 391
 - save_iono_xml, 391
 - save_ref_location_xml, 391
 - save_ref_time_xml, 391
 - save_utc_xml, 391
- Gnss_Sdr_Valve, 393
- Gnss_Signal, 393
 - get_satellite, 394
 - get_signal_str, 394
 - operator<<, 394
 - operator==, 395
- Gnss_Synchro, 395
 - ~Gnss_Synchro, 397
 - Acq_delay_samples, 399
 - Acq_doppler_hz, 399
 - Acq_doppler_step, 399
 - Acq_samplestamp_samples, 399
 - CN0_dB_hz, 400
 - Carrier_Doppler_hz, 399
 - Carrier_phase_rads, 400
 - Channel_ID, 400
 - Code_phase_samples, 400
 - correlation_length_ms, 401
 - Flag_valid_acquisition, 401

- Flag_valid_pseudorange, [401](#)
- Flag_valid_symbol_output, [401](#)
- Flag_valid_word, [402](#)
- fs, [402](#)
- Gnss_Synchro, [397](#)
- interp_TOW_ms, [402](#)
- operator=, [398](#)
- PRN, [402](#)
- Prompt_I, [403](#)
- Prompt_Q, [403](#)
- Pseudorange_m, [403](#)
- RX_time, [403](#)
- serialize, [398](#)
- Signal, [404](#)
- System, [404](#)
- TOW_at_current_symbol_ms, [404](#)
- Tracking_sample_counter, [404](#)
- Gnss_Synchro_Udp_Sink, [406](#)
- gnss_block_factory.h, [919](#)
- gnss_block_interface.h, [919](#)
- Gnss_circular_deque
 - at, [376](#)
 - back, [376](#)
 - clear, [377](#)
 - front, [377](#)
 - get, [377](#)
 - Gnss_circular_deque, [376](#)
 - pop_front, [377](#)
 - push_back, [378](#)
 - reset, [378](#)
 - size, [378](#)
- Gnss_circular_deque< T >, [375](#)
- gnss_circular_deque.h, [920](#)
- gnss_flowgraph.h, [921](#)
- gnss_frequencies.h, [921](#)
 - DFRQ1_GLO, [923](#)
 - DFRQ2_GLO, [923](#)
 - FREQ1, [923](#)
 - FREQ1_BDS, [923](#)
 - FREQ1_GLO, [923](#)
 - FREQ2, [923](#)
 - FREQ2_BDS, [924](#)
 - FREQ2_GLO, [924](#)
 - FREQ3_BDS, [924](#)
 - FREQ3_GLO, [924](#)
 - FREQ5, [924](#)
 - FREQ6, [925](#)
 - FREQ7, [925](#)
 - FREQ8, [925](#)
 - FREQ9, [925](#)
- gnss_obs_codes.h, [926](#)
 - CODE_L1A, [928](#)
 - CODE_L1B, [928](#)
 - CODE_L1C, [929](#)
 - CODE_L1E, [929](#)
 - CODE_L1I, [929](#)
 - CODE_L1L, [929](#)
 - CODE_L1M, [929](#)
 - CODE_L1N, [930](#)
 - CODE_L1P, [930](#)
 - CODE_L1Q, [930](#)
 - CODE_L1S, [930](#)
 - CODE_L1W, [930](#)
 - CODE_L1X, [931](#)
 - CODE_L1Y, [931](#)
 - CODE_L1Z, [931](#)
 - CODE_L2C, [931](#)
 - CODE_L2D, [931](#)
 - CODE_L2I, [932](#)
 - CODE_L2L, [932](#)
 - CODE_L2M, [932](#)
 - CODE_L2N, [932](#)
 - CODE_L2P, [932](#)
 - CODE_L2Q, [933](#)
 - CODE_L2S, [933](#)
 - CODE_L2W, [933](#)
 - CODE_L2X, [933](#)
 - CODE_L2Y, [933](#)
 - CODE_L3I, [934](#)
 - CODE_L3Q, [934](#)
 - CODE_L3X, [934](#)
 - CODE_L5A, [934](#)
 - CODE_L5B, [934](#)
 - CODE_L5C, [935](#)
 - CODE_L5I, [935](#)
 - CODE_L5Q, [935](#)
 - CODE_L5X, [935](#)
 - CODE_L6A, [935](#)
 - CODE_L6B, [936](#)
 - CODE_L6C, [936](#)
 - CODE_L6I, [936](#)
 - CODE_L6L, [936](#)
 - CODE_L6Q, [936](#)
 - CODE_L6S, [937](#)
 - CODE_L6X, [937](#)
 - CODE_L6Z, [937](#)
 - CODE_L7I, [937](#)
 - CODE_L7Q, [937](#)
 - CODE_L7X, [938](#)
 - CODE_L8I, [938](#)
 - CODE_L8Q, [938](#)
 - CODE_L8X, [938](#)
 - CODE_L9A, [938](#)
 - CODE_L9B, [939](#)
 - CODE_L9C, [939](#)
 - CODE_L9X, [939](#)
 - CODE_NONE, [939](#)
 - MAXCODE, [939](#)
- gnss_satellite.h, [940](#)
- gnss_sdr_create_directory.h, [940](#)
- gnss_sdr_flags.h, [941](#)
 - DECLARE_bool, [942](#)
 - DECLARE_double, [942](#), [943](#)
 - DECLARE_int32, [943](#), [944](#)
 - DECLARE_string, [944](#), [945](#)
- gnss_sdr_fpga_sample_counter, [383](#)

- gnss_sdr_fpga_sample_counter.h, 945
- gnss_sdr_make_unique.h, 946
- gnss_sdr_sample_counter, 384
- gnss_sdr_sample_counter.h, 947
- gnss_sdr_supl_client.h, 948
- gnss_sdr_time_counter, 392
- gnss_sdr_time_counter.h, 949
- gnss_sdr_valve.h, 949
- gnss_signal.h, 950
- gnss_signal_processing.h, 951
 - complex_exp_gen, 952
 - complex_exp_gen_conj, 952
 - hex_to_binary_converter, 952
 - resampler, 952
- gnss_synchro.h, 953
- gnss_synchro_monitor, 405
 - ~gnss_synchro_monitor, 406
- gnss_synchro_monitor.h, 953
- gnss_synchro_udp_sink.h, 954
- Gnuplot, 413
 - Gnuplot, 414
 - operator<<, 415
 - replot, 415
 - unset_title, 415
- gnuplot_i.h, 955
- GnuplotException, 415
- Gps_Acq_Assist, 416
 - Azimuth, 417
 - Code_Phase, 417
 - Code_Phase_int, 417
 - Code_Phase_window, 417
 - d_Doppler0, 418
 - d_Doppler1, 418
 - d_TOW, 418
 - dopplerUncertainty, 418
 - Elevation, 418
 - GPS_Bit_Number, 419
 - Gps_Acq_Assist, 417
 - i_satellite_PRN, 419
- Gps_Almanac, 419
 - d_A_f0, 421
 - d_A_f1, 421
 - d_Delta_i, 421
 - d_M_0, 421
 - d_OMEGA0, 422
 - d_OMEGA_DOT, 422
 - d_OMEGA, 421
 - d_e_eccentricity, 421
 - d_sqrt_A, 422
 - Gps_Almanac, 420
 - i_AS_status, 422
 - i_SV_health, 423
 - i_Toa, 423
 - i_WNa, 423
 - i_satellite_PRN, 422
- Gps_CNAV_Ephemeris, 423
 - b_alert_flag, 428
 - b_antispoofing_flag, 428
 - b_integrity_status_flag, 429
 - d_A_DOT, 429
 - d_A_f0, 429
 - d_A_f1, 429
 - d_A_f2, 430
 - d_Cic, 430
 - d_Cis, 430
 - d_Crc, 430
 - d_Crs, 431
 - d_Cuc, 431
 - d_Cus, 431
 - d_DELTA_DOT_N, 432
 - d_DELTA_OMEGA_DOT, 432
 - d_DELTA_A, 431
 - d_Delta_n, 432
 - d_IDOT, 433
 - d_M_0, 433
 - d_OMEGA0, 434
 - d_OMEGA, 433
 - d_TGD, 435
 - d_TOW, 436
 - d_Toc, 435
 - d_Toe1, 436
 - d_Toe2, 436
 - d_Top, 436
 - d_URA0, 437
 - d_URA1, 437
 - d_URA2, 437
 - d_dtr, 432
 - d_e_eccentricity, 432
 - d_i_0, 433
 - d_satClkDrift, 434
 - d_satpos_X, 434
 - d_satpos_Y, 434
 - d_satpos_Z, 434
 - d_satvel_X, 435
 - d_satvel_Y, 435
 - d_satvel_Z, 435
 - Gps_CNAV_Ephemeris, 426
 - i_GPS_week, 437
 - i_URA, 438
 - i_signal_health, 437
 - satellitePosition, 426
 - serialize, 426
 - sv_clock_drift, 428
 - sv_clock_relativistic_term, 428
- Gps_CNAV_Iono, 438
 - d_alpha0, 439
 - d_alpha1, 439
 - d_alpha2, 440
 - d_alpha3, 440
 - d_beta0, 440
 - d_beta1, 440
 - d_beta2, 441
 - d_beta3, 441
 - Gps_CNAV_Iono, 439
 - serialize, 439
 - valid, 441

- Gps_CNAV_Navigation_Message, 442
 - get_ephemeris, 442
 - get_iono, 443
 - get_utc_model, 443
- Gps_CNAV_Navigation_Message, 442
 - have_new_ephemeris, 443
 - have_new_iono, 443
 - have_new_utc_model, 443
- Gps_CNAV_Utc_Model, 444
 - d_A0, 445
 - d_A1, 445
 - d_A2, 445
 - d_DeltaT_LSF, 445
 - d_DeltaT_LS, 445
 - d_t_OT, 446
- Gps_CNAV_Utc_Model, 444
 - i_DN, 446
 - i_WN_LSF, 446
 - i_WN_T, 446
- Gps_Ephemeris, 447
 - b_L2_P_data_flag, 452
 - b_alert_flag, 452
 - b_antispoofing_flag, 452
 - b_fit_interval_flag, 452
 - b_integrity_status_flag, 452
 - d_A_f0, 453
 - d_A_f1, 453
 - d_A_f2, 453
 - d_Cic, 453
 - d_Cis, 454
 - d_Crc, 454
 - d_Crs, 454
 - d_Cuc, 454
 - d_Cus, 455
 - d_Delta_n, 455
 - d_IDOT, 456
 - d_IODE_SF2, 456
 - d_IODE_SF3, 457
 - d_IODC, 456
 - d_M_0, 457
 - d_OMEGA0, 457
 - d_OMEGA_DOT, 458
 - d_OMEGA, 457
 - d_TGD, 459
 - d_TOW, 460
 - d_Toc, 460
 - d_Toe, 460
 - d_dtr, 455
 - d_e_eccentricity, 455
 - d_i_0, 456
 - d_satClkDrift, 458
 - d_satpos_X, 458
 - d_satpos_Y, 458
 - d_satpos_Z, 458
 - d_satvel_X, 459
 - d_satvel_Y, 459
 - d_satvel_Z, 459
 - d_sqrt_A, 459
- Gps_Ephemeris, 449
 - i_AODO, 460
 - i_GPS_week, 461
 - i_SV_accuracy, 461
 - i_code_on_L2, 461
 - satelliteBlock, 461
 - satellitePosition, 449
 - serialize, 450
 - sv_clock_drift, 451
 - sv_clock_relativistic_term, 451
- Gps_Iono, 462
 - d_alpha0, 463
 - d_alpha1, 463
 - d_alpha2, 464
 - d_alpha3, 464
 - d_beta0, 464
 - d_beta1, 464
 - d_beta2, 465
 - d_beta3, 465
- Gps_Iono, 463
 - serialize, 463
 - valid, 465
- Gps_L1_Ca_Dll_Pll_Tracking_GPU_cc, 466
- Gps_L1_Ca_Kf_Tracking_cc, 466
- Gps_L1_Ca_Tcp_Connector_Tracking_cc, 467
- Gps_Navigation_Message, 472
 - get_GPS_week, 474
 - get_TOW, 475
 - get_ephemeris, 474
 - get_flag_iono_valid, 474
 - get_flag_utc_model_valid, 474
 - get_iono, 474
 - get_satellite_PRN, 474
 - get_utc_model, 475
- Gps_Navigation_Message, 473
 - set_channel, 475
 - set_satellite_PRN, 475
 - subframe_decoder, 475
 - utc_time, 476
- Gps_Utc_Model, 476
 - d_A0, 477
 - d_A1, 477
 - d_A2, 477
 - d_DeltaT_LSF, 478
 - d_DeltaT_LS, 477
 - d_t_OT, 478
- Gps_Utc_Model, 477
 - i_DN, 478
 - i_WN_LSF, 478
 - i_WN_T, 478
- gps_acq_assist.h, 956
- gps_almanac.h, 957
- Gps_cnav_decoder, 69
 - GPS_L2_V27_HISTORY_LENGTH_BITS, 69
 - GPS_L2C_V27_DECODE_BITS, 69
 - GPS_L2C_V27_DELAY_BITS, 69
 - GPS_L2C_V27_INIT_BITS, 70
- gps_cnav_ephemeris.h, 960

- gps_cnav_ephemeris_map
 - Rtklib_Solver, 697
- gps_cnav_iono.h, 961
- gps_cnav_navigation_message.h, 961
- gps_cnav_utc_model.h, 962
- gps_ephemeris.h, 963
- gps_ephemeris_map
 - Rtklib_Solver, 697
- gps_iono.h, 963
- gps_l1_ca_code_gen_complex
 - gps_sdr_signal_processing.h, 1005
- gps_l1_ca_code_gen_complex_sampled
 - gps_sdr_signal_processing.h, 1005, 1006
- gps_l1_ca_code_gen_float
 - gps_sdr_signal_processing.h, 1006
- gps_l1_ca_code_gen_int
 - gps_sdr_signal_processing.h, 1006
- gps_l1_ca_dll_pll_tracking.h, 972
- gps_l1_ca_dll_pll_tracking_fpga.h, 972
- gps_l1_ca_dll_pll_tracking_gpu.h, 973
- gps_l1_ca_dll_pll_tracking_gpu_cc.h, 974
- gps_l1_ca_kf_tracking.h, 975
- gps_l1_ca_kf_tracking_cc.h, 975
- gps_l1_ca_pcps_acquisition.h, 976
- gps_l1_ca_pcps_acquisition_fine_doppler.h, 977
- gps_l1_ca_pcps_acquisition_fpga.h, 978
- gps_l1_ca_pcps_assisted_acquisition.h, 979
- gps_l1_ca_pcps_openc1_acquisition.h, 979
- gps_l1_ca_pcps_quicksync_acquisition.h, 980
- gps_l1_ca_pcps_tong_acquisition.h, 981
- gps_l1_ca_tcp_connector_tracking.h, 981
- gps_l1_ca_tcp_connector_tracking_cc.h, 982
- gps_l1_ca_telemetry_decoder.h, 983
- gps_l1_ca_telemetry_decoder_gs, 468
 - general_work, 469
 - set_channel, 469
 - set_satellite, 469
- gps_l1_ca_telemetry_decoder_gs.h, 984
- gps_l2_m_dll_pll_tracking.h, 985
- gps_l2_m_dll_pll_tracking_fpga.h, 985
- gps_l2_m_pcps_acquisition.h, 986
- gps_l2_m_pcps_acquisition_fpga.h, 987
- gps_l2c_m_code_gen_complex
 - gps_l2c_signal.h, 991
- gps_l2c_m_code_gen_complex_sampled
 - gps_l2c_signal.h, 992
- gps_l2c_signal.h, 991
 - gps_l2c_m_code_gen_complex, 991
 - gps_l2c_m_code_gen_complex_sampled, 992
- gps_l2c_telemetry_decoder.h, 992
- gps_l2c_telemetry_decoder_gs, 470
 - general_work, 470
 - set_channel, 470
 - set_satellite, 471
- gps_l2c_telemetry_decoder_gs.h, 993
- gps_l5_dll_pll_tracking.h, 997
- gps_l5_dll_pll_tracking_fpga.h, 998
- gps_l5_signal.h, 998
 - gps_l5i_code_gen_complex, 999
 - gps_l5i_code_gen_complex_sampled, 999
 - gps_l5i_code_gen_float, 1000
 - gps_l5q_code_gen_complex, 1000
 - gps_l5q_code_gen_complex_sampled, 1000
 - gps_l5q_code_gen_float, 1000
- gps_l5_telemetry_decoder.h, 1001
- gps_l5_telemetry_decoder_gs, 471
 - set_channel, 472
 - set_satellite, 472
- gps_l5_telemetry_decoder_gs.h, 1001
- gps_l5i_code_gen_complex
 - gps_l5_signal.h, 999
- gps_l5i_code_gen_complex_sampled
 - gps_l5_signal.h, 999
- gps_l5i_code_gen_float
 - gps_l5_signal.h, 1000
- gps_l5i_pcps_acquisition.h, 1002
- gps_l5i_pcps_acquisition_fpga.h, 1003
- gps_l5q_code_gen_complex
 - gps_l5_signal.h, 1000
- gps_l5q_code_gen_complex_sampled
 - gps_l5_signal.h, 1000
- gps_l5q_code_gen_float
 - gps_l5_signal.h, 1000
- gps_navigation_message.h, 1004
- gps_sdr_signal_processing.h, 1004
 - gps_l1_ca_code_gen_complex, 1005
 - gps_l1_ca_code_gen_complex_sampled, 1005, 1006
 - gps_l1_ca_code_gen_float, 1006
 - gps_l1_ca_code_gen_int, 1006
- gps_utc_model.h, 1006
- GpsL1CaDllPllTracking, 479
 - implementation, 480
 - set_channel, 480
 - set_gnss_synchro, 480
 - stop_tracking, 480
- GpsL1CaDllPllTrackingFpga, 481
 - ~GpsL1CaDllPllTrackingFpga, 482
 - connect, 482
 - disconnect, 483
 - get_left_block, 483
 - get_right_block, 483
 - GpsL1CaDllPllTrackingFpga, 482
 - implementation, 483
 - item_size, 483
 - role, 484
 - set_channel, 484
 - set_gnss_synchro, 484
 - start_tracking, 484
 - stop_tracking, 485
- GpsL1CaDllPllTrackingGPU, 485
 - implementation, 486
 - set_channel, 486
 - set_gnss_synchro, 486
 - stop_tracking, 486
- GpsL1CaKfTracking, 487

- implementation, 488
- set_channel, 488
- set_gnss_synchro, 488
- stop_tracking, 488
- GpsL1CaPcpsAcquisition, 489
 - implementation, 490
 - init, 490
 - mag, 490
 - reset, 491
 - set_channel, 491
 - set_channel_fsm, 491
 - set_doppler_center, 491
 - set_doppler_max, 491
 - set_doppler_step, 492
 - set_gnss_synchro, 492
 - set_local_code, 492
 - set_resampler_latency, 492
 - set_state, 492
 - set_threshold, 493
 - stop_acquisition, 493
- GpsL1CaPcpsAcquisitionFineDoppler, 493
 - implementation, 494
 - init, 495
 - mag, 495
 - reset, 495
 - set_channel, 495
 - set_channel_fsm, 495
 - set_doppler_max, 496
 - set_doppler_step, 496
 - set_gnss_synchro, 496
 - set_state, 496
 - set_threshold, 497
 - stop_acquisition, 497
- GpsL1CaPcpsAcquisitionFpga, 497
 - ~GpsL1CaPcpsAcquisitionFpga, 499
 - connect, 499
 - disconnect, 499
 - get_left_block, 500
 - get_right_block, 500
 - GpsL1CaPcpsAcquisitionFpga, 499
 - implementation, 500
 - init, 500
 - item_size, 500
 - mag, 501
 - reset, 501
 - role, 501
 - set_channel, 501
 - set_channel_fsm, 501
 - set_doppler_center, 502
 - set_doppler_max, 502
 - set_doppler_step, 502
 - set_gnss_synchro, 502
 - set_local_code, 502
 - set_resampler_latency, 503
 - set_state, 503
 - set_threshold, 503
 - stop_acquisition, 503
- GpsL1CaPcpsAssistedAcquisition, 504
 - implementation, 505
 - init, 505
 - mag, 505
 - reset, 505
 - set_channel, 505
 - set_channel_fsm, 506
 - set_doppler_max, 506
 - set_doppler_step, 506
 - set_gnss_synchro, 506
 - set_threshold, 507
 - stop_acquisition, 507
- GpsL1CaPcpsOpenCIAcquisition, 507
 - implementation, 508
 - init, 509
 - mag, 509
 - reset, 509
 - set_channel, 509
 - set_channel_fsm, 509
 - set_doppler_max, 510
 - set_doppler_step, 510
 - set_gnss_synchro, 510
 - set_local_code, 510
 - set_threshold, 511
 - stop_acquisition, 511
- GpsL1CaPcpsQuickSyncAcquisition, 511
 - implementation, 512
 - init, 513
 - mag, 513
 - reset, 513
 - set_channel, 513
 - set_channel_fsm, 513
 - set_doppler_max, 514
 - set_doppler_step, 514
 - set_gnss_synchro, 514
 - set_local_code, 514
 - set_state, 515
 - set_threshold, 515
 - stop_acquisition, 515
- GpsL1CaPcpsTongAcquisition, 516
 - implementation, 517
 - init, 517
 - mag, 517
 - reset, 517
 - set_channel, 517
 - set_channel_fsm, 518
 - set_doppler_max, 518
 - set_doppler_step, 518
 - set_gnss_synchro, 518
 - set_local_code, 519
 - set_state, 519
 - set_threshold, 519
 - stop_acquisition, 519
- GpsL1CaTcpConnectorTracking, 520
 - implementation, 520
 - set_channel, 521
 - set_gnss_synchro, 521
 - stop_tracking, 521
- GpsL1CaTelemetryDecoder, 522

- implementation, [522](#)
- GpsL2CTelemetryDecoder, [523](#)
 - implementation, [523](#)
- GpsL2MDIIPITracking, [524](#)
 - implementation, [524](#)
 - set_channel, [525](#)
 - set_gnss_synchro, [525](#)
 - stop_tracking, [525](#)
- GpsL2MDIIPITrackingFpga, [526](#)
 - implementation, [526](#)
 - set_channel, [527](#)
 - set_gnss_synchro, [527](#)
 - stop_tracking, [527](#)
- GpsL2MPcpsAcquisition, [528](#)
 - implementation, [529](#)
 - init, [529](#)
 - mag, [529](#)
 - reset, [529](#)
 - set_channel, [530](#)
 - set_channel_fsm, [530](#)
 - set_doppler_center, [530](#)
 - set_doppler_max, [530](#)
 - set_doppler_step, [530](#)
 - set_gnss_synchro, [531](#)
 - set_local_code, [531](#)
 - set_resampler_latency, [531](#)
 - set_state, [531](#)
 - set_threshold, [531](#)
 - stop_acquisition, [532](#)
- GpsL2MPcpsAcquisitionFpga, [532](#)
 - implementation, [533](#)
 - init, [534](#)
 - mag, [534](#)
 - reset, [534](#)
 - set_channel, [534](#)
 - set_channel_fsm, [534](#)
 - set_doppler_max, [535](#)
 - set_doppler_step, [535](#)
 - set_gnss_synchro, [535](#)
 - set_local_code, [535](#)
 - set_state, [536](#)
 - set_threshold, [536](#)
 - stop_acquisition, [536](#)
- GpsL5DIIPITracking, [537](#)
 - implementation, [537](#)
 - set_channel, [538](#)
 - set_gnss_synchro, [538](#)
 - stop_tracking, [538](#)
- GpsL5DIIPITrackingFpga, [539](#)
 - ~GpsL5DIIPITrackingFpga, [540](#)
 - connect, [540](#)
 - disconnect, [540](#)
 - get_left_block, [540](#)
 - get_right_block, [541](#)
 - GpsL5DIIPITrackingFpga, [540](#)
 - implementation, [541](#)
 - item_size, [541](#)
 - role, [541](#)
 - set_channel, [541](#)
 - set_gnss_synchro, [542](#)
 - start_tracking, [542](#)
 - stop_tracking, [542](#)
- GpsL5TelemetryDecoder, [554](#)
 - implementation, [554](#)
- GpsL5IPcpsAcquisition, [543](#)
 - implementation, [544](#)
 - init, [544](#)
 - mag, [544](#)
 - reset, [544](#)
 - set_channel, [545](#)
 - set_channel_fsm, [545](#)
 - set_doppler_center, [545](#)
 - set_doppler_max, [545](#)
 - set_doppler_step, [545](#)
 - set_gnss_synchro, [546](#)
 - set_local_code, [546](#)
 - set_resampler_latency, [546](#)
 - set_state, [546](#)
 - set_threshold, [546](#)
 - stop_acquisition, [547](#)
- GpsL5IPcpsAcquisitionFpga, [547](#)
 - ~GpsL5IPcpsAcquisitionFpga, [549](#)
 - connect, [549](#)
 - disconnect, [549](#)
 - get_left_block, [550](#)
 - get_right_block, [550](#)
 - GpsL5IPcpsAcquisitionFpga, [549](#)
 - implementation, [550](#)
 - init, [550](#)
 - item_size, [550](#)
 - mag, [551](#)
 - reset, [551](#)
 - role, [551](#)
 - set_channel, [551](#)
 - set_channel_fsm, [551](#)
 - set_doppler_center, [552](#)
 - set_doppler_max, [552](#)
 - set_doppler_step, [552](#)
 - set_gnss_synchro, [552](#)
 - set_local_code, [552](#)
 - set_resampler_latency, [553](#)
 - set_state, [553](#)
 - set_threshold, [553](#)
 - stop_acquisition, [553](#)
- Gpx_Printer, [556](#)
- gpx_printer.h, [1007](#)
- Gr_Complex_Ip_Packet_Source, [556](#)
- gr_complex_ip_packet_source.h, [1008](#)
- Gravity_ECEF
 - geofunctions.h, [881](#)
- great_circle_distance
 - geofunctions.h, [882](#)
- gtime_t, [557](#)
- HALF_PI
 - MATH_CONSTANTS.h, [1030](#)
- HION

- rtklib.h, [1089](#)
- half_cyc_tag, [557](#)
- have_new_almanac
 - Beidou_Dnav_Navigation_Message, [112](#)
 - Glonass_Gnav_Navigation_Message, [342](#)
- have_new_ephemeris
 - Beidou_Dnav_Navigation_Message, [113](#)
 - Glonass_Gnav_Navigation_Message, [342](#)
 - Gps_CNAV_Navigation_Message, [443](#)
- have_new_iono
 - Beidou_Dnav_Navigation_Message, [113](#)
 - Gps_CNAV_Navigation_Message, [443](#)
- have_new_utc_model
 - Beidou_Dnav_Navigation_Message, [113](#)
 - Glonass_Gnav_Navigation_Message, [342](#)
 - Gps_CNAV_Navigation_Message, [443](#)
- hex_to_bin
 - Rtcm, [672](#)
- hex_to_binary_converter
 - gnss_signal_processing.h, [952](#)
- hex_to_int
 - Rtcm, [672](#)
- hex_to_uint
 - Rtcm, [673](#)
- hybrid_observables.h, [1008](#)
- hybrid_observables_gs, [558](#)
- hybrid_observables_gs.h, [1009](#)
- HybridObservables, [558](#)
 - implementation, [559](#)
 - item_size, [559](#)
- i_0_2
 - Galileo_Ephemeris, [219](#)
- i_AODO
 - Beidou_Dnav_Ephemeris, [105](#)
 - Gps_Ephemeris, [460](#)
- i_AS_status
 - Gps_Almanac, [422](#)
- i_BEIDOU_week
 - Beidou_Dnav_Ephemeris, [105](#)
- i_DeltaT_LS
 - Beidou_Dnav_Utc_Model, [116](#)
- i_DN
 - Beidou_Dnav_Utc_Model, [117](#)
 - Gps_CNAV_Utc_Model, [446](#)
 - Gps_Utc_Model, [478](#)
- i_GPS_week
 - Gps_CNAV_Ephemeris, [437](#)
 - Gps_Ephemeris, [461](#)
- i_SV_accuracy
 - Beidou_Dnav_Ephemeris, [106](#)
 - Gps_Ephemeris, [461](#)
- i_SV_health
 - Beidou_Dnav_Almanac, [91](#)
 - Gps_Almanac, [423](#)
- i_Toa
 - Gps_Almanac, [423](#)
- i_URA
 - Gps_CNAV_Ephemeris, [438](#)
- i_WN_LSF
 - Beidou_Dnav_Utc_Model, [117](#)
 - Gps_CNAV_Utc_Model, [446](#)
 - Gps_Utc_Model, [478](#)
- i_WN_T
 - Gps_CNAV_Utc_Model, [446](#)
 - Gps_Utc_Model, [478](#)
- i_WNa
 - Gps_Almanac, [423](#)
- i_code_on_L2
 - Gps_Ephemeris, [461](#)
- i_nav_type
 - Beidou_Dnav_Ephemeris, [105](#)
- i_prn
 - Sbas_Ephemeris, [704](#)
- i_satellite_PRN
 - Beidou_Dnav_Almanac, [91](#)
 - Beidou_Dnav_Ephemeris, [106](#)
 - Galileo_Almanac, [201](#)
 - Galileo_Ephemeris, [219](#)
 - Glionass_Gnav_Almanac, [324](#)
 - Glionass_Gnav_Ephemeris, [339](#)
 - Gps_Acq_Assist, [419](#)
 - Gps_Almanac, [422](#)
- i_satellite_freq_channel
 - Glionass_Gnav_Almanac, [324](#)
 - Glionass_Gnav_Ephemeris, [338](#)
- i_satellite_slot_number
 - Glionass_Gnav_Almanac, [324](#)
 - Glionass_Gnav_Ephemeris, [339](#)
- i_sig_type
 - Beidou_Dnav_Ephemeris, [106](#)
- i_signal_health
 - Gps_CNAV_Ephemeris, [437](#)
- i_sv_ura
 - Sbas_Ephemeris, [704](#)
- i_t0
 - Sbas_Ephemeris, [704](#)
- iDot_2
 - Galileo_Ephemeris, [219](#)
- IGPBAND1
 - rtklib_sbas.h, [1143](#)
- IGPBAND2
 - rtklib_sbas.h, [1143](#)
- INIReader, [563](#)
 - Get, [563](#)
 - GetInteger, [563](#)
 - INIReader, [563](#)
 - ParseError, [564](#)
- INIReader.h, [1013](#)
- INT_SWAP_STAT
 - rtklib.h, [1090](#)
- INT_SWAP_TRAC
 - rtklib.h, [1090](#)
- IONOOPT_BRDC
 - rtklib.h, [1090](#)
- IONOOPT_EST
 - rtklib.h, [1090](#)

- IONOOPT_IFLC
 - rtklib.h, [1090](#)
- IONOOPT_LEX
 - rtklib.h, [1091](#)
- IONOOPT_OFF
 - rtklib.h, [1091](#)
- IONOOPT_QZS
 - rtklib.h, [1091](#)
- IONOOPT_SBAS
 - rtklib.h, [1091](#)
- IONOOPT_STEC
 - rtklib.h, [1091](#)
- IONOOPT_TEC
 - rtklib.h, [1092](#)
- ibyte_to_cbyte.h, [1010](#)
- ibyte_to_complex.h, [1011](#)
- ibyte_to_cshort.h, [1011](#)
- lbyteToCbyte, [560](#)
 - implementation, [560](#)
- lbyteToComplex, [561](#)
 - implementation, [561](#)
- lbyteToCshort, [562](#)
 - implementation, [562](#)
- implementation
 - Ad9361FpgaSignalSource, [76](#)
 - ArraySignalConditioner, [82](#)
 - BeamformerFilter, [84](#)
 - BeidouB1iPcpsAcquisition, [120](#)
 - BeidouB1iTelemetryDecoder, [124](#)
 - BeidouB3iPcpsAcquisition, [127](#)
 - BeidouB3iTelemetryDecoder, [131](#)
 - ByteToShort, [133](#)
 - Channel, [136](#)
 - CustomUDPSignalSource, [162](#)
 - DirectResamplerConditioner, [166](#)
 - FileSignalSource, [179](#)
 - FirFilter, [181](#)
 - FlexibandSignalSource, [182](#)
 - Fmcomms2SignalSource, [183](#)
 - FreqXlatingFirFilter, [196](#)
 - GalileoE1BTelemetryDecoder, [239](#)
 - GalileoE1DIIPiIVemlTracking, [240](#)
 - GalileoE1DIIPiIVemlTrackingFpga, [243](#)
 - GalileoE1Pcps8msAmbiguousAcquisition, [246](#)
 - GalileoE1PcpsAmbiguousAcquisition, [250](#)
 - GalileoE1PcpsAmbiguousAcquisitionFpga, [256](#)
 - GalileoE1PcpsCccwsrAmbiguousAcquisition, [261](#)
 - GalileoE1PcpsQuickSyncAmbiguousAcquisition, [265](#)
 - GalileoE1PcpsTongAmbiguousAcquisition, [269](#)
 - GalileoE1TcpConnectorTracking, [273](#)
 - GalileoE5aDIIPiITracking, [275](#)
 - GalileoE5aDIIPiITrackingFpga, [279](#)
 - GalileoE5aNoncoherentIQAcquisitionCaf, [282](#)
 - GalileoE5aPcpsAcquisitionFpga, [292](#)
 - GalileoE5aTelemetryDecoder, [297](#)
 - GalileoE5bDIIPiITracking, [299](#)
 - GalileoE5bPcpsAcquisition, [303](#)
 - GalileoE5bPcpsAcquisitionFpga, [309](#)
 - GalileoE5bTelemetryDecoder, [315](#)
 - GenSignalSource, [317](#)
 - GlionassL1CaDIIPiITracking, [356](#)
 - GlionassL1CaDIIPiITracking, [358](#)
 - GlionassL1CaPcpsAcquisition, [360](#)
 - GlionassL1CaTelemetryDecoder, [364](#)
 - GlionassL2CaDIIPiITracking, [365](#)
 - GlionassL2CaDIIPiITracking, [367](#)
 - GlionassL2CaPcpsAcquisition, [369](#)
 - GlionassL2CaTelemetryDecoder, [373](#)
 - Gn3sSignalSource, [374](#)
 - GnMaxSignalSource, [375](#)
 - GpsL1CaDIIPiITracking, [480](#)
 - GpsL1CaDIIPiITrackingFpga, [483](#)
 - GpsL1CaDIIPiITrackingGPU, [486](#)
 - GpsL1CaKfTracking, [488](#)
 - GpsL1CaPcpsAcquisition, [490](#)
 - GpsL1CaPcpsAcquisitionFineDoppler, [494](#)
 - GpsL1CaPcpsAcquisitionFpga, [500](#)
 - GpsL1CaPcpsAssistedAcquisition, [505](#)
 - GpsL1CaPcpsOpenCIAcquisition, [508](#)
 - GpsL1CaPcpsQuickSyncAcquisition, [512](#)
 - GpsL1CaPcpsTongAcquisition, [517](#)
 - GpsL1CaTcpConnectorTracking, [520](#)
 - GpsL1CaTelemetryDecoder, [522](#)
 - GpsL2CTelemetryDecoder, [523](#)
 - GpsL2MDIIPiITracking, [524](#)
 - GpsL2MDIIPiITrackingFpga, [526](#)
 - GpsL2MPcpsAcquisition, [529](#)
 - GpsL2MPcpsAcquisitionFpga, [533](#)
 - GpsL5DIIPiITracking, [537](#)
 - GpsL5DIIPiITrackingFpga, [541](#)
 - GpsL5TelemetryDecoder, [554](#)
 - GpsL5iPcpsAcquisition, [544](#)
 - GpsL5iPcpsAcquisitionFpga, [550](#)
 - HybridObservables, [559](#)
 - lbyteToCbyte, [560](#)
 - lbyteToComplex, [561](#)
 - lbyteToCshort, [562](#)
 - lshortToComplex, [568](#)
 - lshortToCshort, [569](#)
 - LabsatSignalSource, [571](#)
 - MultichannelFileSignalSource, [577](#)
 - NotchFilter, [582](#)
 - NotchFilterLite, [583](#)
 - NsrFileSignalSource, [585](#)
 - OsmosdrSignalSource, [589](#)
 - Pass_Through, [590](#)
 - PlutosdrSignalSource, [634](#)
 - PulseBlankingFilter, [638](#)
 - RawArraySignalSource, [647](#)
 - Rtklib_Pvt, [691](#)
 - RtlTcpSignalSource, [701](#)
 - SbasL1TelemetryDecoder, [707](#)
 - SignalConditioner, [720](#)
 - SignalGenerator, [721](#)
 - SpirFileSignalSource, [726](#)

- TwoBitCpxFileSignalSource, [745](#)
- TwoBitPackedFileSignalSource, [747](#)
- UhdSignalSource, [748](#)
- in_memory_configuration.h, [1012](#)
- InMemoryConfiguration, [564](#)
- ini.h, [1013](#)
- init
 - BeidouB1iPcpsAcquisition, [120](#)
 - BeidouB3iPcpsAcquisition, [127](#)
 - cnav_v27_part_t, [147](#)
 - galileo_e5a_noncoherentIQ_acquisition_caf_cc, [206](#)
 - galileo_pcps_8ms_acquisition_cc, [230](#)
 - GalileoE1Pcps8msAmbiguousAcquisition, [247](#)
 - GalileoE1PcpsAmbiguousAcquisition, [251](#)
 - GalileoE1PcpsAmbiguousAcquisitionFpga, [256](#)
 - GalileoE1PcpsCccwsrAmbiguousAcquisition, [261](#)
 - GalileoE1PcpsQuickSyncAmbiguousAcquisition, [265](#)
 - GalileoE1PcpsTongAmbiguousAcquisition, [270](#)
 - GalileoE5aNoncoherentIQAcquisitionCaf, [282](#)
 - GalileoE5aPcpsAcquisition, [286](#)
 - GalileoE5aPcpsAcquisitionFpga, [292](#)
 - GalileoE5bPcpsAcquisition, [303](#)
 - GalileoE5bPcpsAcquisitionFpga, [310](#)
 - GlonassL1CaPcpsAcquisition, [360](#)
 - GlonassL2CaPcpsAcquisition, [369](#)
 - GpsL1CaPcpsAcquisition, [490](#)
 - GpsL1CaPcpsAcquisitionFineDoppler, [495](#)
 - GpsL1CaPcpsAcquisitionFpga, [500](#)
 - GpsL1CaPcpsAssistedAcquisition, [505](#)
 - GpsL1CaPcpsOpenCIAcquisition, [509](#)
 - GpsL1CaPcpsQuickSyncAcquisition, [513](#)
 - GpsL1CaPcpsTongAcquisition, [517](#)
 - GpsL2MPcpsAcquisition, [529](#)
 - GpsL2MPcpsAcquisitionFpga, [534](#)
 - GpsL5iPcpsAcquisition, [544](#)
 - GpsL5iPcpsAcquisitionFpga, [550](#)
 - pcps_acquisition, [592](#)
 - pcps_acquisition_fine_doppler_cc, [597](#)
 - pcps_acquisition_fpga, [603](#)
 - pcps_assisted_acquisition_cc, [609](#)
 - pcps_cccwsr_acquisition_cc, [613](#)
 - pcps_opencl_acquisition_cc, [618](#)
 - pcps_quicksync_acquisition_cc, [624](#)
 - pcps_tong_acquisition_cc, [629](#)
- initialize
 - Tracking_2nd_DLL_filter, [738](#)
- initialize_secondary_code
 - Fpga_Multicorrelator_8sc, [191](#)
- interleaved_byte_to_complex_byte, [565](#)
- interleaved_byte_to_complex_byte.h, [1014](#)
- interleaved_byte_to_complex_short, [566](#)
- interleaved_byte_to_complex_short.h, [1015](#)
- interleaved_short_to_complex_short, [566](#)
- interleaved_short_to_complex_short.h, [1016](#)
- interp_TOW_ms
 - Gnss_Synchro, [402](#)
- invert
 - cnav_v27_part_t, [147](#)
- is_server_running
 - Rtcm, [673](#)
- ishort_to_complex.h, [1017](#)
- ishort_to_cshort.h, [1017](#)
- IshortToComplex, [567](#)
 - implementation, [568](#)
- IshortToCshort, [568](#)
 - implementation, [569](#)
- item_size
 - GalileoE1DIIPIIVemlTrackingFpga, [243](#)
 - GalileoE1PcpsAmbiguousAcquisitionFpga, [257](#)
 - GalileoE5aDIIPIITrackingFpga, [279](#)
 - GalileoE5aPcpsAcquisitionFpga, [292](#)
 - GalileoE5bPcpsAcquisition, [303](#)
 - GalileoE5bPcpsAcquisitionFpga, [310](#)
 - GpsL1CaDIIPIITrackingFpga, [483](#)
 - GpsL1CaPcpsAcquisitionFpga, [500](#)
 - GpsL5DIIPIITrackingFpga, [541](#)
 - GpsL5iPcpsAcquisitionFpga, [550](#)
 - HybridObservables, [559](#)
 - Rtklib_Pvt, [691](#)
- item_type_helpers.h, [1018](#)
 - item_type_is_complex, [1019](#)
 - item_type_size, [1019](#)
 - item_type_valid, [1019](#)
 - make_vector_converter, [1019](#)
- item_type_is_complex
 - item_type_helpers.h, [1019](#)
- item_type_size
 - item_type_helpers.h, [1019](#)
- item_type_valid
 - item_type_helpers.h, [1019](#)
- kernel_info_t, [569](#)
- Kml_Printer, [570](#)
- kml_printer.h, [1020](#)
- LAM_CARR
 - rtklib.h, [1092](#)
- labsat23_source, [570](#)
- labsat23_source.h, [1021](#)
- labsat_signal_source.h, [1021](#)
- LabsatSignalSource, [571](#)
 - implementation, [571](#)
- lex_t, [572](#)
- lexeph_t, [572](#)
- lexion_t, [573](#)
- lexmsg_t, [573](#)
- load_cnav_ephemeris_xml
 - Gnss_Sdr_Supl_Client, [386](#)
- load_cnav_utc_xml
 - Gnss_Sdr_Supl_Client, [387](#)
- load_ephemeris_xml
 - Gnss_Sdr_Supl_Client, [387](#)
- load_gal_almanac_xml
 - Gnss_Sdr_Supl_Client, [387](#)
- load_gal_ephemeris_xml

- Gnss_Sdr_Supl_Client, [387](#)
- load_gal_iono_xml
 - Gnss_Sdr_Supl_Client, [387](#)
- load_gal_utc_xml
 - Gnss_Sdr_Supl_Client, [387](#)
- load_glo_utc_xml
 - Gnss_Sdr_Supl_Client, [388](#)
- load_gnav_ephemeris_xml
 - Gnss_Sdr_Supl_Client, [388](#)
- load_gps_almanac_xml
 - Gnss_Sdr_Supl_Client, [388](#)
- load_iono_xml
 - Gnss_Sdr_Supl_Client, [388](#)
- load_ref_location_xml
 - Gnss_Sdr_Supl_Client, [388](#)
- load_ref_time_xml
 - Gnss_Sdr_Supl_Client, [388](#)
- load_utc_xml
 - Gnss_Sdr_Supl_Client, [389](#)
- lock_channel
 - Fpga_Multicorrelator_8sc, [191](#)
- lock_detectors.h, [1022](#)
 - carrier_lock_detector, [1023](#)
 - cn0_m2m4_estimator, [1023](#)
 - cn0_svn_estimator, [1023](#)
- lock_time
 - Rtcm, [673](#), [674](#)
 - Rtcm_Printer, [685](#)
- log_rinex_nav
 - Rinex_Printer, [654–656](#)
- log_rinex_obs
 - Rinex_Printer, [656–659](#)
- M0_1
 - Galileo_Ephemeris, [220](#)
- MATH_CONSTANTS.h, [1024](#)
 - AS2R, [1028](#)
 - AU, [1028](#)
 - BEIDOU_GM, [1028](#)
 - BEIDOU_OMEGA_EARTH_DOT, [1028](#)
 - BEIDOU_F, [1028](#)
 - D2R, [1028](#)
 - GALILEO_GM, [1029](#)
 - GALILEO_F, [1029](#)
 - GLONASS_GM, [1029](#)
 - GLONASS_OMEGA_EARTH_DOT, [1029](#)
 - GNSS_OMEGA_EARTH_DOT, [1029](#)
 - GNSS_PI, [1030](#)
 - GPS_GM, [1030](#)
 - GPS_F, [1030](#)
 - HALF_PI, [1030](#)
 - PI_TWO_N19, [1030](#)
 - PI_TWO_N23, [1031](#)
 - PI_TWO_N31, [1031](#)
 - PI_TWO_N38, [1031](#)
 - PI_TWO_N43, [1031](#)
 - R2D, [1031](#)
 - SC2RAD, [1032](#)
 - SPEED_OF_LIGHT_M_MS, [1032](#)
 - SPEED_OF_LIGHT_M_S, [1032](#)
 - TWO_N10, [1032](#)
 - TWO_N11, [1032](#)
 - TWO_N14, [1033](#)
 - TWO_N15, [1033](#)
 - TWO_N16, [1033](#)
 - TWO_N17, [1033](#)
 - TWO_N18, [1033](#)
 - TWO_N19, [1034](#)
 - TWO_N2, [1034](#)
 - TWO_N20, [1034](#)
 - TWO_N21, [1034](#)
 - TWO_N23, [1034](#)
 - TWO_N24, [1035](#)
 - TWO_N25, [1035](#)
 - TWO_N27, [1035](#)
 - TWO_N29, [1035](#)
 - TWO_N30, [1035](#)
 - TWO_N31, [1036](#)
 - TWO_N32, [1036](#)
 - TWO_N33, [1036](#)
 - TWO_N34, [1036](#)
 - TWO_N35, [1036](#)
 - TWO_N38, [1037](#)
 - TWO_N39, [1037](#)
 - TWO_N40, [1037](#)
 - TWO_N43, [1037](#)
 - TWO_N44, [1037](#)
 - TWO_N46, [1038](#)
 - TWO_N48, [1038](#)
 - TWO_N5, [1038](#)
 - TWO_N50, [1038](#)
 - TWO_N51, [1038](#)
 - TWO_N55, [1039](#)
 - TWO_N57, [1039](#)
 - TWO_N59, [1039](#)
 - TWO_N6, [1039](#)
 - TWO_N60, [1039](#)
 - TWO_N66, [1040](#)
 - TWO_N68, [1040](#)
 - TWO_N8, [1040](#)
 - TWO_N9, [1040](#)
 - TWO_P11, [1040](#)
 - TWO_P12, [1041](#)
 - TWO_P14, [1041](#)
 - TWO_P16, [1041](#)
 - TWO_P19, [1041](#)
 - TWO_P3, [1041](#)
 - TWO_P31, [1042](#)
 - TWO_P32, [1042](#)
 - TWO_P4, [1042](#)
 - TWO_P56, [1042](#)
 - TWO_P57, [1042](#)
 - TWO_PI, [1043](#)
 - MAX_TOA_DELAY_MS
 - GPS_L1_CA.h, [971](#)
 - MAXANT
 - rtklib.h, [1092](#)

- MAXBAND
 - rtklib.h, [1092](#)
- MAXCLI
 - rtklib.h, [1093](#)
- MAXCODE
 - gnss_obs_codes.h, [939](#)
- MAXDTOE_BDS
 - rtklib.h, [1093](#)
- MAXDTOE_GAL
 - rtklib.h, [1093](#)
- MAXDTOE_GLO
 - rtklib.h, [1093](#)
- MAXDTOE_QZS
 - rtklib.h, [1094](#)
- MAXDTOE_SBS
 - rtklib.h, [1094](#)
- MAXDTOE_S
 - rtklib.h, [1094](#)
- MAXDTOE
 - rtklib.h, [1093](#)
- MAXERRMSG
 - rtklib.h, [1094](#)
- MAXEXFILE
 - rtklib.h, [1094](#)
- MAXFREQ
 - rtklib.h, [1095](#)
- MAXGDOP
 - rtklib.h, [1095](#)
- MAXITR
 - rtklib_pntpos.h, [1120](#)
- MAXLEAPS
 - rtklib.h, [1095](#)
- MAXNGEO
 - rtklib.h, [1095](#)
- MAXNIGP
 - rtklib.h, [1095](#)
- MAXOBSBUF
 - rtklib.h, [1096](#)
- MAXOBSTYPE
 - rtklib.h, [1096](#)
- MAXOBS
 - rtklib.h, [1096](#)
- MAXPRNBDS
 - rtklib.h, [1096](#)
- MAXPRNGAL
 - rtklib.h, [1096](#)
- MAXPRNGLO
 - rtklib.h, [1097](#)
- MAXPRNGPS
 - rtklib.h, [1097](#)
- MAXPRNSBS
 - rtklib.h, [1097](#)
- MAXRAWLEN
 - rtklib.h, [1097](#)
- MAXRCV
 - rtklib.h, [1097](#)
- MAXSBSAGEF
 - rtklib.h, [1098](#)
- MAXSBSAGEL
 - rtklib.h, [1098](#)
- MAXSBSMSG
 - rtklib.h, [1098](#)
- MAXSBSURA
 - rtklib.h, [1098](#)
- MAXSOLBUF
 - rtklib.h, [1098](#)
- MAXSOLMSG
 - rtklib.h, [1099](#)
- MAXSOLQ
 - rtklib.h, [1099](#)
- MAXSTATMSG
 - rtklib.h, [1099](#)
- MAXSTRMSG
 - rtklib.h, [1099](#)
- MAXSTRPATH
 - rtklib.h, [1099](#)
- MINPRNBDS
 - rtklib.h, [1100](#)
- MINPRNGAL
 - rtklib.h, [1100](#)
- MINPRNGLO
 - rtklib.h, [1100](#)
- MINPRNGPS
 - rtklib.h, [1100](#)
- MINPRNSBS
 - rtklib.h, [1100](#)
- mag
 - BeidouB1iPcpsAcquisition, [120](#)
 - BeidouB3iPcpsAcquisition, [127](#)
 - galileo_e5a_noncoherentIQ_acquisition_caf_cc, [206](#)
 - galileo_pcps_8ms_acquisition_cc, [230](#)
 - GalileoE1Pcps8msAmbiguousAcquisition, [247](#)
 - GalileoE1PcpsAmbiguousAcquisition, [251](#)
 - GalileoE1PcpsAmbiguousAcquisitionFpga, [257](#)
 - GalileoE1PcpsCccwsrAmbiguousAcquisition, [261](#)
 - GalileoE1PcpsQuickSyncAmbiguousAcquisition, [265](#)
 - GalileoE1PcpsTongAmbiguousAcquisition, [270](#)
 - GalileoE5aNoncoherentIQAcquisitionCaf, [282](#)
 - GalileoE5aPcpsAcquisition, [286](#)
 - GalileoE5aPcpsAcquisitionFpga, [293](#)
 - GalileoE5bPcpsAcquisition, [303](#)
 - GalileoE5bPcpsAcquisitionFpga, [310](#)
 - GlonassL1CaPcpsAcquisition, [360](#)
 - GlonassL2CaPcpsAcquisition, [369](#)
 - GpsL1CaPcpsAcquisition, [490](#)
 - GpsL1CaPcpsAcquisitionFineDoppler, [495](#)
 - GpsL1CaPcpsAcquisitionFpga, [501](#)
 - GpsL1CaPcpsAssistedAcquisition, [505](#)
 - GpsL1CaPcpsOpenCIAcquisition, [509](#)
 - GpsL1CaPcpsQuickSyncAcquisition, [513](#)
 - GpsL1CaPcpsTongAcquisition, [517](#)
 - GpsL2MPcpsAcquisition, [529](#)
 - GpsL2MPcpsAcquisitionFpga, [534](#)
 - GpsL5iPcpsAcquisition, [544](#)

- GpsL5iPcpsAcquisitionFpga, 551
- pcps_acquisition, 592
- pcps_acquisition_fine_doppler_cc, 597
- pcps_acquisition_fpga, 603
- pcps_assisted_acquisition_cc, 609
- pcps_cccwsr_acquisition_cc, 613
- pcps_openc1_acquisition_cc, 618
- pcps_quicksync_acquisition_cc, 624
- pcps_tong_acquisition_cc, 629
- make_vector_converter
 - item_type_helpers.h, 1019
- message_lock
 - cnav_v27_part_t, 148
- mmse_resampler_conditioner.h, 1043
- MmseResamplerConditioner, 574
- ModelFunction, 574
- Monitor_Pvt, 575
 - serialize, 576
- Monitor_Pvt_Udp_Sink, 576
- monitor_pvt.h, 1044
- monitor_pvt_udp_sink.h, 1044
- msg_id
 - cnav_msg_t, 145
- msm_h_t, 576
- multichannel_file_signal_source.h, 1045
- MultichannelFileSignalSource, 577
 - implementation, 577
- n_crc_fail
 - cnav_v27_part_t, 148
- n_decoded
 - cnav_v27_part_t, 148
- n_symbols
 - cnav_v27_part_t, 148
- NEXOBS
 - rtklib.h, 1101
- NFREQGLO
 - rtklib.h, 1101
- NFREQ
 - rtklib.h, 1101
- NSATBDS
 - rtklib.h, 1101
- NSATGAL
 - rtklib.h, 1101
- NSATGLO
 - rtklib.h, 1102
- NSATGPS
 - rtklib.h, 1102
- NSATSBS
 - rtklib.h, 1102
- NSYS
 - rtklib.h, 1102
- nav_t, 578
- navBdsFile
 - Rinex_Printer, 666
- navFile
 - Rinex_Printer, 666
- navGalFile
 - Rinex_Printer, 667
- navGloFile
 - Rinex_Printer, 667
- navMixFile
 - Rinex_Printer, 667
- nextPowerOf2
 - pcps_acquisition_fine_doppler_cc, 597
- Nmea_Printer, 579
 - ~Nmea_Printer, 580
 - Nmea_Printer, 580
 - Print_Nmea_Line, 580
- nmea_printer.h, 1046
- nonlinear_tracking.h, 1046
- Notch, 580
- notch_cc.h, 1047
- notch_filter.h, 1048
- notch_filter_lite.h, 1049
- notch_lite_cc.h, 1049
- NotchFilter, 581
 - implementation, 582
- NotchFilterLite, 582
 - implementation, 583
- NotchLite, 583
- nsc_enc_bit
 - convolutional.h, 807
- nsc_transit
 - convolutional.h, 807
- nsr_file_signal_source.h, 1050
- NsrFileSignalSource, 584
 - implementation, 585
- ntrip_t, 585
- NX
 - rtklib_pntpos.h, 1120
- OMEGA_0_2
 - Galileo_Ephemeris, 220
- OMEGA_dot_3
 - Galileo_Ephemeris, 220
- Obs_Conf, 585
- obs_conf.h, 1051
- obs_t, 586
- obsFile
 - Rinex_Printer, 667
- obsd_t, 586
- obsdiff_flags.h, 1052
- observable_tests_flags.h, 1052
- Observables_Dump_Reader, 587
- observables_dump_reader.h, 1053
- observables_interface.h, 1054
- ObservablesInterface, 587
- omega_2
 - Galileo_Ephemeris, 220
- open_device
 - Fpga_Acquisition, 185
- operator<<
 - Gnss_Satellite, 383
 - Gnss_Signal, 394
 - Gnuplot, 415
- operator=
 - Acquisition_Dump_Reader, 73

- Exponential_Smoother, 175
- Gnss_Satellite, 382
- Gnss_Synchro, 398
- Serdes_Gnss_Synchro, 713
- Serdes_Monitor_Pvt, 716
- Tracking_loop_filter, 742
- operator==
 - Gnss_Satellite, 383
 - Gnss_Signal, 395
- opt_t, 588
- osmosdr_signal_source.h, 1054
- OsmosdrSignalSource, 588
 - implementation, 589
- PI_TWO_N19
 - MATH_CONSTANTS.h, 1030
- PI_TWO_N23
 - MATH_CONSTANTS.h, 1031
- PI_TWO_N31
 - MATH_CONSTANTS.h, 1031
- PI_TWO_N38
 - MATH_CONSTANTS.h, 1031
- PI_TWO_N43
 - MATH_CONSTANTS.h, 1031
- PMODE_DGPS
 - rtklib.h, 1102
- PMODE_FIXED
 - rtklib.h, 1103
- PMODE_KINEMA
 - rtklib.h, 1103
- PMODE_MOVEB
 - rtklib.h, 1103
- PMODE_PPP_FIXED
 - rtklib.h, 1103
- PMODE_PPP_KINEMA
 - rtklib.h, 1103
- PMODE_PPP_STATIC
 - rtklib.h, 1104
- PMODE_SINGLE
 - rtklib.h, 1104
- PMODE_STATIC
 - rtklib.h, 1104
- POLYCRC24Q
 - rtklib.h, 1104
- POLYCRC32
 - rtklib.h, 1104
- POSOPT_RINEX
 - rtklib.h, 1105
- PRCOPT_DEFAULT
 - rtklib_rtksvr.h, 1141
- PRN_HWBIAS
 - rtklib.h, 1105
- PRN
 - Gnss_Synchro, 402
- parity_counter
 - convolutional.h, 807
- ParseError
 - INIReader, 564
- part1
 - cnav_msg_decoder_t, 144
- part2
 - cnav_msg_decoder_t, 144
- Pass_Through, 589
 - implementation, 590
- pass_through.h, 1055
- pclk_t, 590
- pcps_acquisition, 591
 - general_work, 592
 - init, 592
 - mag, 592
 - set_active, 592
 - set_channel, 593
 - set_channel_fsm, 593
 - set_doppler_center, 593
 - set_doppler_max, 593
 - set_doppler_step, 594
 - set_gnss_synchro, 594
 - set_local_code, 594
 - set_state, 595
 - set_threshold, 595
- pcps_acquisition.h, 1056
- pcps_acquisition_fine_doppler_cc, 595
 - ~pcps_acquisition_fine_doppler_cc, 596
 - general_work, 597
 - init, 597
 - mag, 597
 - nextPowerOf2, 597
 - set_active, 598
 - set_channel, 598
 - set_channel_fsm, 598
 - set_doppler_max, 598
 - set_doppler_step, 600
 - set_gnss_synchro, 600
 - set_local_code, 600
 - set_state, 601
 - set_threshold, 601
- pcps_acquisition_fine_doppler_cc.h, 1057
- pcps_acquisition_fpga, 601
 - ~pcps_acquisition_fpga, 602
 - init, 603
 - mag, 603
 - reset_acquisition, 603
 - set_active, 603
 - set_channel, 603
 - set_channel_fsm, 605
 - set_doppler_center, 605
 - set_doppler_max, 605
 - set_doppler_step, 605
 - set_gnss_synchro, 606
 - set_local_code, 606
 - set_state, 606
 - set_threshold, 607
- pcps_acquisition_fpga.h, 1058
- pcps_assisted_acquisition_cc, 607
 - ~pcps_assisted_acquisition_cc, 608
 - general_work, 608
 - init, 609

- mag, 609
- set_active, 609
- set_channel, 609
- set_channel_fsm, 610
- set_doppler_max, 610
- set_doppler_step, 610
- set_gnss_synchro, 610
- set_local_code, 611
- set_threshold, 611
- pcps_assisted_acquisition_cc.h, 1059
- pcps_cccwsr_acquisition_cc, 612
 - ~pcps_cccwsr_acquisition_cc, 613
 - general_work, 613
 - init, 613
 - mag, 613
 - set_active, 613
 - set_channel, 614
 - set_channel_fsm, 614
 - set_doppler_max, 614
 - set_doppler_step, 614
 - set_gnss_synchro, 615
 - set_local_code, 615
 - set_state, 615
 - set_threshold, 616
- pcps_cccwsr_acquisition_cc.h, 1060
- pcps_openc1_acquisition_cc, 616
 - ~pcps_openc1_acquisition_cc, 617
 - general_work, 618
 - init, 618
 - mag, 618
 - set_active, 618
 - set_channel, 619
 - set_channel_fsm, 619
 - set_doppler_max, 619
 - set_doppler_step, 619
 - set_gnss_synchro, 620
 - set_local_code, 620
 - set_state, 620
 - set_threshold, 622
- pcps_openc1_acquisition_cc.h, 1061
- pcps_quicksync_acquisition_cc, 622
 - ~pcps_quicksync_acquisition_cc, 623
 - general_work, 624
 - init, 624
 - mag, 624
 - set_active, 624
 - set_channel, 624
 - set_channel_fsm, 625
 - set_doppler_max, 625
 - set_doppler_step, 625
 - set_gnss_synchro, 626
 - set_local_code, 626
 - set_state, 626
 - set_threshold, 627
- pcps_quicksync_acquisition_cc.h, 1063
- pcps_tong_acquisition_cc, 627
 - ~pcps_tong_acquisition_cc, 628
 - general_work, 628
 - init, 629
 - mag, 629
 - set_active, 629
 - set_channel, 629
 - set_channel_fsm, 630
 - set_doppler_max, 630
 - set_doppler_step, 630
 - set_gnss_synchro, 630
 - set_local_code, 631
 - set_state, 631
 - set_threshold, 631
- pcps_tong_acquisition_cc.h, 1064
- pcpsconf_fpga_t, 632
- pcv_t, 632
- pcvs_t, 633
- peph_t, 633
- phase_unwrap
 - tracking_discriminators.h, 1173
- pll_cloop_two_quadrant_atan
 - tracking_discriminators.h, 1174
- pll_four_quadrant_atan
 - tracking_discriminators.h, 1174
- plutosdr_signal_source.h, 1066
- PlutosdrSignalSource, 634
 - implementation, 634
- pntpos
 - rtklib_pntpos.h, 1119
- pop_front
 - Gnss_circular_deque, 377
- position_test_flags.h, 1066
- pppcorr_t, 635
- prcopt_t, 635
- preamble_seen
 - cnav_v27_part_t, 148
- print_MSM_1
 - Rtcm, 674
- print_MSM_2
 - Rtcm, 674
- print_MSM_3
 - Rtcm, 675
- print_MSM_4
 - Rtcm, 675
- print_MSM_5
 - Rtcm, 675
- print_MSM_6
 - Rtcm, 676
- print_MSM_7
 - Rtcm, 676
- print_MT1001
 - Rtcm, 676
- print_MT1002
 - Rtcm, 677
- print_MT1003
 - Rtcm, 677
- print_MT1004
 - Rtcm, 677
- print_MT1005
 - Rtcm, 677

- print_MT1005_test
 - Rtcm, [678](#)
 - Rtcm_Printer, [685](#)
- print_MT1006
 - Rtcm, [678](#)
- print_MT1008
 - Rtcm, [678](#)
- print_MT1009
 - Rtcm, [678](#)
- print_MT1010
 - Rtcm, [679](#)
- print_MT1011
 - Rtcm, [680](#)
- print_MT1012
 - Rtcm, [680](#)
- print_MT1019
 - Rtcm, [681](#)
- print_MT1020
 - Rtcm, [681](#)
- print_MT1029
 - Rtcm, [681](#)
- print_MT1045
 - Rtcm, [682](#)
- Print_Nmea_Line
 - Nmea_Printer, [580](#)
- Print_Rtcm_MT1009
 - Rtcm_Printer, [686](#)
- Print_Rtcm_MT1010
 - Rtcm_Printer, [686](#)
- Print_Rtcm_MT1011
 - Rtcm_Printer, [687](#)
- Print_Rtcm_MT1012
 - Rtcm_Printer, [687](#)
- Print_Rtcm_MT1019
 - Rtcm_Printer, [688](#)
- Print_Rtcm_MT1020
 - Rtcm_Printer, [688](#)
- Print_Rtcm_MT1045
 - Rtcm_Printer, [689](#)
- priorize_satellites
 - GNSSFlowgraph, [411](#)
- prn
 - cnav_msg_t, [145](#)
- Prompt_I
 - Gnss_Synchro, [403](#)
- Prompt_Q
 - Gnss_Synchro, [403](#)
- Pseudorange_m
 - Gnss_Synchro, [403](#)
- pulse_blanking_cc, [636](#)
- pulse_blanking_cc.h, [1067](#)
- pulse_blanking_filter.h, [1068](#)
- PulseBlankingFilter, [637](#)
 - implementation, [638](#)
- push_back
 - Gnss_circular_deque, [378](#)
- pv_Geo_to_ECEF
 - geofunctions.h, [882](#)
- Pvt_Conf, [638](#)
- Pvt_Solution, [639](#)
 - get_avg_height, [641](#)
 - get_avg_latitude, [641](#)
 - get_avg_longitude, [641](#)
 - get_clock_drift_ppm, [641](#)
 - get_course_over_ground, [641](#)
 - get_height, [641](#)
 - get_latitude, [642](#)
 - get_longitude, [642](#)
 - get_num_valid_observations, [642](#)
 - get_speed_over_ground, [642](#)
 - get_time_offset_s, [642](#)
 - set_averaging_depth, [642](#)
 - set_clock_drift_ppm, [643](#)
 - set_course_over_ground, [643](#)
 - set_num_valid_observations, [643](#)
 - set_pre_2009_file, [643](#)
 - set_rx_pos, [643](#)
 - set_rx_vel, [643](#)
 - set_speed_over_ground, [644](#)
 - set_time_offset_s, [644](#)
- pvt_conf.h, [1069](#)
- pvt_interface.h, [1070](#)
- pvt_solution.h, [1070](#)
- PvtInterface, [644](#)
- R2D
 - MATH_CONSTANTS.h, [1031](#)
- RE_WGS84
 - rtklib.h, [1105](#)
- REL_HUMI
 - rtklib.h, [1105](#)
- RTL_TCP_COMMAND
 - rtl_tcp_commands.h, [1151](#)
- RX_time
 - Gnss_Synchro, [403](#)
- raw_array_signal_source.h, [1071](#)
- raw_msg
 - cnav_msg_t, [145](#)
- raw_t, [645](#)
- RawArraySignalSource, [646](#)
 - implementation, [647](#)
- read_MT1005
 - Rtcm, [682](#)
- read_MT1019
 - Rtcm, [682](#)
- read_MT1020
 - Rtcm, [682](#)
- read_MT1045
 - Rtcm, [683](#)
- read_acquisition_results
 - Fpga_Acquisition, [185](#)
- read_fpga_total_scale_factor
 - Fpga_Acquisition, [185](#)
- read_sample_counter
 - Fpga_Multicorrelator_8sc, [192](#)
- readProtobuffer
 - Serdes_Gnss_Synchro, [713](#)

- Serdes_Monitor_Pvt, 716
- Region1_flag_5
 - Galileo_Iono, 226
- Region2_flag_5
 - Galileo_Iono, 226
- Region3_flag_5
 - Galileo_Iono, 227
- Region4_flag_5
 - Galileo_Iono, 227
- Region5_flag_5
 - Galileo_Iono, 227
- replot
 - Gnuplot, 415
- resampler
 - gnss_signal_processing.h, 952
- reset
 - BeidouB1iPcpsAcquisition, 121
 - BeidouB3iPcpsAcquisition, 128
 - dll_pll_veml_tracking_fpga, 171
 - GalileoE1Pcps8msAmbiguousAcquisition, 247
 - GalileoE1PcpsAmbiguousAcquisition, 251
 - GalileoE1PcpsAmbiguousAcquisitionFpga, 257
 - GalileoE1PcpsCccwsrAmbiguousAcquisition, 262
 - GalileoE1PcpsQuickSyncAmbiguousAcquisition, 266
 - GalileoE1PcpsTongAmbiguousAcquisition, 270
 - GalileoE5aNoncoherentIQAcquisitionCaf, 282
 - GalileoE5aPcpsAcquisition, 286
 - GalileoE5aPcpsAcquisitionFpga, 293
 - GalileoE5bPcpsAcquisition, 303
 - GalileoE5bPcpsAcquisitionFpga, 310
 - GlonassL1CaPcpsAcquisition, 361
 - GlonassL2CaPcpsAcquisition, 370
 - Gnss_circular_deque, 378
 - GpsL1CaPcpsAcquisition, 491
 - GpsL1CaPcpsAcquisitionFineDoppler, 495
 - GpsL1CaPcpsAcquisitionFpga, 501
 - GpsL1CaPcpsAssistedAcquisition, 505
 - GpsL1CaPcpsOpenCIAcquisition, 509
 - GpsL1CaPcpsQuickSyncAcquisition, 513
 - GpsL1CaPcpsTongAcquisition, 517
 - GpsL2MPcpsAcquisition, 529
 - GpsL2MPcpsAcquisitionFpga, 534
 - GpsL5iPcpsAcquisition, 544
 - GpsL5iPcpsAcquisitionFpga, 551
- reset_acquisition
 - Fpga_Acquisition, 185
 - pcps_acquisition_fpga, 603
- Rinex_Printer, 647
 - ~Rinex_Printer, 652
 - compute_BDS_time, 652
 - compute_GPS_time, 653
 - compute_Galileo_time, 653
 - compute_UTC_time, 653
 - get_leap_second, 654
 - log_rinex_nav, 654–656
 - log_rinex_obs, 656–659
 - navBdsFile, 666
 - navFile, 666
 - navGalFile, 667
 - navGloFile, 667
 - navMixFile, 667
 - obsFile, 667
 - Rinex_Printer, 652
 - rinex_nav_header, 659–662
 - rinex_obs_header, 662–665
 - rinex_sbs_header, 665
 - sbsFile, 667
 - to_date_time, 666
 - update_nav_header, 666
- rinex_nav_header
 - Rinex_Printer, 659–662
- rinex_obs_header
 - Rinex_Printer, 662–665
- rinex_printer.h, 1072
- rinex_sbs_header
 - Rinex_Printer, 665
- role
 - GalileoE1DIIPIIVemlTrackingFpga, 244
 - GalileoE1PcpsAmbiguousAcquisitionFpga, 257
 - GalileoE5aDIIPIITrackingFpga, 279
 - GalileoE5aPcpsAcquisitionFpga, 293
 - GalileoE5bPcpsAcquisition, 304
 - GalileoE5bPcpsAcquisitionFpga, 310
 - GpsL1CaDIIPIITrackingFpga, 484
 - GpsL1CaPcpsAcquisitionFpga, 501
 - GpsL5DIIPIITrackingFpga, 541
 - GpsL5iPcpsAcquisitionFpga, 551
- Rtcm, 668
 - bin_to_binary_data, 671
 - bin_to_double, 671
 - bin_to_hex, 672
 - bin_to_uint, 672
 - binary_data_to_bin, 672
 - check_CRC, 672
 - hex_to_bin, 672
 - hex_to_int, 672
 - hex_to_uint, 673
 - is_server_running, 673
 - lock_time, 673, 674
 - print_MSM_1, 674
 - print_MSM_2, 674
 - print_MSM_3, 675
 - print_MSM_4, 675
 - print_MSM_5, 675
 - print_MSM_6, 676
 - print_MSM_7, 676
 - print_MT1001, 676
 - print_MT1002, 677
 - print_MT1003, 677
 - print_MT1004, 677
 - print_MT1005, 677
 - print_MT1005_test, 678
 - print_MT1006, 678
 - print_MT1008, 678
 - print_MT1009, 678

- print_MT1010, [679](#)
- print_MT1011, [680](#)
- print_MT1012, [680](#)
- print_MT1019, [681](#)
- print_MT1020, [681](#)
- print_MT1029, [681](#)
- print_MT1045, [682](#)
- read_MT1005, [682](#)
- read_MT1019, [682](#)
- read_MT1020, [682](#)
- read_MT1045, [683](#)
- Rtcm, [671](#)
- run_server, [683](#)
- send_message, [683](#)
- stop_server, [683](#)
- rtcm.h, [1073](#)
- Rtcm_Printer, [684](#)
 - ~Rtcm_Printer, [685](#)
 - lock_time, [685](#)
 - print_MT1005_test, [685](#)
 - Print_Rtcm_MT1009, [686](#)
 - Print_Rtcm_MT1010, [686](#)
 - Print_Rtcm_MT1011, [687](#)
 - Print_Rtcm_MT1012, [687](#)
 - Print_Rtcm_MT1019, [688](#)
 - Print_Rtcm_MT1020, [688](#)
 - Print_Rtcm_MT1045, [689](#)
 - Rtcm_Printer, [685](#)
- rtcm_printer.h, [1074](#)
- rtcm_t, [689](#)
- rtk_t, [690](#)
- rtklib.h, [1074](#)
 - ARMODE_CONT, [1084](#)
 - ARMODE_FIXHOLD, [1084](#)
 - ARMODE_INST, [1084](#)
 - ARMODE_OFF, [1084](#)
 - ARMODE_PPPAR_ILS, [1085](#)
 - ARMODE_PPPAR, [1085](#)
 - CHISQR, [1085](#)
 - DTTOL, [1085](#)
 - EFACT_BDS, [1086](#)
 - EFACT_GAL, [1086](#)
 - EFACT_GLO, [1086](#)
 - EFACT_GPS, [1086](#)
 - EFACT_IRN, [1086](#)
 - EFACT_QZS, [1087](#)
 - EFACT_SBS, [1087](#)
 - EPHOPT_BRDC, [1087](#)
 - EPHOPT_LEX, [1087](#)
 - EPHOPT_PREC, [1087](#)
 - EPHOPT_SBAS, [1088](#)
 - EPHOPT_SSRAPC, [1088](#)
 - EPHOPT_SSRCOM, [1088](#)
 - ERR_BRDCI, [1088](#)
 - ERR_CBIAS, [1088](#)
 - ERR_SAAS, [1089](#)
 - FE_WGS84, [1089](#)
 - FTP_TIMEOUT, [1089](#)
 - fatalfunc_t, [1084](#)
 - GAP_RESION, [1089](#)
 - HION, [1089](#)
 - INT_SWAP_STAT, [1090](#)
 - INT_SWAP_TRAC, [1090](#)
 - IONOOPT_BRDC, [1090](#)
 - IONOOPT_EST, [1090](#)
 - IONOOPT_IFLC, [1090](#)
 - IONOOPT_LEX, [1091](#)
 - IONOOPT_OFF, [1091](#)
 - IONOOPT_QZS, [1091](#)
 - IONOOPT_SBAS, [1091](#)
 - IONOOPT_STEC, [1091](#)
 - IONOOPT_TEC, [1092](#)
 - LAM_CARR, [1092](#)
 - MAXANT, [1092](#)
 - MAXBAND, [1092](#)
 - MAXCLI, [1093](#)
 - MAXDToe_BDS, [1093](#)
 - MAXDToe_GAL, [1093](#)
 - MAXDToe_GLO, [1093](#)
 - MAXDToe_QZS, [1094](#)
 - MAXDToe_SBS, [1094](#)
 - MAXDToe_S, [1094](#)
 - MAXDToe, [1093](#)
 - MAXERRMSG, [1094](#)
 - MAXEXFILE, [1094](#)
 - MAXFREQ, [1095](#)
 - MAXGDOP, [1095](#)
 - MAXLEAPS, [1095](#)
 - MAXNGEO, [1095](#)
 - MAXNIGP, [1095](#)
 - MAXOBSBUF, [1096](#)
 - MAXOBSSTYPE, [1096](#)
 - MAXOBS, [1096](#)
 - MAXPRNBDS, [1096](#)
 - MAXPRNGAL, [1096](#)
 - MAXPRNGLO, [1097](#)
 - MAXPRNGPS, [1097](#)
 - MAXPRNSBS, [1097](#)
 - MAXRAWLEN, [1097](#)
 - MAXRCV, [1097](#)
 - MAXSBSAGEF, [1098](#)
 - MAXSBSAGEL, [1098](#)
 - MAXSBSMSG, [1098](#)
 - MAXSBSURA, [1098](#)
 - MAXSOLBUF, [1098](#)
 - MAXSOLMSG, [1099](#)
 - MAXSOLQ, [1099](#)
 - MAXSTATMSG, [1099](#)
 - MAXSTRMSG, [1099](#)
 - MAXSTRPATH, [1099](#)
 - MINPRNBDS, [1100](#)
 - MINPRNGAL, [1100](#)
 - MINPRNGLO, [1100](#)
 - MINPRNGPS, [1100](#)
 - MINPRNSBS, [1100](#)
 - NEXOBS, [1101](#)

- NFREQGLO, [1101](#)
- NFREQ, [1101](#)
- NSATBDS, [1101](#)
- NSATGAL, [1101](#)
- NSATGLO, [1102](#)
- NSATGPS, [1102](#)
- NSATSBS, [1102](#)
- NSYS, [1102](#)
- PMODE_DGPS, [1102](#)
- PMODE_FIXED, [1103](#)
- PMODE_KINEMA, [1103](#)
- PMODE_MOVEB, [1103](#)
- PMODE_PPP_FIXED, [1103](#)
- PMODE_PPP_KINEMA, [1103](#)
- PMODE_PPP_STATIC, [1104](#)
- PMODE_SINGLE, [1104](#)
- PMODE_STATIC, [1104](#)
- POLYCRC24Q, [1104](#)
- POLYCRC32, [1104](#)
- POSOPT_RINEX, [1105](#)
- PRN_HWBIAS, [1105](#)
- RE_WGS84, [1105](#)
- REL_HUMI, [1105](#)
- SERIBUFFSIZE, [1105](#)
- SOLF_ENU, [1106](#)
- SOLF_GSIF, [1106](#)
- SOLF_LLH, [1106](#)
- SOLF_NMEA, [1106](#)
- SOLF_STAT, [1106](#)
- SOLF_XYZ, [1107](#)
- SOLQ_DGPS, [1107](#)
- SOLQ_DR, [1107](#)
- SOLQ_FIX, [1107](#)
- SOLQ_FLOAT, [1107](#)
- SOLQ_NONE, [1108](#)
- SOLQ_PPP, [1108](#)
- SOLQ_SBAS, [1108](#)
- SOLQ_SINGLE, [1108](#)
- SYS_ALL, [1108](#)
- SYS_BDS, [1109](#)
- SYS_GAL, [1109](#)
- SYS_GLO, [1109](#)
- SYS_GPS, [1109](#)
- SYS_IRN, [1109](#)
- SYS_LEO, [1110](#)
- SYS_NONE, [1110](#)
- SYS_QZS, [1110](#)
- SYS_SBS, [1110](#)
- TIMES_GPST, [1110](#)
- TIMES_JST, [1111](#)
- TIMES_UTC, [1111](#)
- TIMETAGH_LEN, [1111](#)
- TINTACT, [1111](#)
- TROPOPT_CORG, [1112](#)
- TROPOPT_COR, [1111](#)
- TROPOPT_ESTG, [1112](#)
- TROPOPT_EST, [1112](#)
- TROPOPT_OFF, [1112](#)
- TROPOPT_SAAS, [1112](#)
- TROPOPT_SBAS, [1113](#)
- Rtklib_Pvt, [690](#)
 - implementation, [691](#)
 - item_size, [691](#)
- Rtklib_Solver, [695](#)
 - beidou_dnav_ephemeris_map, [696](#)
 - galileo_ephemeris_map, [696](#)
 - glonass_gnav_almanac, [696](#)
 - glonass_gnav_ephemeris_map, [697](#)
 - glonass_gnav_utc_model, [697](#)
 - gps_cnav_ephemeris_map, [697](#)
 - gps_ephemeris_map, [697](#)
- Rtklib_Solver_Dump_Reader, [698](#)
- rtklib_conversions.h, [1113](#)
 - eph_to_rtklib, [1114](#)
- rtklib_ephemeris.h, [1114](#)
- rtklib_ionex.h, [1115](#)
- rtklib_lambda.h, [1116](#)
 - SWAP_LAMBDA, [1117](#)
- rtklib_pntpos.h, [1118](#)
 - ERR_ION, [1120](#)
 - ERR_TROP, [1120](#)
 - MAXITR, [1120](#)
 - NX, [1120](#)
 - pntpos, [1119](#)
- rtklib_ppp.h, [1121](#)
 - SWAP_D, [1123](#)
 - SWAP_I, [1123](#)
- rtklib_preceph.h, [1123](#)
- rtklib_pvt.h, [1125](#)
- rtklib_pvt_gs, [692](#)
 - ~rtklib_pvt_gs, [693](#)
 - clear_ephemeris, [693](#)
 - get_beidou_dnav_almanac_map, [693](#)
 - get_beidou_dnav_ephemeris_map, [693](#)
 - get_galileo_almanac_map, [693](#)
 - get_galileo_ephemeris_map, [694](#)
 - get_gps_almanac_map, [694](#)
 - get_gps_ephemeris_map, [694](#)
 - get_latest_PVT, [694](#)
 - work, [694](#)
- rtklib_pvt_gs.h, [1125](#)
- rtklib_rtcn.h, [1126](#)
- rtklib_rtcn2.h, [1127](#)
- rtklib_rtcn3.h, [1128](#)
 - CODES_BDS, [1131](#)
 - CODES_GAL, [1131](#)
 - CODES_GLO, [1131](#)
 - CODES_GPS, [1131](#)
 - CODES_QZS, [1132](#)
 - CODES_SBS, [1132](#)
 - SSRUDINT, [1132](#)
- rtklib_rtkcmn.h, [1133](#)
 - Rx, [1136](#)
 - Ry, [1136](#)
 - Rz, [1137](#)
- rtklib_rtkpos.h, [1137](#)

- rtklib_rtksvr.h, [1139](#)
 - PRCOPT_DEFAULT, [1141](#)
 - SOLOPT_DEFAULT, [1141](#)
- rtklib_sbas.h, [1141](#)
 - IGPBAND1, [1143](#)
 - IGPBAND2, [1143](#)
- rtklib_solution.h, [1144](#)
- rtklib_solver.h, [1146](#)
- rtklib_solver_dump_reader.h, [1147](#)
- rtklib_stream.h, [1147](#)
- rtklib_tides.h, [1150](#)
- rtksvr_t, [698](#)
- Rtl_Tcp_Dongle_Info, [699](#)
- rtl_tcp_command
 - rtl_tcp_commands.h, [1152](#)
- rtl_tcp_commands.h, [1151](#)
 - RTL_TCP_COMMAND, [1151](#)
 - rtl_tcp_command, [1152](#)
- rtl_tcp_dongle_info.h, [1152](#)
- rtl_tcp_signal_source.h, [1153](#)
- rtl_tcp_signal_source_c, [700](#)
- rtl_tcp_signal_source_c.h, [1154](#)
- RtlTcpSignalSource, [701](#)
 - implementation, [701](#)
- run
 - ControlThread, [157](#)
- run_acquisition
 - Fpga_Acquisition, [186](#)
- run_server
 - Rtcm, [683](#)
- Rx
 - rtklib_rtkcmn.h, [1136](#)
- Ry
 - rtklib_rtkcmn.h, [1136](#)
- Rz
 - rtklib_rtkcmn.h, [1137](#)
- SC2RAD
 - MATH_CONSTANTS.h, [1032](#)
- SERIBUFFSIZE
 - rtklib.h, [1105](#)
- SOLF_ENU
 - rtklib.h, [1106](#)
- SOLF_GSIF
 - rtklib.h, [1106](#)
- SOLF_LLH
 - rtklib.h, [1106](#)
- SOLF_NMEA
 - rtklib.h, [1106](#)
- SOLF_STAT
 - rtklib.h, [1106](#)
- SOLF_XYZ
 - rtklib.h, [1107](#)
- SOLOPT_DEFAULT
 - rtklib_rtksvr.h, [1141](#)
- SOLQ_DGPS
 - rtklib.h, [1107](#)
- SOLQ_DR
 - rtklib.h, [1107](#)
- SOLQ_FIX
 - rtklib.h, [1107](#)
- SOLQ_FLOAT
 - rtklib.h, [1107](#)
- SOLQ_NONE
 - rtklib.h, [1108](#)
- SOLQ_PPP
 - rtklib.h, [1108](#)
- SOLQ_SBAS
 - rtklib.h, [1108](#)
- SOLQ_SINGLE
 - rtklib.h, [1108](#)
- SPEED_OF_LIGHT_M_MS
 - MATH_CONSTANTS.h, [1032](#)
- SPEED_OF_LIGHT_M_S
 - MATH_CONSTANTS.h, [1032](#)
- SSRUDINT
 - rtklib_rtc3.h, [1132](#)
- SWAP_LAMBDA
 - rtklib_lambda.h, [1117](#)
- SWAP_D
 - rtklib_ppp.h, [1123](#)
- SWAP_I
 - rtklib_ppp.h, [1123](#)
- SYS_ALL
 - rtklib.h, [1108](#)
- SYS_BDS
 - rtklib.h, [1109](#)
- SYS_GAL
 - rtklib.h, [1109](#)
- SYS_GLO
 - rtklib.h, [1109](#)
- SYS_GPS
 - rtklib.h, [1109](#)
- SYS_IRN
 - rtklib.h, [1109](#)
- SYS_LEO
 - rtklib.h, [1110](#)
- SYS_NONE
 - rtklib.h, [1110](#)
- SYS_QZS
 - rtklib.h, [1110](#)
- SYS_SBS
 - rtklib.h, [1110](#)
- satelliteBlock
 - Beidou_Dnav_Ephemeris, [106](#)
 - Gps_Ephemeris, [461](#)
- satellitePosition
 - Beidou_Dnav_Ephemeris, [94](#)
 - Beidou_Dnav_Navigation_Message, [113](#)
 - Galileo_Ephemeris, [212](#)
 - Gps_CNAV_Ephemeris, [426](#)
 - Gps_Ephemeris, [449](#)
- save_cnav_ephemeris_map_xml
 - Gnss_Sdr_Supl_Client, [389](#)
- save_cnav_utc_xml
 - Gnss_Sdr_Supl_Client, [389](#)
- save_ephemeris_map_xml

- Gnss_Sdr_Supl_Client, 389
- save_gal_almanac_xml
 - Gnss_Sdr_Supl_Client, 389
- save_gal_ephemeris_map_xml
 - Gnss_Sdr_Supl_Client, 390
- save_gal_iono_xml
 - Gnss_Sdr_Supl_Client, 390
- save_gal_utc_xml
 - Gnss_Sdr_Supl_Client, 390
- save_glo_utc_xml
 - Gnss_Sdr_Supl_Client, 390
- save_gnav_ephemeris_map_xml
 - Gnss_Sdr_Supl_Client, 390
- save_gps_almanac_xml
 - Gnss_Sdr_Supl_Client, 391
- save_iono_xml
 - Gnss_Sdr_Supl_Client, 391
- save_ref_location_xml
 - Gnss_Sdr_Supl_Client, 391
- save_ref_time_xml
 - Gnss_Sdr_Supl_Client, 391
- save_utc_xml
 - Gnss_Sdr_Supl_Client, 391
- Sbas_Ephemeris, 702
 - b_sv_do_not_use, 703
 - d_acc, 703
 - d_af0, 703
 - d_af1, 703
 - d_pos, 703
 - d_tof, 704
 - d_vel, 704
 - i_prn, 704
 - i_sv_ura, 704
 - i_t0, 704
- sbas_ephemeris.h, 1155
- sbas_l1_telemetry_decoder.h, 1155
- sbas_l1_telemetry_decoder_gs, 705
 - general_work, 706
 - set_channel, 706
 - set_satellite, 706
- sbas_l1_telemetry_decoder_gs.h, 1156
- SbasL1TelemetryDecoder, 706
 - implementation, 707
- sbs_t, 707
- sbsFile
 - Rinex_Printer, 667
- sbsfcrr_t, 708
- sbsigp_t, 708
- sbsigpband_t, 709
- sbsion_t, 709
- sbslcorr_t, 709
- sbsmsg_t, 710
- sbssat_t, 710
- sbssatp_t, 711
- send_message
 - Rtcm, 683
- send_telemetry_msg
 - GNSSFlowgraph, 411
- seph_t, 711
- Serdes_Gnss_Synchro, 711
 - createProtobuffer, 712
 - operator=, 713
 - readProtobuffer, 713
 - Serdes_Gnss_Synchro, 712
- Serdes_Monitor_Pvt, 714
 - createProtobuffer, 715
 - operator=, 716
 - readProtobuffer, 716
 - Serdes_Monitor_Pvt, 714
- serdes_gnss_synchro.h, 1157
- serdes_monitor_pvt.h, 1158
- serial_t, 716
- serialize
 - Agnss_Ref_Location, 78
 - Agnss_Ref_Time, 79
 - Beidou_Dnav_Ephemeris, 94
 - Beidou_Dnav_Iono, 108
 - Galileo_Ephemeris, 212
 - Galileo_Iono, 225
 - Galileo_Utc_Model, 237
 - Glonass_Gnav_Almanac, 320
 - Glonass_Gnav_Ephemeris, 329
 - Glonass_Gnav_Utc_Model, 344
 - Gnss_Synchro, 398
 - Gps_CNAV_Ephemeris, 426
 - Gps_CNAV_Iono, 439
 - Gps_Ephemeris, 450
 - Gps_Iono, 463
 - Monitor_Pvt, 576
- set_DLL_BW
 - Tracking_2nd_DLL_filter, 738
- set_PLL_BW
 - Tracking_2nd_PLL_filter, 739
- set_active
 - galileo_e5a_noncoherentIQ_acquisition_caf_cc, 206
 - galileo_pcps_8ms_acquisition_cc, 230
 - pcps_acquisition, 592
 - pcps_acquisition_fine_doppler_cc, 598
 - pcps_acquisition_fpga, 603
 - pcps_assisted_acquisition_cc, 609
 - pcps_cccwsr_acquisition_cc, 613
 - pcps_opencl_acquisition_cc, 618
 - pcps_quicksync_acquisition_cc, 624
 - pcps_tong_acquisition_cc, 629
- set_alpha
 - Exponential_Smoother, 175
- set_averaging_depth
 - Pvt_Solution, 642
- set_block_exp
 - Fpga_Acquisition, 186
- set_channel
 - beidou_b1i_telemetry_decoder_gs, 86
 - beidou_b3i_telemetry_decoder_gs, 87
 - BeidouB1iDIIPIITracking, 118
 - BeidouB1iPcpsAcquisition, 121

- BeidouB3iDIIPIITracking, 125
- BeidouB3iPcpsAcquisition, 128
- dll_pll_veml_tracking_fpga, 171
- Fpga_Multicorrelator_8sc, 192
- galileo_e5a_noncoherentIQ_acquisition_caf_cc, 207
- galileo_pcps_8ms_acquisition_cc, 231
- galileo_telemetry_decoder_gs, 235
- GalileoE1DIIPIIVemlTracking, 240
- GalileoE1DIIPIIVemlTrackingFpga, 244
- GalileoE1Pcps8msAmbiguousAcquisition, 247
- GalileoE1PcpsAmbiguousAcquisition, 251
- GalileoE1PcpsAmbiguousAcquisitionFpga, 257
- GalileoE1PcpsCccwsrAmbiguousAcquisition, 262
- GalileoE1PcpsQuickSyncAmbiguousAcquisition, 266
- GalileoE1PcpsTongAmbiguousAcquisition, 270
- GalileoE1TcpConnectorTracking, 274
- GalileoE5aDIIPIITracking, 276
- GalileoE5aDIIPIITrackingFpga, 279
- GalileoE5aNoncoherentIQAcquisitionCaf, 282
- GalileoE5aPcpsAcquisition, 287
- GalileoE5aPcpsAcquisitionFpga, 293
- GalileoE5bDIIPIITracking, 299
- GalileoE5bPcpsAcquisition, 304
- GalileoE5bPcpsAcquisitionFpga, 311
- glonass_l1_ca_telemetry_decoder_gs, 350
- glonass_l2_ca_telemetry_decoder_gs, 354
- GlonassL1CaDIIPIICaidTracking, 356
- GlonassL1CaDIIPIITracking, 358
- GlonassL1CaPcpsAcquisition, 361
- GlonassL2CaDIIPIICaidTracking, 365
- GlonassL2CaDIIPIITracking, 367
- GlonassL2CaPcpsAcquisition, 370
- Gps_Navigation_Message, 475
- gps_l1_ca_telemetry_decoder_gs, 469
- gps_l2c_telemetry_decoder_gs, 470
- gps_l5_telemetry_decoder_gs, 472
- GpsL1CaDIIPIITracking, 480
- GpsL1CaDIIPIITrackingFpga, 484
- GpsL1CaDIIPIITrackingGPU, 486
- GpsL1CaKfTracking, 488
- GpsL1CaPcpsAcquisition, 491
- GpsL1CaPcpsAcquisitionFineDoppler, 495
- GpsL1CaPcpsAcquisitionFpga, 501
- GpsL1CaPcpsAssistedAcquisition, 505
- GpsL1CaPcpsOpenCIAcquisition, 509
- GpsL1CaPcpsQuickSyncAcquisition, 513
- GpsL1CaPcpsTongAcquisition, 517
- GpsL1CaTcpConnectorTracking, 521
- GpsL2MDIIPIITracking, 525
- GpsL2MDIIPIITrackingFpga, 527
- GpsL2MPcpsAcquisition, 530
- GpsL2MPcpsAcquisitionFpga, 534
- GpsL5DIIPIITracking, 538
- GpsL5DIIPIITrackingFpga, 541
- GpsL5iPcpsAcquisition, 545
- GpsL5iPcpsAcquisitionFpga, 551
- pcps_acquisition, 593
- pcps_acquisition_fine_doppler_cc, 598
- pcps_acquisition_fpga, 603
- pcps_assisted_acquisition_cc, 609
- pcps_cccwsr_acquisition_cc, 614
- pcps_opencl_acquisition_cc, 619
- pcps_quicksync_acquisition_cc, 624
- pcps_tong_acquisition_cc, 629
- sbas_l1_telemetry_decoder_gs, 706
- set_channel_fsm
 - BeidouB1iPcpsAcquisition, 121
 - BeidouB3iPcpsAcquisition, 128
 - galileo_e5a_noncoherentIQ_acquisition_caf_cc, 207
 - galileo_pcps_8ms_acquisition_cc, 231
 - GalileoE1Pcps8msAmbiguousAcquisition, 247
 - GalileoE1PcpsAmbiguousAcquisition, 251
 - GalileoE1PcpsAmbiguousAcquisitionFpga, 258
 - GalileoE1PcpsCccwsrAmbiguousAcquisition, 262
 - GalileoE1PcpsQuickSyncAmbiguousAcquisition, 266
 - GalileoE1PcpsTongAmbiguousAcquisition, 270
 - GalileoE5aNoncoherentIQAcquisitionCaf, 283
 - GalileoE5aPcpsAcquisition, 287
 - GalileoE5aPcpsAcquisitionFpga, 293
 - GalileoE5bPcpsAcquisition, 304
 - GalileoE5bPcpsAcquisitionFpga, 311
 - GlonassL1CaPcpsAcquisition, 361
 - GlonassL2CaPcpsAcquisition, 370
 - GpsL1CaPcpsAcquisition, 491
 - GpsL1CaPcpsAcquisitionFineDoppler, 495
 - GpsL1CaPcpsAcquisitionFpga, 501
 - GpsL1CaPcpsAssistedAcquisition, 506
 - GpsL1CaPcpsOpenCIAcquisition, 509
 - GpsL1CaPcpsQuickSyncAcquisition, 513
 - GpsL1CaPcpsTongAcquisition, 518
 - GpsL2MPcpsAcquisition, 530
 - GpsL2MPcpsAcquisitionFpga, 534
 - GpsL5iPcpsAcquisition, 545
 - GpsL5iPcpsAcquisitionFpga, 551
- pcps_acquisition, 593
- pcps_acquisition_fine_doppler_cc, 598
- pcps_acquisition_fpga, 605
- pcps_assisted_acquisition_cc, 610
- pcps_cccwsr_acquisition_cc, 614
- pcps_opencl_acquisition_cc, 619
- pcps_quicksync_acquisition_cc, 625
- pcps_tong_acquisition_cc, 630
- set_clock_drift_ppm
 - Pvt_Solution, 643
- set_configuration
 - FrontEndCal, 197
 - GNSSFlowgraph, 412
- set_control_queue
 - ControlThread, 157
- set_course_over_ground
 - Pvt_Solution, 643
- set_doppler_center

- GalileoE1PcpsAmbiguousAcquisition, [252](#)
- GalileoE1PcpsAmbiguousAcquisitionFpga, [258](#)
- GalileoE5aPcpsAcquisition, [287](#)
- GalileoE5aPcpsAcquisitionFpga, [294](#)
- GalileoE5bPcpsAcquisition, [304](#)
- GalileoE5bPcpsAcquisitionFpga, [311](#)
- GpsL1CaPcpsAcquisition, [491](#)
- GpsL1CaPcpsAcquisitionFpga, [502](#)
- GpsL2MPcpsAcquisition, [530](#)
- GpsL5iPcpsAcquisition, [545](#)
- GpsL5iPcpsAcquisitionFpga, [552](#)
- pcps_acquisition, [593](#)
- pcps_acquisition_fpga, [605](#)
- set_doppler_max
 - BeidouB1iPcpsAcquisition, [121](#)
 - BeidouB3iPcpsAcquisition, [128](#)
 - Fpga_Acquisition, [186](#)
 - galileo_e5a_noncoherentIQ_acquisition_caf_cc, [207](#)
 - galileo_pcps_8ms_acquisition_cc, [231](#)
 - GalileoE1Pcps8msAmbiguousAcquisition, [248](#)
 - GalileoE1PcpsAmbiguousAcquisition, [252](#)
 - GalileoE1PcpsAmbiguousAcquisitionFpga, [258](#)
 - GalileoE1PcpsCccwsrAmbiguousAcquisition, [262](#)
 - GalileoE1PcpsQuickSyncAmbiguousAcquisition, [266](#)
 - GalileoE1PcpsTongAmbiguousAcquisition, [271](#)
 - GalileoE5aNoncoherentIQAcquisitionCaf, [283](#)
 - GalileoE5aPcpsAcquisition, [287](#)
 - GalileoE5aPcpsAcquisitionFpga, [294](#)
 - GalileoE5bPcpsAcquisition, [305](#)
 - GalileoE5bPcpsAcquisitionFpga, [311](#)
 - GlonassL1CaPcpsAcquisition, [361](#)
 - GlonassL2CaPcpsAcquisition, [370](#)
 - GpsL1CaPcpsAcquisition, [491](#)
 - GpsL1CaPcpsAcquisitionFineDoppler, [496](#)
 - GpsL1CaPcpsAcquisitionFpga, [502](#)
 - GpsL1CaPcpsAssistedAcquisition, [506](#)
 - GpsL1CaPcpsOpenCIAcquisition, [510](#)
 - GpsL1CaPcpsQuickSyncAcquisition, [514](#)
 - GpsL1CaPcpsTongAcquisition, [518](#)
 - GpsL2MPcpsAcquisition, [530](#)
 - GpsL2MPcpsAcquisitionFpga, [535](#)
 - GpsL5iPcpsAcquisition, [545](#)
 - GpsL5iPcpsAcquisitionFpga, [552](#)
 - pcps_acquisition, [594](#)
 - pcps_acquisition_fine_doppler_cc, [600](#)
 - pcps_acquisition_fpga, [605](#)
 - pcps_assisted_acquisition_cc, [610](#)
 - pcps_cccwsr_acquisition_cc, [614](#)
 - pcps_openc1_acquisition_cc, [619](#)
 - pcps_quicksync_acquisition_cc, [625](#)
 - pcps_tong_acquisition_cc, [630](#)
- set_doppler_step
 - BeidouB1iPcpsAcquisition, [121](#)
 - BeidouB3iPcpsAcquisition, [129](#)
 - Fpga_Acquisition, [186](#)
 - galileo_e5a_noncoherentIQ_acquisition_caf_cc, [207](#)
 - galileo_pcps_8ms_acquisition_cc, [231](#)
 - GalileoE1Pcps8msAmbiguousAcquisition, [248](#)
 - GalileoE1PcpsAmbiguousAcquisition, [252](#)
 - GalileoE1PcpsAmbiguousAcquisitionFpga, [258](#)
 - GalileoE1PcpsCccwsrAmbiguousAcquisition, [263](#)
 - GalileoE1PcpsQuickSyncAmbiguousAcquisition, [267](#)
 - GalileoE1PcpsTongAmbiguousAcquisition, [271](#)
 - GalileoE1TcpConnectorTracking, [274](#)
 - GalileoE5aDIIPIITracking, [276](#)
- galileo_e5a_noncoherentIQ_acquisition_caf_cc, [207](#)
- galileo_pcps_8ms_acquisition_cc, [231](#)
- GalileoE1Pcps8msAmbiguousAcquisition, [248](#)
- GalileoE1PcpsAmbiguousAcquisition, [252](#)
- GalileoE1PcpsAmbiguousAcquisitionFpga, [258](#)
- GalileoE1PcpsCccwsrAmbiguousAcquisition, [263](#)
- GalileoE1PcpsQuickSyncAmbiguousAcquisition, [267](#)
- GalileoE1PcpsTongAmbiguousAcquisition, [271](#)
- GalileoE5aNoncoherentIQAcquisitionCaf, [283](#)
- GalileoE5aPcpsAcquisition, [287](#)
- GalileoE5aPcpsAcquisitionFpga, [294](#)
- GalileoE5bPcpsAcquisition, [305](#)
- GalileoE5bPcpsAcquisitionFpga, [312](#)
- GlonassL1CaPcpsAcquisition, [362](#)
- GlonassL2CaPcpsAcquisition, [371](#)
- GpsL1CaPcpsAcquisition, [492](#)
- GpsL1CaPcpsAcquisitionFineDoppler, [496](#)
- GpsL1CaPcpsAcquisitionFpga, [502](#)
- GpsL1CaPcpsAssistedAcquisition, [506](#)
- GpsL1CaPcpsOpenCIAcquisition, [510](#)
- GpsL1CaPcpsQuickSyncAcquisition, [514](#)
- GpsL1CaPcpsTongAcquisition, [518](#)
- GpsL2MPcpsAcquisition, [530](#)
- GpsL2MPcpsAcquisitionFpga, [535](#)
- GpsL5iPcpsAcquisition, [545](#)
- GpsL5iPcpsAcquisitionFpga, [552](#)
- pcps_acquisition, [594](#)
- pcps_acquisition_fine_doppler_cc, [600](#)
- pcps_acquisition_fpga, [605](#)
- pcps_assisted_acquisition_cc, [610](#)
- pcps_cccwsr_acquisition_cc, [614](#)
- pcps_openc1_acquisition_cc, [619](#)
- pcps_quicksync_acquisition_cc, [625](#)
- pcps_tong_acquisition_cc, [630](#)
- set_doppler_sweep
 - Fpga_Acquisition, [187](#)
- set_gnss_synchro
 - BeidouB1iDIIPIITracking, [118](#)
 - BeidouB1iPcpsAcquisition, [122](#)
 - BeidouB3iDIIPIITracking, [125](#)
 - BeidouB3iPcpsAcquisition, [129](#)
 - dll_pll_veml_tracking_fpga, [171](#)
 - galileo_e5a_noncoherentIQ_acquisition_caf_cc, [208](#)
 - galileo_pcps_8ms_acquisition_cc, [232](#)
 - GalileoE1DIIPIIVemlTracking, [240](#)
 - GalileoE1DIIPIIVemlTrackingFpga, [244](#)
 - GalileoE1Pcps8msAmbiguousAcquisition, [248](#)
 - GalileoE1PcpsAmbiguousAcquisition, [252](#)
 - GalileoE1PcpsAmbiguousAcquisitionFpga, [258](#)
 - GalileoE1PcpsCccwsrAmbiguousAcquisition, [263](#)
 - GalileoE1PcpsQuickSyncAmbiguousAcquisition, [267](#)
 - GalileoE1PcpsTongAmbiguousAcquisition, [271](#)
 - GalileoE1TcpConnectorTracking, [274](#)
 - GalileoE5aDIIPIITracking, [276](#)

- GalileoE5aDIIPIITrackingFpga, [280](#)
- GalileoE5aNoncoherentIQAcquisitionCaf, [283](#)
- GalileoE5aPcpsAcquisition, [288](#)
- GalileoE5aPcpsAcquisitionFpga, [294](#)
- GalileoE5bDIIPIITracking, [299](#)
- GalileoE5bPcpsAcquisition, [305](#)
- GalileoE5bPcpsAcquisitionFpga, [312](#)
- GlionassL1CaDIIPIICAidTracking, [356](#)
- GlionassL1CaDIIPIITracking, [358](#)
- GlionassL1CaPcpsAcquisition, [362](#)
- GlionassL2CaDIIPIICAidTracking, [365](#)
- GlionassL2CaDIIPIITracking, [367](#)
- GlionassL2CaPcpsAcquisition, [371](#)
- GpsL1CaDIIPIITracking, [480](#)
- GpsL1CaDIIPIITrackingFpga, [484](#)
- GpsL1CaDIIPIITrackingGPU, [486](#)
- GpsL1CaKfTracking, [488](#)
- GpsL1CaPcpsAcquisition, [492](#)
- GpsL1CaPcpsAcquisitionFineDoppler, [496](#)
- GpsL1CaPcpsAcquisitionFpga, [502](#)
- GpsL1CaPcpsAssistedAcquisition, [506](#)
- GpsL1CaPcpsOpenCIAcquisition, [510](#)
- GpsL1CaPcpsQuickSyncAcquisition, [514](#)
- GpsL1CaPcpsTongAcquisition, [518](#)
- GpsL1CaTcpConnectorTracking, [521](#)
- GpsL2MDIIPIITracking, [525](#)
- GpsL2MDIIPIITrackingFpga, [527](#)
- GpsL2MPcpsAcquisition, [531](#)
- GpsL2MPcpsAcquisitionFpga, [535](#)
- GpsL5DIIPIITracking, [538](#)
- GpsL5DIIPIITrackingFpga, [542](#)
- GpsL5iPcpsAcquisition, [546](#)
- GpsL5iPcpsAcquisitionFpga, [552](#)
- pcps_acquisition, [594](#)
- pcps_acquisition_fine_doppler_cc, [600](#)
- pcps_acquisition_fpga, [606](#)
- pcps_assisted_acquisition_cc, [610](#)
- pcps_cccwsr_acquisition_cc, [615](#)
- pcps_openc1_acquisition_cc, [620](#)
- pcps_quicksync_acquisition_cc, [626](#)
- pcps_tong_acquisition_cc, [630](#)
- set_initial_sample
 - Fpga_Multicorrelator_8sc, [192](#)
- set_local_code
 - BeidouB1iPcpsAcquisition, [122](#)
 - BeidouB3iPcpsAcquisition, [129](#)
 - Fpga_Acquisition, [187](#)
 - galileo_e5a_noncoherentIQ_acquisition_caf_cc, [208](#)
 - galileo_pcps_8ms_acquisition_cc, [232](#)
 - GalileoE1Pcps8msAmbiguousAcquisition, [248](#)
 - GalileoE1PcpsAmbiguousAcquisition, [252](#)
 - GalileoE1PcpsAmbiguousAcquisitionFpga, [259](#)
 - GalileoE1PcpsQuickSyncAmbiguousAcquisition, [267](#)
 - GalileoE1PcpsTongAmbiguousAcquisition, [271](#)
 - GalileoE5aNoncoherentIQAcquisitionCaf, [284](#)
 - GalileoE5aPcpsAcquisition, [288](#)
 - GalileoE5aPcpsAcquisitionFpga, [294](#)
 - GalileoE5bPcpsAcquisition, [305](#)
 - GalileoE5bPcpsAcquisitionFpga, [312](#)
 - GlionassL1CaPcpsAcquisition, [362](#)
 - GlionassL2CaPcpsAcquisition, [371](#)
 - GpsL1CaPcpsAcquisition, [492](#)
 - GpsL1CaPcpsAcquisitionFpga, [502](#)
 - GpsL1CaPcpsOpenCIAcquisition, [510](#)
 - GpsL1CaPcpsQuickSyncAcquisition, [514](#)
 - GpsL1CaPcpsTongAcquisition, [519](#)
 - GpsL2MPcpsAcquisition, [531](#)
 - GpsL2MPcpsAcquisitionFpga, [535](#)
 - GpsL5iPcpsAcquisition, [546](#)
 - GpsL5iPcpsAcquisitionFpga, [552](#)
 - pcps_acquisition, [594](#)
 - pcps_acquisition_fine_doppler_cc, [600](#)
 - pcps_acquisition_fpga, [606](#)
 - pcps_assisted_acquisition_cc, [611](#)
 - pcps_cccwsr_acquisition_cc, [615](#)
 - pcps_openc1_acquisition_cc, [620](#)
 - pcps_quicksync_acquisition_cc, [626](#)
 - pcps_tong_acquisition_cc, [631](#)
 - set_local_code_and_taps
 - Fpga_Multicorrelator_8sc, [192](#)
 - set_num_valid_observations
 - Pvt_Solution, [643](#)
 - set_output_vectors
 - Fpga_Multicorrelator_8sc, [192](#)
 - set_pdi
 - Tracking_2nd_DLL_filter, [738](#)
 - Tracking_2nd_PLL_filter, [739](#)
 - set_pre_2009_file
 - Pvt_Solution, [643](#)
 - set_resampler_latency
 - BeidouB1iPcpsAcquisition, [122](#)
 - BeidouB3iPcpsAcquisition, [129](#)
 - GalileoE1PcpsAmbiguousAcquisition, [253](#)
 - GalileoE1PcpsAmbiguousAcquisitionFpga, [259](#)
 - GalileoE5aPcpsAcquisition, [288](#)
 - GalileoE5aPcpsAcquisitionFpga, [295](#)
 - GalileoE5bPcpsAcquisition, [305](#)
 - GalileoE5bPcpsAcquisitionFpga, [312](#)
 - GpsL1CaPcpsAcquisition, [492](#)
 - GpsL1CaPcpsAcquisitionFpga, [503](#)
 - GpsL2MPcpsAcquisition, [531](#)
 - GpsL5iPcpsAcquisition, [546](#)
 - GpsL5iPcpsAcquisitionFpga, [553](#)
 - set_rx_pos
 - Pvt_Solution, [643](#)
 - set_rx_vel
 - Pvt_Solution, [643](#)
 - set_samples_for_initialization
 - Exponential_Smoother, [175](#)
 - set_satellite
 - beidou_b1i_telemetry_decoder_gs, [86](#)
 - beidou_b3i_telemetry_decoder_gs, [88](#)
 - galileo_telemetry_decoder_gs, [235](#)
 - glonass_l1_ca_telemetry_decoder_gs, [350](#)

- glonass_l2_ca_telemetry_decoder_gs, 354
- gps_l1_ca_telemetry_decoder_gs, 469
- gps_l2c_telemetry_decoder_gs, 471
- gps_l5_telemetry_decoder_gs, 472
- sbas_l1_telemetry_decoder_gs, 706
- set_satellite_PRN
 - Beidou_Dnav_Navigation_Message, 113
 - Gps_Navigation_Message, 475
- set_secondary_code_lengths
 - Fpga_Multicorrelator_8sc, 193
- set_signal
 - Channel, 136
- set_single_doppler_flag
 - GalileoE5aPcpsAcquisitionFpga, 295
 - GalileoE5bPcpsAcquisitionFpga, 312
- set_speed_over_ground
 - Pvt_Solution, 644
- set_state
 - BeidouB1iPcpsAcquisition, 122
 - BeidouB3iPcpsAcquisition, 129
 - galileo_e5a_noncoherentIQ_acquisition_caf_cc, 208
 - galileo_pcps_8ms_acquisition_cc, 232
 - GalileoE1PcpsAmbiguousAcquisition, 253
 - GalileoE1PcpsAmbiguousAcquisitionFpga, 259
 - GalileoE1PcpsCccwsrAmbiguousAcquisition, 263
 - GalileoE1PcpsQuickSyncAmbiguousAcquisition, 267
 - GalileoE1PcpsTongAmbiguousAcquisition, 272
 - GalileoE5aNoncoherentIQAcquisitionCaf, 284
 - GalileoE5aPcpsAcquisition, 288
 - GalileoE5aPcpsAcquisitionFpga, 295
 - GalileoE5bPcpsAcquisition, 306
 - GalileoE5bPcpsAcquisitionFpga, 313
 - GlonassL1CaPcpsAcquisition, 362
 - GlonassL2CaPcpsAcquisition, 371
 - GpsL1CaPcpsAcquisition, 492
 - GpsL1CaPcpsAcquisitionFineDoppler, 496
 - GpsL1CaPcpsAcquisitionFpga, 503
 - GpsL1CaPcpsQuickSyncAcquisition, 515
 - GpsL1CaPcpsTongAcquisition, 519
 - GpsL2MPcpsAcquisition, 531
 - GpsL2MPcpsAcquisitionFpga, 536
 - GpsL5iPcpsAcquisition, 546
 - GpsL5iPcpsAcquisitionFpga, 553
 - pcps_acquisition, 595
 - pcps_acquisition_fine_doppler_cc, 601
 - pcps_acquisition_fpga, 606
 - pcps_cccwsr_acquisition_cc, 615
 - pcps_opencl_acquisition_cc, 620
 - pcps_quicksync_acquisition_cc, 626
 - pcps_tong_acquisition_cc, 631
- set_switch_position
 - Fpga_Switch, 194
- set_threshold
 - BeidouB1iPcpsAcquisition, 122
 - BeidouB3iPcpsAcquisition, 130
 - galileo_e5a_noncoherentIQ_acquisition_caf_cc, 209
 - galileo_pcps_8ms_acquisition_cc, 234
 - GalileoE1Pcps8msAmbiguousAcquisition, 249
 - GalileoE1PcpsAmbiguousAcquisition, 253
 - GalileoE1PcpsAmbiguousAcquisitionFpga, 259
 - GalileoE1PcpsCccwsrAmbiguousAcquisition, 263
 - GalileoE1PcpsQuickSyncAmbiguousAcquisition, 267
 - GalileoE1PcpsTongAmbiguousAcquisition, 272
 - GalileoE5aNoncoherentIQAcquisitionCaf, 284
 - GalileoE5aPcpsAcquisition, 289
 - GalileoE5aPcpsAcquisitionFpga, 295
 - GalileoE5bPcpsAcquisition, 306
 - GalileoE5bPcpsAcquisitionFpga, 313
 - GlonassL1CaPcpsAcquisition, 362
 - GlonassL2CaPcpsAcquisition, 371
 - GpsL1CaPcpsAcquisition, 493
 - GpsL1CaPcpsAcquisitionFineDoppler, 497
 - GpsL1CaPcpsAcquisitionFpga, 503
 - GpsL1CaPcpsAssistedAcquisition, 507
 - GpsL1CaPcpsOpenCIAcquisition, 511
 - GpsL1CaPcpsQuickSyncAcquisition, 515
 - GpsL1CaPcpsTongAcquisition, 519
 - GpsL2MPcpsAcquisition, 531
 - GpsL2MPcpsAcquisitionFpga, 536
 - GpsL5iPcpsAcquisition, 546
 - GpsL5iPcpsAcquisitionFpga, 553
 - pcps_acquisition, 595
 - pcps_acquisition_fine_doppler_cc, 601
 - pcps_acquisition_fpga, 607
 - pcps_assisted_acquisition_cc, 611
 - pcps_cccwsr_acquisition_cc, 616
 - pcps_opencl_acquisition_cc, 622
 - pcps_quicksync_acquisition_cc, 627
 - pcps_tong_acquisition_cc, 631
- set_time_offset_s
 - Pvt_Solution, 644
- short_x2_to_cshort, 717
- short_x2_to_cshort.h, 1158
- Signal
 - Gnss_Synchro, 404
 - signal_conditioner.h, 1159
 - signal_generator.h, 1160
 - signal_generator_c, 718
 - signal_make_generator_c, 718
 - signal_generator_c.h, 1160
 - signal_make_generator_c, 1161
 - signal_generator_flags.h, 1162
 - signal_make_generator_c
 - signal_generator_c, 718
 - signal_generator_c.h, 1161
- SignalConditioner, 719
 - ~SignalConditioner, 720
 - implementation, 720
 - SignalConditioner, 720
- SignalGenerator, 721
 - implementation, 721

- size
 - Gnss_circular_deque, [378](#)
- Skew_symmetric
 - geofunctions.h, [882](#)
- snrmask_t, [722](#)
- sol_t, [722](#)
- solbuf_t, [722](#)
- solopt_t, [723](#)
- solstat_t, [723](#)
- solstatbuf_t, [724](#)
- spir_file_signal_source.h, [1163](#)
- spir_gss6450_file_signal_source.h, [1163](#)
- SpirFileSignalSource, [726](#)
 - implementation, [726](#)
- SpirGSS6450FileSignalSource, [727](#)
- Spirent_Motion_Csv_Dump_Reader, [724](#)
- spirent_motion_csv_dump_reader.h, [1164](#)
- ssat_t, [728](#)
- ssr_t, [728](#)
- sta_t, [729](#)
- start
 - GNSSFlowgraph, [412](#)
- start_acquisition
 - Channel, [136](#)
- start_tracking
 - dll_pll_veml_tracking_fpga, [171](#)
 - GalileoE1DIIPIIVemlTrackingFpga, [244](#)
 - GalileoE5aDIIPIITrackingFpga, [280](#)
 - GpsL1CaDIIPIITrackingFpga, [484](#)
 - GpsL5DIIPIITrackingFpga, [542](#)
- stec_t, [729](#)
- stop
 - GNSSFlowgraph, [412](#)
- stop_acquisition
 - BeidouB1iPcpsAcquisition, [123](#)
 - BeidouB3iPcpsAcquisition, [130](#)
 - GalileoE1Pcps8msAmbiguousAcquisition, [249](#)
 - GalileoE1PcpsAmbiguousAcquisition, [253](#)
 - GalileoE1PcpsAmbiguousAcquisitionFpga, [259](#)
 - GalileoE1PcpsCccwsrAmbiguousAcquisition, [263](#)
 - GalileoE1PcpsQuickSyncAmbiguousAcquisition, [268](#)
 - GalileoE1PcpsTongAmbiguousAcquisition, [272](#)
 - GalileoE5aNoncoherentIQAquisitionCaf, [284](#)
 - GalileoE5aPcpsAcquisition, [289](#)
 - GalileoE5aPcpsAcquisitionFpga, [296](#)
 - GalileoE5bPcpsAcquisition, [306](#)
 - GalileoE5bPcpsAcquisitionFpga, [313](#)
 - GlonassL1CaPcpsAcquisition, [363](#)
 - GlonassL2CaPcpsAcquisition, [372](#)
 - GpsL1CaPcpsAcquisition, [493](#)
 - GpsL1CaPcpsAcquisitionFineDoppler, [497](#)
 - GpsL1CaPcpsAcquisitionFpga, [503](#)
 - GpsL1CaPcpsAssistedAcquisition, [507](#)
 - GpsL1CaPcpsOpenCIAcquisition, [511](#)
 - GpsL1CaPcpsQuickSyncAcquisition, [515](#)
 - GpsL1CaPcpsTongAcquisition, [519](#)
 - GpsL2MPcpsAcquisition, [532](#)
 - GpsL2MPcpsAcquisitionFpga, [536](#)
 - GpsL5iPcpsAcquisition, [547](#)
 - GpsL5iPcpsAcquisitionFpga, [553](#)
- stop_channel
 - Channel, [136](#)
- stop_server
 - Rtcm, [683](#)
- stop_tracking
 - BeidouB1iDIIPIITracking, [118](#)
 - BeidouB3iDIIPIITracking, [125](#)
 - dll_pll_veml_tracking_fpga, [172](#)
 - GalileoE1DIIPIIVemlTracking, [240](#)
 - GalileoE1DIIPIIVemlTrackingFpga, [245](#)
 - GalileoE1TcpConnectorTracking, [274](#)
 - GalileoE5aDIIPIITracking, [276](#)
 - GalileoE5aDIIPIITrackingFpga, [280](#)
 - GalileoE5bDIIPIITracking, [300](#)
 - GlonassL1CaDIIPIICAidTracking, [356](#)
 - GlonassL1CaDIIPIITracking, [358](#)
 - GlonassL2CaDIIPIICAidTracking, [366](#)
 - GlonassL2CaDIIPIITracking, [367](#)
 - GpsL1CaDIIPIITracking, [480](#)
 - GpsL1CaDIIPIITrackingFpga, [485](#)
 - GpsL1CaDIIPIITrackingGPU, [486](#)
 - GpsL1CaKfTracking, [488](#)
 - GpsL1CaTcpConnectorTracking, [521](#)
 - GpsL2MDIIPIITracking, [525](#)
 - GpsL2MDIIPIITrackingFpga, [527](#)
 - GpsL5DIIPIITracking, [538](#)
 - GpsL5DIIPIITrackingFpga, [542](#)
- stream_cfg, [730](#)
- stream_t, [730](#)
- string_converter.h, [1165](#)
- string_decoder
 - Glonass_Gnav_Navigation_Message, [342](#)
- StringConverter, [731](#)
- subframe_decoder
 - Gps_Navigation_Message, [475](#)
- sv_clock_correction
 - Beidou_Dnav_Navigation_Message, [113](#)
- sv_clock_drift
 - Beidou_Dnav_Ephemeris, [96](#)
 - Galileo_Ephemeris, [212](#)
 - Glonass_Gnav_Ephemeris, [330](#)
 - Gps_CNAV_Ephemeris, [428](#)
 - Gps_Ephemeris, [451](#)
- sv_clock_relativistic_term
 - Beidou_Dnav_Ephemeris, [96](#)
 - Galileo_Ephemeris, [212](#)
 - Gps_CNAV_Ephemeris, [428](#)
 - Gps_Ephemeris, [451](#)
- swift_common.h, [1166](#)
- symbols
 - cnav_v27_part_t, [149](#)
- System
 - Gnss_Synchro, [404](#)
- t0c_4
 - Galileo_Ephemeris, [221](#)

t0e_1
 Galileo_Ephemeris, [221](#)
 t0t_6
 Galileo_Utc_Model, [237](#)
 T_OA
 GPS_L1_CA.h, [968](#)
 TIMES_GPST
 rtklib.h, [1110](#)
 TIMES_JST
 rtklib.h, [1111](#)
 TIMES_UTC
 rtklib.h, [1111](#)
 TIMETAGH_LEN
 rtklib.h, [1111](#)
 TINTACT
 rtklib.h, [1111](#)
 TOW_5
 Galileo_Ephemeris, [221](#)
 Galileo_Iono, [227](#)
 TOW_at_current_symbol_ms
 Gnss_Synchro, [404](#)
 TROPOPT_CORG
 rtklib.h, [1112](#)
 TROPOPT_COR
 rtklib.h, [1111](#)
 TROPOPT_ESTG
 rtklib.h, [1112](#)
 TROPOPT_EST
 rtklib.h, [1112](#)
 TROPOPT_OFF
 rtklib.h, [1112](#)
 TROPOPT_SAAS
 rtklib.h, [1112](#)
 TROPOPT_SBAS
 rtklib.h, [1113](#)
 TWO_N10
 MATH_CONSTANTS.h, [1032](#)
 TWO_N11
 MATH_CONSTANTS.h, [1032](#)
 TWO_N14
 MATH_CONSTANTS.h, [1033](#)
 TWO_N15
 MATH_CONSTANTS.h, [1033](#)
 TWO_N16
 MATH_CONSTANTS.h, [1033](#)
 TWO_N17
 MATH_CONSTANTS.h, [1033](#)
 TWO_N18
 MATH_CONSTANTS.h, [1033](#)
 TWO_N19
 MATH_CONSTANTS.h, [1034](#)
 TWO_N2
 MATH_CONSTANTS.h, [1034](#)
 TWO_N20
 MATH_CONSTANTS.h, [1034](#)
 TWO_N21
 MATH_CONSTANTS.h, [1034](#)
 TWO_N23
 MATH_CONSTANTS.h, [1034](#)
 TWO_N24
 MATH_CONSTANTS.h, [1035](#)
 TWO_N25
 MATH_CONSTANTS.h, [1035](#)
 TWO_N27
 MATH_CONSTANTS.h, [1035](#)
 TWO_N29
 MATH_CONSTANTS.h, [1035](#)
 TWO_N30
 MATH_CONSTANTS.h, [1035](#)
 TWO_N31
 MATH_CONSTANTS.h, [1036](#)
 TWO_N32
 MATH_CONSTANTS.h, [1036](#)
 TWO_N33
 MATH_CONSTANTS.h, [1036](#)
 TWO_N34
 MATH_CONSTANTS.h, [1036](#)
 TWO_N35
 MATH_CONSTANTS.h, [1036](#)
 TWO_N38
 MATH_CONSTANTS.h, [1037](#)
 TWO_N39
 MATH_CONSTANTS.h, [1037](#)
 TWO_N40
 MATH_CONSTANTS.h, [1037](#)
 TWO_N43
 MATH_CONSTANTS.h, [1037](#)
 TWO_N44
 MATH_CONSTANTS.h, [1037](#)
 TWO_N46
 MATH_CONSTANTS.h, [1038](#)
 TWO_N48
 MATH_CONSTANTS.h, [1038](#)
 TWO_N5
 MATH_CONSTANTS.h, [1038](#)
 TWO_N50
 MATH_CONSTANTS.h, [1038](#)
 TWO_N51
 MATH_CONSTANTS.h, [1038](#)
 TWO_N55
 MATH_CONSTANTS.h, [1039](#)
 TWO_N57
 MATH_CONSTANTS.h, [1039](#)
 TWO_N59
 MATH_CONSTANTS.h, [1039](#)
 TWO_N6
 MATH_CONSTANTS.h, [1039](#)
 TWO_N60
 MATH_CONSTANTS.h, [1039](#)
 TWO_N66
 MATH_CONSTANTS.h, [1040](#)
 TWO_N68
 MATH_CONSTANTS.h, [1040](#)
 TWO_N8
 MATH_CONSTANTS.h, [1040](#)
 TWO_N9

- MATH_CONSTANTS.h, [1040](#)
- TWO_P11
 - MATH_CONSTANTS.h, [1040](#)
- TWO_P12
 - MATH_CONSTANTS.h, [1041](#)
- TWO_P14
 - MATH_CONSTANTS.h, [1041](#)
- TWO_P16
 - MATH_CONSTANTS.h, [1041](#)
- TWO_P19
 - MATH_CONSTANTS.h, [1041](#)
- TWO_P3
 - MATH_CONSTANTS.h, [1041](#)
- TWO_P31
 - MATH_CONSTANTS.h, [1042](#)
- TWO_P32
 - MATH_CONSTANTS.h, [1042](#)
- TWO_P4
 - MATH_CONSTANTS.h, [1042](#)
- TWO_P56
 - MATH_CONSTANTS.h, [1042](#)
- TWO_P57
 - MATH_CONSTANTS.h, [1042](#)
- TWO_PI
 - MATH_CONSTANTS.h, [1043](#)
- Tcp_Communication, [731](#)
- Tcp_Packet_Data, [732](#)
- tcp_cmd_interface.h, [1166](#)
- tcp_communication.h, [1167](#)
- tcp_packet_data.h, [1168](#)
- tcp_t, [733](#)
- TcpCmdInterface, [733](#)
 - get_LLH, [734](#)
 - get_utc_time, [734](#)
- tcpcli_t, [733](#)
- tcpsvr_t, [734](#)
- tec_t, [735](#)
- telemetry_decoder_interface.h, [1168](#)
- TelemetryDecoderInterface, [735](#)
- test_flags.h, [1169](#)
- tle_t, [736](#)
- tled_t, [736](#)
- Tlm_Dump_Reader, [737](#)
- tlm_dump_reader.h, [1170](#)
- to_date_time
 - Rinex_Printer, [666](#)
- togeo
 - geofunctions.h, [882](#)
- topocent
 - geofunctions.h, [883](#)
- tow
 - cnav_msg_t, [146](#)
- Tracking_2nd_DLL_filter, [737](#)
 - get_code_nco, [738](#)
 - initialize, [738](#)
 - set_DLL_BW, [738](#)
 - set_pdi, [738](#)
- tracking_2nd_DLL_filter.h, [1170](#)
- Tracking_2nd_PLL_filter, [739](#)
 - set_PLL_BW, [739](#)
 - set_pdi, [739](#)
- tracking_2nd_PLL_filter.h, [1171](#)
- Tracking_Dump_Reader, [740](#)
- Tracking_FLL_PLL_filter, [740](#)
- tracking_FLL_PLL_filter.h, [1175](#)
- Tracking_True_Obs_Reader, [742](#)
- tracking_discriminators.h, [1171](#)
 - dll_nc_e_minus_l_normalized, [1172](#)
 - dll_nc_vemlp_normalized, [1173](#)
 - fil_four_quadrant_atan, [1173](#)
 - phase_unwrap, [1173](#)
 - pll_cloop_two_quadrant_atan, [1174](#)
 - pll_four_quadrant_atan, [1174](#)
- tracking_dump_reader.h, [1174](#)
- tracking_interface.h, [1175](#)
- Tracking_loop_filter, [741](#)
 - operator=, [742](#)
 - Tracking_loop_filter, [742](#)
- tracking_loop_filter.h, [1176](#)
- Tracking_sample_counter
 - Gnss_Synchro, [404](#)
- tracking_tests_flags.h, [1177](#)
- tracking_true_obs_reader.h, [1178](#)
- TrackingInterface, [743](#)
- trop_t, [744](#)
- True_Observables_Reader, [744](#)
- true_observables_reader.h, [1179](#)
- two_bit_cpx_file_signal_source.h, [1179](#)
- two_bit_packed_file_signal_source.h, [1180](#)
- TwoBitCpxFileSignalSource, [745](#)
 - implementation, [745](#)
- TwoBitPackedFileSignalSource, [746](#)
 - implementation, [747](#)
- uhd_signal_source.h, [1181](#)
- UhdSignalSource, [747](#)
 - implementation, [748](#)
- unlock_channel
 - Fpga_Multicorrelator_8sc, [193](#)
- unpack_2bit_samples, [748](#)
- unpack_2bit_samples.h, [1182](#)
- unpack_byte_2bit_cpx_samples, [749](#)
- unpack_byte_2bit_cpx_samples.h, [1183](#)
- unpack_byte_2bit_samples, [750](#)
- unpack_byte_2bit_samples.h, [1184](#)
- unpack_byte_4bit_samples, [750](#)
- unpack_byte_4bit_samples.h, [1184](#)
- unpack_intspir_1bit_samples, [751](#)
- unpack_intspir_1bit_samples.h, [1185](#)
- unpack_spir_gss6450_samples, [752](#)
- unpack_spir_gss6450_samples.h, [1186](#)
- UnscentedFilter, [752](#)
- unset_title
 - Gnuplot, [415](#)
- update_PRN
 - Gnss_Satellite, [382](#)
- update_local_code

- Fpga_Multicorrelator_8sc, [193](#)
- update_nav_header
 - Rinex_Printer, [666](#)
- update_prn_code_length
 - Fpga_Multicorrelator_8sc, [193](#)
- url_t, [753](#)
- utc_time
 - Beidou_Dnav_Navigation_Message, [114](#)
 - Glionass_Gnav_Utc_Model, [344](#)
 - Gps_Navigation_Message, [476](#)
- v27_decision_t, [753](#)
- v27_poly_t, [753](#)
- v27_t, [754](#)
- valid
 - Beidou_Dnav_Iono, [110](#)
 - Gps_CNAV_Iono, [441](#)
 - Gps_Iono, [465](#)
- Viterbi
 - convolutional.h, [808](#)
- Viterbi_Decoder, [754](#)
 - decode_block, [755](#)
- viterbi_decoder.h, [1187](#)
- WN_5
 - Galileo_Ephemeris, [221](#)
 - Galileo_Iono, [228](#)
- WNot_6
 - Galileo_Utc_Model, [237](#)
- wait
 - GNSSFlowgraph, [412](#)
- what_block
 - Gnss_Satellite, [382](#)
- work
 - rtklib_pvt_gs, [694](#)
- write_local_code
 - Fpga_Acquisition, [187](#)