

IndoorNaviHardware

Generated by Doxygen 1.8.14

Contents

1	Readme file	1
2	Deprecated List	3
3	Data Structure Index	5
3.1	Data Structures	5
4	File Index	7
4.1	File List	7
5	Data Structure Documentation	9
5.1	agc_cfg_struct Struct Reference	9
5.1.1	Field Documentation	9
5.1.1.1	lo32	9
5.1.1.2	target	9
5.2	carry_instance_t Struct Reference	10
5.2.1	Detailed Description	10
5.2.2	Field Documentation	10
5.2.2.1	isConnectedToServer	10
5.2.2.2	target	11
5.2.2.3	toSinkId	11
5.3	carry_settings_t Struct Reference	11
5.3.1	Detailed Description	11
5.3.2	Field Documentation	11
5.3.2.1	trace_max_fail_cnt	11

5.3.2.2	trace_max_inactive_time	12
5.4	carry_target_t Struct Reference	12
5.4.1	Detailed Description	12
5.4.2	Field Documentation	12
5.4.2.1	addr	13
5.4.2.2	last_update_time	13
5.4.2.3	trace	13
5.5	carry_trace_t Struct Reference	13
5.5.1	Detailed Description	13
5.5.2	Field Documentation	14
5.5.2.1	fail_cnt	14
5.5.2.2	last_update_time	14
5.5.2.3	pass_cnt	14
5.5.2.4	path	14
5.5.2.5	path_len	14
5.6	dwt_cb_data_t Struct Reference	15
5.6.1	Field Documentation	15
5.6.1.1	dataLength	15
5.6.1.2	fctrl	15
5.6.1.3	rx_flags	15
5.6.1.4	status	15
5.7	dwt_config_t Struct Reference	15
5.7.1	Detailed Description	16
5.7.2	Field Documentation	16
5.7.2.1	chan	16
5.7.2.2	dataRate	16
5.7.2.3	nsSFD	17
5.7.2.4	phrMode	17
5.7.2.5	prf	17
5.7.2.6	rxCode	17

5.7.2.7	rxPAC	17
5.7.2.8	sfdTO	17
5.7.2.9	txCode	18
5.7.2.10	txPreambLength	18
5.8	dwt_devicecnts_t Struct Reference	18
5.8.1	Field Documentation	18
5.8.1.1	ARFE	18
5.8.1.2	CRCB	19
5.8.1.3	CRCG	19
5.8.1.4	HPW	19
5.8.1.5	OVER	19
5.8.1.6	PHE	19
5.8.1.7	PTO	19
5.8.1.8	RSL	19
5.8.1.9	RTO	19
5.8.1.10	SFDTO	20
5.8.1.11	TXF	20
5.8.1.12	TXW	20
5.9	dwt_local_data_t Struct Reference	20
5.9.1	Detailed Description	21
5.9.2	Field Documentation	21
5.9.2.1	cbData	21
5.9.2.2	cbRxErr	21
5.9.2.3	cbRxOk	21
5.9.2.4	cbRxTo	22
5.9.2.5	cbTxDone	22
5.9.2.6	dblbuffer	22
5.9.2.7	init_xtrim	22
5.9.2.8	longFrames	22
5.9.2.9	lotID	22

5.9.2.10	otprev	22
5.9.2.11	partID	22
5.9.2.12	sleep_mode	23
5.9.2.13	sysCFGreg	23
5.9.2.14	txFCTRL	23
5.9.2.15	wait4resp	23
5.10	dwt_rxdiag_t Struct Reference	23
5.10.1	Field Documentation	23
5.10.1.1	firstPath	24
5.10.1.2	firstPathAmp1	24
5.10.1.3	firstPathAmp2	24
5.10.1.4	firstPathAmp3	24
5.10.1.5	maxGrowthCIR	24
5.10.1.6	maxNoise	24
5.10.1.7	rxPreamCount	24
5.10.1.8	stdNoise	25
5.11	dwt_txconfig_t Struct Reference	25
5.11.1	Field Documentation	25
5.11.1.1	PGdly	25
5.11.1.2	power	25
5.12	FC_BEACON_s Struct Reference	25
5.12.1	Detailed Description	26
5.12.2	Field Documentation	26
5.12.2.1	FC	26
5.12.2.2	len	26
5.12.2.3	serial	26
5.13	FC_CARRY_s Struct Reference	26
5.13.1	Detailed Description	27
5.13.2	Field Documentation	27
5.13.2.1	flag_hops	27

5.13.2.2 hops	27
5.13.2.3 src_addr	27
5.14 FC_STAT_s Struct Reference	28
5.14.1 Detailed Description	28
5.14.2 Field Documentation	28
5.14.2.1 battery_mV	28
5.14.2.2 err_cnt	28
5.14.2.3 FC	29
5.14.2.4 len	29
5.14.2.5 rx_cnt	29
5.14.2.6 to_cnt	29
5.14.2.7 tx_cnt	29
5.15 FC_SYNC_FIN_s Struct Reference	29
5.15.1 Field Documentation	30
5.15.1.1 FC	30
5.15.1.2 len	30
5.15.1.3 slot_num	30
5.15.1.4 tree_level	30
5.15.1.5 TsFinTxBuf	31
5.15.1.6 TsOffset	31
5.15.1.7 TsPollTx	31
5.15.1.8 TsRespRx	31
5.16 FC_SYNC_POLL_s Struct Reference	31
5.16.1 Field Documentation	32
5.16.1.1 FC	32
5.16.1.2 len	32
5.16.1.3 num_poll_anchor	32
5.16.1.4 poll_addr	32
5.17 FC_SYNC_RESP_s Struct Reference	32
5.17.1 Field Documentation	33

5.17.1.1	FC	33
5.17.1.2	len	33
5.17.1.3	TsPollRx	33
5.17.1.4	TsRespTx	33
5.18	FC_TURN_OFF_s Struct Reference	33
5.18.1	Detailed Description	34
5.18.2	Field Documentation	34
5.18.2.1	FC	34
5.18.2.2	len	34
5.18.2.3	reason	34
5.19	FC_TURN_ON_s Struct Reference	34
5.19.1	Detailed Description	34
5.19.2	Field Documentation	35
5.19.2.1	FC	35
5.19.2.2	len	35
5.20	FC_VERSION_s Struct Reference	35
5.20.1	Detailed Description	35
5.20.2	Field Documentation	36
5.20.2.1	FC	36
5.20.2.2	fMajor	36
5.20.2.3	fMinor	36
5.20.2.4	hash	36
5.20.2.5	hMajor	36
5.20.2.6	hMinor	36
5.20.2.7	len	37
5.20.2.8	role	37
5.20.2.9	serial	37
5.21	FU_instance_t Struct Reference	37
5.21.1	Field Documentation	37
5.21.1.1	fileSize	37

5.21.1.2 newCrc	38
5.21.1.3 newHash	38
5.21.1.4 newVer	38
5.21.1.5 sesionPacketCounter	38
5.22 FU_prot Struct Reference	38
5.22.1 Detailed Description	39
5.22.2 Field Documentation	39
5.22.2.1 data	39
5.22.2.2 extra	39
5.22.2.3 FC	39
5.22.2.4 frameLen	39
5.22.2.5 hash	39
5.22.2.6 opcode	40
5.23 FU_SOT_prot Struct Reference	40
5.23.1 Detailed Description	40
5.23.2 Field Documentation	40
5.23.2.1 FC	41
5.23.2.2 fileSize	41
5.23.2.3 firmwareCRC	41
5.23.2.4 frameCRC	41
5.23.2.5 frameLen	41
5.23.2.6 fversion	41
5.23.2.7 hash	42
5.23.2.8 opcode	42
5.24 mac_buf_t Struct Reference	42
5.24.1 Field Documentation	42
5.24.1.1 "@1	43
5.24.1.2 buf	43
5.24.1.3 control	43
5.24.1.4 data	43

5.24.1.5 dPtr	43
5.24.1.6 dst	43
5.24.1.7 frame	43
5.24.1.8 isRangingFrame	44
5.24.1.9 last_update_time	44
5.24.1.10 pan	44
5.24.1.11 retransmit_fail_cnt	44
5.24.1.12 rx_len	44
5.24.1.13 seq_num	44
5.24.1.14 src	44
5.24.1.15 state	45
5.25 mac_instance_t Struct Reference	45
5.25.1 Field Documentation	45
5.25.1.1 beacon_timer_timestamp	46
5.25.1.2 buf	46
5.25.1.3 buf_get_ind	46
5.25.1.4 frame_under_tx_is_ranging	46
5.25.1.5 last_rx_ts	46
5.25.1.6 rx_buf	46
5.25.1.7 seq_num	46
5.25.1.8 slot_number	46
5.25.1.9 slot_time_offset	47
5.26 mac_settings_t Struct Reference	47
5.26.1 Field Documentation	48
5.26.1.1 addr	48
5.26.1.2 max_buf_inactive_time	48
5.26.1.3 max_frame_fail_cnt	48
5.26.1.4 pan	48
5.26.1.5 rapport_anchor_anchor_distance	48
5.26.1.6 role	49

5.26.1.7 slot_guard_time_us	49
5.26.1.8 slot_time_us	49
5.26.1.9 slots_sum_time_us	49
5.26.1.10 sync_dly	49
5.27 prot_cb_t Struct Reference	50
5.27.1 Detailed Description	50
5.27.2 Field Documentation	50
5.27.2.1 cb	50
5.27.2.2 FC	51
5.28 prot_packet_info_t Struct Reference	51
5.28.1 Detailed Description	51
5.28.2 Field Documentation	51
5.28.2.1 carry	52
5.28.2.2 direct_src	52
5.29 settings_otp_t Struct Reference	52
5.29.1 Detailed Description	52
5.29.2 Field Documentation	52
5.29.2.1 h_major	53
5.29.2.2 h_minor	53
5.29.2.3 serial	53
5.30 settings_t Struct Reference	53
5.30.1 Detailed Description	54
5.30.2 Field Documentation	54
5.30.2.1 carry	54
5.30.2.2 mac	54
5.30.2.3 transceiver	54
5.30.2.4 version	54
5.31 settings_version_t Struct Reference	55
5.31.1 Detailed Description	55
5.31.2 Field Documentation	55

5.31.2.1	boot_reserved	55
5.31.2.2	f_hash	55
5.31.2.3	f_major	56
5.31.2.4	f_minor	56
5.31.2.5	h_version	56
5.32	sync_instance_t Struct Reference	56
5.32.1	Field Documentation	57
5.32.1.1	local_obj	57
5.32.1.2	neighbour	57
5.32.1.3	toa	57
5.32.1.4	toa_ts_poll_rx_raw	57
5.32.1.5	tree_level	58
5.33	sync_neighbour_t Struct Reference	58
5.33.1	Field Documentation	58
5.33.1.1	addr	58
5.33.1.2	drift	59
5.33.1.3	sync_ready	59
5.33.1.4	time_coeffP	59
5.33.1.5	time_coeffP_raw	59
5.33.1.6	time_drift_sum	59
5.33.1.7	time_offset	59
5.33.1.8	tof_dw	60
5.33.1.9	tree_level	60
5.33.1.10	update_ts	60
5.34	toa_core_t Struct Reference	60
5.34.1	Field Documentation	61
5.34.1.1	addr_tab	61
5.34.1.2	anc_in_poll_cnt	61
5.34.1.3	initiator	61
5.34.1.4	prev_state	61

5.34.1.5 resp_ind	61
5.34.1.6 state	61
5.34.1.7 TsFinRx	62
5.34.1.8 TsFinTx	62
5.34.1.9 TsPollRx	62
5.34.1.10 TsPollTx	62
5.34.1.11 TsRespRx	62
5.34.1.12 TsRespTx	62
5.35 toa_settings_t Struct Reference	62
5.35.1 Field Documentation	63
5.35.1.1 fin_dly_us	63
5.35.1.2 guard_time_us	63
5.35.1.3 resp_dly_us	63
5.35.1.4 rx_after_tx_offset_us	63
5.36 transceiver_settings_t Struct Reference	64
5.36.1 Detailed Description	64
5.36.2 Field Documentation	64
5.36.2.1 ant_dly_rx	64
5.36.2.2 ant_dly_tx	65
5.36.2.3 dwt_config	65
5.36.2.4 dwt_txconfig	65
5.36.2.5 low_power_mode	65
5.36.2.6 set_pac_from_settings	65
5.37 txt_buf_t Struct Reference	65
5.37.1 Detailed Description	66
5.37.2 Field Documentation	66
5.37.2.1 cmd	66
5.37.2.2 cnt	66
5.37.2.3 end	66
5.37.2.4 start	66
5.38 txt_cb_t Struct Reference	66
5.38.1 Detailed Description	67
5.38.2 Field Documentation	67
5.38.2.1 cb	67
5.38.2.2 cmd	67

6 File Documentation	69
6.1 base64.c File Reference	69
6.1.1 Function Documentation	69
6.1.1.1 BASE64_Decode()	70
6.1.1.2 BASE64_Encode()	70
6.1.1.3 BASE64_TextSize()	71
6.2 base64.h File Reference	71
6.2.1 Detailed Description	72
6.2.2 Function Documentation	72
6.2.2.1 BASE64_Decode()	72
6.2.2.2 BASE64_Encode()	72
6.2.2.3 BASE64_TextSize()	73
6.3 bin_const.h File Reference	73
6.3.1 Detailed Description	74
6.3.2 Enumeration Type Documentation	74
6.3.2.1 FC_t	74
6.4 bin_parser.c File Reference	75
6.4.1 Function Documentation	76
6.4.1.1 BIN_Parse()	76
6.4.1.2 BIN_ParseSingle()	77
6.5 bin_parser.h File Reference	77
6.5.1 Detailed Description	79
6.5.2 Macro Definition Documentation	79
6.5.2.1 BIN_ASSERT	79
6.5.3 Typedef Documentation	79
6.5.3.1 prot_parser_cb	79
6.5.4 Function Documentation	80
6.5.4.1 BIN_Parse()	80
6.5.4.2 BIN_ParseSingle()	81
6.6 bin_parser_cb.c File Reference	81

6.6.1	Function Documentation	82
6.6.1.1	BIN_SEND_RESP()	82
6.6.1.2	FC_BEACON_cb()	83
6.6.1.3	FC_STAT_ASK_cb()	84
6.6.1.4	FC_TURN_ON_cb()	84
6.6.1.5	FC_VERSION_ASK_cb()	84
6.6.2	Variable Documentation	84
6.6.2.1	prot_cb_len	85
6.6.2.2	prot_cb_tab	85
6.7	bin_struct.h File Reference	85
6.7.1	Detailed Description	86
6.8	carry.c File Reference	87
6.8.1	Function Documentation	87
6.8.1.1	_CARRY_FindTarget()	88
6.8.1.2	_CARRY_FindTrace()	88
6.8.1.3	_CARRY_ProtFill()	88
6.8.1.4	CARRY_Init()	89
6.8.1.5	CARRY_ParseMessage()	89
6.8.1.6	CARRY_PrepBufTo()	90
6.8.1.7	CARRY_WriteTrace()	91
6.8.2	Variable Documentation	91
6.8.2.1	carry	92
6.9	carry.h File Reference	92
6.9.1	Detailed Description	94
6.9.2	Macro Definition Documentation	94
6.9.2.1	CARRY_HEAD_MIN_LEN	94
6.9.3	Function Documentation	94
6.9.3.1	CARRY_GetBufTo()	94
6.9.3.2	CARRY_Init()	95
6.9.3.3	CARRY_ParseMessage()	95

6.9.3.4	CARRY_PrepBufTo()	96
6.9.3.5	CARRY_WriteTrace()	97
6.10	carry_const.h File Reference	98
6.10.1	Macro Definition Documentation	98
6.10.1.1	CARRY_FLAG_ACK_REQ	98
6.10.1.2	CARRY_FLAG_REROUTE	98
6.10.1.3	CARRY_FLAG_TARGET_DEV	99
6.10.1.4	CARRY_FLAG_TARGET_MASK	99
6.10.1.5	CARRY_FLAG_TARGET_SERVER	99
6.10.1.6	CARRY_FLAG_TARGET_SINK	99
6.10.1.7	CARRY_HOPS_NUM_MASK	99
6.10.1.8	CARRY_MAX_HOPS	99
6.11	carry_settings.h File Reference	100
6.11.1	Detailed Description	101
6.11.2	Macro Definition Documentation	102
6.11.2.1	CARRY_ASSERT	102
6.11.2.2	CARRY_MAX_TARGETS	102
6.11.2.3	CARRY_MAX_TRACE	102
6.11.2.4	CARRY_SETTINGS_DEF	102
6.12	deca_device.c File Reference	103
6.12.1	Detailed Description	106
6.12.2	Macro Definition Documentation	106
6.12.2.1	B20_SIGN_EXTEND_MASK	106
6.12.2.2	B20_SIGN_EXTEND_TEST	107
6.12.2.3	ENABLE_ALL_SEQ	107
6.12.2.4	FCTRL_ACK_REQ_MASK	107
6.12.2.5	FCTRL_LEN_MAX	107
6.12.2.6	FORCE_LDE	107
6.12.2.7	FORCE OTP OFF	107
6.12.2.8	FORCE OTP ON	107

6.12.2.9 FORCE_SYS_PLL	107
6.12.2.10 FORCE_SYS_XTI	108
6.12.2.11 FORCE_TX_PLL	108
6.12.2.12 LDOTUNE_ADDRESS	108
6.12.2.13 LOTID_ADDRESS	108
6.12.2.14 PARTID_ADDRESS	108
6.12.2.15 READ_ACC_OFF	108
6.12.2.16 READ_ACC_ON	108
6.12.2.17 VBAT_ADDRESS	108
6.12.2.18 VTEMP_ADDRESS	109
6.12.2.19 XTRIM_ADDRESS	109
6.12.3 Function Documentation	109
6.12.3.1 _dwt_aonarrayupload()	109
6.12.3.2 _dwt_aonconfigupload()	110
6.12.3.3 _dwt_computetxpowersetting()	110
6.12.3.4 _dwt_configlde()	111
6.12.3.5 _dwt_disablesequencing()	111
6.12.3.6 _dwt_enableclocks()	112
6.12.3.7 _dwt_loaducodefromrom()	112
6.12.3.8 _dwt_otpprogword32()	113
6.12.3.9 _dwt_otpread()	114
6.12.3.10 _dwt_otpsetmrregs()	114
6.12.3.11 dwt_calcbandwidthtempadj()	115
6.12.3.12 dwt_calcpgcount()	115
6.12.3.13 dwt_calcpowertempadj()	116
6.12.3.14 dwt_calibratesleepcnt()	116
6.12.3.15 dwt_checkirq()	117
6.12.3.16 dwt_configcontinuousframemode()	117
6.12.3.17 dwt_configcwmode()	117
6.12.3.18 dwt_configeventcounters()	118

6.12.3.19 dwt_configure()	118
6.12.3.20 dwt_configuresleep()	118
6.12.3.21 dwt_configuresleepcnt()	119
6.12.3.22 dwt_configuretxrf()	119
6.12.3.23 dwt_enableautoack()	119
6.12.3.24 dwt_enableframefilter()	120
6.12.3.25 dwt_entersleep()	120
6.12.3.26 dwt_entersleepaftertx()	120
6.12.3.27 dwt_forcetxoff()	121
6.12.3.28 dwt_geteui()	122
6.12.3.29 dwt_getinitxtaltrim()	122
6.12.3.30 dwt_getlotid()	122
6.12.3.31 dwt_getpartid()	123
6.12.3.32 dwt_initialise()	123
6.12.3.33 dwt_isr()	123
6.12.3.34 dwt_loadopsettabfromotp()	124
6.12.3.35 dwt_lowpowerlistenisr()	124
6.12.3.36 dwt_otpread()	124
6.12.3.37 dwt_otprevision()	125
6.12.3.38 dwt_otpwriteandverify()	125
6.12.3.39 dwt_read16bitoffsetreg()	125
6.12.3.40 dwt_read32bitoffsetreg()	126
6.12.3.41 dwt_read8bitoffsetreg()	127
6.12.3.42 dwt_readaccdata()	127
6.12.3.43 dwt_readcarrierintegrator()	128
6.12.3.44 dwt_readdevid()	128
6.12.3.45 dwt_readdiagnostics()	129
6.12.3.46 dwt_readeventcounters()	129
6.12.3.47 dwt_readfromdevice()	130
6.12.3.48 dwt_readrxdata()	131

6.12.3.49 <code>dwt_readrxtimestamp()</code>	132
6.12.3.50 <code>dwt_readrxtimestamphi32()</code>	133
6.12.3.51 <code>dwt_readrxtimestamplo32()</code>	133
6.12.3.52 <code>dwt_readsystime()</code>	133
6.12.3.53 <code>dwt_readsystimestamphi32()</code>	134
6.12.3.54 <code>dwt_readtempvbat()</code>	134
6.12.3.55 <code>dwt_readtxtimestamp()</code>	135
6.12.3.56 <code>dwt_readtxtimestamphi32()</code>	135
6.12.3.57 <code>dwt_readtxtimestamplo32()</code>	136
6.12.3.58 <code>dwt_readwakeuptemp()</code>	136
6.12.3.59 <code>dwt_readwakeupvbat()</code>	136
6.12.3.60 <code>dwt_rxenable()</code>	137
6.12.3.61 <code>dwt_rxreset()</code>	137
6.12.3.62 <code>dwt_setaddress16()</code>	138
6.12.3.63 <code>dwt_setcallbacks()</code>	138
6.12.3.64 <code>dwt_setdblrxbuffmode()</code>	139
6.12.3.65 <code>dwt_setdelayedtrxtime()</code>	139
6.12.3.66 <code>dwt_seteui()</code>	140
6.12.3.67 <code>dwt_setfinegrainxseq()</code>	140
6.12.3.68 <code>dwt_setgpiodirection()</code>	141
6.12.3.69 <code>dwt_setgpiovalue()</code>	141
6.12.3.70 <code>dwt_setinterrupt()</code>	141
6.12.3.71 <code>dwt_setleds()</code>	142
6.12.3.72 <code>dwt_setlnapamode()</code>	142
6.12.3.73 <code>dwt_setlocaldataptr()</code>	143
6.12.3.74 <code>dwt_setlowpowerlistening()</code>	143
6.12.3.75 <code>dwt_setpanid()</code>	144
6.12.3.76 <code>dwt_setpreambledetecttimeout()</code>	144
6.12.3.77 <code>dwt_setraftertxdelay()</code>	145
6.12.3.78 <code>dwt_setrxantennadelay()</code>	145

6.12.3.79 <code>dwt_setrxtimeout()</code>	146
6.12.3.80 <code>dwt_setsmarttxpower()</code>	146
6.12.3.81 <code>dwt_setsniffmode()</code>	146
6.12.3.82 <code>dwt_setsnoozetime()</code>	147
6.12.3.83 <code>dwt_setxantennadelay()</code>	148
6.12.3.84 <code>dwt_setxtaltrim()</code>	148
6.12.3.85 <code>dwt_softreset()</code>	149
6.12.3.86 <code>dwt_spicswakeups()</code>	149
6.12.3.87 <code>dwt_starttx()</code>	150
6.12.3.88 <code>dwt_syncrxbufptrs()</code>	150
6.12.3.89 <code>dwt_write16bitoffsetreg()</code>	151
6.12.3.90 <code>dwt_write32bitoffsetreg()</code>	152
6.12.3.91 <code>dwt_write8bitoffsetreg()</code>	153
6.12.3.92 <code>dwt_writetodevice()</code>	154
6.12.3.93 <code>dwt_writetxdata()</code>	155
6.12.3.94 <code>dwt_writetxfctrl()</code>	156
6.13 <code>deca_device_api.h</code> File Reference	156
6.13.1 Detailed Description	162
6.13.2 Macro Definition Documentation	162
6.13.2.1 <code>_DECA_INT16_</code>	162
6.13.2.2 <code>_DECA_INT32_</code>	162
6.13.2.3 <code>_DECA_INT8_</code>	162
6.13.2.4 <code>_DECA_UINT16_</code>	162
6.13.2.5 <code>_DECA_UINT32_</code>	162
6.13.2.6 <code>_DECA_UINT8_</code>	163
6.13.2.7 <code>DWT_BR_110K</code>	163
6.13.2.8 <code>DWT_BR_6M8</code>	163
6.13.2.9 <code>DWT_BR_850K</code>	163
6.13.2.10 <code>DWT_CB_DATA_RX_FLAG_RNG</code>	163
6.13.2.11 <code>DWT_CONFIG</code>	163

6.13.2.12 DWT_DEVICE_ID	163
6.13.2.13 DWT_ERROR	164
6.13.2.14 DWT_FF_ACK_EN	164
6.13.2.15 DWT_FF_BEACON_EN	164
6.13.2.16 DWT_FF_COORD_EN	164
6.13.2.17 DWT_FF_DATA_EN	164
6.13.2.18 DWT_FF_MAC_EN	164
6.13.2.19 DWT_FF_NOTYPE_EN	164
6.13.2.20 DWT_FF_RSVD_EN	164
6.13.2.21 DWT_IDLE_ON_DLY_ERR	165
6.13.2.22 DWT_INT_ARFE	165
6.13.2.23 DWT_INT_LDED	165
6.13.2.24 DWT_INT_RFCE	165
6.13.2.25 DWT_INT_RFCG	165
6.13.2.26 DWT_INT_RFSL	165
6.13.2.27 DWT_INT_RFTO	165
6.13.2.28 DWT_INT_RPHE	165
6.13.2.29 DWT_INT_RXOVRR	166
6.13.2.30 DWT_INT_RXPTO	166
6.13.2.31 DWT_INT_SFDT	166
6.13.2.32 DWT_INT_TFRS	166
6.13.2.33 DWT_LEDS_DISABLE	166
6.13.2.34 DWT_LEDS_ENABLE	166
6.13.2.35 DWT_LEDS_INIT_BLINK	166
6.13.2.36 DWT_LOADEUI	166
6.13.2.37 DWT_LOADNONE	167
6.13.2.38 DWT_LOADOPSET	167
6.13.2.39 DWT_LOADUCODE	167
6.13.2.40 DWT_NO_SYNC_PTRS	167
6.13.2.41 DWT_NUM_DW_DEV	167

6.13.2.42 DWT_OPSET_64LEN	167
6.13.2.43 DWT_OPSET_DEFLT	167
6.13.2.44 DWT_OPSET_TIGHT	167
6.13.2.45 DWT_PAC16	168
6.13.2.46 DWT_PAC32	168
6.13.2.47 DWT_PAC64	168
6.13.2.48 DWT_PAC8	168
6.13.2.49 DWT_PHRMODE_EXT	168
6.13.2.50 DWT_PHRMODE_STD	168
6.13.2.51 DWT_PLEN_1024	168
6.13.2.52 DWT_PLEN_128	169
6.13.2.53 DWT_PLEN_1536	169
6.13.2.54 DWT_PLEN_2048	169
6.13.2.55 DWT_PLEN_256	169
6.13.2.56 DWT_PLEN_4096	169
6.13.2.57 DWT_PLEN_512	169
6.13.2.58 DWT_PLEN_64	169
6.13.2.59 DWT_PRESRV_SLEEP	170
6.13.2.60 DWT_PRF_16M	170
6.13.2.61 DWT_PRF_64M	170
6.13.2.62 dwt_read32bitreg	170
6.13.2.63 DWT_RESPONSE_EXPECTED	170
6.13.2.64 DWT_RX_EN	170
6.13.2.65 DWT_SFDTOC_DEF	170
6.13.2.66 DWT_SLP_EN	171
6.13.2.67 DWT_START_RX_DELAYED	171
6.13.2.68 DWT_START_RX_IMMEDIATE	171
6.13.2.69 DWT_START_TX_DELAYED	171
6.13.2.70 DWT_START_TX_IMMEDIATE	171
6.13.2.71 DWT_SUCCESS	171

6.13.2.72 DWT_TANDV	171
6.13.2.73 DWT_TIME_UNITS	172
6.13.2.74 DWT_WAKE_CS	172
6.13.2.75 DWT_WAKE_SLPCNT	172
6.13.2.76 DWT_WAKE_WK	172
6.13.2.77 dwt_write32bitreg	172
6.13.2.78 DWT_XTAL_EN	172
6.13.2.79 FREQ_OFFSET_MULTIPLIER	172
6.13.2.80 FREQ_OFFSET_MULTIPLIER_110KB	173
6.13.2.81 HERTZ_TO_PPM_MULTIPLIER_CHAN_1	173
6.13.2.82 HERTZ_TO_PPM_MULTIPLIER_CHAN_2	173
6.13.2.83 HERTZ_TO_PPM_MULTIPLIER_CHAN_3	173
6.13.2.84 HERTZ_TO_PPM_MULTIPLIER_CHAN_5	173
6.13.3 Typedef Documentation	173
6.13.3.1 decalrqStatus_t	173
6.13.3.2 dwt_cb_t	173
6.13.3.3 int16	174
6.13.3.4 int32	174
6.13.3.5 int8	174
6.13.3.6 uint16	174
6.13.3.7 uint32	174
6.13.3.8 uint8	174
6.13.4 Function Documentation	174
6.13.4.1 deca_sleep()	175
6.13.4.2 decamutexoff()	175
6.13.4.3 decamutexon()	176
6.13.4.4 dwt_calcbandwidthtempadj()	177
6.13.4.5 dwt_calcpgcount()	177
6.13.4.6 dwt_calcpowertempadj()	178
6.13.4.7 dwt_calibratesleepcnt()	178

6.13.4.8 <code>dwt_checkirq()</code>	179
6.13.4.9 <code>dwt_configcontinuousframemode()</code>	179
6.13.4.10 <code>dwt_configcwmode()</code>	179
6.13.4.11 <code>dwt_configeventcounters()</code>	180
6.13.4.12 <code>dwt_configure()</code>	180
6.13.4.13 <code>dwt_configuresleep()</code>	180
6.13.4.14 <code>dwt_configuresleepcnt()</code>	181
6.13.4.15 <code>dwt_configuretxrf()</code>	181
6.13.4.16 <code>dwt_enableautoack()</code>	181
6.13.4.17 <code>dwt_enableframefilter()</code>	182
6.13.4.18 <code>dwt_entersleep()</code>	182
6.13.4.19 <code>dwt_entersleepaftertx()</code>	182
6.13.4.20 <code>dwt_forcetxoff()</code>	183
6.13.4.21 <code>dwt_geteui()</code>	184
6.13.4.22 <code>dwt_getinitxtaltrim()</code>	184
6.13.4.23 <code>dwt_getlotid()</code>	184
6.13.4.24 <code>dwt_getpartid()</code>	185
6.13.4.25 <code>dwt_initialise()</code>	185
6.13.4.26 <code>dwt_isr()</code>	185
6.13.4.27 <code>dwt_loadopsettabfromotp()</code>	186
6.13.4.28 <code>dwt_lowpowerlistenisr()</code>	186
6.13.4.29 <code>dwt_otpread()</code>	186
6.13.4.30 <code>dwt_otprevision()</code>	187
6.13.4.31 <code>dwt_otpwriteandverify()</code>	187
6.13.4.32 <code>dwt_read16bitoffsetreg()</code>	187
6.13.4.33 <code>dwt_read32bitoffsetreg()</code>	188
6.13.4.34 <code>dwt_read8bitoffsetreg()</code>	189
6.13.4.35 <code>dwt_readaccdata()</code>	189
6.13.4.36 <code>dwt_readcarrierintegrator()</code>	190
6.13.4.37 <code>dwt_readdevid()</code>	190

6.13.4.38 dwt_readdiagnostics()	191
6.13.4.39 dwt_readeventcounters()	191
6.13.4.40 dwt_readfromdevice()	192
6.13.4.41 dwt_readrxdata()	193
6.13.4.42 dwt_readrxtimestamp()	194
6.13.4.43 dwt_readrxtimestamphi32()	195
6.13.4.44 dwt_readrxtimestamplo32()	195
6.13.4.45 dwt_readsystime()	195
6.13.4.46 dwt_readsystimestamphi32()	196
6.13.4.47 dwt_readtempvbat()	196
6.13.4.48 dwt_readtxtimestamp()	197
6.13.4.49 dwt_readtxtimestamphi32()	197
6.13.4.50 dwt_readtxtimestamplo32()	198
6.13.4.51 dwt_readwakeuptemp()	198
6.13.4.52 dwt_readwakeupvbat()	198
6.13.4.53 dwt_rxenable()	199
6.13.4.54 dwt_rxreset()	199
6.13.4.55 dwt_setaddress16()	200
6.13.4.56 dwt_setcallbacks()	200
6.13.4.57 dwt_setdblrxbuffmode()	201
6.13.4.58 dwt_setdelayedtrxtime()	201
6.13.4.59 dwt_seteui()	202
6.13.4.60 dwt_setfinegraintrxseq()	202
6.13.4.61 dwt_setgpiodirection()	203
6.13.4.62 dwt_setgpiovalue()	203
6.13.4.63 dwt_setinterrupt()	203
6.13.4.64 dwt_setleds()	204
6.13.4.65 dwt_setlnapemode()	204
6.13.4.66 dwt_setlocaldataaptr()	205
6.13.4.67 dwt_setlowpowerlistening()	205

6.13.4.68 dwt_setpanid()	206
6.13.4.69 dwt_setpreambledetecttimeout()	206
6.13.4.70 dwt_setrxaftertxdelay()	207
6.13.4.71 dwt_setrxantennadelay()	207
6.13.4.72 dwt_setrxtimeout()	208
6.13.4.73 dwt_setsmarttxpower()	208
6.13.4.74 dwt_setsniffmode()	208
6.13.4.75 dwt_setsnoozetime()	209
6.13.4.76 dwt_setxantennadelay()	210
6.13.4.77 dwt_setxtaltrim()	210
6.13.4.78 dwt_softreset()	211
6.13.4.79 dwt_spicswakeup()	211
6.13.4.80 dwt_starttx()	212
6.13.4.81 dwt_syncrxbufptrs()	212
6.13.4.82 dwt_write16bitoffsetreg()	213
6.13.4.83 dwt_write32bitoffsetreg()	214
6.13.4.84 dwt_write8bitoffsetreg()	215
6.13.4.85 dwt_writetodevice()	216
6.13.4.86 dwt_writetxdata()	217
6.13.4.87 dwt_writetxfctrl()	218
6.13.4.88 readfromspi()	218
6.13.4.89 writetospi()	219
6.14 deca_param_types.h File Reference	220
6.14.1 Detailed Description	222
6.14.2 Macro Definition Documentation	222
6.14.2.1 DA_ATTN_STEP	222
6.14.2.2 LDE_PARAM1	222
6.14.2.3 LDE_PARAM3_16	222
6.14.2.4 LDE_PARAM3_64	223
6.14.2.5 MIXER_GAIN_STEP	223

6.14.2.6	N_STD_FACTOR	223
6.14.2.7	NUM_BR	223
6.14.2.8	NUM_BW	223
6.14.2.9	NUM_CH	223
6.14.2.10	NUM_CH_SUPPORTED	223
6.14.2.11	NUM_PACS	223
6.14.2.12	NUM_PRF	224
6.14.2.13	NUM_SFD	224
6.14.2.14	PCODES	224
6.14.2.15	PEAK_MULTIPLIER	224
6.14.2.16	XMLPARAMS_VERSION	224
6.14.3	Variable Documentation	224
6.14.3.1	agc_config	224
6.14.3.2	chan_idx	224
6.14.3.3	digital_bb_config	225
6.14.3.4	dtune1	225
6.14.3.5	dwnsSFDlen	225
6.14.3.6	fs_pll_cfg	225
6.14.3.7	fs_pll_tune	225
6.14.3.8	lde_replicaCoeff	225
6.14.3.9	rx_config	225
6.14.3.10	sftsh	225
6.14.3.11	tx_config	226
6.14.3.12	txpwr_compensation	226
6.15	deca_params_init.c File Reference	226
6.15.1	Detailed Description	227
6.15.2	Variable Documentation	227
6.15.2.1	agc_config	227
6.15.2.2	chan_idx	227
6.15.2.3	digital_bb_config	227

6.15.2.4	dtune1	228
6.15.2.5	dwnsSFDlen	228
6.15.2.6	fs_pll_cfg	228
6.15.2.7	fs_pll_tune	228
6.15.2.8	lde_replicaCoeff	229
6.15.2.9	rx_config	229
6.15.2.10	sftsh	229
6.15.2.11	tx_config	230
6.15.2.12	txpwr_compensation	230
6.16	deca_regs.h File Reference	230
6.16.1	Detailed Description	251
6.16.2	Macro Definition Documentation	251
6.16.2.1	ACC_MEM_ID	251
6.16.2.2	ACC_MEM_LEN	251
6.16.2.3	ACK_RESP_T_ACK_TIM_MASK	251
6.16.2.4	ACK_RESP_T_ACK_TIM_OFFSET	252
6.16.2.5	ACK_RESP_T_ID	252
6.16.2.6	ACK_RESP_T_LEN	252
6.16.2.7	ACK_RESP_T_MASK	252
6.16.2.8	ACK_RESP_T_W4R_TIM_MASK	252
6.16.2.9	ACK_RESP_T_W4R_TIM_OFFSET	252
6.16.2.10	ACK_TIM_MASK	252
6.16.2.11	AGC_CFG_STS_ID	253
6.16.2.12	AGC_CTRL1_DIS_AM	253
6.16.2.13	AGC_CTRL1_LEN	253
6.16.2.14	AGC_CTRL1_MASK	253
6.16.2.15	AGC_CTRL1_OFFSET	253
6.16.2.16	AGC_CTRL_ID	253
6.16.2.17	AGC_CTRL_LEN	253
6.16.2.18	AGC_STAT1_EDG1_MASK	254

6.16.2.19 AGC_STAT1_EDG2_MASK	254
6.16.2.20 AGC_STAT1_LEN	254
6.16.2.21 AGC_STAT1_MASK	254
6.16.2.22 AGC_STAT1_OFFSET	254
6.16.2.23 AGC_TUNE1_16M	254
6.16.2.24 AGC_TUNE1_64M	254
6.16.2.25 AGC_TUNE1_LEN	255
6.16.2.26 AGC_TUNE1_MASK	255
6.16.2.27 AGC_TUNE1_OFFSET	255
6.16.2.28 AGC_TUNE2_LEN	255
6.16.2.29 AGC_TUNE2_MASK	255
6.16.2.30 AGC_TUNE2_OFFSET	255
6.16.2.31 AGC_TUNE2_VAL	255
6.16.2.32 AGC_TUNE3_LEN	255
6.16.2.33 AGC_TUNE3_MASK	256
6.16.2.34 AGC_TUNE3_OFFSET	256
6.16.2.35 AGC_TUNE3_VAL	256
6.16.2.36 AON_ADDR_LEN	256
6.16.2.37 AON_ADDR_LPOSC_CAL_0	256
6.16.2.38 AON_ADDR_LPOSC_CAL_1	256
6.16.2.39 AON_ADDR_OFFSET	256
6.16.2.40 AON_CFG0_LEN	257
6.16.2.41 AON_CFG0_LPCLKDIVA_MASK	257
6.16.2.42 AON_CFG0_LPCLKDIVA_SHIFT	257
6.16.2.43 AON_CFG0_LPDIV_EN	257
6.16.2.44 AON_CFG0_OFFSET	257
6.16.2.45 AON_CFG0_SLEEP_EN	257
6.16.2.46 AON_CFG0_SLEEP_SHIFT	257
6.16.2.47 AON_CFG0_SLEEP_TIM	258
6.16.2.48 AON_CFG0_SLEEP_TIM_OFFSET	258

6.16.2.49 AON_CFG0_WAKE_CNT	258
6.16.2.50 AON_CFG0_WAKE_PIN	258
6.16.2.51 AON_CFG0_WAKE_SPI	258
6.16.2.52 AON_CFG1_LEN	258
6.16.2.53 AON_CFG1_LPOSC_CAL	258
6.16.2.54 AON_CFG1_MASK	259
6.16.2.55 AON_CFG1_OFFSET	259
6.16.2.56 AON_CFG1_SLEEP_CEN	259
6.16.2.57 AON_CFG1_SMXX	259
6.16.2.58 AON_CTRL_DCA_ENAB	259
6.16.2.59 AON_CTRL_DCA_READ	259
6.16.2.60 AON_CTRL_LEN	259
6.16.2.61 AON_CTRL_MASK	260
6.16.2.62 AON_CTRL_OFFSET	260
6.16.2.63 AON_CTRL_RESTORE	260
6.16.2.64 AON_CTRL_SAVE	260
6.16.2.65 AON_CTRL_UPL_CFG	260
6.16.2.66 AON_ID	260
6.16.2.67 AON_LEN	260
6.16.2.68 AON_RDAT_LEN	261
6.16.2.69 AON_RDAT_OFFSET	261
6.16.2.70 AON_WCFG_LEN	261
6.16.2.71 AON_WCFG_MASK	261
6.16.2.72 AON_WCFG_OFFSET	261
6.16.2.73 AON_WCFG_ONW_L64P	261
6.16.2.74 AON_WCFG_ONW_LDC	261
6.16.2.75 AON_WCFG_ONW_LEUI	262
6.16.2.76 AON_WCFG_ONW_LLDE	262
6.16.2.77 AON_WCFG_ONW_LLDO	262
6.16.2.78 AON_WCFG_ONW_RADC	262

6.16.2.79 AON_WCFG_ONW_RX	262
6.16.2.80 AON_WCFG_PRES_SLEEP	262
6.16.2.81 BOOSTNORM_MASK	262
6.16.2.82 BOOSTP125_MASK	263
6.16.2.83 BOOSTP250_MASK	263
6.16.2.84 BOOSTP500_MASK	263
6.16.2.85 CHAN_CTRL_DWSFD	263
6.16.2.86 CHAN_CTRL_DWSFD_SHIFT	263
6.16.2.87 CHAN_CTRL_ID	263
6.16.2.88 CHAN_CTRL_LEN	263
6.16.2.89 CHAN_CTRL_MASK	264
6.16.2.90 CHAN_CTRL_RNSSFD	264
6.16.2.91 CHAN_CTRL_RNSSFD_SHIFT	264
6.16.2.92 CHAN_CTRL_RX_CHAN_MASK	264
6.16.2.93 CHAN_CTRL_RX_CHAN_SHIFT	264
6.16.2.94 CHAN_CTRL_RX_PCOD_MASK	264
6.16.2.95 CHAN_CTRL_RX_PCOD_SHIFT	264
6.16.2.96 CHAN_CTRL_RXFPRF_16	265
6.16.2.97 CHAN_CTRL_RXFPRF_4	265
6.16.2.98 CHAN_CTRL_RXFPRF_64	265
6.16.2.99 CHAN_CTRL_RXFPRF_MASK	265
6.16.2.100CHAN_CTRL_RXFPRF_SHIFT	265
6.16.2.101CHAN_CTRL_TNSSFD	265
6.16.2.102CHAN_CTRL_TNSSFD_SHIFT	265
6.16.2.103CHAN_CTRL_TX_CHAN_MASK	266
6.16.2.104CHAN_CTRL_TX_CHAN_SHIFT	266
6.16.2.105CHAN_CTRL_TX_PCOD_MASK	266
6.16.2.106CHAN_CTRL_TX_PCOD_SHIFT	266
6.16.2.107CIR_MXG_MASK	266
6.16.2.108CIR_MXG_SHIFT	266

6.16.2.109DEV_ID_ID	266
6.16.2.110DEV_ID_LEN	267
6.16.2.111DEV_ID_MODEL_MASK	267
6.16.2.112DEV_ID_REV_MASK	267
6.16.2.113DEV_ID_RIDTAG_MASK	267
6.16.2.114DEV_ID_VER_MASK	267
6.16.2.115DIAG_TMC_LEN	267
6.16.2.116DIAG_TMC_MASK	267
6.16.2.117DIAG_TMC_OFFSET	268
6.16.2.118DIAG_TMC_TX_PSTM	268
6.16.2.119DIG_DIAG_ID	268
6.16.2.120DIG_DIAG_LEN	268
6.16.2.121DRX_CARRIER_INT_LEN	268
6.16.2.122DRX_CARRIER_INT_MASK	268
6.16.2.123DRX_CARRIER_INT_OFFSET	268
6.16.2.124DRX_CONF_ID	269
6.16.2.125DRX_CONF_LEN	269
6.16.2.126DRX_PRETOC_LEN	269
6.16.2.127DRX_PRETOC_MASK	269
6.16.2.128DRX_PRETOC_OFFSET	269
6.16.2.129DRX_SFDTOC_LEN	269
6.16.2.130DRX_SFDTOC_MASK	269
6.16.2.131DRX_SFDTOC_OFFSET	270
6.16.2.132DRX_TUNE0b_110K_NSTD	270
6.16.2.133DRX_TUNE0b_110K_STD	270
6.16.2.134DRX_TUNE0b_6M8_NSTD	270
6.16.2.135DRX_TUNE0b_6M8_STD	270
6.16.2.136DRX_TUNE0b_850K_NSTD	270
6.16.2.137DRX_TUNE0b_850K_STD	270
6.16.2.138DRX_TUNE0b_LEN	270

6.16.2.139DRX_TUNE0b_MASK	271
6.16.2.140DRX_TUNE0b_OFFSET	271
6.16.2.141DRX_TUNE1a_LEN	271
6.16.2.142DRX_TUNE1a_MASK	271
6.16.2.143DRX_TUNE1a_OFFSET	271
6.16.2.144DRX_TUNE1a_PRF16	271
6.16.2.145DRX_TUNE1a_PRF64	271
6.16.2.146DRX_TUNE1b_110K	271
6.16.2.147DRX_TUNE1b_6M8_PRE64	272
6.16.2.148DRX_TUNE1b_850K_6M8	272
6.16.2.149DRX_TUNE1b_LEN	272
6.16.2.150DRX_TUNE1b_MASK	272
6.16.2.151DRX_TUNE1b_OFFSET	272
6.16.2.152DRX_TUNE2_LEN	272
6.16.2.153DRX_TUNE2_MASK	272
6.16.2.154DRX_TUNE2_OFFSET	272
6.16.2.155DRX_TUNE2_PRF16_PAC16	273
6.16.2.156DRX_TUNE2_PRF16_PAC32	273
6.16.2.157DRX_TUNE2_PRF16_PAC64	273
6.16.2.158DRX_TUNE2_PRF16_PAC8	273
6.16.2.159DRX_TUNE2_PRF64_PAC16	273
6.16.2.160DRX_TUNE2_PRF64_PAC32	273
6.16.2.161DRX_TUNE2_PRF64_PAC64	273
6.16.2.162DRX_TUNE2_PRF64_PAC8	273
6.16.2.163DRX_TUNE4H_LEN	274
6.16.2.164DRX_TUNE4H_MASK	274
6.16.2.165DRX_TUNE4H_OFFSET	274
6.16.2.166DRX_TUNE4H_PRE128PLUS	274
6.16.2.167DRX_TUNE4H_PRE64	274
6.16.2.168DW_NS_SFD_LEN_110K	274

6.16.2.169DW_NS_SFD_LEN_6M8	274
6.16.2.170DW_NS_SFD_LEN_850K	274
6.16.2.171DX_TIME_ID	275
6.16.2.172DX_TIME_LEN	275
6.16.2.173EC_CTRL_LEN	275
6.16.2.174EC_CTRL_MASK	275
6.16.2.175EC_CTRL_OFFSET	275
6.16.2.176EC_CTRL_OSRSM	275
6.16.2.177EC_CTRL_OSTRM	275
6.16.2.178EC_CTRL_OSTM	276
6.16.2.179EC_CTRL_PLLCK	276
6.16.2.180EC_CTRL_WAIT_MASK	276
6.16.2.181EC_GOLP	276
6.16.2.182EC_GOLP_LEN	276
6.16.2.183EC_GOLP_MASK	276
6.16.2.184EC_GOLP_OFFSET_EXT_MASK	276
6.16.2.185EC_RXTC_LEN	277
6.16.2.186EC_RXTC_MASK	277
6.16.2.187EC_RXTC_OFFSET	277
6.16.2.188EUI_64_ID	277
6.16.2.189EUI_64_LEN	277
6.16.2.190EUI_64_OFFSET	277
6.16.2.191EVC_CLR	277
6.16.2.192EVC_CTRL_LEN	278
6.16.2.193EVC_CTRL_MASK	278
6.16.2.194EVC_CTRL_OFFSET	278
6.16.2.195EVC_EN	278
6.16.2.196EVC_FCE_LEN	278
6.16.2.197EVC_FCE_MASK	278
6.16.2.198EVC_FCE_OFFSET	278

6.16.2.199	EVC_FCG_LEN	279
6.16.2.200	EVC_FCG_MASK	279
6.16.2.201	EVC_FCG_OFFSET	279
6.16.2.202	EVC_FFR_LEN	279
6.16.2.203	EVC_FFR_MASK	279
6.16.2.204	EVC_FFR_OFFSET	279
6.16.2.205	EVC_FWTO_LEN	279
6.16.2.206	EVC_FWTO_MASK	280
6.16.2.207	EVC_FWTO_OFFSET	280
6.16.2.208	EVC_HPW_LEN	280
6.16.2.209	EVC_HPW_MASK	280
6.16.2.210	EVC_HPW_OFFSET	280
6.16.2.211	EVC_OVR_LEN [1/2]	280
6.16.2.212	EVC_OVR_LEN [2/2]	280
6.16.2.213	EVC_OVR_MASK [1/2]	281
6.16.2.214	EVC_OVR_MASK [2/2]	281
6.16.2.215	EVC_OVR_OFFSET	281
6.16.2.216	EVC_PHE_LEN	281
6.16.2.217	EVC_PHE_MASK	281
6.16.2.218	EVC_PHE_OFFSET	281
6.16.2.219	EVC_PTO_LEN	281
6.16.2.220	EVC_PTO_MASK	282
6.16.2.221	EVC_PTO_OFFSET	282
6.16.2.222	EVC_RES1_OFFSET	282
6.16.2.223	EVC_RSE_LEN	282
6.16.2.224	EVC_RSE_MASK	282
6.16.2.225	EVC_RSE_OFFSET	282
6.16.2.226	EVC_STO_OFFSET	282
6.16.2.227	EVC_TPW_LEN	283
6.16.2.228	EVC_TPW_MASK	283

6.16.2.229EVC_TPW_OFFSET	283
6.16.2.230EVC_TXFS_LEN	283
6.16.2.231EVC_TXFS_MASK	283
6.16.2.232EVC_TXFS_OFFSET	283
6.16.2.233EXT_SYNC_ID	283
6.16.2.234EXT_SYNC_LEN	284
6.16.2.235FP_AMPL2_MASK	284
6.16.2.236FP_AMPL2_SHIFT	284
6.16.2.237FP_AMPL3_MASK	284
6.16.2.238FP_AMPL3_SHIFT	284
6.16.2.239FS_CTRL_ID	284
6.16.2.240FS_CTRL_LEN	284
6.16.2.241FS_PLLCFG_CH1	285
6.16.2.242FS_PLLCFG_CH2	285
6.16.2.243FS_PLLCFG_CH3	285
6.16.2.244FS_PLLCFG_CH4	285
6.16.2.245FS_PLLCFG_CH5	285
6.16.2.246FS_PLLCFG_CH7	285
6.16.2.247FS_PLLCFG_LEN	285
6.16.2.248FS_PLLCFG_OFFSET	285
6.16.2.249FS_PLLTUNE_CH1	286
6.16.2.250FS_PLLTUNE_CH2	286
6.16.2.251FS_PLLTUNE_CH3	286
6.16.2.252FS_PLLTUNE_CH4	286
6.16.2.253FS_PLLTUNE_CH5	286
6.16.2.254FS_PLLTUNE_CH7	286
6.16.2.255FS_PLLTUNE_LEN	286
6.16.2.256FS_PLLTUNE_OFFSET	286
6.16.2.257FS_RES1_LEN	287
6.16.2.258FS_RES1_OFFSET	287

6.16.2.259FS_RES2_LEN	287
6.16.2.260FS_RES2_OFFSET	287
6.16.2.261FS_RES3_LEN	287
6.16.2.262FS_RES3_OFFSET	287
6.16.2.263FS_XTALT_LEN	287
6.16.2.264FS_XTALT_MASK	288
6.16.2.265FS_XTALT_MIDRANGE	288
6.16.2.266FS_XTALT_OFFSET	288
6.16.2.267GDM0	288
6.16.2.268GDM1	288
6.16.2.269GDM2	288
6.16.2.270GDM3	288
6.16.2.271GDM4	289
6.16.2.272GDM5	289
6.16.2.273GDM6	289
6.16.2.274GDM7	289
6.16.2.275GDM8	289
6.16.2.276GDP0	289
6.16.2.277GDP1	289
6.16.2.278GDP2	290
6.16.2.279GDP3	290
6.16.2.280GDP4	290
6.16.2.281GDP5	290
6.16.2.282GDP6	290
6.16.2.283GDP7	290
6.16.2.284GDP8	290
6.16.2.285GIBES0	290
6.16.2.286GIBES1	291
6.16.2.287GIBES2	291
6.16.2.288GIBES3	291

6.16.2.289GIBES4	291
6.16.2.290GIBES5	291
6.16.2.291GIBES6	291
6.16.2.292GIBES7	291
6.16.2.293GIBES8	291
6.16.2.294GICLR0	292
6.16.2.295GICLR1	292
6.16.2.296GICLR2	292
6.16.2.297GICLR3	292
6.16.2.298GICLR4	292
6.16.2.299GICLR5	292
6.16.2.300GICLR6	292
6.16.2.301GICLR7	293
6.16.2.302GICLR8	293
6.16.2.303GIDBE0	293
6.16.2.304GIDBE1	293
6.16.2.305GIDBE2	293
6.16.2.306GIDBE3	293
6.16.2.307GIDBE4	293
6.16.2.308GIDBE5	294
6.16.2.309GIDBE6	294
6.16.2.310GIDBE7	294
6.16.2.311GIDBE8	294
6.16.2.312GIMOD0	294
6.16.2.313GIMOD1	294
6.16.2.314GIMOD2	294
6.16.2.315GIMOD3	295
6.16.2.316GIMOD4	295
6.16.2.317GIMOD5	295
6.16.2.318GIMOD6	295

6.16.2.319GIMOD7	295
6.16.2.320GIMOD8	295
6.16.2.321GIRQE0	295
6.16.2.322GIRQE1	295
6.16.2.323GIRQE2	296
6.16.2.324GIRQE3	296
6.16.2.325GIRQE4	296
6.16.2.326GIRQE5	296
6.16.2.327GIRQE6	296
6.16.2.328GIRQE7	296
6.16.2.329GIRQE8	296
6.16.2.330GIRQx0	296
6.16.2.331GIRQx1	297
6.16.2.332GIRQx2	297
6.16.2.333GIRQx3	297
6.16.2.334GIRQx4	297
6.16.2.335GIRQx5	297
6.16.2.336GIRQx6	297
6.16.2.337GIRQx7	297
6.16.2.338GIRQx8	297
6.16.2.339GISENO	298
6.16.2.340GISEN1	298
6.16.2.341GISEN2	298
6.16.2.342GISEN3	298
6.16.2.343GISEN4	298
6.16.2.344GISEN5	298
6.16.2.345GISEN6	298
6.16.2.346GISEN7	299
6.16.2.347GISEN8	299
6.16.2.348GPIO_CTRL_ID	299

6.16.2.349GPIO_CTRL_LEN	299
6.16.2.350GPIO_DIR_LEN	299
6.16.2.351GPIO_DIR_MASK	299
6.16.2.352GPIO_DIR_OFFSET	299
6.16.2.353GPIO_DOUT_LEN	300
6.16.2.354GPIO_DOUT_MASK	300
6.16.2.355GPIO_DOUT_OFFSET	300
6.16.2.356GPIO_IBES_LEN	300
6.16.2.357GPIO_IBES_MASK	300
6.16.2.358GPIO_IBES_OFFSET	300
6.16.2.359GPIO_ICLR_LEN	300
6.16.2.360GPIO_ICLR_MASK	301
6.16.2.361GPIO_ICLR_OFFSET	301
6.16.2.362GPIO_IDBE_LEN	301
6.16.2.363GPIO_IDBE_MASK	301
6.16.2.364GPIO_IDBE_OFFSET	301
6.16.2.365GPIO_IMODE_LEN	301
6.16.2.366GPIO_IMODE_MASK	301
6.16.2.367GPIO_IMODE_OFFSET	302
6.16.2.368GPIO_IRQE_LEN	302
6.16.2.369GPIO_IRQE_MASK	302
6.16.2.370GPIO_IRQE_OFFSET	302
6.16.2.371GPIO_ISEN_LEN	302
6.16.2.372GPIO_ISEN_MASK	302
6.16.2.373GPIO_ISEN_OFFSET	302
6.16.2.374GPIO_MODE_LEN	303
6.16.2.375GPIO_MODE_MASK	303
6.16.2.376GPIO_MODE_OFFSET	303
6.16.2.377GPIO_MSGP0_MASK	303
6.16.2.378GPIO_MSGP1_MASK	303

6.16.2.379GPIO_MSGP2_MASK	303
6.16.2.380GPIO_MSGP3_MASK	303
6.16.2.381GPIO_MSGP4_MASK	303
6.16.2.382GPIO_MSGP5_MASK	304
6.16.2.383GPIO_MSGP6_MASK	304
6.16.2.384GPIO_MSGP7_MASK	304
6.16.2.385GPIO_MSGP8_MASK	304
6.16.2.386GPIO_PIN2_RXLED	304
6.16.2.387GPIO_PIN3_TXLED	304
6.16.2.388GPIO_PIN4_EXTPA	304
6.16.2.389GPIO_PIN5_EXTTXE	304
6.16.2.390GPIO_PIN6_EXTRXE	305
6.16.2.391GPIO_RAW_LEN	305
6.16.2.392GPIO_RAW_MASK	305
6.16.2.393GPIO_RAW_OFFSET	305
6.16.2.394GRAWP0	305
6.16.2.395GRAWP1	305
6.16.2.396GRAWP2	305
6.16.2.397GRAWP3	306
6.16.2.398GRAWP4	306
6.16.2.399GRAWP5	306
6.16.2.400GRAWP6	306
6.16.2.401GRAWP7	306
6.16.2.402GRAWP8	306
6.16.2.403GxM0	306
6.16.2.404GxM1	306
6.16.2.405GxM2	307
6.16.2.406GxM3	307
6.16.2.407GxM4	307
6.16.2.408GxM5	307

6.16.2.409GxM6	307
6.16.2.410GxM7	307
6.16.2.411GxM8	307
6.16.2.412GxP0	307
6.16.2.413GxP1	308
6.16.2.414GxP2	308
6.16.2.415GxP3	308
6.16.2.416GxP4	308
6.16.2.417GxP5	308
6.16.2.418GxP6	308
6.16.2.419GxP7	308
6.16.2.420GxP8	308
6.16.2.421LDE_CFG1_LEN	309
6.16.2.422LDE_CFG1_NSTDEV_MASK	309
6.16.2.423LDE_CFG1_OFFSET	309
6.16.2.424LDE_CFG1_PMULT_MASK	309
6.16.2.425LDE_CFG2_LEN	309
6.16.2.426LDE_CFG2_OFFSET	309
6.16.2.427LDE_IF_ID	309
6.16.2.428LDE_IF_LEN	310
6.16.2.429LDE_PPAMPL_LEN	310
6.16.2.430LDE_PPAMPL_OFFSET	310
6.16.2.431LDE_PPINDEX_LEN	310
6.16.2.432LDE_PPINDEX_OFFSET	310
6.16.2.433LDE_REPC_LEN	310
6.16.2.434LDE_REPC_OFFSET	310
6.16.2.435LDE_REPC_PCODE_1	311
6.16.2.436LDE_REPC_PCODE_10	311
6.16.2.437LDE_REPC_PCODE_11	311
6.16.2.438LDE_REPC_PCODE_12	311

6.16.2.439LDE_REPC_PCODE_13	311
6.16.2.440LDE_REPC_PCODE_14	311
6.16.2.441LDE_REPC_PCODE_15	311
6.16.2.442LDE_REPC_PCODE_16	311
6.16.2.443LDE_REPC_PCODE_17	312
6.16.2.444LDE_REPC_PCODE_18	312
6.16.2.445LDE_REPC_PCODE_19	312
6.16.2.446LDE_REPC_PCODE_2	312
6.16.2.447LDE_REPC_PCODE_20	312
6.16.2.448LDE_REPC_PCODE_21	312
6.16.2.449LDE_REPC_PCODE_22	312
6.16.2.450LDE_REPC_PCODE_23	312
6.16.2.451LDE_REPC_PCODE_24	313
6.16.2.452LDE_REPC_PCODE_3	313
6.16.2.453LDE_REPC_PCODE_4	313
6.16.2.454LDE_REPC_PCODE_5	313
6.16.2.455LDE_REPC_PCODE_6	313
6.16.2.456LDE_REPC_PCODE_7	313
6.16.2.457LDE_REPC_PCODE_8	313
6.16.2.458LDE_REPC_PCODE_9	313
6.16.2.459LDE_RXANTD_LEN	314
6.16.2.460LDE_RXANTD_OFFSET	314
6.16.2.461LDE_THRESH_LEN	314
6.16.2.462LDE_THRESH_OFFSET	314
6.16.2.463OTP_ADDR	314
6.16.2.464OTP_ADDR_LEN	314
6.16.2.465OTP_ADDR_MASK	314
6.16.2.466OTP_CTRL	315
6.16.2.467OTP_CTRL_LDELOAD	315
6.16.2.468OTP_CTRL_LEN	315

6.16.2.469OTP_CTRL_MASK	315
6.16.2.470OTP_CTRL_OTPPROG	315
6.16.2.471OTP_CTRL_OTPRDEN	315
6.16.2.472OTP_CTRL_OTPREAD	315
6.16.2.473OTP_IF_ID	316
6.16.2.474OTP_IF_LEN	316
6.16.2.475OTP_RDAT	316
6.16.2.476OTP_RDAT_LEN	316
6.16.2.477OTP_SF	316
6.16.2.478OTP_SF_LDO_KICK	316
6.16.2.479OTP_SF_LEN	316
6.16.2.480OTP_SF_MASK	317
6.16.2.481OTP_SF_OPS_KICK	317
6.16.2.482OTP_SF_OPS_SEL_L64	317
6.16.2.483OTP_SF_OPS_SEL_MASK	317
6.16.2.484OTP_SF_OPS_SEL_SHFT	317
6.16.2.485OTP_SF_OPS_SEL_TIGHT	317
6.16.2.486OTP_SRDAT	317
6.16.2.487OTP_SRDAT_LEN	318
6.16.2.488OTP_STAT	318
6.16.2.489OTP_STAT_LEN	318
6.16.2.490OTP_STAT_MASK	318
6.16.2.491OTP_STAT_OTPPRGD	318
6.16.2.492OTP_STAT_OTPVPOK	318
6.16.2.493OTP_WDAT	318
6.16.2.494OTP_WDAT_LEN	319
6.16.2.495PANADR_ID	319
6.16.2.496PANADR_LEN	319
6.16.2.497PANADR_PAN_ID_MASK	319
6.16.2.498PANADR_PAN_ID_OFFSET	319

6.16.2.499PANADR_SHORT_ADDR_MASK	319
6.16.2.500PANADR_SHORT_ADDR_OFFSET	319
6.16.2.501PMSC_CTRL0_FACE	320
6.16.2.502PMSC_CTRL0_GPDCE	320
6.16.2.503PMSC_CTRL0_KHZCLEN	320
6.16.2.504PMSC_CTRL0_LEN	320
6.16.2.505PMSC_CTRL0_MASK	320
6.16.2.506PMSC_CTRL0_OFFSET	320
6.16.2.507PMSC_CTRL0_PLL2_SEQ_EN	320
6.16.2.508PMSC_CTRL0_RESET_ALL	320
6.16.2.509PMSC_CTRL0_RESET_CLEAR	321
6.16.2.510PMSC_CTRL0_RESET_RX	321
6.16.2.511PMSC_CTRL0_RXCLKS_125M	321
6.16.2.512PMSC_CTRL0_RXCLKS_19M	321
6.16.2.513PMSC_CTRL0_RXCLKS_AUTO	321
6.16.2.514PMSC_CTRL0_RXCLKS_OFF	321
6.16.2.515PMSC_CTRL0_SOFTRESET_OFFSET	321
6.16.2.516PMSC_CTRL0_SYSCLKS_125M	322
6.16.2.517PMSC_CTRL0_SYSCLKS_19M	322
6.16.2.518PMSC_CTRL0_SYSCLKS_AUTO	322
6.16.2.519PMSC_CTRL0_TXCLKS_125M	322
6.16.2.520PMSC_CTRL0_TXCLKS_19M	322
6.16.2.521PMSC_CTRL0_TXCLKS_AUTO	322
6.16.2.522PMSC_CTRL0_TXCLKS_OFF	322
6.16.2.523PMSC_CTRL1_ARX2INIT	323
6.16.2.524PMSC_CTRL1_ARXSLP	323
6.16.2.525PMSC_CTRL1_ATXSLP	323
6.16.2.526PMSC_CTRL1_KHZCLKDIV_MASK	323
6.16.2.527PMSC_CTRL1_LDERUNE	323
6.16.2.528PMSC_CTRL1_LEN	323

6.16.2.529PMSC_CTRL1_MASK	323
6.16.2.530PMSC_CTRL1_OFFSET	324
6.16.2.531PMSC_CTRL1_PKTSEQ_DISABLE	324
6.16.2.532PMSC_CTRL1_PKTSEQ_ENABLE	324
6.16.2.533PMSC_CTRL1_PLLSYN	324
6.16.2.534PMSC_CTRL1_SNOZE	324
6.16.2.535PMSC_CTRL1_SNOZR	324
6.16.2.536PMSC_ID	324
6.16.2.537PMSC_LEDC_BLINK_NOW_ALL	325
6.16.2.538PMSC_LEDC_BLINK_TIM_MASK	325
6.16.2.539PMSC_LEDC_BLINK_TIME_DEF	325
6.16.2.540PMSC_LEDC_BLNKEN	325
6.16.2.541PMSC_LEDC_LEN	325
6.16.2.542PMSC_LEDC_MASK	325
6.16.2.543PMSC_LEDC_OFFSET	325
6.16.2.544PMSC_LEN	326
6.16.2.545PMSC_RES1_OFFSET	326
6.16.2.546PMSC_RES2_OFFSET	326
6.16.2.547PMSC_RES3_OFFSET	326
6.16.2.548PMSC_SNOZT_LEN	326
6.16.2.549PMSC_SNOZT_OFFSET	326
6.16.2.550PMSC_TXFINESEQ_DISABLE	326
6.16.2.551PMSC_TXFINESEQ_ENABLE	327
6.16.2.552PMSC_TXFINESEQ_OFFSET	327
6.16.2.553REG_05_ID_RESERVED	327
6.16.2.554REG_07_ID_RESERVED	327
6.16.2.555REG_0B_ID_RESERVED	327
6.16.2.556REG_16_ID_RESERVED	327
6.16.2.557REG_1B_ID_RESERVED	328
6.16.2.558REG_1C_ID_RESERVED	328

6.16.2.559REG_20_ID_RESERVED	328
6.16.2.560REG_22_ID_RESERVED	328
6.16.2.561REG_29_ID_RESERVED	328
6.16.2.562REG_30_ID_RESERVED	328
6.16.2.563REG_31_ID_RESERVED	329
6.16.2.564REG_32_ID_RESERVED	329
6.16.2.565REG_33_ID_RESERVED	329
6.16.2.566REG_34_ID_RESERVED	329
6.16.2.567REG_35_ID_RESERVED	329
6.16.2.568REG_37_ID_RESERVED	329
6.16.2.569REG_38_ID_RESERVED	329
6.16.2.570REG_39_ID_RESERVED	330
6.16.2.571REG_3A_ID_RESERVED	330
6.16.2.572REG_3B_ID_RESERVED	330
6.16.2.573REG_3C_ID_RESERVED	330
6.16.2.574REG_3D_ID_RESERVED	330
6.16.2.575REG_3E_ID_RESERVED	330
6.16.2.576REG_3F_ID_RESERVED	330
6.16.2.577RF_CONF_ID	331
6.16.2.578RF_CONF_LEN	331
6.16.2.579RF_CONF_PGMIXBIASEN_MASK	331
6.16.2.580RF_CONF_PLLEN_MASK	331
6.16.2.581RF_CONF_RXEN_MASK	331
6.16.2.582RF_CONF_TXALLEN_MASK	331
6.16.2.583RF_CONF_TXBLOCKSEN_MASK	331
6.16.2.584RF_CONF_TXEN_MASK	332
6.16.2.585RF_CONF_TXPLLPOWEN_MASK	332
6.16.2.586RF_CONF_TXPOW_MASK	332
6.16.2.587RF_RXCTRLH_LEN	332
6.16.2.588RF_RXCTRLH_NBW	332

6.16.2.589RF_RXCTRLH_OFFSET	332
6.16.2.590RF_RXCTRLH_WBW	332
6.16.2.591RF_STATUS_OFFSET	332
6.16.2.592RF_TXCTRL_CH1	333
6.16.2.593RF_TXCTRL_CH2	333
6.16.2.594RF_TXCTRL_CH3	333
6.16.2.595RF_TXCTRL_CH4	333
6.16.2.596RF_TXCTRL_CH5	333
6.16.2.597RF_TXCTRL_CH7	333
6.16.2.598RF_TXCTRL_LEN	333
6.16.2.599RF_TXCTRL_OFFSET	334
6.16.2.600RF_TXCTRL_TXMTUNE_MASK	334
6.16.2.601RF_TXCTRL_TXTXMQ_MASK	334
6.16.2.602RX_BUFFER_ID	334
6.16.2.603RX_BUFFER_LEN	334
6.16.2.604RX_EQUAL_CIR_MXG_MASK	334
6.16.2.605RX_EQUAL_CIR_MXG_SHIFT	334
6.16.2.606RX_EQUAL_FP_AMPL2_MASK	335
6.16.2.607RX_EQUAL_FP_AMPL2_SHIFT	335
6.16.2.608RX_EQUAL_FP_AMPL3_MASK	335
6.16.2.609RX_EQUAL_FP_AMPL3_SHIFT	335
6.16.2.610RX_EQUAL_STD_NOISE_MASK	335
6.16.2.611RX_EQUAL_STD_NOISE_SHIFT	335
6.16.2.612RX_FINFO_ID	335
6.16.2.613RX_FINFO_LEN	336
6.16.2.614RX_FINFO_MASK_32	336
6.16.2.615RX_FINFO_OFFSET	336
6.16.2.616RX_FINFO_RNG	336
6.16.2.617RX_FINFO_RNG_SHIFT	336
6.16.2.618RX_FINFO_RXBR_110k	336

6.16.2.619RX_FINFO_RXBR_6M	336
6.16.2.620RX_FINFO_RXBR_850k	337
6.16.2.621RX_FINFO_RXBR_MASK	337
6.16.2.622RX_FINFO_RXBR_SHIFT	337
6.16.2.623RX_FINFO_RXFL_MASK_1023	337
6.16.2.624RX_FINFO_RXFLE_MASK	337
6.16.2.625RX_FINFO_RXFLEN_MASK	337
6.16.2.626RX_FINFO_RXNSPL_MASK	337
6.16.2.627RX_FINFO_RXPACC_MASK	338
6.16.2.628RX_FINFO_RXPACC_SHIFT	338
6.16.2.629RX_FINFO_RXPEL_1024	338
6.16.2.630RX_FINFO_RXPEL_128	338
6.16.2.631RX_FINFO_RXPEL_1536	338
6.16.2.632RX_FINFO_RXPEL_2048	338
6.16.2.633RX_FINFO_RXPEL_256	338
6.16.2.634RX_FINFO_RXPEL_4096	338
6.16.2.635RX_FINFO_RXPEL_512	339
6.16.2.636RX_FINFO_RXPEL_64	339
6.16.2.637RX_FINFO_RXPEL_MASK	339
6.16.2.638RX_FINFO_RXPRF_16M	339
6.16.2.639RX_FINFO_RXPRF_64M	339
6.16.2.640RX_FINFO_RXPRF_MASK	339
6.16.2.641RX_FINFO_RXPRF_SHIFT	339
6.16.2.642RX_FINFO_RXPSR_MASK	339
6.16.2.643RX_FQUAL_ID	340
6.16.2.644RX_FQUAL_LEN	340
6.16.2.645RX_FWTO_ID	340
6.16.2.646RX_FWTO_LEN	340
6.16.2.647RX_FWTO_MASK	340
6.16.2.648RX_FWTO_OFFSET	340

6.16.2.649RX_SNIFF_ID	340
6.16.2.650RX_SNIFF_LEN	341
6.16.2.651RX_SNIFF_MASK	341
6.16.2.652RX_SNIFF_OFFSET	341
6.16.2.653RX_SNIFF_OFFT_MASK	341
6.16.2.654RX_SNIFF_SNIFT_ONT_MASK	341
6.16.2.655RX_STAMP_LEN	341
6.16.2.656RX_TIME_FP_AMPL1_OFFSET	341
6.16.2.657RX_TIME_FP_INDEX_OFFSET	342
6.16.2.658RX_TIME_FP_RAWST_OFFSET	342
6.16.2.659RX_TIME_ID	342
6.16.2.660RX_TIME_LLEN	342
6.16.2.661RX_TIME_RX_STAMP_LEN	342
6.16.2.662RX_TIME_RX_STAMP_OFFSET	342
6.16.2.663RX_TTCKI_ID	342
6.16.2.664RX_TTCKI_LEN	343
6.16.2.665RX_TTCKO_ID	343
6.16.2.666RX_TTCKO_LEN	343
6.16.2.667RX_TTCKO_MASK_32	343
6.16.2.668RX_TTCKO_RCPHASE_MASK	343
6.16.2.669RX_TTCKO_RSMPDEL_MASK	343
6.16.2.670RX_TTCKO_RXTOFS_MASK	343
6.16.2.671SNIFF_OFFT_MASK	344
6.16.2.672SNIFF_ONT_MASK	344
6.16.2.673STD_NOISE_MASK	344
6.16.2.674STD_NOISE_SHIFT	344
6.16.2.675SYS_CFG_AACKPEND	344
6.16.2.676SYS_CFG_AUTOACK	344
6.16.2.677SYS_CFG_DIS_DRXB	344
6.16.2.678SYS_CFG_DIS_FCE	344

6.16.2.679SYS_CFG_DIS_PHE	345
6.16.2.680SYS_CFG_DIS_RSDE	345
6.16.2.681SYS_CFG_DIS_STXP	345
6.16.2.682SYS_CFG_FCS_INIT2F	345
6.16.2.683SYS_CFG_FF_ALL_EN	345
6.16.2.684SYS_CFG_FFA4	345
6.16.2.685SYS_CFG_FFA5	345
6.16.2.686SYS_CFG_FFAA	346
6.16.2.687SYS_CFG_FFAB	346
6.16.2.688SYS_CFG_FFAD	346
6.16.2.689SYS_CFG_FFAM	346
6.16.2.690SYS_CFG_FFAR	346
6.16.2.691SYS_CFG_FFBC	346
6.16.2.692SYS_CFG_FFE	346
6.16.2.693SYS_CFG_HIRQ_POL	347
6.16.2.694SYS_CFG_ID	347
6.16.2.695SYS_CFG_LEN	347
6.16.2.696SYS_CFG_MASK	347
6.16.2.697SYS_CFG_PHR_MODE_00	347
6.16.2.698SYS_CFG_PHR_MODE_11	347
6.16.2.699SYS_CFG_PHR_MODE_SHFT	347
6.16.2.700SYS_CFG_RXAUTR	348
6.16.2.701SYS_CFG_RXM110K	348
6.16.2.702SYS_CFG_RXWTOE	348
6.16.2.703SYS_CFG_SPI_EDGE	348
6.16.2.704SYS_CTRL_CANSFCS	348
6.16.2.705SYS_CTRL_HRB _T	348
6.16.2.706SYS_CTRL_HRB _T _OFFSET	348
6.16.2.707SYS_CTRL_HSRBT_TOGGLE	349
6.16.2.708SYS_CTRL_ID	349

6.16.2.70SYS_CTRL_LEN	349
6.16.2.71SYS_CTRL_MASK_32	349
6.16.2.711SYS_CTRL_OFFSET	349
6.16.2.712SYS_CTRL_RXDLYE	349
6.16.2.713SYS_CTRL_RXENAB	349
6.16.2.714SYS_CTRL_SFCST	350
6.16.2.715SYS_CTRL_TRXOFF	350
6.16.2.716SYS_CTRL_TXDLYS	350
6.16.2.717SYS_CTRL_TXSTRT	350
6.16.2.718SYS_CTRL_WAIT4RESP	350
6.16.2.719SYS_MASK_ID	350
6.16.2.720SYS_MASK_LEN	350
6.16.2.721SYS_MASK_MAAT	351
6.16.2.722SYS_MASK_MAFFREJ	351
6.16.2.723SYS_MASK_MASK_32	351
6.16.2.724SYS_MASK_MCPLLL	351
6.16.2.725SYS_MASK_MCPLOCK	351
6.16.2.726SYS_MASK_MESYNCR	351
6.16.2.727SYS_MASK_MGPIORQ	351
6.16.2.728SYS_MASK_MHPDWARN	352
6.16.2.729SYS_MASK_MLDEDONE	352
6.16.2.730SYS_MASK_MLDEERR	352
6.16.2.731SYS_MASK_MRFPLLL	352
6.16.2.732SYS_MASK_MRXdFR	352
6.16.2.733SYS_MASK_MRXFCE	352
6.16.2.734SYS_MASK_MRXFCE	352
6.16.2.735SYS_MASK_MRxoVRR	352
6.16.2.736SYS_MASK_MRXPHD	353
6.16.2.737SYS_MASK_MRXPHE	353
6.16.2.738SYS_MASK_MRXP RD	353

6.16.2.739SYS_MASK_MRXP TO	353
6.16.2.740SYS_MASK_MRXRFS L	353
6.16.2.741SYS_MASK_MRXRFT O	353
6.16.2.742SYS_MASK_MRXSFD D	353
6.16.2.743SYS_MASK_MRXSFDTO	353
6.16.2.744SYS_MASK_MSLP2INIT	354
6.16.2.745SYS_MASK_MTXBERR	354
6.16.2.746SYS_MASK_MTXF RB	354
6.16.2.747SYS_MASK_MTXF RS	354
6.16.2.748SYS_MASK_MTXPHS	354
6.16.2.749SYS_MASK_MTXPRS	354
6.16.2.750SYS_STATE_ID	354
6.16.2.751SYS_STATE_LEN	355
6.16.2.752SYS_STATUS_AAT	355
6.16.2.753SYS_STATUS_AFFREJ	355
6.16.2.754SYS_STATUS_ALL_DBLBUFF	355
6.16.2.755SYS_STATUS_ALL_RX_ERR	355
6.16.2.756SYS_STATUS_ALL_RX_GOOD	355
6.16.2.757SYS_STATUS_ALL_RX_TO	356
6.16.2.758SYS_STATUS_ALL_TX	356
6.16.2.759SYS_STATUS_CLKPLL_LL	356
6.16.2.760SYS_STATUS_CPLock	356
6.16.2.761SYS_STATUS_ESYNCR	356
6.16.2.762SYS_STATUS_GPIOIRQ	356
6.16.2.763SYS_STATUS_HPDWARN	356
6.16.2.764SYS_STATUS_HSRBP	357
6.16.2.765SYS_STATUS_ICRBP	357
6.16.2.766SYS_STATUS_ID	357
6.16.2.767SYS_STATUS_IRQS	357
6.16.2.768SYS_STATUS_LDEDONE	357

6.16.2.76SYS_STATUS_LDEERR	357
6.16.2.77SYS_STATUS_LEN	357
6.16.2.771SYS_STATUS_MASK_32	358
6.16.2.772SYS_STATUS_OFFSET	358
6.16.2.773SYS_STATUS_reserved	358
6.16.2.774SYS_STATUS_RFPLL_LL	358
6.16.2.775SYS_STATUS_RXDFR	358
6.16.2.776SYS_STATUS_RXFCE	358
6.16.2.777SYS_STATUS_RXFCG	358
6.16.2.778SYS_STATUS_RXOVRR	359
6.16.2.779SYS_STATUS_RXPHD	359
6.16.2.780SYS_STATUS_RXPHE	359
6.16.2.781SYS_STATUS_RXPRD	359
6.16.2.782SYS_STATUS_RXPREJ	359
6.16.2.783SYS_STATUS_RXPTO	359
6.16.2.784SYS_STATUS_RXRFSL	359
6.16.2.785SYS_STATUS_RXRFTO	359
6.16.2.786SYS_STATUS_RXRSCS	360
6.16.2.787SYS_STATUS_RXSFDD	360
6.16.2.788SYS_STATUS_RXSFDTO	360
6.16.2.789SYS_STATUS_SLP2INIT	360
6.16.2.790SYS_STATUS_TXBERR	360
6.16.2.791SYS_STATUS_TXERR	360
6.16.2.792SYS_STATUS_TXFRB	360
6.16.2.793SYS_STATUS_TXFRS	361
6.16.2.794SYS_STATUS_TXPHS	361
6.16.2.795SYS_STATUS_TXPRS	361
6.16.2.796SYS_STATUS_TXPUTE	361
6.16.2.797SYS_TIME_ID	361
6.16.2.798SYS_TIME_LEN	361

6.16.2.799SYS_TIME_OFFSET	361
6.16.2.800TC_PGCAL_STATUS_DELAY_MASK	362
6.16.2.801TC_PGCAL_STATUS_LEN	362
6.16.2.802TC_PGCAL_STATUS_OFFSET	362
6.16.2.803TC_PGCCTRL_AUTOCAL	362
6.16.2.804TC_PGCCTRL_CALSTART	362
6.16.2.805TC_PGCCTRL_DIR_CONV	362
6.16.2.806TC_PGCCTRL_LEN	362
6.16.2.807TC_PGCCTRL_OFFSET	362
6.16.2.808TC_PGCCTRL_ON_TX	363
6.16.2.809TC_PGCCTRL_TMEAS_MASK	363
6.16.2.810TC_PGDELAY_CH1	363
6.16.2.811TC_PGDELAY_CH2	363
6.16.2.812TC_PGDELAY_CH3	363
6.16.2.813TC_PGDELAY_CH4	363
6.16.2.814TC_PGDELAY_CH5	363
6.16.2.815TC_PGDELAY_CH7	364
6.16.2.816TC_PGDELAY_LEN	364
6.16.2.817TC_PGDELAY_OFFSET	364
6.16.2.818TC_PGTTEST_CW	364
6.16.2.819TC_PGTTEST_LEN	364
6.16.2.820TC_PGTTEST_NORMAL	364
6.16.2.821TC_PGTTEST_OFFSET	364
6.16.2.822TC_SARL_SAR_C	364
6.16.2.823TC_SARL_SAR_LTEMP_OFFSET	365
6.16.2.824TC_SARL_SAR_LVBAT_OFFSET	365
6.16.2.825TC_SARW_SAR_WTEMP_OFFSET	365
6.16.2.826TC_SARW_SAR_WVBAT_OFFSET	365
6.16.2.827TX_ANTD_ID	365
6.16.2.828TX_ANTD_LEN	365

6.16.2.829TX_ANTD_OFFSET	365
6.16.2.830TX_BUFFER_ID	366
6.16.2.831TX_BUFFER_LEN	366
6.16.2.832TX_CAL_ID	366
6.16.2.833TX_CAL_LEN	366
6.16.2.834TX_FCTRL_FLE_MASK	366
6.16.2.835TX_FCTRL_ID	366
6.16.2.836TX_FCTRL_IFSDELAY_MASK	366
6.16.2.837TX_FCTRL_LEN	367
6.16.2.838TX_FCTRL_PE_MASK	367
6.16.2.839TX_FCTRL_PE_SHFT	367
6.16.2.840TX_FCTRL_SAFE_MASK_32	367
6.16.2.841TX_FCTRL_TFLE_MASK	367
6.16.2.842TX_FCTRL_TFLEN_MASK	367
6.16.2.843TX_FCTRL_TR	367
6.16.2.844TX_FCTRL_TR_SHFT	368
6.16.2.845TX_FCTRL_TXBOFFS_MASK	368
6.16.2.846TX_FCTRL_TXBOFFS_SHFT	368
6.16.2.847TX_FCTRL_TXBR_110k	368
6.16.2.848TX_FCTRL_TXBR_6M	368
6.16.2.849TX_FCTRL_TXBR_850k	368
6.16.2.850TX_FCTRL_TXBR_MASK	368
6.16.2.851TX_FCTRL_TXBR_SHFT	369
6.16.2.852TX_FCTRL_TXPRF_16M	369
6.16.2.853TX_FCTRL_TXPRF_4M	369
6.16.2.854TX_FCTRL_TXPRF_64M	369
6.16.2.855TX_FCTRL_TXPRF_MASK	369
6.16.2.856TX_FCTRL_TXPRF_SHFT	369
6.16.2.857TX_FCTRL_TXPSR_MASK	369
6.16.2.858TX_FCTRL_TXPSR_PE_1024	370

6.16.2.859TX_FCTRL_TXPSR_PE_128	370
6.16.2.860TX_FCTRL_TXPSR_PE_1536	370
6.16.2.861TX_FCTRL_TXPSR_PE_16	370
6.16.2.862TX_FCTRL_TXPSR_PE_2048	370
6.16.2.863TX_FCTRL_TXPSR_PE_256	370
6.16.2.864TX_FCTRL_TXPSR_PE_4096	370
6.16.2.865TX_FCTRL_TXPSR_PE_512	370
6.16.2.866TX_FCTRL_TXPSR_PE_64	371
6.16.2.867TX_FCTRL_TXPSR_PE_MASK	371
6.16.2.868TX_FCTRL_TXPSR_SHFT	371
6.16.2.869TX_POWER_BOOSTNORM_MASK	371
6.16.2.870TX_POWER_BOOSTNORM_SHIFT	371
6.16.2.871TX_POWER_BOOSTP125_MASK	371
6.16.2.872TX_POWER_BOOSTP125_SHIFT	371
6.16.2.873TX_POWER_BOOSTP250_MASK	372
6.16.2.874TX_POWER_BOOSTP250_SHIFT	372
6.16.2.875TX_POWER_BOOSTP500_MASK	372
6.16.2.876TX_POWER_BOOSTP500_SHIFT	372
6.16.2.877TX_POWER_ID	372
6.16.2.878TX_POWER_LEN	372
6.16.2.879TX_POWER_MAN_DEFAULT	372
6.16.2.880TX_POWER_TXPOWPHR_MASK	373
6.16.2.881TX_POWER_TXPOWSD_MASK	373
6.16.2.882TX_STAMP_LEN	373
6.16.2.883TX_TIME_ID	373
6.16.2.884TX_TIME_LLEN	373
6.16.2.885TX_TIME_TX_RAWST_OFFSET	373
6.16.2.886TX_TIME_TX_STAMP_LEN	373
6.16.2.887TX_TIME_TX_STAMP_OFFSET	374
6.16.2.888USR_SFD_ID	374

6.16.2.889	USR_SFD_LEN	374
6.16.2.890	W4R_TIM_MASK	374
6.17	deca_types.h File Reference	374
6.17.1	Detailed Description	375
6.17.2	Macro Definition Documentation	375
6.17.2.1	_DECA_INT16_	375
6.17.2.2	_DECA_INT32_	375
6.17.2.3	_DECA_INT8_	376
6.17.2.4	_DECA_UINT16_	376
6.17.2.5	_DECA_UINT32_	376
6.17.2.6	_DECA_UINT8_	376
6.17.2.7	NULL	376
6.17.3	Typedef Documentation	376
6.17.3.1	int16	376
6.17.3.2	int32	376
6.17.3.3	int8	377
6.17.3.4	uint16	377
6.17.3.5	uint32	377
6.17.3.6	uint8	377
6.18	deca_version.h File Reference	377
6.18.1	Detailed Description	378
6.18.2	Macro Definition Documentation	378
6.18.2.1	DW1000_DEVICE_DRIVER_VER_STRING	378
6.18.2.2	DW1000_DRIVER_VERSION	378
6.19	fu.c File Reference	378
6.19.1	Function Documentation	379
6.19.1.1	FU_AcceptFirmware()	379
6.19.1.2	FU_AcceptFirmwareVersion()	379
6.19.1.3	FU_AcceptHardwareVersion()	380
6.19.1.4	FU_GetCurrentFlashBase()	380

6.19.1.5 FU_HandleAsDevice()	380
6.19.1.6 FU_Init()	381
6.20 fu.h File Reference	381
6.20.1 Detailed Description	384
6.20.2 Macro Definition Documentation	387
6.20.2.1 FU_ASSERT	388
6.20.2.2 FU_BLOCK_SIZE	388
6.20.2.3 FU_DESTINATION_1	388
6.20.2.4 FU_DESTINATION_2	388
6.20.2.5 FU_ERR_BAD_F_VERSION	388
6.20.2.6 FU_ERR_BAD_FILE_SIZE	388
6.20.2.7 FU_ERR_BAD_FLASH_CRC	388
6.20.2.8 FU_ERR_BAD_FRAME_CRC	389
6.20.2.9 FU_ERR_BAD_FRAME_HASH	389
6.20.2.10 FU_ERR_BAD_FRAME_LEN	389
6.20.2.11 FU_ERR_BAD_H_VERSION	389
6.20.2.12 FU_ERR_BAD_OFFSET	389
6.20.2.13 FU_ERR_BAD_OPCODE_SET	389
6.20.2.14 FU_ERR_BAD_PROT_VER	390
6.20.2.15 FU_ERR_FIR_NOT_ACCEPTED_YET	390
6.20.2.16 FU_ERR_FLASH_ERASING	390
6.20.2.17 FU_ERR_FLASH_WRITING	390
6.20.2.18 FU_ERR_VERSION_IN_PACKAGE	390
6.20.2.19 FU_MAX_DATA_SIZE	391
6.20.2.20 FU_MAX_PROGRAM_SIZE	391
6.20.2.21 FU_OPCODE_ABORT	391
6.20.2.22 FU_OPCODE_ACK	391
6.20.2.23 FU_OPCODE_DATA	391
6.20.2.24 FU_OPCODE_EOT	391
6.20.2.25 FU_OPCODE_SOT	392

6.20.2.26 FU_PROT_HEAD_SIZE	392
6.20.2.27 FU_PROT_VERSION	392
6.20.2.28 FU_VERSION_LOC	392
6.20.3 Function Documentation	392
6.20.3.1 FU_AcceptFirmware()	392
6.20.3.2 FU_GetCurrentFlashBase()	393
6.20.3.3 FU_HandleAsDevice()	393
6.20.3.4 FU_Init()	394
6.20.4 Variable Documentation	394
6.20.4.1 PROG_DESTINATION1	394
6.20.4.2 PROG_DESTINATION2	394
6.20.4.3 PROG_SETTINGS1	395
6.20.4.4 PROG_START1	395
6.21 gitversion.h File Reference	395
6.21.1 Macro Definition Documentation	396
6.21.1.1 __F_HASH__	396
6.21.1.2 __F_MAJOR__	396
6.21.1.3 __F_MINOR__	396
6.22 iassert.h File Reference	396
6.22.1 Macro Definition Documentation	397
6.22.1.1 IASSERT	397
6.22.2 Function Documentation	397
6.22.2.1 PORT_iassert_fun()	397
6.23 listener_parser.c File Reference	397
6.23.1 Macro Definition Documentation	398
6.23.1.1 LISTENER_CASE	398
6.23.2 Function Documentation	398
6.23.2.1 listener_isr()	398
6.23.2.2 listener_parse()	399
6.24 logs.h File Reference	400

6.24.1 Detailed Description	402
6.24.2 Macro Definition Documentation	402
6.24.2.1 LOG_BASE64	402
6.24.2.2 LOG_CRIT	402
6.24.2.3 LOG_DBG	402
6.24.2.4 LOG_ERR	402
6.24.2.5 LOG_INF	403
6.24.2.6 LOG_TEST	403
6.24.2.7 LOG_WRN	403
6.24.3 Function Documentation	403
6.24.3.1 LOG_Bin()	403
6.24.3.2 LOG_Text()	404
6.25 mac.c File Reference	404
6.25.1 Function Documentation	406
6.25.1.1 _MAC_BufferReset()	406
6.25.1.2 _MAC_BufGetOldestToTx()	407
6.25.1.3 MAC_AckFramesr()	407
6.25.1.4 MAC_BeaconTimerGetMs()	407
6.25.1.5 MAC_BeaconTimerReset()	408
6.25.1.6 MAC_Buffer()	409
6.25.1.7 MAC_BufferPrepare()	410
6.25.1.8 MAC_BufLen()	411
6.25.1.9 MAC_FillFrameTo()	411
6.25.1.10 MAC_Free()	412
6.25.1.11 MAC_Init()	412
6.25.1.12 MAC_Read()	413
6.25.1.13 MAC_Read8()	414
6.25.1.14 MAC_Send()	414
6.25.1.15 MAC_SendRanging()	415
6.25.1.16 MAC_SetFrameType()	416

6.25.1.17 MAC_ToSlotsTimeUs()	416
6.25.1.18 MAC_TryTransmitFrameInSlot()	417
6.25.1.19 MAC_UpdateSlotTimer()	417
6.25.1.20 MAC_UsFromRx()	418
6.25.1.21 MAC_Write()	419
6.25.1.22 MAC_Write8()	419
6.25.1.23 MAC_YourSlotIsr()	420
6.25.2 Variable Documentation	420
6.25.2.1 mac	420
6.26 mac.h File Reference	421
6.26.1 Detailed Description	423
6.26.2 Macro Definition Documentation	424
6.26.2.1 MAC_HEAD_LENGTH	424
6.26.2.2 MAC_TRACE	424
6.26.2.3 MAC_TRACE_ENABLED	424
6.26.2.4 MAC_USAGE_BUF_START	424
6.26.2.5 MAC_USAGE_BUF_STOP	424
6.26.3 Function Documentation	424
6.26.3.1 listener_isr()	424
6.26.3.2 MAC_AckFrameIsr()	425
6.26.3.3 MAC_BeaconTimerGetMs()	426
6.26.3.4 MAC_BeaconTimerReset()	426
6.26.3.5 MAC_Buffer()	427
6.26.3.6 MAC_BufferPrepare()	428
6.26.3.7 MAC_BufLen()	429
6.26.3.8 MAC_FillFrameTo()	430
6.26.3.9 MAC_Free()	430
6.26.3.10 MAC_Init()	431
6.26.3.11 MAC_Read()	432
6.26.3.12 MAC_Read8()	432

6.26.3.13 MAC_Send()	432
6.26.3.14 MAC_SendRanging()	433
6.26.3.15 MAC_SetFrameType()	434
6.26.3.16 MAC_ToSlotsTime()	435
6.26.3.17 MAC_UsFromRx()	435
6.26.3.18 MAC_Write()	436
6.26.3.19 MAC_Write8()	436
6.26.3.20 MAC_YourSlotIsr()	437
6.27 mac_const.h File Reference	438
6.27.1 Macro Definition Documentation	439
6.27.1.1 ADDR_ANCHOR_FLAG	439
6.27.1.2 ADDR_BROADCAST	439
6.27.1.3 FR_CR_ACK	439
6.27.1.4 FR_CR_BEACON	439
6.27.1.5 FR_CR_DATA	439
6.27.1.6 FR_CR_MAC	440
6.27.1.7 FR_CR_PAN	440
6.27.1.8 FR_CR_PENDING	440
6.27.1.9 FR_CR_RACK	440
6.27.1.10 FR_CR_SECURE	440
6.27.1.11 FR_CR_TYPE_MASK	440
6.27.1.12 FR_CRH_DA_LONG	440
6.27.1.13 FR_CRH_DA_NONE	440
6.27.1.14 FR_CRH_DA_SHORT	441
6.27.1.15 FR_CRH_FVER0	441
6.27.1.16 FR_CRH_FVER1	441
6.27.1.17 FR_CRH_SA_LONG	441
6.27.1.18 FR_CRH_SA_NONE	441
6.27.1.19 FR_CRH_SA_SHORT	441
6.27.2 Typedef Documentation	441

6.27.2.1 dev_addr_t	441
6.27.2.2 pan_dev_addr_t	441
6.27.3 Enumeration Type Documentation	441
6.27.3.1 mac_buf_state	441
6.28 mac_port.h File Reference	442
6.28.1 Macro Definition Documentation	443
6.28.1.1 MAC_ASSERT	443
6.28.1.2 MAC_LOG_ERR	444
6.28.2 Typedef Documentation	444
6.28.2.1 mac_buff_time_t	444
6.29 mac_settings.h File Reference	444
6.29.1 Macro Definition Documentation	446
6.29.1.1 _DEF_SLOT_CNT	446
6.29.1.2 _DEF_SLOT_SUM_TIME	446
6.29.1.3 _DEF_SLOT_TIME	446
6.29.1.4 MAC_BUF_CNT	446
6.29.1.5 MAC_BUF_LEN	446
6.29.1.6 MAC_SETTINGS_DEF	446
6.29.1.7 SYNC_MAC_NEIGHBOURS	447
6.29.1.8 TOA_MAX_DEV_IN_POLL	447
6.29.2 Enumeration Type Documentation	447
6.29.2.1 rtls_role	447
6.30 port.h File Reference	447
6.30.1 Detailed Description	450
6.30.2 Macro Definition Documentation	450
6.30.2.1 DBG	450
6.30.2.2 PORT_ASSERT	450
6.30.3 Function Documentation	450
6.30.3.1 decamutexoff()	450
6.30.3.2 decamutexon()	451

6.30.3.3 PORT_BatteryMeasure()	451
6.30.3.4 PORT_BatteryVoltage()	452
6.30.3.5 PORT_BkpRegisterRead()	452
6.30.3.6 PORT_BkpRegisterWrite()	453
6.30.3.7 PORT_CrcFeed()	454
6.30.3.8 PORT_CrcReset()	454
6.30.3.9 PORT_EnterStopMode()	455
6.30.3.10 PORT_FlashErase()	455
6.30.3.11 PORT_FlashSave()	456
6.30.3.12 PORT_lassert_fun()	456
6.30.3.13 PORT_Init()	457
6.30.3.14 PORT_LedOff()	457
6.30.3.15 PORT_LedOn()	458
6.30.3.16 PORT_Reboot()	459
6.30.3.17 PORT_ResetTransceiver()	459
6.30.3.18 PORT_SetSlotTimerPeriodUs()	459
6.30.3.19 PORT_SleepMs()	460
6.30.3.20 PORT_SlotTimerSetUsOffset()	461
6.30.3.21 PORT_SlotTimerTickUs()	461
6.30.3.22 PORT_SpiSpeedSlow()	462
6.30.3.23 PORT_TickHr()	462
6.30.3.24 PORT_TickHrToUs()	463
6.30.3.25 PORT_TickMs()	463
6.30.3.26 PORT_TimeStartTimers()	464
6.30.3.27 PORT_WakeupTransceiver()	464
6.30.3.28 PORT_WatchdogInit()	465
6.30.3.29 PORT_WatchdogRefresh()	465
6.30.3.30 readfromspi()	465
6.30.3.31 writetospi()	466
6.31 printer.c File Reference	466

6.31.1 Function Documentation	467
6.31.1.1 PRINT_Stat()	467
6.31.1.2 PRINT_Version()	468
6.32 printer.h File Reference	468
6.32.1 Detailed Description	469
6.32.2 Function Documentation	470
6.32.2.1 PRINT_Stat()	470
6.32.2.2 PRINT_Version()	470
6.33 README.md File Reference	471
6.34 settings.c File Reference	471
6.34.1 Function Documentation	472
6.34.1.1 SETTINGS_Init()	472
6.34.1.2 settings_is_otp_erased()	472
6.34.2 Variable Documentation	473
6.34.2.1 _settings_otp	473
6.34.2.2 settings	473
6.34.2.3 settings_otp	473
6.35 settings.h File Reference	474
6.35.1 Macro Definition Documentation	475
6.35.1.1 __H_VERSION__	475
6.35.1.2 DEF_SETTINGS	476
6.35.1.3 H_MAJOR_CALC	476
6.35.1.4 H_VERSION_CALC	476
6.35.1.5 VERSION_SETTINGS_DEF	477
6.35.2 Function Documentation	477
6.35.2.1 SETTINGS_Init()	477
6.35.3 Variable Documentation	478
6.35.3.1 settings	478
6.35.3.2 settings_otp	478
6.36 sync.c File Reference	478

6.36.1 Macro Definition Documentation	479
6.36.1.1 MOVE_ARRAY_ELEM	479
6.36.1.2 PROT_CHECK_LEN	480
6.36.2 Function Documentation	480
6.36.2.1 FC_SYNC_FIN_cb()	480
6.36.2.2 FC_SYNC_POLL_cb()	481
6.36.2.3 FC_SYNC_RESP_cb()	481
6.36.2.4 SYNC_CalcTimeCoeff()	482
6.36.2.5 SYNC_FindOrCreateNeighbour()	483
6.36.2.6 SYNC_GlobTime()	483
6.36.2.7 SYNC_GlobTimeNeig()	484
6.36.2.8 SYNC_Init()	485
6.36.2.9 SYNC_InitNeighbour()	485
6.36.2.10 SYNC_RxCb()	485
6.36.2.11 SYNC_RxToCb()	486
6.36.2.12 SYNC_SendFinal()	487
6.36.2.13 SYNC_SendPoll()	488
6.36.2.14 SYNC_SendResp()	488
6.36.2.15 SYNC_Smooth()	489
6.36.2.16 SYNC_TrimDrift()	489
6.36.2.17 SYNC_TxCb()	490
6.36.2.18 SYNC_Update()	490
6.36.2.19 SYNC_UpdateLocalTimeParams()	491
6.36.2.20 SYNC_UpdateNeighbour()	491
6.36.3 Variable Documentation	492
6.36.3.1 bad_len_msg	492
6.36.3.2 mac	492
6.36.3.3 sync	492
6.37 sync.h File Reference	492
6.37.1 Detailed Description	494

6.37.2 Macro Definition Documentation	495
6.37.2.1 SYNC_ASSERT	495
6.37.2.2 SYNC_TIME_DUMP	495
6.37.2.3 SYNC_TIME_DUMP_ENABLED	495
6.37.2.4 SYNC_TRACE	495
6.37.2.5 SYNC_TRACE_ENABLED	495
6.37.2.6 SYNC_TRACE_TOA	495
6.37.2.7 SYNC_TRACE_TOA_ENABLED	496
6.37.3 Function Documentation	496
6.37.3.1 SYNC_FindOrCreateNeighbour()	496
6.37.3.2 SYNC_GlobTime()	497
6.37.3.3 SYNC_Init()	498
6.37.3.4 SYNC_RxCb()	498
6.37.3.5 SYNC_RxToCb()	499
6.37.3.6 SYNC_SendPoll()	500
6.37.3.7 SYNC_TxCb()	500
6.38 toa.c File Reference	501
6.38.1 Function Documentation	502
6.38.1.1 _TOA_GetRangeBias()	502
6.38.1.2 TOA_AddMeasure()	503
6.38.1.3 TOA_CalcTofconst()	503
6.38.1.4 TOA_CalcTofDwTu()	504
6.38.1.5 TOA_EnableRxBeforeFin()	504
6.38.1.6 TOA_FindAddrIndexInResp()	505
6.38.1.7 TOA_GetRangeBias()	506
6.38.1.8 TOA_Read40bValue()	506
6.38.1.9 TOA_SetTxTime()	507
6.38.1.10 TOA_State()	507
6.38.1.11 TOA_TofToCm()	508
6.38.1.12 TOA_Write40bValue()	509

6.39 toa.h File Reference	509
6.39.1 Macro Definition Documentation	511
6.39.1.1 SPEED_OF_LIGHT	511
6.39.2 Enumeration Type Documentation	512
6.39.2.1 toa_state_t	512
6.39.3 Function Documentation	512
6.39.3.1 TOA_AddMeasure()	512
6.39.3.2 TOA_CalcTof()	513
6.39.3.3 TOA_CalcTofDwTu()	513
6.39.3.4 TOA_EnableRxBeforeFin()	514
6.39.3.5 TOA_FindAddrIndexInResp()	515
6.39.3.6 TOA_Read40bValue()	515
6.39.3.7 TOA_SetTxTime()	516
6.39.3.8 TOA_State()	517
6.39.3.9 TOA_TofToCm()	518
6.39.3.10 TOA_Write40bValue()	518
6.40 toa_calib.c File Reference	519
6.40.1 Macro Definition Documentation	520
6.40.1.1 CM_OFFSET_16M_NB	520
6.40.1.2 CM_OFFSET_16M_WB	520
6.40.1.3 CM_OFFSET_64M_NB	520
6.40.1.4 CM_OFFSET_64M_WB	520
6.40.1.5 NUM_16M_OFFSET	520
6.40.1.6 NUM_16M_OFFSETWB	521
6.40.1.7 NUM_64M_OFFSET	521
6.40.1.8 NUM_64M_OFFSETWB	521
6.40.2 Function Documentation	521
6.40.2.1 _TOA_GetRangeBias()	521
6.40.3 Variable Documentation	522
6.40.3.1 chan_idxnb	522

6.40.3.2 chan_idxwb	522
6.40.3.3 range25cm16PRFnb	522
6.40.3.4 range25cm16PRFwb	523
6.40.3.5 range25cm64PRFnb	523
6.40.3.6 range25cm64PRFwb	524
6.41 tools.c File Reference	524
6.42 tools.h File Reference	524
6.42.1 Detailed Description	526
6.42.2 Macro Definition Documentation	526
6.42.2.1 ALL_UNUSED	526
6.42.2.2 ALL_UNUSED_IMPL	526
6.42.2.3 ALL_UNUSED_IMPL_	526
6.42.2.4 DECREMENT_CYCLE	526
6.42.2.5 DECREMENT_MOD	527
6.42.2.6 INCREMENT_CYCLE	527
6.42.2.7 INCREMENT_MOD	527
6.42.2.8 UNUSED_1	527
6.42.2.9 UNUSED_2	527
6.42.2.10 UNUSED_3	528
6.42.2.11 UNUSED_4	528
6.42.2.12 UNUSED_5	528
6.42.2.13 UNUSED_6	528
6.42.2.14 UNUSED_7	528
6.42.2.15 UNUSED_8	529
6.42.2.16 VA_NUM_ARGS	529
6.42.2.17 VA_NUM_ARGS_IMPL	529
6.43 transceiver.c File Reference	529
6.43.1 Function Documentation	531
6.43.1.1 TRANSCEIVER_DefaultRx()	531
6.43.1.2 TRANSCEIVER_EnterDeepSleep()	532

6.43.1.3	TRANSCEIVER_EstimateTxTimeUs()	532
6.43.1.4	TRANSCEIVER_GetRxTimestamp()	533
6.43.1.5	TRANSCEIVER_GetTime()	534
6.43.1.6	TRANSCEIVER_GetTxTimestamp()	534
6.43.1.7	TRANSCEIVER_Init()	535
6.43.1.8	TRANSCEIVER_Read()	535
6.43.1.9	TRANSCEIVER_ReadDiagnostic()	536
6.43.1.10	TRANSCEIVER_Send()	537
6.43.1.11	TRANSCEIVER_SendRanging()	537
6.43.1.12	TRANSCEIVER_SetAddr()	538
6.43.1.13	TRANSCEIVER_SetCb()	539
6.43.1.14	TRANSCEIVER_WakeUp()	540
6.43.2	Variable Documentation	540
6.43.2.1	transceiver_br	540
6.43.2.2	transceiver_pac	540
6.43.2.3	transceiver_plen	541
6.43.2.4	transceiver_sfd	541
6.44	transceiver.h File Reference	541
6.44.1	Detailed Description	543
6.44.2	Macro Definition Documentation	543
6.44.2.1	MASK_40BIT	543
6.44.2.2	UUS_TO_DWT_TIME	543
6.44.3	Function Documentation	543
6.44.3.1	TRANSCEIVER_DefaultRx()	544
6.44.3.2	TRANSCEIVER_EnterDeepSleep()	544
6.44.3.3	TRANSCEIVER_EstimateTxTimeUs()	545
6.44.3.4	TRANSCEIVER_GetRxTimestamp()	545
6.44.3.5	TRANSCEIVER_GetTime()	546
6.44.3.6	TRANSCEIVER_GetTxTimestamp()	547
6.44.3.7	TRANSCEIVER_Init()	548

6.44.3.8 TRANSCEIVER_Read()	548
6.44.3.9 TRANSCEIVER_ReadDiagnostic()	549
6.44.3.10 TRANSCEIVER_Send()	549
6.44.3.11 TRANSCEIVER_SendRanging()	550
6.44.3.12 TRANSCEIVER_SetAddr()	551
6.44.3.13 TRANSCEIVER_SetCb()	552
6.45 transceiver_settings.h File Reference	552
6.45.1 Detailed Description	554
6.45.2 Macro Definition Documentation	555
6.45.2.1 TRANSCEIVER_ASSERT	555
6.45.2.2 TRANSCEIVER_SETTINGS_DEF	555
6.46 transceiver_translator.c File Reference	555
6.46.1 Function Documentation	556
6.46.1.1 deca_sleep()	556
6.47 txt_parser.c File Reference	557
6.47.1 Function Documentation	557
6.47.1.1 TXT_Atol()	558
6.47.1.2 TXT_Control()	558
6.47.1.3 TXT_GetParam()	559
6.47.1.4 TXT_Input()	559
6.47.1.5 TXT_Parse()	560
6.47.1.6 TXT_PointParamNumber()	560
6.47.1.7 TXT_StartsWith()	561
6.47.2 Variable Documentation	561
6.47.2.1 _txt_buf_raw	561
6.48 txt_parser.h File Reference	561
6.48.1 Detailed Description	563
6.48.2 Typedef Documentation	563
6.48.2.1 cchar	563
6.48.2.2 txt_parser_cb	563

6.48.3 Function Documentation	563
6.48.3.1 TXT_Atol()	563
6.48.3.2 TXT_Control()	564
6.48.3.3 TXT_GetParam()	564
6.48.3.4 TXT_Input()	565
6.48.3.5 TXT_PointParamNumber()	566
6.48.3.6 TXT_StartsWith()	566
6.49 txt_parser_cb.c File Reference	567
6.49.1 Function Documentation	567
6.49.1.1 _TXT_Ask()	568
6.49.1.2 _TXT_Finalize()	568
6.49.1.3 TXT_HangCb()	569
6.49.1.4 TXT_RFSet()	569
6.49.1.5 TXT_StatCb()	570
6.49.1.6 TXT_TestCb()	570
6.49.1.7 TXT_VersionCb()	570
6.49.2 Variable Documentation	571
6.49.2.1 txt_cb_len	571
6.49.2.2 txt_cb_tab	571
6.50 uwb_main.c File Reference	571
6.50.1 Function Documentation	572
6.50.1.1 BatteryControl()	572
6.50.1.2 BeaconSender()	573
6.50.1.3 CheckSleepMode()	573
6.50.1.4 Desynchronize()	574
6.50.1.5 diagnostic()	574
6.50.1.6 RangingControl()	575
6.50.1.7 SendBeaconMessage()	575
6.50.1.8 SendTurnOffMessage()	576
6.50.1.9 SendTurnOnMessage()	576
6.50.1.10 str_append()	577
6.50.1.11 TurnOff()	577
6.50.1.12 UwbMain()	578
6.51 uwb_main.h File Reference	578
6.51.1 Detailed Description	579
6.51.2 Function Documentation	580
6.51.2.1 UwbMain()	580

Chapter 1

Readme file

short description

- **Code organization**

Code is divided into platform dependent and independent part. First one is in folder platform and each of those function has prefix 'PORT_', rest of files comes from second part.

Huge part of code is called from interrupts and there is also main program loop for functionalities without process time restrictions.

To build code GNU cross tool chain has been used.

Interrupts

There should be implemented interrupt from:

1. DW1000 IRQ pin
1. Slot timer timeout

Code style

```
int PREFIX_FunctionName(int arg_name1, int arg_name2);
```

Files names should contains only small letters, underscores, dot and number.

Bootloader

Firmware upgrade functionality need to have installed special bootloader on the target platform. Moreover there is a few variables imported from linker script to firmware upgrade module.

Authors

Hardware:

1. L4V0.1 - Karol Trzciński

1. L4V0.2 - Karol Trzciński

1. L4V0.3 - Karol Trzciński

1. L4V0.4 - Karol Trzciński

1. L4V0.5 - Karol Trzciński

1. NrfV1.1 - DevaWave

1. AnV2.0 - Andrzej Kuczwara

Firmware:

1. Karol Trzciński:

- system architecture
- DW1000 usage
- trilateration algorithms
- radio protocols

2. Dawid Pepliński:

- inertial navigation

Chapter 2

Deprecated List

Global CARRY_MAX_TARGETS

change structure to anchor->parent

Class carry_trace_t

use anchor->parent model

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

<code>agc_cfg_struct</code>	9
<code>carry_instance_t</code>	
Global singleton	10
<code>carry_settings_t</code>	
Carry settings typedef	11
<code>carry_target_t</code>	
Target info struct	12
<code>carry_trace_t</code>	
Data about some trace to target	13
<code>dwt_cb_data_t</code>	15
<code>dwt_config_t</code>	15
<code>dwt_deviceentcnts_t</code>	18
<code>dwt_local_data_t</code>	20
<code>dwt_rxdiag_t</code>	23
<code>dwt_txconfig_t</code>	25
<code>FC_BEACON_s</code>	
See <code>FC_t</code> description	25
<code>FC_CARRY_s</code>	
Protocol struct	26
<code>FC_STAT_s</code>	
See <code>FC_t</code> description	28
<code>FC_SYNC_FIN_s</code>	29
<code>FC_SYNC_POLL_s</code>	31
<code>FC_SYNC_RESP_s</code>	32
<code>FC_TURN_OFF_s</code>	
See <code>FC_t</code> description	33
<code>FC_TURN_ON_s</code>	
See <code>FC_t</code> description	34
<code>FC_VERSION_s</code>	
See <code>FC_t</code> description	35
<code>FU_instance_t</code>	37
<code>FU_prot</code>	
Raw firmware upgrade protocol struct	38
<code>FU_SOT_prot</code>	
Start of frame struct	40

mac_buf_t	42
mac_instance_t	45
mac_settings_t	47
prot_cb_t	
Struct of function code callbacks	50
prot_packet_info_t	
Packet extra information struct	51
settings_otp_t	
Internal One Time Programmable settings structure	52
settings_t	
Main setting container	53
settings_version_t	
Current firmware and hardware description structure	55
sync_instance_t	56
sync_neighbour_t	58
toa_core_t	60
toa_settings_t	62
transceiver_settings_t	
Transceiver settings	64
txt_buf_t	
Text parser engine circular buffer structure	65
txt_cb_t	
Text parser engine callback struct	66

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

base64.c	69
base64.h	
Base64 coder and decoder in C	71
bin_const.h	
Define numerical function codes for binary data	73
bin_parser.c	75
bin_parser.h	
Binary data parser engine	77
bin_parser_cb.c	81
bin_struct.h	
Binary data structures definition	85
carry.c	87
carry.h	
Carry is a transport/routing layer	92
carry_const.h	98
carry_settings.h	
Carry settings typedef and default values	100
deca_device.c	
Decawave device configuration and control functions	103
deca_device_api.h	
DW1000 API Functions	156
deca_param_types.h	
Decawave general type definitions for configuration structures	220
deca_params_init.c	
DW1000 configuration parameters	226
deca_regs.h	
DW1000 Register Definitions This file supports assembler and C development for DW1000 enabled devices	230
deca_types.h	
Decawave general type definitions	374
deca_version.h	
Defines the version info for the DW1000 device driver including its API	377
fu.c	378
fu.h	
Firmware upgrade protocol	381

gitversion.h	395
iassert.h	396
listener_parser.c	397
logs.h	
Interactive assertion module	400
mac.c	404
mac.h	
Medium Access Control layer 802.15.4	421
mac_const.h	438
mac_port.h	442
mac_settings.h	444
port.h	
Portable module, strictly hardware dependent	447
printer.c	466
printer.h	
Common messages printers	468
settings.c	471
settings.h	474
sync.c	478
sync.h	
Time synchronization module	492
toa.c	501
toa.h	509
toa_calib.c	519
tools.c	524
tools.h	
File for small code snippets	524
transceiver.c	529
transceiver.h	
Transceiver	541
transceiver_settings.h	
Transceiver settings struct typedef with default values	552
transceiver_translator.c	555
txt_parser.c	557
txt_parser.h	
Text parser engine	561
txt_parser_cb.c	567
uwb_main.c	571
uwb_main.h	
Main core function	578

Chapter 5

Data Structure Documentation

5.1 agc_cfg_struct Struct Reference

```
#include <deca_param_types.h>
```

Data Fields

- `uint32 lo32`
- `uint16 target [NUM_PRF]`

5.1.1 Field Documentation

5.1.1.1 `lo32`

```
uint32 lo32
```

5.1.1.2 `target`

```
uint16 target [NUM_PRF]
```

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

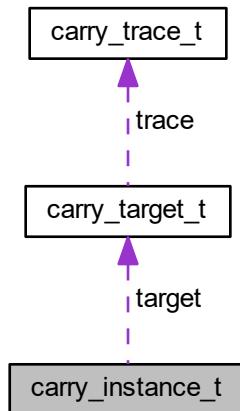
- `deca_param_types.h`

5.2 carry_instance_t Struct Reference

global singleton

```
#include <carry.h>
```

Collaboration diagram for carry_instance_t:



Data Fields

- `carry_target_t target [CARRY_MAX_TARGETS]`
- `bool isConnectedToServer`
- `dev_addr_t toSinkId`

5.2.1 Detailed Description

global singleton

5.2.2 Field Documentation

5.2.2.1 isConnectedToServer

```
bool isConnectedToServer
```

5.2.2.2 target

```
carry_target_t target[CARRY_MAX_TARGETS]
```

5.2.2.3 toSinkId

```
dev_addr_t toSinkId
```

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

- [carry.h](#)

5.3 carry_settings_t Struct Reference

carry settings typedef

```
#include <carry_settings.h>
```

Data Fields

- `mac_buff_time_t trace_max_inactive_time`
max time from last message to be kept
- `int trace_max_fail_cnt`
trace retransmit/delete threshold

5.3.1 Detailed Description

carry settings typedef

5.3.2 Field Documentation

5.3.2.1 trace_max_fail_cnt

```
int trace_max_fail_cnt
```

trace retransmit/delete threshold

5.3.2.2 trace_max_inactive_time

```
mac_buff_time_t trace_max_inactive_time
```

max time from last message to be kept

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

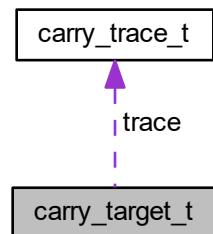
- [carry_settings.h](#)

5.4 carry_target_t Struct Reference

target info struct

```
#include <carry.h>
```

Collaboration diagram for carry_target_t:



Data Fields

- [dev_addr_t addr](#)
- [carry_trace_t trace \[CARRY_MAX_TRACE\]](#)
- [mac_buff_time_t last_update_time](#)

5.4.1 Detailed Description

target info struct

5.4.2 Field Documentation

5.4.2.1 addr

```
dev_addr_t addr
```

5.4.2.2 last_update_time

```
mac_buff_time_t last_update_time
```

5.4.2.3 trace

```
carry_trace_t trace[CARRY_MAX_TRACE]
```

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

- [carry.h](#)

5.5 carry_trace_t Struct Reference

data about some trace to target

```
#include <carry.h>
```

Data Fields

- [dev_addr_t path \[CARRY_MAX_HOPS\]](#)
trace path
- [int path_len](#)
number of hops in path
- [int pass_cnt](#)
number of successful packet transmission via this path
- [int fail_cnt](#)
number of failed packet transmission via this path
- [mac_buff_time_t last_update_time](#)
trace last active time

5.5.1 Detailed Description

data about some trace to target

Deprecated use anchor->parent model

5.5.2 Field Documentation

5.5.2.1 fail_cnt

```
int fail_cnt
```

number of failed packet transmission via this path

5.5.2.2 last_update_time

```
mac_buff_time_t last_update_time
```

trace last active time

5.5.2.3 pass_cnt

```
int pass_cnt
```

number of successful packet transmission via this path

5.5.2.4 path

```
dev_addr_t path[CARRY_MAX_HOPS]
```

trace path

5.5.2.5 path_len

```
int path_len
```

number of hops in path

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

- [carry.h](#)

5.6 dwt_cb_data_t Struct Reference

```
#include <deca_device_api.h>
```

Data Fields

- `uint32 status`
- `uint16 datalength`
- `uint8 fctrl [2]`
- `uint8 rx_flags`

5.6.1 Field Documentation

5.6.1.1 `datalength`

```
uint16 datalength
```

5.6.1.2 `fctrl`

```
uint8 fctrl[2]
```

5.6.1.3 `rx_flags`

```
uint8 rx_flags
```

5.6.1.4 `status`

```
uint32 status
```

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

- `deca_device_api.h`

5.7 dwt_config_t Struct Reference

```
#include <deca_device_api.h>
```

Data Fields

- `uint8 chan`
`channel number {1, 2, 3, 4, 5, 7}`
- `uint8 prf`
`Pulse Repetition Frequency {DWT_PRF_16M or DWT_PRF_64M}.`
- `uint8 txPreambLength`
`DWT_PLLEN_64..DWT_PLLEN_4096.`
- `uint8 rxPAC`
`Acquisition Chunk Size (Relates to RX preamble length)`
- `uint8 txCode`
`TX preamble code.`
- `uint8 rxCode`
`RX preamble code.`
- `uint8 nsSFD`
`Boolean should we use non-standard SFD for better performance.`
- `uint8 dataRate`
`Data Rate {DWT_BR_110K, DWT_BR_850K or DWT_BR_6M8}.`
- `uint8 phrMode`
`PHR mode {0x0 - standard DWT_PHRMODE_STD, 0x3 - extended frames DWT_PHRMODE_EXT}.`
- `uint16 sfdTO`
`SFD timeout value (in symbols)`

5.7.1 Detailed Description

Structure typedef: `dwt_config_t`

Structure for setting device configuration via `dwt_configure()` function

5.7.2 Field Documentation

5.7.2.1 chan

`uint8 chan`

`channel number {1, 2, 3, 4, 5, 7 }`

5.7.2.2 dataRate

`uint8 dataRate`

`Data Rate {DWT_BR_110K, DWT_BR_850K or DWT_BR_6M8}.`

5.7.2.3 nsSFD

```
uint8 nsSFD
```

Boolean should we use non-standard SFD for better performance.

5.7.2.4 phrMode

```
uint8 phrMode
```

PHR mode {0x0 - standard DWT_PHRMODE_STD, 0x3 - extended frames DWT_PHRMODE_EXT}.

5.7.2.5 prf

```
uint8 prf
```

Pulse Repetition Frequency {DWT_PRF_16M or DWT_PRF_64M}.

5.7.2.6 rxCode

```
uint8 rxCode
```

RX preamble code.

5.7.2.7 rxPAC

```
uint8 rxPAC
```

Acquisition Chunk Size (Relates to RX preamble length)

5.7.2.8 sfdTO

```
uint16 sfdTO
```

SFD timeout value (in symbols)

5.7.2.9 txCode

```
uint8 txCode
```

TX preamble code.

5.7.2.10 txPreambLength

```
uint8 txPreambLength
```

DWT_PLEN_64..DWT_PLEN_4096.

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

- [deca_device_api.h](#)

5.8 dwt_deviceentcnts_t Struct Reference

```
#include <deca_device_api.h>
```

Data Fields

- [uint16 PHE](#)
- [uint16 RSL](#)
- [uint16 CRCG](#)
- [uint16 CRCB](#)
- [uint16 ARFE](#)
- [uint16 OVER](#)
- [uint16 SFDTO](#)
- [uint16 PTO](#)
- [uint16 RTO](#)
- [uint16 TXF](#)
- [uint16 HPW](#)
- [uint16 TXW](#)

5.8.1 Field Documentation

5.8.1.1 ARFE

```
uint16 ARFE
```

5.8.1.2 CRCB

```
uint16 CRCB
```

5.8.1.3 CRCG

```
uint16 CRCG
```

5.8.1.4 HPW

```
uint16 HPW
```

5.8.1.5 OVER

```
uint16 OVER
```

5.8.1.6 PHE

```
uint16 PHE
```

5.8.1.7 PTO

```
uint16 PTO
```

5.8.1.8 RSL

```
uint16 RSL
```

5.8.1.9 RTO

```
uint16 RTO
```

5.8.1.10 SFDTO

```
uint16 SFDTO
```

5.8.1.11 TXF

```
uint16 TXF
```

5.8.1.12 TXW

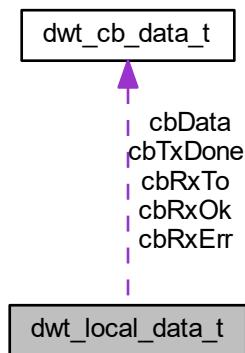
```
uint16 TXW
```

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

- [deca_device_api.h](#)

5.9 dwt_local_data_t Struct Reference

Collaboration diagram for dwt_local_data_t:



Data Fields

- `uint32 partID`
- `uint32 lotID`
- `uint8 longFrames`
- `uint8 otprev`
- `uint32 txFCTRL`
- `uint8 init_xtrim`
- `uint8 dblbuffon`
- `uint32 sysCFGreg`
- `uint16 sleep_mode`
- `uint8 wait4resp`
- `dwt_cb_data_t cbData`
- `dwt_cb_t cbTxDone`
- `dwt_cb_t cbRxOk`
- `dwt_cb_t cbRxTo`
- `dwt_cb_t cbRxErr`

5.9.1 Detailed Description

Static data for DW1000 DecaWave Transceiver control

5.9.2 Field Documentation

5.9.2.1 cbData

`dwt_cb_data_t cbData`

5.9.2.2 cbRxErr

`dwt_cb_t cbRxErr`

5.9.2.3 cbRxOk

`dwt_cb_t cbRxOk`

5.9.2.4 cbRxTo

```
dwt_cb_t cbRxTo
```

5.9.2.5 cbTxDone

```
dwt_cb_t cbTxDone
```

5.9.2.6 dblbuffon

```
uint8 dblbuffon
```

5.9.2.7 init_xtrim

```
uint8 init_xtrim
```

5.9.2.8 longFrames

```
uint8 longFrames
```

5.9.2.9 lotID

```
uint32 lotID
```

5.9.2.10 otprev

```
uint8 otprev
```

5.9.2.11 partID

```
uint32 partID
```

5.9.2.12 sleep_mode

```
uint16 sleep_mode
```

5.9.2.13 sysCFGreg

```
uint32 sysCFGreg
```

5.9.2.14 txFCTRL

```
uint32 txFCTRL
```

5.9.2.15 wait4resp

```
uint8 wait4resp
```

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

- [deca_device.c](#)

5.10 dwt_rxdiag_t Struct Reference

```
#include <deca_device_api.h>
```

Data Fields

- `uint16 maxNoise`
- `uint16 firstPathAmp1`
- `uint16 stdNoise`
- `uint16 firstPathAmp2`
- `uint16 firstPathAmp3`
- `uint16 maxGrowthCIR`
- `uint16 rxPreamCount`
- `uint16 firstPath`

5.10.1 Field Documentation

5.10.1.1 firstPath

```
uint16 firstPath
```

5.10.1.2 firstPathAmp1

```
uint16 firstPathAmp1
```

5.10.1.3 firstPathAmp2

```
uint16 firstPathAmp2
```

5.10.1.4 firstPathAmp3

```
uint16 firstPathAmp3
```

5.10.1.5 maxGrowthCIR

```
uint16 maxGrowthCIR
```

5.10.1.6 maxNoise

```
uint16 maxNoise
```

5.10.1.7 rxPreamCount

```
uint16 rxPreamCount
```

5.10.1.8 stdNoise

```
uint16 stdNoise
```

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

- [deca_device_api.h](#)

5.11 dwt_txconfig_t Struct Reference

```
#include <deca_device_api.h>
```

Data Fields

- [uint8 PGdly](#)
- [uint32 power](#)

5.11.1 Field Documentation

5.11.1.1 PGdly

```
uint8 PGdly
```

5.11.1.2 power

```
uint32 power
```

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

- [deca_device_api.h](#)

5.12 FC_BEACON_s Struct Reference

see [FC_t](#) description

```
#include <bin_struct.h>
```

Data Fields

- `uint8_t FC`
- `uint8_t len`
- `uint64_t serial`
device serial number from settings.version.serial

5.12.1 Detailed Description

see [FC_t](#) description

5.12.2 Field Documentation

5.12.2.1 FC

`uint8_t FC`

5.12.2.2 len

`uint8_t len`

5.12.2.3 serial

`uint64_t serial`

device serial number from settings.version.serial

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

- [bin_struct.h](#)

5.13 FC_CARRY_s Struct Reference

protocol struct

```
#include <carry.h>
```

Data Fields

- `unsigned char flag_hops`
extra flags field, CARRY_FLAG_xx
- `dev_addr_t src_addr`
source address
- `dev_addr_t hops [0]`
destination .. next hop

5.13.1 Detailed Description

protocol struct

5.13.2 Field Documentation

5.13.2.1 flag_hops

`unsigned char flag_hops`
extra flags field, CARRY_FLAG_xx

5.13.2.2 hops

`dev_addr_t hops [0]`
destination .. next hop

5.13.2.3 src_addr

`dev_addr_t src_addr`
source address

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

- `carry.h`

5.14 FC_STAT_s Struct Reference

see [FC_t](#) description

```
#include <bin_struct.h>
```

Data Fields

- `uint8_t FC`
- `uint8_t len`
- `uint16_t battery_mV`
last measured battery voltage in millivolt
- `uint16_t err_cnt`
radio transmission and reception error counter
- `uint16_t to_cnt`
radio transmission timeout counter
- `uint16_t rx_cnt`
radio reception counter
- `uint16_t tx_cnt`
radio successful transmission counter

5.14.1 Detailed Description

see [FC_t](#) description

5.14.2 Field Documentation

5.14.2.1 battery_mV

```
uint16_t battery_mV
```

last measured battery voltage in millivolt

5.14.2.2 err_cnt

```
uint16_t err_cnt
```

radio transmission and reception error counter

5.14.2.3 FC

uint8_t FC

5.14.2.4 len

uint8_t len

5.14.2.5 rx_cnt

uint16_t rx_cnt

radio reception counter

5.14.2.6 to_cnt

uint16_t to_cnt

radio transmission timeout counter

5.14.2.7 tx_cnt

uint16_t tx_cnt

radio successful transmission counter

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

- [bin_struct.h](#)

5.15 FC_SYNC_FIN_s Struct Reference

```
#include <sync.h>
```

Data Fields

- `uint8_t FC`
- `uint8_t len`
- `uint8_t tree_level`
device level in tree
- `uint8_t slot_num`
device slot number
- `uint32_t TsPollTx`
poll transmit timestamp in dw time units
- `uint8_t TsFinTxBuf [5]`
40b fin transmit timestamp in dw time units
- `uint8_t TsOffset [5]`
40b device local to global time offset
- `uint32_t TsRespRx [0]`
list of response receive timestamps in dtu

5.15.1 Field Documentation

5.15.1.1 FC

`uint8_t FC`

5.15.1.2 len

`uint8_t len`

5.15.1.3 slot_num

`uint8_t slot_num`

device slot number

5.15.1.4 tree_level

`uint8_t tree_level`

device level in tree

5.15.1.5 TsFinTxBuf

uint8_t TsFinTxBuf[5]

40b fin transmit timestamp in dw time units

5.15.1.6 TsOffset

uint8_t TsOffset[5]

40b device local to global time offset

5.15.1.7 TsPollTx

uint32_t TsPollTx

poll transmit timestamp in dw time units

5.15.1.8 TsRespRx

uint32_t TsRespRx[0]

list of response receive timestamps in dtu

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

- [sync.h](#)

5.16 FC_SYNC_POLL_s Struct Reference

```
#include <sync.h>
```

Data Fields

- uint8_t FC
- uint8_t len
- uint8_t num_poll_anchor
 - number of addresses in array poll_addr*
- [dev_addr_t poll_addr \[0\]](#)
 - list of anchors addresses to poll*

5.16.1 Field Documentation

5.16.1.1 FC

```
uint8_t FC
```

5.16.1.2 len

```
uint8_t len
```

5.16.1.3 num_poll_anchor

```
uint8_t num_poll_anchor
```

number of addresses in array poll_addr

5.16.1.4 poll_addr

```
dev_addr_t poll_addr[0]
```

list of anchors addresses to poll

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

- [sync.h](#)

5.17 FC_SYNC_RESP_s Struct Reference

```
#include <sync.h>
```

Data Fields

- uint8_t FC
- uint8_t len
- uint32_t TsPollRx
 - poll receive timestamp in dw time units*
- uint32_t TsRespTx
 - response transmit timestamp in dw time units*

5.17.1 Field Documentation

5.17.1.1 FC

```
uint8_t FC
```

5.17.1.2 len

```
uint8_t len
```

5.17.1.3 TsPollRx

```
uint32_t TsPollRx
```

poll receive timestamp in dw time units

5.17.1.4 TsRespTx

```
uint32_t TsRespTx
```

response transmit timestamp in dw time units

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

- [sync.h](#)

5.18 FC_TURN_OFF_s Struct Reference

see [FC_t](#) description

```
#include <bin_struct.h>
```

Data Fields

- `uint8_t FC`
- `uint8_t len`
- `uint8_t reason`

reason for turn OFF

5.18.1 Detailed Description

see [FC_t](#) description

5.18.2 Field Documentation

5.18.2.1 FC

```
uint8_t FC
```

5.18.2.2 len

```
uint8_t len
```

5.18.2.3 reason

```
uint8_t reason
```

reason for turn OFF

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

- [bin_struct.h](#)

5.19 FC_TURN_ON_s Struct Reference

see [FC_t](#) description

```
#include <bin_struct.h>
```

Data Fields

- `uint8_t FC`
- `uint8_t len`

5.19.1 Detailed Description

see [FC_t](#) description

5.19.2 Field Documentation

5.19.2.1 FC

```
uint8_t FC
```

5.19.2.2 len

```
uint8_t len
```

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

- [bin_struct.h](#)

5.20 FC_VERSION_s Struct Reference

see [FC_t](#) description

```
#include <bin_struct.h>
```

Data Fields

- `uint8_t FC`
hardware major version
- `uint8_t len`
hardware minor version
- `uint8_t hMajor`
hardware major version
- `uint8_t hMinor`
hardware minor version
- `uint8_t role`
device role, rtls_role
- `uint8_t fMajor`
firmware major version
- `uint16_t fMinor`
firmware minor version
- `uint64_t serial`
device serial number
- `uint64_t hash`
firmware git hash commit

5.20.1 Detailed Description

see [FC_t](#) description

5.20.2 Field Documentation

5.20.2.1 FC

`uint8_t FC`

5.20.2.2 fMajor

`uint8_t fMajor`

firmware major version

5.20.2.3 fMinor

`uint16_t fMinor`

firmware minor version

5.20.2.4 hash

`uint64_t hash`

firmware git hash commit

5.20.2.5 hMajor

`uint8_t hMajor`

hardware major version

5.20.2.6 hMinor

`uint8_t hMinor`

hardware minor version

5.20.2.7 len

```
uint8_t len
```

5.20.2.8 role

```
uint8_t role
```

device role, [rtls_role](#)

5.20.2.9 serial

```
uint64_t serial
```

device serial number

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

- [bin_struct.h](#)

5.21 FU_instance_t Struct Reference

Data Fields

- int [sesionPacketCounter](#)
- uint16_t [newCrc](#)
- uint32_t [newVer](#)
- uint8_t [newHash](#)
- uint32_t [fileSize](#)

5.21.1 Field Documentation

5.21.1.1 fileSize

```
uint32_t fileSize
```

5.21.1.2 newCrc

```
uint16_t newCrc
```

5.21.1.3 newHash

```
uint8_t newHash
```

5.21.1.4 newVer

```
uint32_t newVer
```

5.21.1.5 sesionPacketCounter

```
int sesionPacketCounter
```

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

- [fu.c](#)

5.22 FU_prot Struct Reference

raw firmware upgrade protocol struct

```
#include <fu.h>
```

Data Fields

- `uint8_t FC`
function code FU
- `uint8_t frameLen`
- `uint8_t opcode`
- `uint8_t hash`
mix of hardware and software version in this packet
- `uint16_t extra`
- `uint8_t data [FU_MAX_DATA_SIZE]`
the end

5.22.1 Detailed Description

raw firmware upgrade protocol struct

definition of field extra may vary depending on opcode

5.22.2 Field Documentation

5.22.2.1 data

```
uint8_t data[FU_MAX_DATA_SIZE]
```

the end

extra data and two bytes of frame CRC at

5.22.2.2 extra

```
uint16_t extra
```

BigEndian, optional 16 bit data field with value depending from opcode

5.22.2.3 FC

```
uint8_t FC
```

function code FU

5.22.2.4 frameLen

```
uint8_t frameLen
```

size in bytes from start of [FU_prot](#) address to last byte of CRC

5.22.2.5 hash

```
uint8_t hash
```

mix of hardware and software version in this packet

5.22.2.6 opcode

```
uint8_t opcode
```

operation code[lobyte] and protocol version [hibyte]

See also

- macro with FU_OPCODE_ prefix
- SFU_MakeOpcode

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

- [fu.h](#)

5.23 FU_SOT_prot Struct Reference

start of frame struct

```
#include <fu.h>
```

Data Fields

- uint8_t **FC**
function code FU
- uint8_t **frameLen**
- uint8_t **opcode**
operation code, FU_OPCODE_SOT
- uint8_t **hash**
mix of hardware and software version in this packet
- uint32_t **fversion**
- uint16_t **firmwareCRC**
BigEndian.
- uint32_t **fileSize**
BigEndian.
- uint16_t **frameCRC**
BigEndian.

5.23.1 Detailed Description

start of frame struct

5.23.2 Field Documentation

5.23.2.1 FC

uint8_t FC

function code FU

5.23.2.2 fileSize

uint32_t fileSize

BigEndian.

5.23.2.3 firmwareCRC

uint16_t firmwareCRC

BigEndian.

5.23.2.4 frameCRC

uint16_t frameCRC

BigEndian.

5.23.2.5 frameLen

uint8_t frameLen

size in bytes from start of [FU_prot](#) address to last byte of CRC

5.23.2.6 fversion

uint32_t fversion

full version (device) or transferred firmware version (from host) [4b:hMajor, 4b:hMinor, 8b:fMajor, 16b:fMinor]

5.23.2.7 hash

`uint8_t hash`

mix of hardware and software version in this packet

5.23.2.8 opcode

`uint8_t opcode`

operation code, [FU_OPCODE_SOT](#)

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

- [fu.h](#)

5.24 mac_buf_t Struct Reference

`#include <mac.h>`

Data Fields

- union {

 unsigned char `buf` [`MAC_BUF_LEN`]

 struct `_packed` {

 unsigned char `control` [2]

 unsigned char `seq_num`

 `pan_dev_addr_t pan`

 `dev_addr_t dst`

 `dev_addr_t src`

 unsigned char `data` [64]

 } `frame`

};
- `unsigned char * dPtr`
read/write pointer
- `mac_buf_state state`
current buf state
- `int rx_len`
number of received bytes including 80
- `bool isRangingFrame`
- `short retransmit_fail_cnt`
increased when after ack receive timeout
- `unsigned int last_update_time`

5.24.1 Field Documentation

5.24.1.1 "@1

```
union { ... }
```

5.24.1.2 buf

```
unsigned char buf[MAC_BUF_LEN]
```

5.24.1.3 control

```
unsigned char control[2]
```

5.24.1.4 data

```
unsigned char data[64]
```

5.24.1.5 dPtr

```
unsigned char* dPtr
```

read/write pointer

5.24.1.6 dst

```
dev_addr_t dst
```

5.24.1.7 frame

```
struct { ... } ::_packed frame
```

5.24.1.8 isRangingFrame

bool isRangingFrame

5.24.1.9 last_update_time

unsigned int last_update_time

5.24.1.10 pan

[pan_dev_addr_t](#) pan

5.24.1.11 retransmit_fail_cnt

short retransmit_fail_cnt

increased when after ack receive timeout

5.24.1.12 rx_len

int rx_len

number of received bytes includeing 80

5.24.1.13 seq_num

unsigned char seq_num

5.24.1.14 src

[dev_addr_t](#) src

5.24.1.15 state

```
mac_buf_state state
```

current buf state

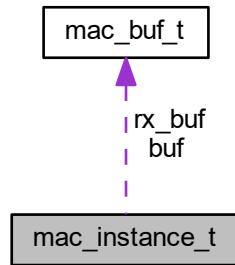
The documentation for this struct was generated from the following file:

- [mac.h](#)

5.25 mac_instance_t Struct Reference

```
#include <mac.h>
```

Collaboration diagram for mac_instance_t:



Data Fields

- int [slot_number](#)
- [mac_buf_t buf \[MAC_BUF_CNT\]](#)
- uint8_t [seq_num](#)
- [mac_buf_t rx_buf](#)
- short [buf_get_ind](#)
- int64_t [slot_time_offset](#)
- bool [frame_under_tx_is_ranging](#)
- unsigned int [last_rx_ts](#)
- unsigned int [beacon_timer_timestamp](#)

5.25.1 Field Documentation

5.25.1.1 beacon_timer_timestamp

```
unsigned int beacon_timer_timestamp
```

5.25.1.2 buf

```
mac_buf_t buf[MAC_BUF_CNT]
```

5.25.1.3 buf_get_ind

```
short buf_get_ind
```

5.25.1.4 frame_under_tx_is_ranging

```
bool frame_under_tx_is_ranging
```

5.25.1.5 last_rx_ts

```
unsigned int last_rx_ts
```

5.25.1.6 rx_buf

```
mac_buf_t rx_buf
```

5.25.1.7 seq_num

```
uint8_t seq_num
```

5.25.1.8 slot_number

```
int slot_number
```

5.25.1.9 slot_time_offset

```
int64_t slot_time_offset
```

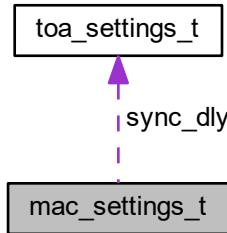
The documentation for this struct was generated from the following file:

- [mac.h](#)

5.26 mac_settings_t Struct Reference

```
#include <mac_settings.h>
```

Collaboration diagram for mac_settings_t:



Data Fields

- [dev_addr_t addr](#)
local device address
- [pan_dev_addr_t pan](#)
personal area network
- int [slot_time_us](#)
one slot time in us (including guard time)
- int [slot_guard_time_us](#)
guard time between slots
- int [slots_sum_time_us](#)
slots sum time in us
- int [max_frame_fail_cnt](#)
frame retransmit/delete threshold
- [mac_buff_time_t max_buf_inactive_time](#)
maximal buf inactive time
- [toa_settings_t sync_dly](#)
SYNC TOA delay settings.
- [rtls_role role](#)
local device
- bool [raport_anchor_anchor_distance](#)
true

5.26.1 Field Documentation

5.26.1.1 addr

`dev_addr_t` `addr`

local device address

5.26.1.2 max_buf_inactive_time

`mac_buff_time_t` `max_buf_inactive_time`

maximal buf inactive time

5.26.1.3 max_frame_fail_cnt

`int` `max_frame_fail_cnt`

frame retransmit/delete threshold

5.26.1.4 pan

`pan_dev_addr_t` `pan`

personal area network

5.26.1.5 report_anchor_anchor_distance

`bool` `report_anchor_anchor_distance`

true

5.26.1.6 role

`rtls_role` role

local device

5.26.1.7 slot_guard_time_us

`int slot_guard_time_us`

guard time between slots

5.26.1.8 slot_time_us

`int slot_time_us`

one slot time in us (including guard time)

5.26.1.9 slots_sum_time_us

`int slots_sum_time_us`

slots sum time in us

5.26.1.10 sync_dly

`toa_settings_t sync_dly`

SYNC TOA delay settings.

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

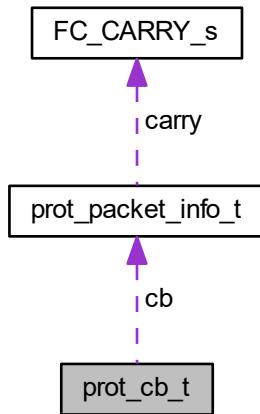
- [mac_settings.h](#)

5.27 prot_cb_t Struct Reference

struct of function code callbacks

```
#include <bin_parser.h>
```

Collaboration diagram for prot_cb_t:



Data Fields

- [FC_t FC](#)
Function Code.
- [prot_parser_cb cb](#)
associated callback function pointer

5.27.1 Detailed Description

struct of function code callbacks

5.27.2 Field Documentation

5.27.2.1 cb

[prot_parser_cb cb](#)

associated callback function pointer

5.27.2.2 FC

[FC_t](#) FC

Function Code.

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

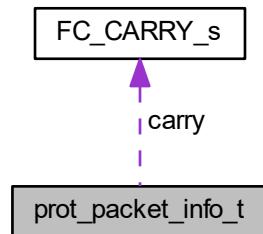
- [bin_parser.h](#)

5.28 prot_packet_info_t Struct Reference

packet extra information struct

```
#include <mac_const.h>
```

Collaboration diagram for prot_packet_info_t:



Data Fields

- [dev_addr_t direct_src](#)
- struct [FC_CARRY_s](#) * carry

5.28.1 Detailed Description

packet extra information struct

5.28.2 Field Documentation

5.28.2.1 carry

```
struct FC_CARRY_s* carry
```

5.28.2.2 direct_src

```
dev_addr_t direct_src
```

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

- [mac_const.h](#)

5.29 settings_otp_t Struct Reference

Internal One Time Programmable settings structure.

```
#include <settings.h>
```

Data Fields

- [uint8_t h_major](#)
hardware major version number
- [uint8_t h_minor](#)
hardware minor version number
- [uint64_t serial](#)
hardware serial number

5.29.1 Detailed Description

Internal One Time Programmable settings structure.

This kind of settings is specially usable for hardware and serial number description because it should be constant value from production without possibility of change.

Note

This structure is packed

5.29.2 Field Documentation

5.29.2.1 h_major

`uint8_t h_major`

hardware major version number

5.29.2.2 h_minor

`uint8_t h_minor`

hardware minor version number

5.29.2.3 serial

`uint64_t serial`

hardware serial number

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

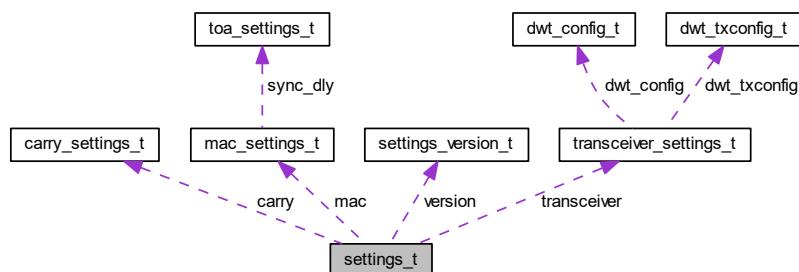
- [settings.h](#)

5.30 settings_t Struct Reference

main setting container.

```
#include <settings.h>
```

Collaboration diagram for settings_t:



Data Fields

- `settings_version_t` `version`
- `transceiver_settings_t` `transceiver`
- `mac_settings_t` `mac`
- `carry_settings_t` `carry`

5.30.1 Detailed Description

main setting container.

This structure consist of settings structure from particular modules.

Note

`settings_version_t` need to be the first field in settings because of reference from bootloader instance of this structure should be global and be located in precisely defined location in non-volatile memory (FLASH)

5.30.2 Field Documentation

5.30.2.1 carry

`carry_settings_t` `carry`

5.30.2.2 mac

`mac_settings_t` `mac`

5.30.2.3 transceiver

`transceiver_settings_t` `transceiver`

5.30.2.4 version

`settings_version_t` `version`

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

- `settings.h`

5.31 settings_version_t Struct Reference

current firmware and hardware description structure

```
#include <settings.h>
```

Data Fields

- `uint32_t boot_reserved`
field reserved for bootloader
- `uint8_t h_version`
hardware version from H_VERSION_CALC
- `uint8_t f_major`
firmware major version number
- `uint16_t f_minor`
firmware minor version number
- `uint64_t f_hash`
firmware hash version number

5.31.1 Detailed Description

current firmware and hardware description structure

Values from this struct should be modified only by firmware upgrade procedure.

Note

It is used also from bootloader to check firmware and hardware compatibility between code versions.
This structure is packed

5.31.2 Field Documentation

5.31.2.1 boot_reserved

```
uint32_t boot_reserved
```

field reserved for bootloader

5.31.2.2 f_hash

```
uint64_t f_hash
```

firmware hash version number

5.31.2.3 f_major

```
uint8_t f_major
```

firmware major version number

5.31.2.4 f_minor

```
uint16_t f_minor
```

firmware minor version number

5.31.2.5 h_version

```
uint8_t h_version
```

hardware version from [H_VERSION_CALC](#)

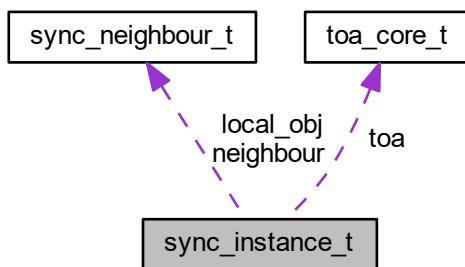
The documentation for this struct was generated from the following file:

- [settings.h](#)

5.32 sync_instance_t Struct Reference

```
#include <sync.h>
```

Collaboration diagram for sync_instance_t:



Data Fields

- `toa_core_t toa`
SYNC toa data.
- `int64_t toa_ts_poll_rx_raw`
used to SYNC TOA_EnableRxBeforeFin
- `uint8_t tree_level`
local device tree level
- `sync_neighbour_t local_obj`
local clock from dwt data
- `sync_neighbour_t neighbour [SYNC_MAC_NEIGHBOURS]`
list of neighbours

5.32.1 Field Documentation

5.32.1.1 local_obj

`sync_neighbour_t local_obj`

local clock from dwt data

5.32.1.2 neighbour

`sync_neighbour_t neighbour [SYNC_MAC_NEIGHBOURS]`

list of neighbours

5.32.1.3 toa

`toa_core_t toa`

SYNC toa data.

5.32.1.4 toa_ts_poll_rx_raw

`int64_t toa_ts_poll_rx_raw`

used to SYNC TOA_EnableRxBeforeFin

5.32.1.5 tree_level

`uint8_t tree_level`

local device tree level

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

- [sync.h](#)

5.33 sync_neighbour_t Struct Reference

```
#include <sync.h>
```

Data Fields

- `dev_addr_t addr`
neigbour address
- `uint8_t tree_level`
number of hops from sink (sink is level 0)
- `uint8_t sync_ready`
sync ready (true or false)
- `int64_t time_offset`
local to global time offset in dtu
- `int64_t update_ts`
last update timestamp in dtu
- `int64_t drift [3]`
list of last clock drifts
- `float time_coeffP [3]`
list of last time drifts over delta dtu after
- `float time_coeffP_raw [3]`
raw list of last time drifts over delta dtu
- `float time_drift_sum`
sum of clock drifts
- `float tof_dw`
time of flight between neighbour and local device in dtu

5.33.1 Field Documentation

5.33.1.1 addr

`dev_addr_t addr`

neigbour address

5.33.1.2 drift

```
int64_t drift[3]
```

list of last clock drifts

5.33.1.3 sync_ready

```
uint8_t sync_ready
```

sync ready (true or false)

5.33.1.4 time_coeffP

```
float time_coeffP[3]
```

list of last time drifts over delta dtu after

5.33.1.5 time_coeffP_raw

```
float time_coeffP_raw[3]
```

raw list of last time drifts over delta dtu

5.33.1.6 time_drift_sum

```
float time_drift_sum
```

sum of clock drifts

5.33.1.7 time_offset

```
int64_t time_offset
```

local to global time offset in dtu

5.33.1.8 tof_dw

float tof_dw

time of flight between neighbour and local device in dtu

5.33.1.9 tree_level

uint8_t tree_level

number of hops from sink (sink is level 0)

5.33.1.10 update_ts

int64_t update_ts

last update timestamp in dtu

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

- [sync.h](#)

5.34 toa_core_t Struct Reference

#include <toa.h>

Data Fields

- [toa_state_t state](#)
- [toa_state_t prev_state](#)
- [int64_t TsPollTx](#)
- [uint32_t TsRespRx \[TOA_MAX_DEV_IN_POLL\]](#)
- [uint32_t TsFinTx](#)
- [uint32_t TsPollRx](#)
- [uint32_t TsRespTx](#)
- [uint32_t TsFinRx](#)
- [uint8_t resp_ind](#)
0..anc_in_poll_cnt-1
- [uint8_t anc_in_poll_cnt](#)
1..TOA_MAX_DEV_IN_POLL
- [dev_addr_t addr_tab \[TOA_MAX_DEV_IN_POLL\]](#)
anchors addresses
- [dev_addr_t initiator](#)
address of measure initiator

5.34.1 Field Documentation

5.34.1.1 addr_tab

`dev_addr_t addr_tab[TOA_MAX_DEV_IN_POLL]`

anchors addresses

5.34.1.2 anc_in_poll_cnt

`uint8_t anc_in_poll_cnt`

1..TOA_MAX_DEV_IN_POLL

5.34.1.3 initiator

`dev_addr_t initiator`

address of measure initiator

5.34.1.4 prev_state

`toa_state_t prev_state`

5.34.1.5 resp_ind

`uint8_t resp_ind`

0..anc_in_poll_cnt-1

5.34.1.6 state

`toa_state_t state`

5.34.1.7 TsFinRx

```
uint32_t TsFinRx
```

5.34.1.8 TsFinTx

```
uint32_t TsFinTx
```

5.34.1.9 TsPollRx

```
uint32_t TsPollRx
```

5.34.1.10 TsPollTx

```
int64_t TsPollTx
```

5.34.1.11 TsRespRx

```
uint32_t TsRespRx[TOA_MAX_DEV_IN_POLL]
```

5.34.1.12 TsRespTx

```
uint32_t TsRespTx
```

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

- [toa.h](#)

5.35 toa_settings_t Struct Reference

```
#include <mac_settings.h>
```

Data Fields

- int [fin_dly_us](#)
- int [resp_dly_us](#) [TOA_MAX_DEV_IN_POLL]
- int [guard_time_us](#)
- int [rx_after_tx_offset_us](#)

5.35.1 Field Documentation

5.35.1.1 [fin_dly_us](#)

```
int fin_dly_us
```

5.35.1.2 [guard_time_us](#)

```
int guard_time_us
```

5.35.1.3 [resp_dly_us](#)

```
int resp_dly_us[TOA_MAX_DEV_IN_POLL]
```

5.35.1.4 [rx_after_tx_offset_us](#)

```
int rx_after_tx_offset_us
```

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

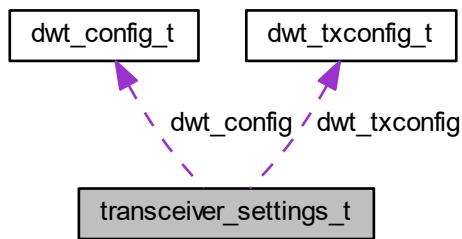
- [mac_settings.h](#)

5.36 transceiver_settings_t Struct Reference

transceiver settings

```
#include <transceiver_settings.h>
```

Collaboration diagram for transceiver_settings_t:



Data Fields

- unsigned short `ant_dly_rx`
- unsigned short `ant_dly_tx`
- char `low_power_mode`
- char `set_pac_from_settings`
- `dwt_config_t dwt_config`
- `dwt_txconfig_t dwt_txconfig`

5.36.1 Detailed Description

transceiver settings

5.36.2 Field Documentation

5.36.2.1 ant_dly_rx

```
unsigned short ant_dly_rx
```

5.36.2.2 ant_dly_tx

```
unsigned short ant_dly_tx
```

5.36.2.3 dwt_config

```
dwt_config_t dwt_config
```

5.36.2.4 dwt_txconfig

```
dwt_txconfig_t dwt_txconfig
```

5.36.2.5 low_power_mode

```
char low_power_mode
```

5.36.2.6 set_pac_from_settings

```
char set_pac_from_settings
```

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

- [transceiver_settings.h](#)

5.37 txt_buf_t Struct Reference

text parser engine circular buffer structure

```
#include <txt_parser.h>
```

Data Fields

- **cchar * cmd**
pointer to start of current message to parse
- **cchar *const start**
pointer to start of buffer
- **cchar *const end**
pointer to start of buffer + buffer length
- **int cnt**
number of messages ready to parse

5.37.1 Detailed Description

text parser engine circular buffer structure

5.37.2 Field Documentation

5.37.2.1 cmd

`cchar*` cmd

pointer to start of current message to parse

5.37.2.2 cnt

`int` cnt

numer of messages ready to parse

5.37.2.3 end

`cchar*` const end

pointer to start of buffer + buffer length

5.37.2.4 start

`cchar*` const start

pointer to start of buffer

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

- [txt_parser.h](#)

5.38 txt_cb_t Struct Reference

text parser engine callback struct

```
#include <txt_parser.h>
```

Data Fields

- `const char * cmd`
command string
- `const txt_parser_cb cb`
callback function routine

5.38.1 Detailed Description

text parser engine callback struct

5.38.2 Field Documentation

5.38.2.1 cb

`const txt_parser_cb cb`

callback function routine

5.38.2.2 cmd

`const char* cmd`

command string

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

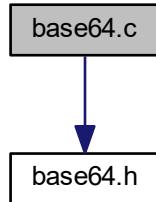
- [txt_parser.h](#)

Chapter 6

File Documentation

6.1 base64.c File Reference

```
#include "base64.h"  
Include dependency graph for base64.c:
```



Functions

- int **BASE64_TextSize** (unsigned int binSize)
calculate target base64 text size
- int **BASE64_Encode** (unsigned char *dst, const unsigned char *src, unsigned short srcSize)
from bin to base64
- int **BASE64_Decode** (unsigned char *dst, const unsigned char *src, unsigned short dstBufCapacity)
from base64 to bin

6.1.1 Function Documentation

6.1.1.1 BASE64_Decode()

```
int BASE64_Decode (
    unsigned char * dst,
    const unsigned char * src,
    unsigned short dstBufCapacity )
```

from base64 to bin

decode src buff to \0 char or to first occurrence of '='

Note

dst can point same place as *dst*

Parameters

out	<i>dst</i>	destination buffer for binary data
in	<i>src</i>	source buffer for
in	<i>dstBufCapacity</i>	maximal destination buffer capacity

Returns

int size of bytes in *dst* buffer without terminating zero

6.1.1.2 BASE64_Encode()

```
int BASE64_Encode (
    unsigned char * dst,
    const unsigned char * src,
    unsigned short srcSize )
```

from bin to base64

Note

function add terminating 0 at the end of *dst*
dst can NOT point same place as *dst*

Parameters

out	<i>dst</i>	destination buffer for base64 string
in	<i>src</i>	source buffer with binary data
in	<i>srcSize</i>	size of source data to encode

Returns

int *dst* buffer size

6.1.1.3 BASE64_TextSize()

```
int BASE64_TextSize (
    unsigned int binSize )
```

calculate target base64 text size

Parameters

<i>binSize</i>	binary data size in bytes
----------------	---------------------------

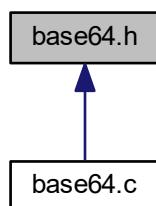
Returns

int base64 text size in bytes (without null terminator)

6.2 base64.h File Reference

base64 coder and decoder in C

This graph shows which files directly or indirectly include this file:



Functions

- int [BASE64_Encode](#) (unsigned char *dst, const unsigned char *src, unsigned short srcSize)
from bin to base64
- int [BASE64_Decode](#) (unsigned char *dst, const unsigned char *src, unsigned short dstBufCapacity)
from base64 to bin
- int [BASE64_TextSize](#) (unsigned int binSize)
calculate target base64 text size

6.2.1 Detailed Description

base64 coder and decoder in C

Date

2018-06-28

Author

: Karol Trzcinski

6.2.2 Function Documentation

6.2.2.1 BASE64_Decode()

```
int BASE64_Decode (
    unsigned char * dst,
    const unsigned char * src,
    unsigned short dstBufCapacity )
```

from base64 to bin

decode src buff to \0 char or to first occurrence of '='

Note

dst can point same place as dst

Parameters

out	<i>dst</i>	destination buffer for binary data
in	<i>src</i>	source buffer for
in	<i>dstBufCapacity</i>	maximal destination buffer capacity

Returns

int size of bytes in dst buffer without terminating zero

6.2.2.2 BASE64_Encode()

```
int BASE64_Encode (
    unsigned char * dst,
    const unsigned char * src,
    unsigned short srcSize )
```

from bin to base64

Note

function add terminating 0 at the end of dst
dst can NOT point same place as dst

Parameters

out	<i>dst</i>	destination buffer for base64 string
in	<i>src</i>	source buffer with binary data
in	<i>srcSize</i>	size of source data to encode

Returns

int dst buffer size

6.2.2.3 BASE64_TextSize()

```
int BASE64_TextSize (
    unsigned int binSize )
```

calculate target base64 text size

Parameters

<i>binSize</i>	binary data size in bytes
----------------	---------------------------

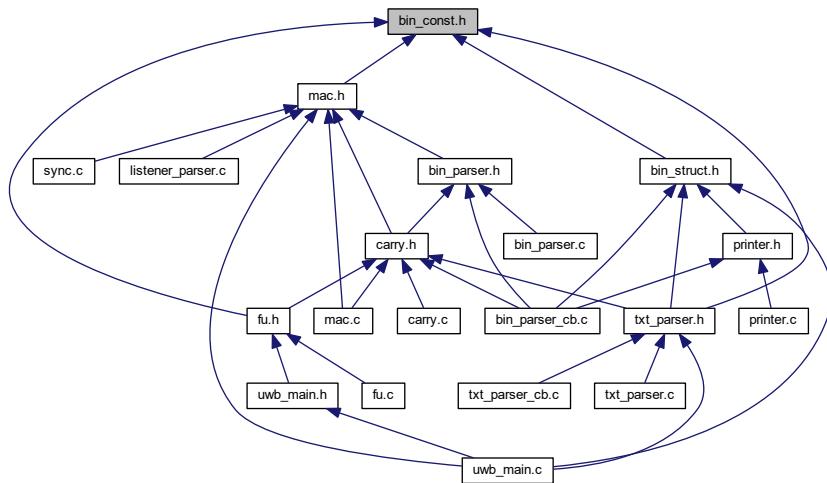
Returns

int base64 text size in bytes (without null terminator)

6.3 bin_const.h File Reference

define numerical function codes for binary data

This graph shows which files directly or indirectly include this file:



Enumerations

- enum `FC_t`{
 `FC_BEACON` = 0x00, `FC_TURN_ON` = 0x01, `FC_TURN_OFF` = (`FC_TURN_ON`+1), `FC_DEV_ACCEPTED` = 0x03,
 `FC_CARRY` = 0x04, `FC_FU` = 0x05, `FC_SYNC_POLL` = 0x11, `FC_SYNC_RESP` = (`FC_SYNC_POLL` + 1),
 `FC_SYNC_FIN` = (`FC_SYNC_POLL` + 2), `FC_VERSION_ASK` = 0x15, `FC_VERSION_RESP` = (`FC_VERSION_ASK` + 1),
 `FC_STAT_ASK` = 0x18,
 `FC_STAT_RESP` = (`FC_STAT_ASK` + 1), `FC_STAT_SET` = (`FC_STAT_ASK` + 2), `FC_TXSET_ASK` = 0x1C,
 `FC_TXSET_RESP` = (`FC_TXSET_ASK` + 1),
 `FC_TXSET_SET` = (`FC_TXSET_ASK` + 2) }

6.3.1 Detailed Description

define numerical function codes for binary data

Author

Karol Trzcinski

Date

2018-06-28

6.3.2 Enumeration Type Documentation

6.3.2.1 FC_t

enum `FC_t`

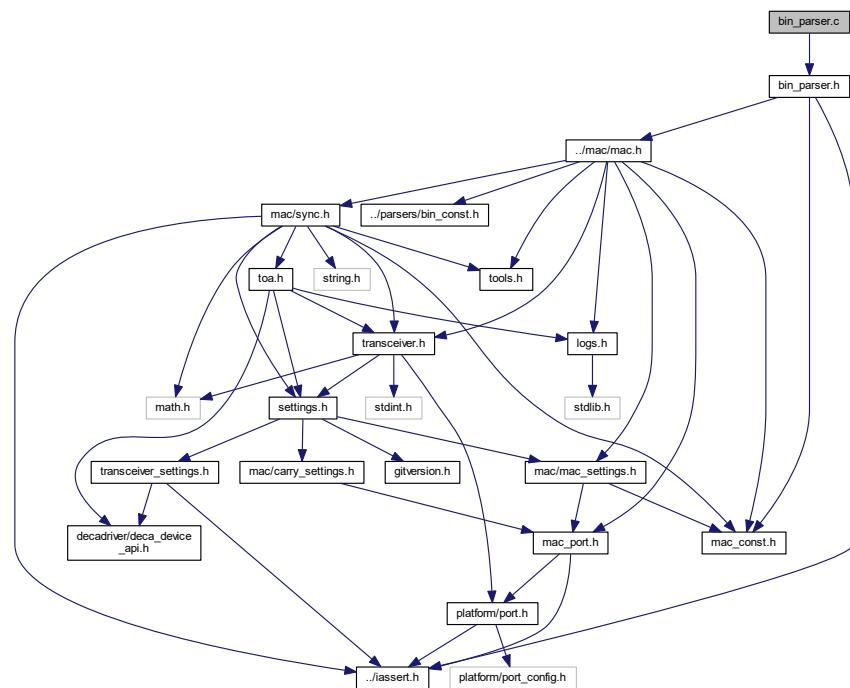
Enumerator

FC_BEACON	beacon message is send when nobody sent message to this device by a few seconds
FC_TURN_ON	when device turn on then it should send turn on message
FC_TURN_OFF	when device turn off then it should send turn off message
FC_DEV_ACCEPTED	device accepted in network by local sink
FC_CARRY	carry message
FC_FU	firmware upgrade message
FC_SYNC_POLL	first ranging message in sync procedure
FC_SYNC RESP	second ranging message in sync procedure
FC_SYNC FIN	third and last message in sync procedure
FC_VERSION ASK	device is asked about version
FC_VERSION RESP	device response version message
FC_STAT ASK	device is asked about status
FC_STAT RESP	device is asked about status
FC_STAT SET	device should change status to new one
FC_TXSET ASK	device is asked about transceiver settings
FC_TXSET RESP	device is asked about transceiver settings
FC_TXSET SET	device should change transceiver settings to new one

6.4 bin_parser.c File Reference

```
#include "bin_parser.h"
```

Include dependency graph for bin_parser.c:



Functions

- `uint8_t BIN_ParseSingle (const uint8_t *buf, const prot_packet_info_t *info)`
parse single message
- `void BIN_Parse (mac_buf_t *buf, const prot_packet_info_t *info, int size)`
parse each message in frame

6.4.1 Function Documentation

6.4.1.1 BIN_Parse()

```
void BIN_Parse (
    mac_buf_t * buf,
    const prot_packet_info_t * info,
    int size )
```

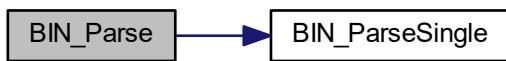
parse each message in frame

This function also keep buf->dPtr consistent so callback functions doesn't have to manage buffer data pointer.

Parameters

in	<i>buf</i>	pointer to buffer with message
in	<i>info</i>	extra informations about frame
	<i>size</i>	number of bytes to parse

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.1.2 BIN_ParseSingle()

```
uint8_t BIN_ParseSingle (
    const uint8_t * buf,
    const prot_packet_info_t * info )
```

parse single message

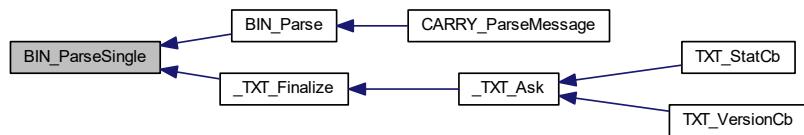
Parameters

in	<i>buf</i>	pointer to function code of message followed by frame len and data
in	<i>info</i>	extra informations about frame

Returns

uint8_t frame len when it was processed, 0 otherwise

Here is the caller graph for this function:

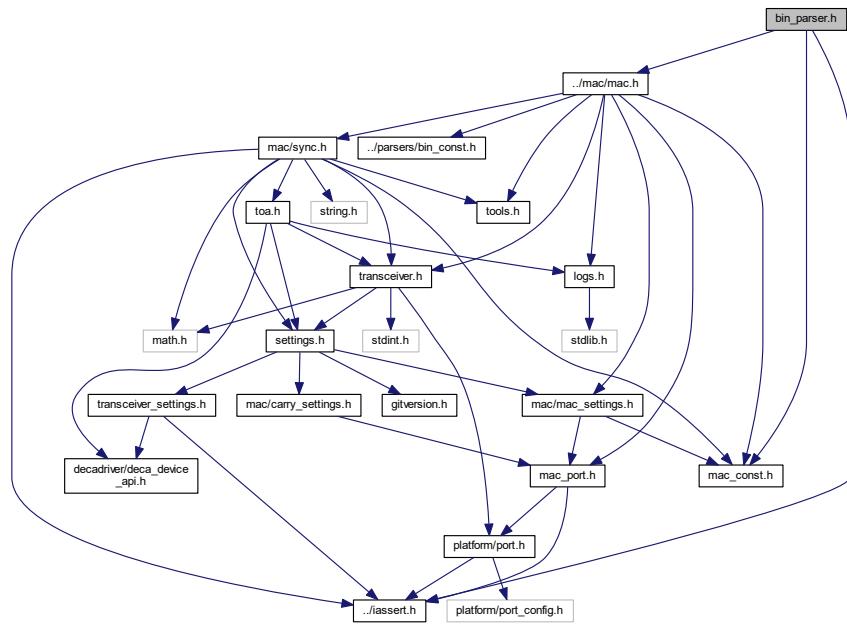


6.5 bin_parser.h File Reference

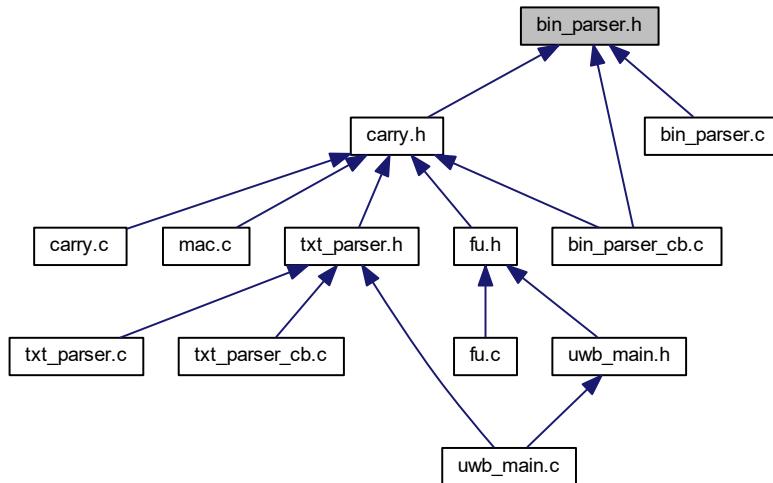
binary data parser engine

```
#include "../iassert.h"
#include "../mac/mac.h"
#include "../mac/mac_const.h"
```

Include dependency graph for bin_parser.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `prot_cb_t`
struct of function code callbacks

Macros

- `#define BIN_ASSERT(expr) IASSERT(expr)`
connect BIN_ASSERT with IASSERT

Typedefs

- `typedef void(* prot_parser_cb) (const void *data, const prot_packet_info_t *info)`
function code callback function template

Functions

- `uint8_t BIN_ParseSingle (const uint8_t *buf, const prot_packet_info_t *info)`
parse single message
- `void BIN_Parse (mac_buf_t *buf, const prot_packet_info_t *info, int size)`
parse each message in frame

6.5.1 Detailed Description

binary data parser engine

core of this engine is in bin_parser_cb where each callback is implemented and added to the callbacks table.

Author

Karol Trzcinski

Date

2018-06-28

6.5.2 Macro Definition Documentation

6.5.2.1 BIN_ASSERT

```
#define BIN_ASSERT(  
    expr ) IASSERT(expr)
```

connect BIN_ASSERT with IASSERT

6.5.3 Typedef Documentation

6.5.3.1 prot_parser_cb

```
typedef void(* prot_parser_cb) (const void *data, const prot_packet_info_t *info)  
function code callback function template
```

Parameters

in	<i>data</i>	pointer to input data, first byte is function code (FC_), second byte is frame length
in	<i>info</i>	is pointer to extra frame information structure

6.5.4 Function Documentation**6.5.4.1 BIN_Parse()**

```
void BIN_Parse (
    mac_buf_t * buf,
    const prot_packet_info_t * info,
    int size )
```

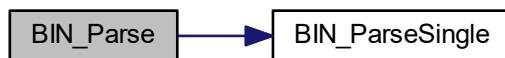
parse each message in frame

This function also keep buf->dPtr consistent so callback functions doesn't have to manage buffer data pointer.

Parameters

in	<i>buf</i>	pointer to buffer with message
in	<i>info</i>	extra informations about frame
	<i>size</i>	number of bytes to parse

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.4.2 BIN_ParseSingle()

```
uint8_t BIN_ParseSingle (
    const uint8_t * buf,
    const prot_packet_info_t * info )
```

parse single message

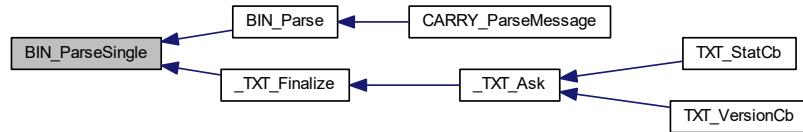
Parameters

in	<i>buf</i>	pointer to function code of message followed by frame len and data
in	<i>info</i>	extra informations about frame

Returns

uint8_t frame len when it was processed, 0 otherwise

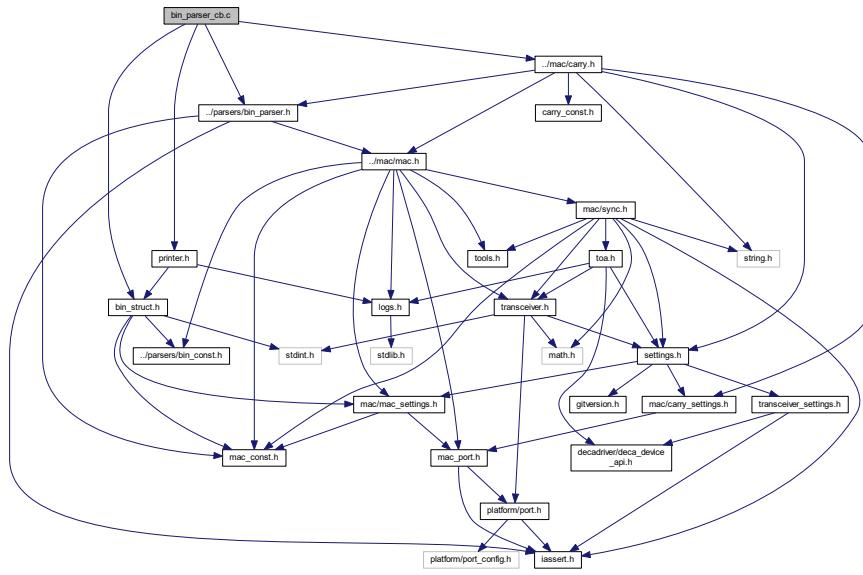
Here is the caller graph for this function:



6.6 bin_parser_cb.c File Reference

```
#include "../mac/carry.h"
#include "bin_parser.h"
#include "bin_struct.h"
#include "printer.h"
```

Include dependency graph for bin_parser_cb.c:



Functions

- void `BIN_SEND_RESP (FC_t FC, const void *data, uint8_t len, const prot_packet_info_t *info)`
- void `FC_TURN_ON_cb (const void *data, const prot_packet_info_t *info)`
- void `FC_BEACON_cb (const void *data, const prot_packet_info_t *info)`
- void `FC_STAT_ASK_cb (const void *data, const prot_packet_info_t *info)`
- void `FC_VERSION_ASK_cb (const void *data, const prot_packet_info_t *info)`

Variables

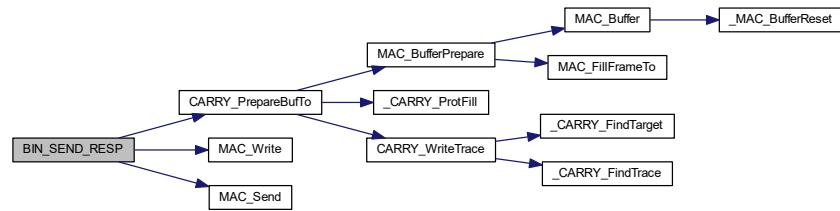
- const `prot_cb_t prot_cb_tab []`
- const int `prot_cb_len = sizeof(prot_cb_tab) / sizeof(*prot_cb_tab)`

6.6.1 Function Documentation

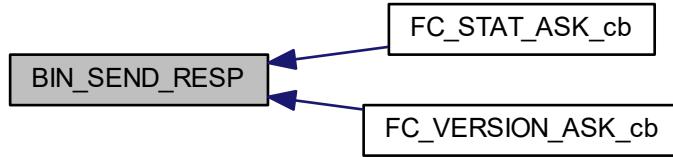
6.6.1.1 BIN_SEND_RESP()

```
void BIN_SEND_RESP (
    FC_t FC,
    const void * data,
    uint8_t len,
    const prot_packet_info_t * info )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.1.2 FC_BEACON_cb()

```

void FC_BEACON_cb (
    const void * data,
    const prot_packet_info_t * info )

```

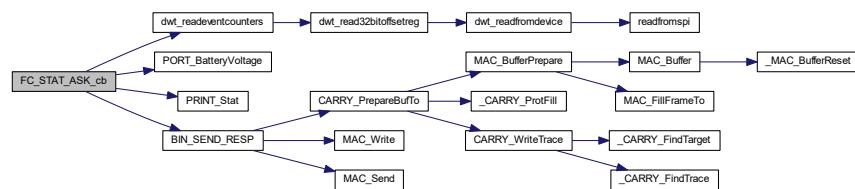
Here is the call graph for this function:



6.6.1.3 FC_STAT_ASK_cb()

```
void FC_STAT_ASK_cb (
    const void * data,
    const prot_packet_info_t * info )
```

Here is the call graph for this function:



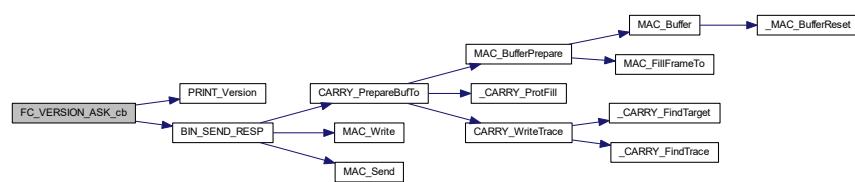
6.6.1.4 FC_TURN_ON_cb()

```
void FC_TURN_ON_cb (
    const void * data,
    const prot_packet_info_t * info )
```

6.6.1.5 FC_VERSION_ASK_cb()

```
void FC_VERSION_ASK_cb (
    const void * data,
    const prot_packet_info_t * info )
```

Here is the call graph for this function:



6.6.2 Variable Documentation

6.6.2.1 prot_cb_len

```
const int prot_cb_len = sizeof(prot_cb_tab) / sizeof(*prot_cb_tab)
```

6.6.2.2 prot_cb_tab

```
const prot_cb_t prot_cb_tab[]
```

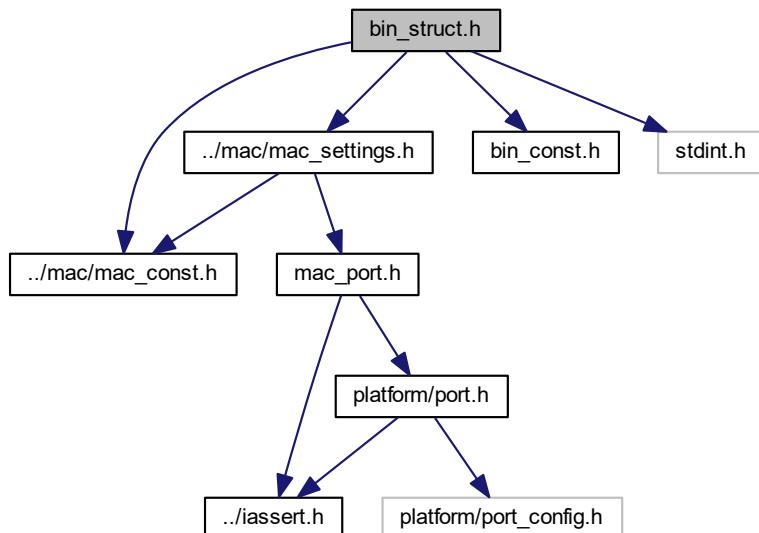
Initial value:

```
= {
    {FC_TURN_ON, FC_TURN_ON_cb},
    {FC_BEACON, FC_BEACON_cb},
    {FC_CARRY, 0},
    {FC_STAT_ASK, FC_STAT_ASK_cb},
    {FC_VERSION_ASK, FC_VERSION_ASK_cb},
}
```

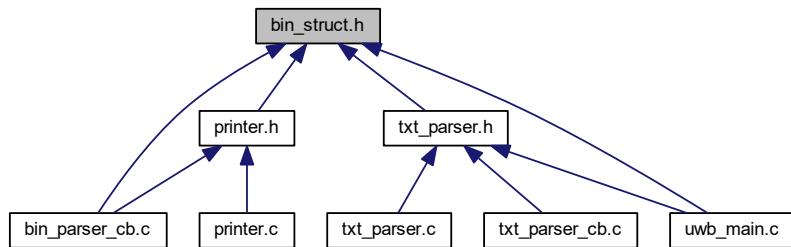
6.7 bin_struct.h File Reference

binary data structures definition

```
#include "../mac/mac_const.h"
#include "../mac/mac_settings.h"
#include "bin_const.h"
#include <stdint.h>
Include dependency graph for bin_struct.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [FC_TURN_ON_s](#)
see [FC_t](#) description
- struct [FC_TURN_OFF_s](#)
see [FC_t](#) description
- struct [FC_BEACON_s](#)
see [FC_t](#) description
- struct [FC_STAT_s](#)
see [FC_t](#) description
- struct [FC_VERSION_s](#)
see [FC_t](#) description

6.7.1 Detailed Description

binary data structures definition

Each data frame need to start with Function Code field (FC 1 byte) followed by frame len (len 1 byte).

Author

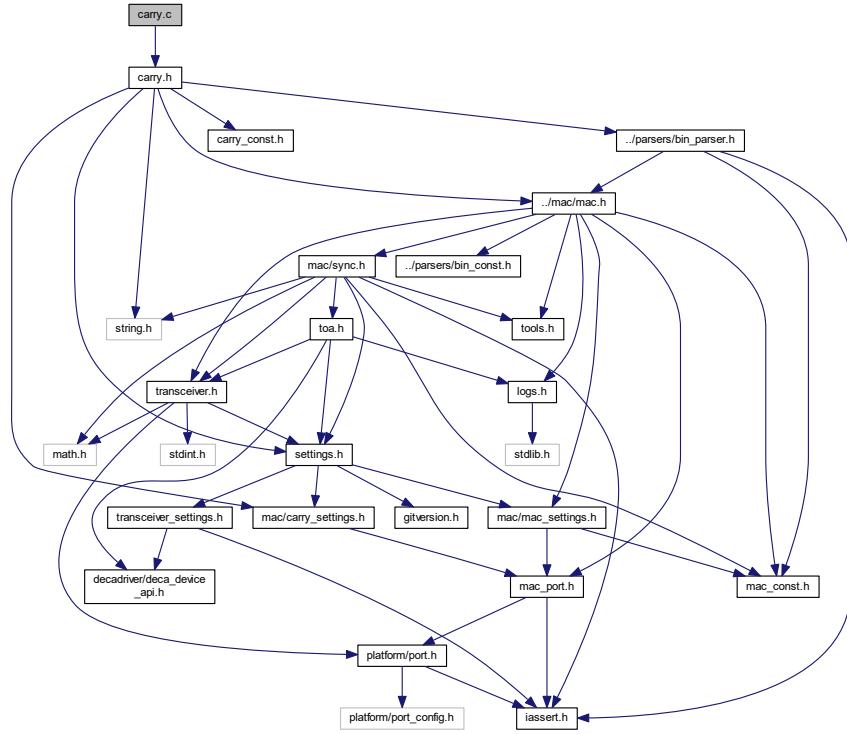
Karol Trzcinski

Date

2018-06-28

6.8 carry.c File Reference

```
#include "carry.h"
Include dependency graph for carry.c:
```



Functions

- void **CARRY_Init** (bool isConnectedToServer)

initialize module data
- **carry_target_t * _CARRY_FindTarget** (dev_addr_t target)
- **carry_trace_t * _CARRY_FindTrace** (carry_target_t *ptarget)
- int **CARRY_WriteTrace** (dev_addr_t *buf, dev_addr_t target)

write trace to target, including target address
- **FC_CARRY_s * _CARRY_ProtoFill** (mac_buf_t *buf)
- **mac_buf_t * CARRY_PrepareBufTo** (dev_addr_t target)

reserve buffer, write headers and set buffer fields default values.
- void **CARRY_ParseMessage** (mac_buf_t *buf)

function called from MAC module after receiving CARRY frame

Variables

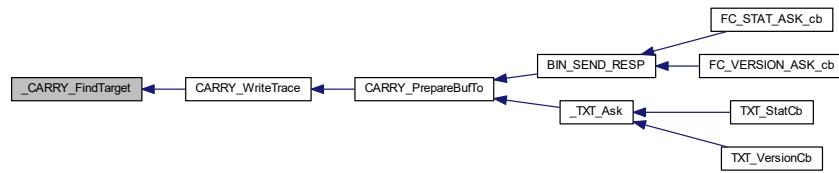
- **carry_instance_t carry**

6.8.1 Function Documentation

6.8.1.1 _CARRY_FindTarget()

```
carry_target_t* _CARRY_FindTarget (
    dev_addr_t target )
```

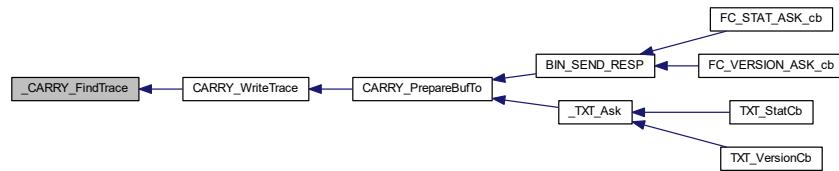
Here is the caller graph for this function:



6.8.1.2 _CARRY_FindTrace()

```
carry_trace_t* _CARRY_FindTrace (
    carry_target_t * pttarget )
```

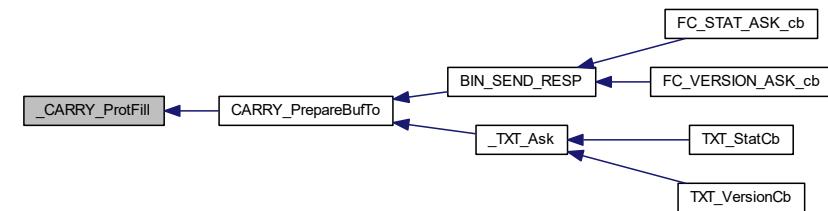
Here is the caller graph for this function:



6.8.1.3 _CARRY_ProtFill()

```
FC_CARRY_s* _CARRY_ProtFill (
    mac_buf_t * buf )
```

Here is the caller graph for this function:



6.8.1.4 CARRY_Init()

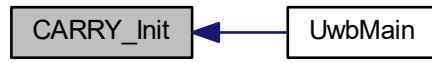
```
void CARRY_Init (
    bool isConnectedToServer )
```

initialize module data

Parameters

in	<i>isConnectedToServer</i>	bool value indicating device connection with remote server
----	----------------------------	--

Here is the caller graph for this function:



6.8.1.5 CARRY_ParseMessage()

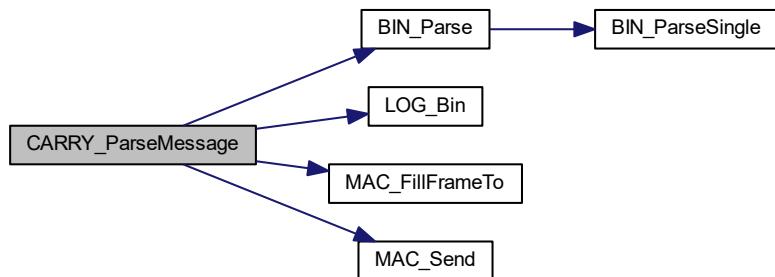
```
void CARRY_ParseMessage (
    mac_buf_t * buf )
```

function called from MAC module after receiving CARRY frame

Parameters

in	<i>buf</i>	
----	------------	--

Here is the call graph for this function:



6.8.1.6 CARRY_PrepBufTo()

```
mac_buf_t* CARRY_PrepBufTo (
    dev_addr_t target )
```

reserve buffer, write headers and set buffer fields default values.

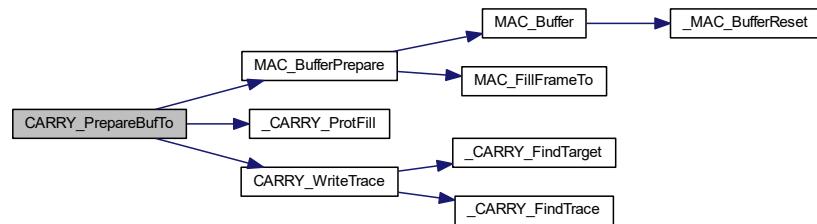
Parameters

in	<i>target</i>	device address
----	---------------	----------------

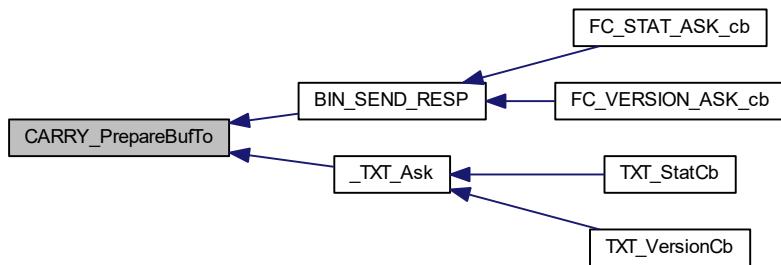
Returns

mac_buf_t* result buffer or null

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.1.7 CARRY_WriteTrace()

```
int CARRY_WriteTrace (
    dev_addr_t * buf,
    dev_addr_t target )
```

write trace to target, including target address

target address is the last one

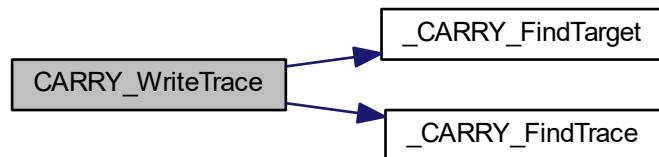
Parameters

in	<i>buf</i>	
in	<i>target</i>	

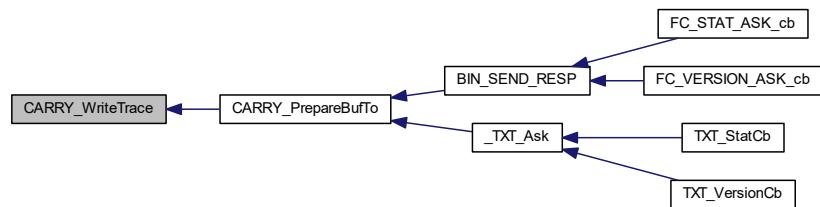
Returns

int number of written addresses or 0 when target is unknown

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.2 Variable Documentation

6.8.2.1 carry

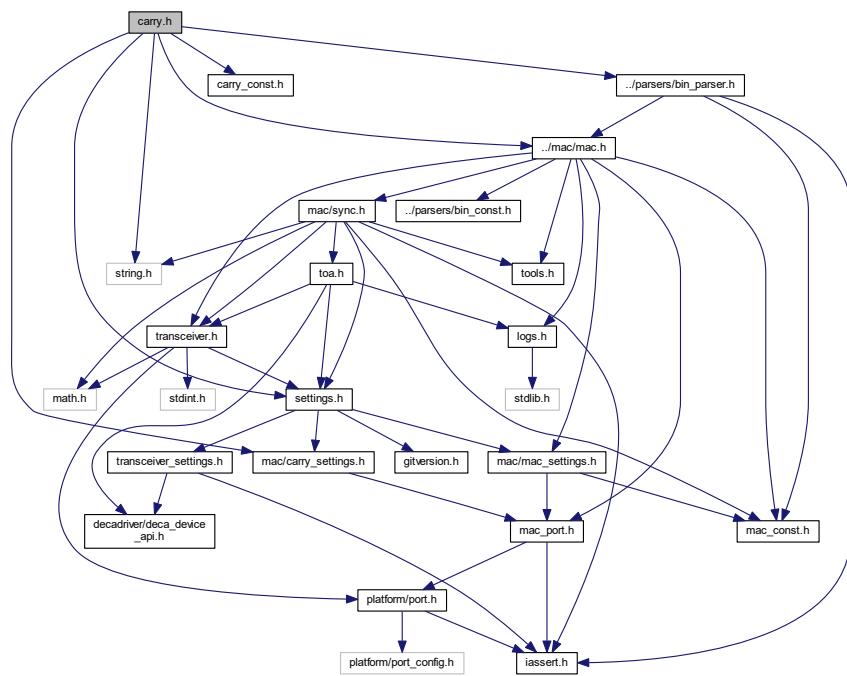
`carry_instance_t` carry

6.9 carry.h File Reference

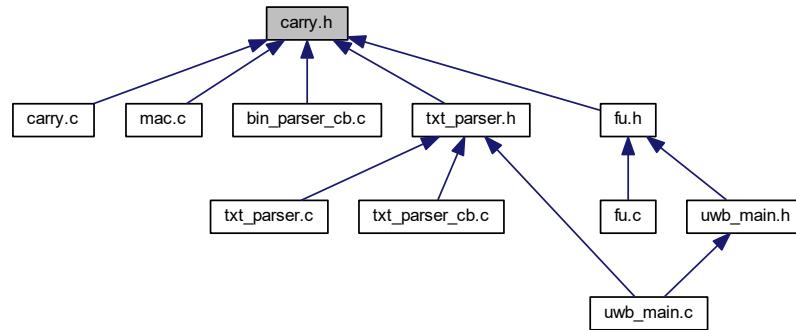
carry is a transport/routing layer

```
#include <string.h>
#include "../mac/mac.h"
#include "../settings.h"
#include "carry_const.h"
#include "carry_settings.h"
#include "../parsers/bin_parser.h"
```

Include dependency graph for carry.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `FC_CARRY_s`
protocol struct
- struct `carry_trace_t`
data about some trace to target
- struct `carry_target_t`
target info struct
- struct `carry_instance_t`
global singleton

Macros

- `#define CARRY_HEAD_MIN_LEN (1 + 1 + sizeof(dev_addr_t) + 1)`
carry head minimal length

Functions

- void `CARRY_Init` (bool isConnectedToServer)
initialize module data
- int `CARRY_WriteTrace` (dev_addr_t *buf, dev_addr_t target)
write trace to target, including target address
- `mac_buf_t * CARRY_PrepareBufTo` (dev_addr_t target)
reserve buffer, write headers and set buffer fields default values.
- `mac_buf_t * CARRY_GetBufTo` (dev_addr_t target)
find or create buffer to the target device
- void `CARRY_ParseMessage` (mac_buf_t *buf)
function called from MAC module after receiving CARRY frame

6.9.1 Detailed Description

carry is a transport/routing layer

module responsible for writing trace to the target device

Author

Karol Trzcinski

Date

2018-07-02

6.9.2 Macro Definition Documentation

6.9.2.1 CARRY_HEAD_MIN_LEN

```
#define CARRY_HEAD_MIN_LEN (1 + 1 + sizeof(dev_addr_t) + 1)
```

carry head minimal length

6.9.3 Function Documentation

6.9.3.1 CARRY_GetBufTo()

```
mac_buf_t* CARRY_GetBufTo (
    dev_addr_t target )
```

find or create buffer to the target device

returned buffer can be partially filled

Parameters

in	<i>target</i>	address
----	---------------	---------

Returns

mac_buf_t* result buffer or null

6.9.3.2 CARRY_Init()

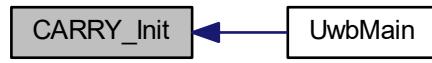
```
void CARRY_Init (
    bool isConnectedToServer )
```

initialize module data

Parameters

in	<i>isConnectedToServer</i>	bool value indicating device connection with remote server
----	----------------------------	--

Here is the caller graph for this function:



6.9.3.3 CARRY_ParseMessage()

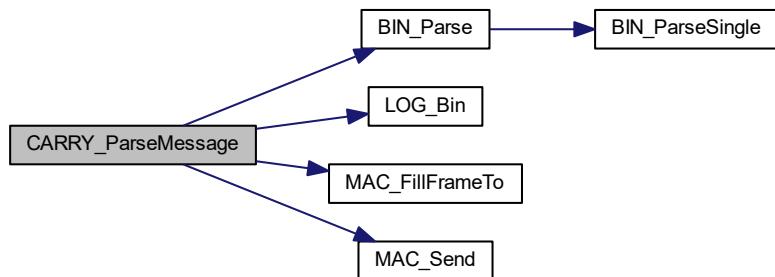
```
void CARRY_ParseMessage (
    mac_buf_t * buf )
```

function called from MAC module after receiving CARRY frame

Parameters

in	<i>buf</i>	
----	------------	--

Here is the call graph for this function:



6.9.3.4 CARRY_PrepBufTo()

```
mac_buf_t* CARRY_PrepBufTo (
    dev_addr_t target )
```

reserve buffer, write headers and set buffer fields default values.

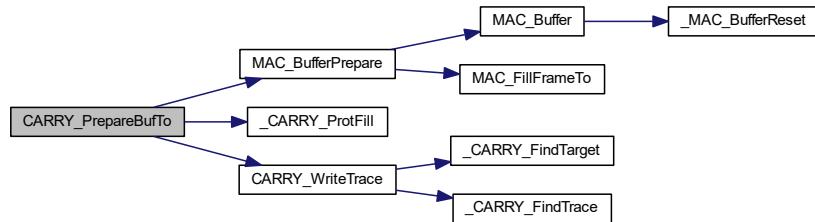
Parameters

in	<i>target</i>	device address
----	---------------	----------------

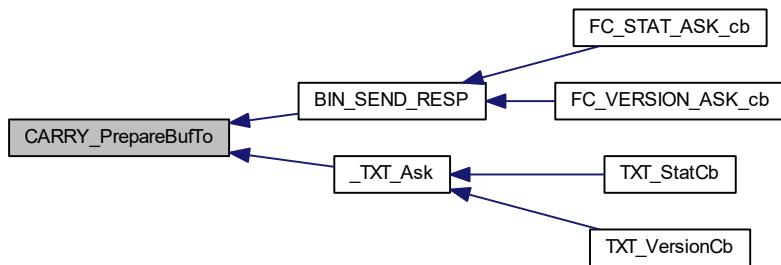
Returns

mac_buf_t* result buffer or null

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.3.5 CARRY_WriteTrace()

```
int CARRY_WriteTrace (
    dev_addr_t * buf,
    dev_addr_t target )
```

write trace to target, including target address

target address is the last one

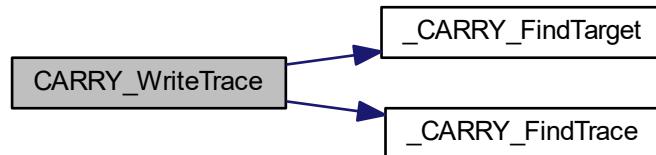
Parameters

in	<i>buf</i>	
in	<i>target</i>	

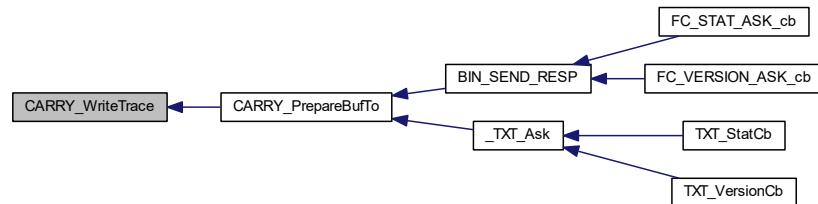
Returns

int number of written addresses or 0 when target is unknown

Here is the call graph for this function:

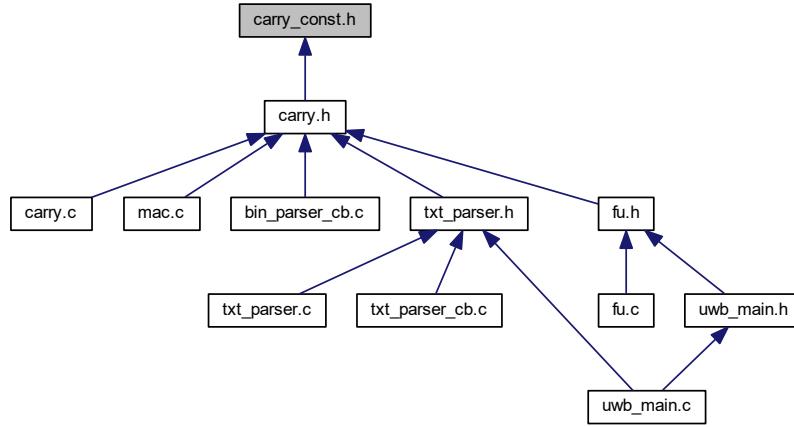


Here is the caller graph for this function:



6.10 carry_const.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define CARRY_FLAG_TARGET_DEV ($0 \lll 6$)
- #define CARRY_FLAG_TARGET_SINK ($1 \lll 6$)
- #define CARRY_FLAG_TARGET_SERVER ($2 \lll 6$)
- #define CARRY_FLAG_TARGET_MASK ($3 \lll 6$)
- #define CARRY_FLAG_ACK_REQ ($1 \lll 4$)
- #define CARRY_FLAG_REROUTE ($2 \lll 4$)
- #define CARRY_HOPS_NUM_MASK (0x07)
- #define CARRY_MAX_HOPS 8

6.10.1 Macro Definition Documentation

6.10.1.1 CARRY_FLAG_ACK_REQ

```
#define CARRY_FLAG_ACK_REQ (1 << 4)
```

6.10.1.2 CARRY_FLAG_REROUTE

```
#define CARRY_FLAG_REROUTE (2 << 4)
```

6.10.1.3 CARRY_FLAG_TARGET_DEV

```
#define CARRY_FLAG_TARGET_DEV (0 << 6)
```

6.10.1.4 CARRY_FLAG_TARGET_MASK

```
#define CARRY_FLAG_TARGET_MASK (3 << 6)
```

6.10.1.5 CARRY_FLAG_TARGET_SERVER

```
#define CARRY_FLAG_TARGET_SERVER (2 << 6)
```

6.10.1.6 CARRY_FLAG_TARGET_SINK

```
#define CARRY_FLAG_TARGET_SINK (1 << 6)
```

6.10.1.7 CARRY_HOPS_NUM_MASK

```
#define CARRY_HOPS_NUM_MASK (0x07)
```

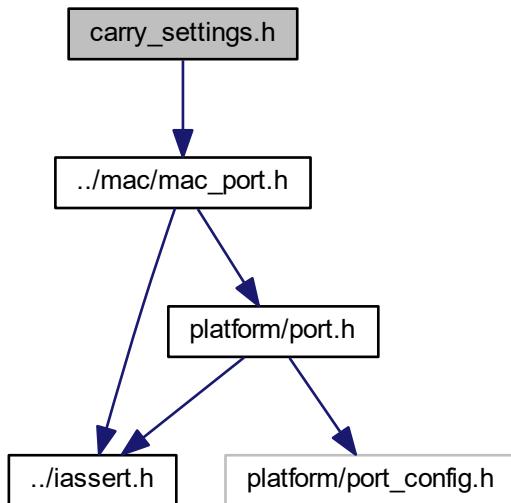
6.10.1.8 CARRY_MAX_HOPS

```
#define CARRY_MAX_HOPS 8
```

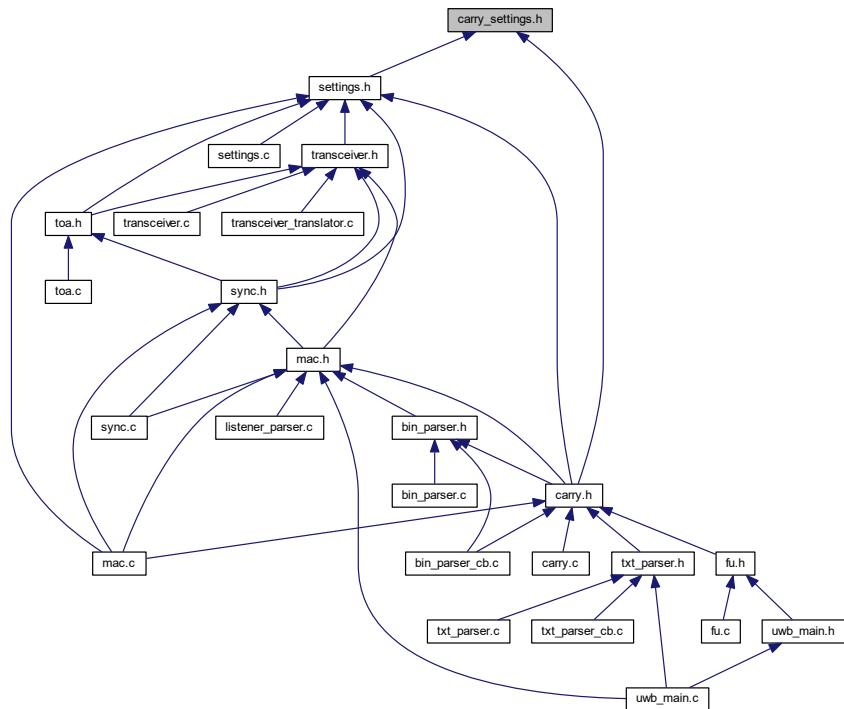
6.11 carry_settings.h File Reference

carry settings typedef and default values

```
#include "../mac/mac_port.h"
Include dependency graph for carry_settings.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `carry_settings_t`
`carry settings typedef`

Macros

- `#define CARRY_ASSERT(expr) IASSERT(expr)`
- `#define CARRY_MAX_TRACE 2`
maximum number of parents for one anchor
- `#define CARRY_MAX_TARGETS 8`
number of anchors traces in sink device
- `#define CARRY_SETTINGS_DEF`

6.11.1 Detailed Description

carry settings typedef and default values

Author

Karol Trzcinski

Date

2018-07-02

6.11.2 Macro Definition Documentation

6.11.2.1 CARRY_ASSERT

```
#define CARRY_ASSERT(  
    expr ) IASSERT(expr)
```

6.11.2.2 CARRY_MAX_TARGETS

```
#define CARRY_MAX_TARGETS 8
```

number of anchors traces in sink device

Deprecated change structure to anchor->parent

6.11.2.3 CARRY_MAX_TRACE

```
#define CARRY_MAX_TRACE 2
```

maximum number of parents for one anchor

6.11.2.4 CARRY_SETTINGS_DEF

```
#define CARRY_SETTINGS_DEF
```

Value:

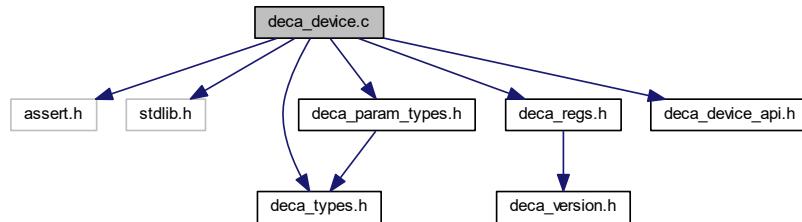
```
{  
    .trace_max_inactive_time = 1000, .trace_max_fail_cnt = 5 \  
}
```

6.12 deca_device.c File Reference

Decawave device configuration and control functions.

```
#include <assert.h>
#include <stdlib.h>
#include "deca_types.h"
#include "deca_param_types.h"
#include "deca_regs.h"
#include "deca_device_api.h"
```

Include dependency graph for deca_device.c:



Data Structures

- struct [dwt_local_data_t](#)

Macros

- #define FORCE_SYS_XTI 0
- #define ENABLE_ALL_SEQ 1
- #define FORCE_SYS_PLL 2
- #define READ_ACC_ON 7
- #define READ_ACC_OFF 8
- #define FORCE OTP ON 11
- #define FORCE OTP OFF 12
- #define FORCE_TX_PLL 13
- #define FORCE_LDE 14
- #define FCTRL_ACK_REQ_MASK 0x20
- #define FCTRL_LEN_MAX 2
- #define LDOTUNE_ADDRESS (0x04)
- #define PARTID_ADDRESS (0x06)
- #define LOTID_ADDRESS (0x07)
- #define VBAT_ADDRESS (0x08)
- #define VTEMP_ADDRESS (0x09)
- #define XTRIM_ADDRESS (0x1E)
- #define B20_SIGN_EXTEND_TEST (0x00100000UL)
- #define B20_SIGN_EXTEND_MASK (0xFFFF0000UL)

Functions

- void `_dwt_enableclocks` (int clocks)
- void `_dwt_configlde` (int prf)
- void `_dwt_loaducodefromrom` (void)

load ucode from OTP MEMORY or ROM
- `uint32 _dwt_otpread` (uint32 address)
- `uint32 _dwt_otpprogword32` (uint32 data, uint16 address)
- void `_dwt_aonarrayupload` (void)

This function uploads always on (AON) data array and configuration. Thus if this function is used, then `_dwt_aonconfigupload` is not necessary. The DW1000 will go to SLEEP straight after this if the DWT_SLP_EN has been set.
- int `dwt_setlocaldataptr` (unsigned int index)
- int `dwt_initialise` (uint16 config)
- `uint8 dwt_otprevision` (void)

This is used to return the read OTP revision.
- void `dwt_setfinegraintxseq` (int enable)
- void `dwt_setlnapemode` (int lna, int pa)
- void `dwt_setgpiodirection` (uint32 gpioNum, uint32 direction)
- void `dwt_setgpiovalue` (uint32 gpioNum, uint32 value)
- `uint32 dwt_getpartid` (void)

This is used to return the read part ID of the device.
- `uint32 dwt_getlotid` (void)

This is used to return the read lot ID of the device.
- `uint32 dwt_readdevid` (void)

This is used to return the read device type and revision information of the DW1000 device (MP part is 0xDECA0130)
- void `dwt_configuretxrf` (`dwt_txconfig_t` *config)
- void `dwt_configure` (`dwt_config_t` *config)
- void `dwt_setrxantennadelay` (uint16 rxDelay)
- void `dwt_settxantennadelay` (uint16 txDelay)
- int `dwt_writetxdata` (uint16 txFrameLength, uint8 *txFrameBytes, uint16 txBufferOffset)
- void `dwt_writetxfctrl` (uint16 txFrameLength, uint16 txBufferOffset, int ranging)
- void `dwt_readdrxdata` (uint8 *buffer, uint16 length, uint16 rxBufferOffset)
- void `dwt_readaccdata` (uint8 *buffer, uint16 len, uint16 accOffset)
- `int32 dwt_readcarrierintegrator` (void)

This is used to read the RX carrier integrator value (relating to the frequency offset of the TX node)
- void `dwt_readdiagnostics` (`dwt_rxdiag_t` *diagnostics)
- void `dwt_readtxtimestamp` (uint8 *timestamp)
- `uint32 dwt_readtxtimestampphi32` (void)

This is used to read the high 32-bits of the TX timestamp (adjusted with the programmed antenna delay)
- `uint32 dwt_readtxtimestamplo32` (void)

This is used to read the low 32-bits of the TX timestamp (adjusted with the programmed antenna delay)
- `uint32 dwt_readrxtimestampphi32` (void)

This is used to read the high 32-bits of the RX timestamp (adjusted with the programmed antenna delay)
- `uint32 dwt_readrxtimestamplo32` (void)

This is used to read the low 32-bits of the RX timestamp (adjusted with the programmed antenna delay)
- `uint32 dwt_readsystimestampphi32` (void)

This is used to read the high 32-bits of the system time.
- void `dwt_readsystime` (uint8 *timestamp)
- void `dwt_writetodevice` (uint16 recordNumber, uint16 index, uint32 length, const uint8 *buffer)
- void `dwt_readfromdevice` (uint16 recordNumber, uint16 index, uint32 length, uint8 *buffer)
- `uint32 dwt_read32bitoffsetreg` (int regFileID, int regOffset)

- `uint16 dwt_read16bitoffsetreg` (int regFileID, int regOffset)
- `uint8 dwt_read8bitoffsetreg` (int regFileID, int regOffset)
- `void dwt_write8bitoffsetreg` (int regFileID, int regOffset, `uint8` regval)
- `void dwt_write16bitoffsetreg` (int regFileID, int regOffset, `uint16` regval)
- `void dwt_write32bitoffsetreg` (int regFileID, int regOffset, `uint32` regval)
- `void dwt_enableframefilter` (`uint16` enable)
- `void dwt_setpanid` (`uint16` panID)
- `void dwt_setaddress16` (`uint16` shortAddress)
- `void dwt_seteui` (`uint8` *eui64)
- `void dwt_geteui` (`uint8` *eui64)
- `void dwt_otpread` (`uint32` address, `uint32` *array, `uint8` length)
- `uint32 dwt_otpsetmrregs` (int mode)
- `int dwt_otpwriteandverify` (`uint32` value, `uint16` address)
- `void _dwt_aonconfigupload` (void)

This function uploads always on (AON) configuration, as set in the AON_CFG0_OFFSET register.

- `void dwt_entersleep` (void)

This function puts the device into deep sleep or sleep. `dwt_configuresleep()` should be called first to configure the sleep and on-wake/wake-up parameters.

- `void dwt_configuresleepcnt` (`uint16` sleepcnt)
- `uint16 dwt_calibratesleepcnt` (void)

calibrates the local oscillator as its frequency can vary between 7 and 13kHz depending on temp and voltage

- `void dwt_configuresleep` (`uint16` mode, `uint8` wake)
- `void dwt_entersleepaftertx` (int enable)

sets the auto TX to sleep bit. This means that after a frame transmission the device will enter deep sleep mode. The `dwt_configuresleep()` function needs to be called before this to configure the on-wake settings

- `int dwt_spicswakeup` (`uint8` *buff, `uint16` length)
- `void dwt_loadopsettabfromotp` (`uint8` ops_sel)
- `void dwt_setsmarttxpower` (int enable)
- `void dwt_enableautoack` (`uint8` responseDelayTime)
- `void dwt_setdblrxbuffmode` (int enable)
- `void dwt_setrxaftertxdelay` (`uint32` rxDelayTime)
- `void dwt_setcallbacks` (`dwt_cb_t` cbTxDone, `dwt_cb_t` cbRxOk, `dwt_cb_t` cbRxTo, `dwt_cb_t` cbRxErr)
- `uint8 dwt_checkirq` (void)

This function checks if the IRQ line is active - this is used instead of interrupt handler.

- `void dwt_isr` (void)

This is the DW1000's general Interrupt Service Routine. It will process/report the following events:

- `void dwt_lowpowerlistenisr` (void)
- `void dwt_setleds` (`uint8` mode)
- `void _dwt_disablesequencing` (void)

This function disables the TX blocks sequencing, it disables PMSC control of RF blocks, system clock is also set to XTAL.

- `void dwt_setdelayedtrxtime` (`uint32` starttime)
- `int dwt_starttx` (`uint8` mode)
- `void dwt_forcetrxoff` (void)

This is used to turn off the transceiver.

- `void dwt_syncrxbufptrs` (void)

this function synchronizes rx buffer pointers need to make sure that the host/IC buffer pointers are aligned before starting RX

- `void dwt_setsniffmode` (int enable, `uint8` timeOn, `uint8` timeOff)
- `void dwt_setlowpowerlistening` (int enable)
- `void dwt_setsnoozetime` (`uint8` snooze_time)
- `int dwt_rxenable` (int mode)
- `void dwt_setrxtimeout` (`uint16` time)
- `void dwt_setpreambledetecttimeout` (`uint16` timeout)

- void `dwt_setinterrupt` (uint32 bitmask, uint8 enable)
- void `dwt_configeventcounters` (int enable)
- void `dwt_readeventcounters` (`dwt_deviceentcnts_t` *counters)
- void `dwt_rxreset` (void)

this function resets the receiver of the DW1000
- void `dwt_softreset` (void)

this function resets the DW1000
- void `dwt_setxtaltrim` (uint8 value)
- uint8 `dwt_getinitxtaltrim` (void)

This function returns the value of XTAL trim that has been applied during initialisation (`dwt_init`). This can be either the value read in OTP memory or a default value.
- void `dwt_configcwmode` (uint8 chan)
- void `dwt_configcontinuousframemode` (uint32 framerepetitionrate)
- uint16 `dwt_readtempvbat` (uint8 fastSPI)
- uint8 `dwt_readwakeuptemp` (void)

this function reads the temperature of the DW1000 that was sampled on waking from Sleep/Deepsleep. They are not current values, but read on last wakeup if DWT_TANDV bit is set in mode parameter of `dwt_configuresleep`
- uint8 `dwt_readwakeupvbat` (void)

this function reads the battery voltage of the DW1000 that was sampled on waking from Sleep/Deepsleep. They are not current values, but read on last wakeup if DWT_TANDV bit is set in mode parameter of `dwt_configuresleep`
- uint32 `dwt_calcbandwidthtempadj` (uint16 target_count)
- uint32 `_dwt_computetxpowersetting` (uint32 ref_powerreg, int32 power_adj)
- uint32 `dwt_calcpowertempadj` (uint8 channel, uint32 ref_powerreg, double curr_temp, double ref_temp)
- uint16 `dwt_calcpccount` (uint8 pgdly)

6.12.1 Detailed Description

Decawave device configuration and control functions.

Attention

Copyright 2013 (c) Decawave Ltd, Dublin, Ireland.

All rights reserved.

6.12.2 Macro Definition Documentation

6.12.2.1 B20_SIGN_EXTEND_MASK

```
#define B20_SIGN_EXTEND_MASK (0xFFFF00000UL)
```

6.12.2.2 B20_SIGN_EXTEND_TEST

```
#define B20_SIGN_EXTEND_TEST (0x00100000UL)
```

6.12.2.3 ENABLE_ALL_SEQ

```
#define ENABLE_ALL_SEQ 1
```

6.12.2.4 FCTRL_ACK_REQ_MASK

```
#define FCTRL_ACK_REQ_MASK 0x20
```

6.12.2.5 FCTRL_LEN_MAX

```
#define FCTRL_LEN_MAX 2
```

6.12.2.6 FORCE_LDE

```
#define FORCE_LDE 14
```

6.12.2.7 FORCE OTP OFF

```
#define FORCE OTP OFF 12
```

6.12.2.8 FORCE OTP ON

```
#define FORCE OTP ON 11
```

6.12.2.9 FORCE_SYS_PLL

```
#define FORCE_SYS_PLL 2
```

6.12.2.10 FORCE_SYS_XTI

```
#define FORCE_SYS_XTI 0
```

6.12.2.11 FORCE_TX_PLL

```
#define FORCE_TX_PLL 13
```

6.12.2.12 LDOTUNE_ADDRESS

```
#define LDOTUNE_ADDRESS (0x04)
```

6.12.2.13 LOTID_ADDRESS

```
#define LOTID_ADDRESS (0x07)
```

6.12.2.14 PARTID_ADDRESS

```
#define PARTID_ADDRESS (0x06)
```

6.12.2.15 READ_ACC_OFF

```
#define READ_ACC_OFF 8
```

6.12.2.16 READ_ACC_ON

```
#define READ_ACC_ON 7
```

6.12.2.17 VBAT_ADDRESS

```
#define VBAT_ADDRESS (0x08)
```

6.12.2.18 VTEMP_ADDRESS

```
#define VTEMP_ADDRESS (0x09)
```

6.12.2.19 XTRIM_ADDRESS

```
#define XTRIM_ADDRESS (0x1E)
```

6.12.3 Function Documentation

6.12.3.1 _dwt_aonarrayupload()

```
void _dwt_aonarrayupload (
    void )
```

This function uploads always on (AON) data array and configuration. Thus if this function is used, then `_dwt_aonconfigupload` is not necessary. The DW1000 will go straight to SLEEP after this if the `DWT_SLP_EN` has been set.

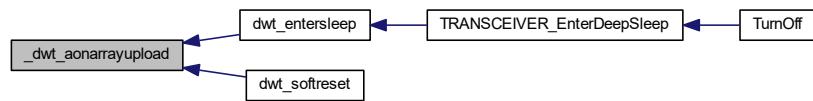
input parameters

output parameters

no return value Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.2 _dwt_aonconfigupload()

```
_dwt_aonconfigupload (
    void )
```

This function uploads always on (AON) configuration, as set in the AON_CFG0_OFFSET register.

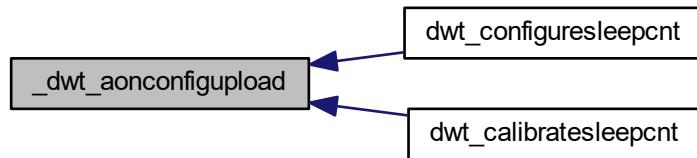
input parameters

output parameters

no return value Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.3 _dwt_computetxpowersetting()

```
uint32 _dwt_computetxpowersetting (
    uint32 ref_powerreg,
    int32 power_adj )
```

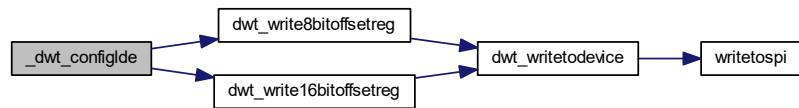
Here is the caller graph for this function:



6.12.3.4 _dwt_configlde()

```
void _dwt_configlde (
    int prf )
```

Here is the call graph for this function:



6.12.3.5 _dwt_disablesequencing()

```
_dwt_disablesequencing (
    void )
```

This function disables the TX blocks sequencing, it disables PMSC control of RF blocks, system clock is also set to XTAL.

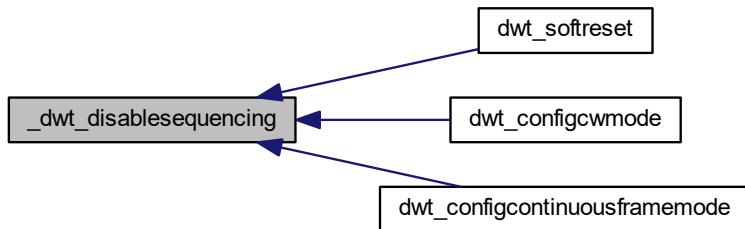
input parameters none

output parameters none

no return value Here is the call graph for this function:



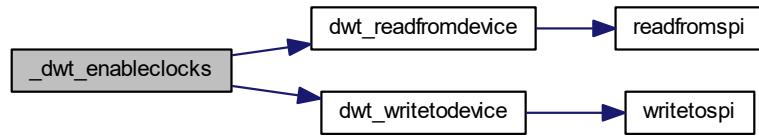
Here is the caller graph for this function:



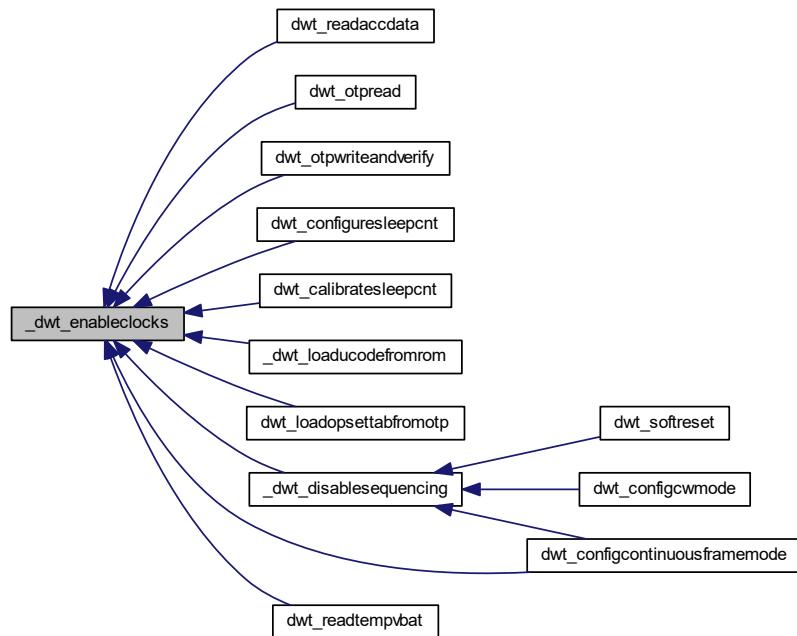
6.12.3.6 `_dwt_enableclocks()`

```
void _dwt_enableclocks (
    int clocks )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.7 `_dwt_loaducodefromrom()`

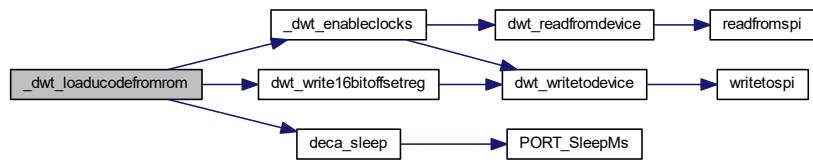
```
void _dwt_loaducodefromrom (
    void )
```

load ucode from OTP MEMORY or ROM

input parameters

output parameters

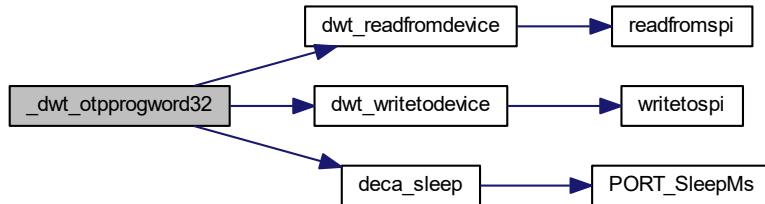
no return value Here is the call graph for this function:



6.12.3.8 `_dwt_otpprogword32()`

```
uint32 _dwt_otpprogword32 (
    uint32 data,
    uint16 address )
```

Here is the call graph for this function:



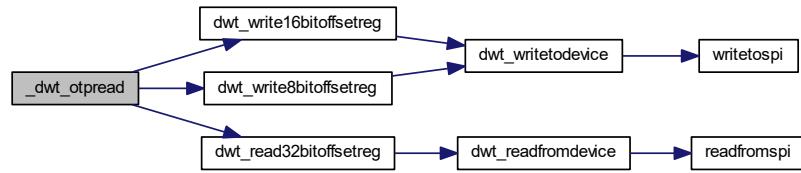
Here is the caller graph for this function:



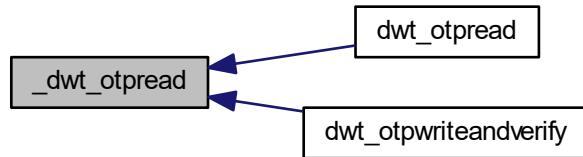
6.12.3.9 _dwt_otpread()

```
uint32 _dwt_otpread (
    uint32 address )
```

Here is the call graph for this function:



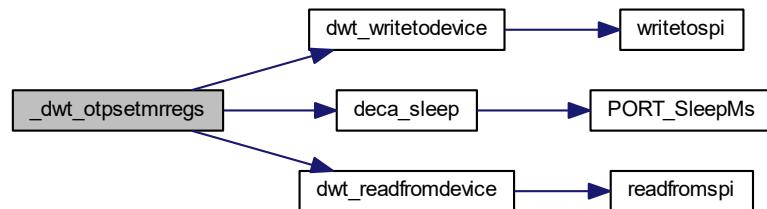
Here is the caller graph for this function:



6.12.3.10 _dwt_otpsetmrregs()

```
uint32 _dwt_otpsetmrregs (
    int mode )
```

Here is the call graph for this function:



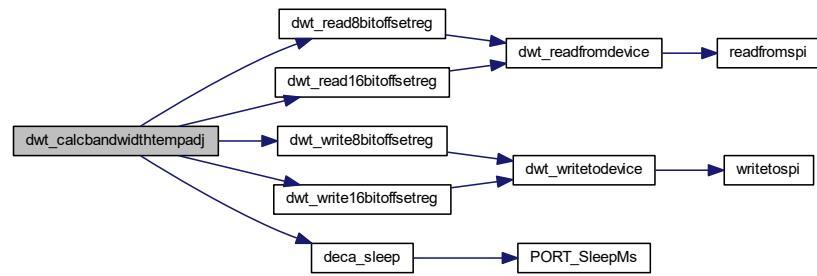
Here is the caller graph for this function:



6.12.3.11 dwt_calcbandwidthtempadj()

```
uint32 dwt_calcbandwidthtempadj (
    uint16 target_count )
```

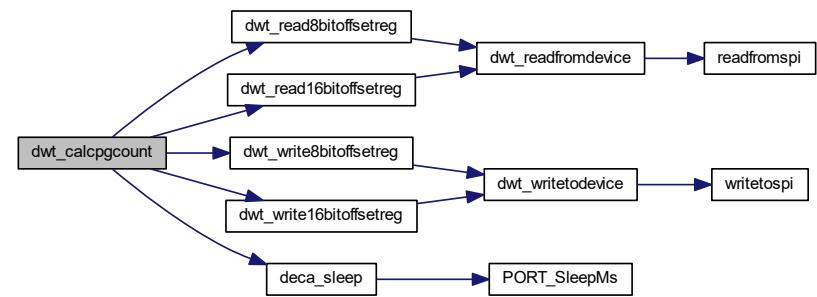
Here is the call graph for this function:



6.12.3.12 dwt_calcpgcount()

```
uint16 dwt_calcpgcount (
    uint8 pgdly )
```

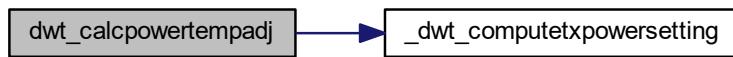
Here is the call graph for this function:



6.12.3.13 dwt_calcpowertempadj()

```
uint32 dwt_calcpowertempadj (
    uint8 channel,
    uint32 ref_powerreg,
    double curr_temp,
    double ref_temp )
```

Here is the call graph for this function:



6.12.3.14 dwt_calibratesleepcnt()

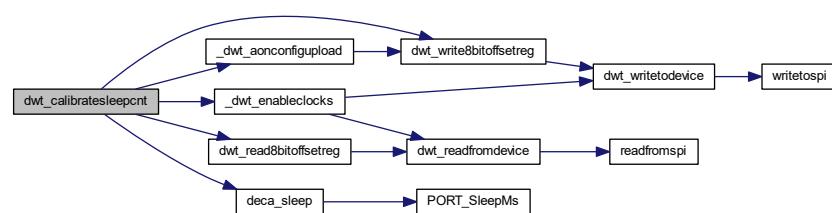
```
dwt_calibratesleepcnt (
    void )
```

calibrates the local oscillator as its frequency can vary between 7 and 13kHz depending on temp and voltage

NOTE: this function needs to be run before dwt_configuresleepcnt, so that we know what the counter units are
input parameters

output parameters

returns the number of XTAL/2 cycles per low-power oscillator cycle. LP OSC frequency = 19.2 MHz/return value
Here is the call graph for this function:



6.12.3.15 dwt_checkirq()

```
dwt_checkirq (
    void  )
```

This function checks if the IRQ line is active - this is used instead of interrupt handler.

input parameters

output parameters

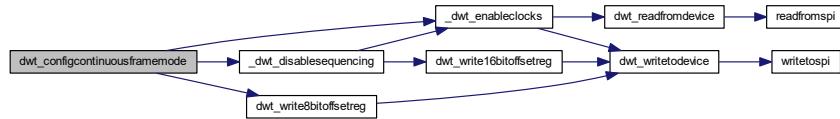
return value is 1 if the IRQS bit is set and 0 otherwise Here is the call graph for this function:



6.12.3.16 dwt_configcontinuousframemode()

```
void dwt_configcontinuousframemode (
    uint32 framerepetitionrate )
```

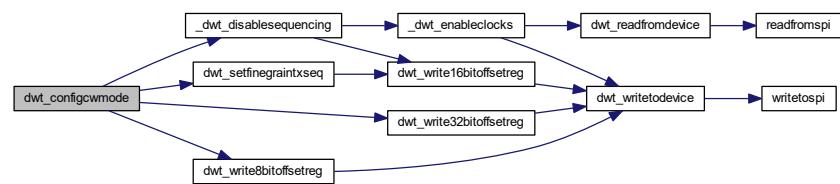
Here is the call graph for this function:



6.12.3.17 dwt_configcwmode()

```
void dwt_configcwmode (
    uint8 chan )
```

Here is the call graph for this function:



6.12.3.18 dwt_configeventcounters()

```
void dwt_configeventcounters (
    int enable )
```

Here is the call graph for this function:



6.12.3.19 dwt_configure()

```
void dwt_configure (
    dwt_config_t * config )
```

6.12.3.20 dwt_configuresleep()

```
void dwt_configuresleep (
    uint16 mode,
    uint8 wake )
```

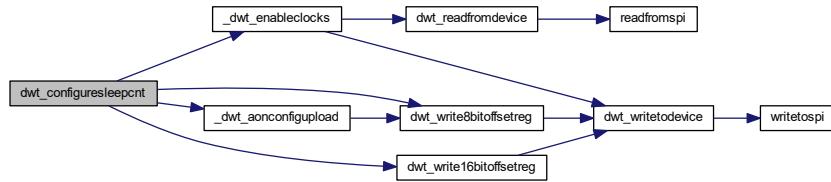
Here is the caller graph for this function:



6.12.3.21 dwt_configuresleepcnt()

```
void dwt_configuresleepcnt (
    uint16 sleepcnt )
```

Here is the call graph for this function:



6.12.3.22 dwt_configuretxrf()

```
void dwt_configuretxrf (
    dwt_txconfig_t * config )
```

Here is the call graph for this function:



6.12.3.23 dwt_enableautoack()

```
void dwt_enableautoack (
    uint8 responseDelayTime )
```

Here is the call graph for this function:



6.12.3.24 dwt_enableframefilter()

```
void dwt_enableframefilter (
    uint16 enable )
```

6.12.3.25 dwt_entersleep()

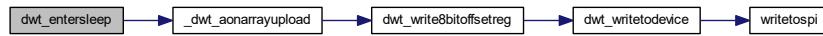
```
dwt_entersleep (
    void )
```

This function puts the device into deep sleep or sleep. [dwt_configuresleep\(\)](#) should be called first to configure the sleep and on-wake/wake-up parameters.

input parameters

output parameters

no return value Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.26 dwt_entersleepaftertx()

```
dwt_entersleepaftertx (
    int enable )
```

sets the auto TX to sleep bit. This means that after a frame transmission the device will enter deep sleep mode. The [dwt_configuresleep\(\)](#) function needs to be called before this to configure the on-wake settings

NOTE: the IRQ line has to be low/inactive (i.e. no pending events)

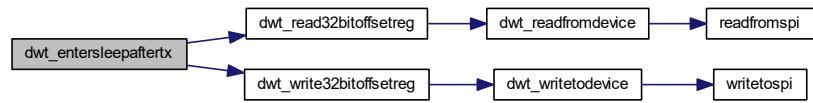
input parameters

Parameters

<code>enable</code>	- 1 to configure the device to enter deep sleep after TX, 0 - disables the configuration
---------------------	--

output parameters

no return value Here is the call graph for this function:

6.12.3.27 `dwt_forcetrxoff()`

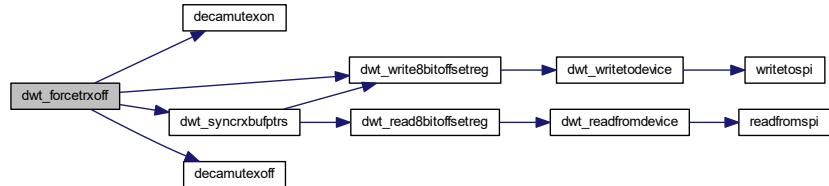
```
dwt_forcetrxoff (
    void )
```

This is used to turn off the transceiver.

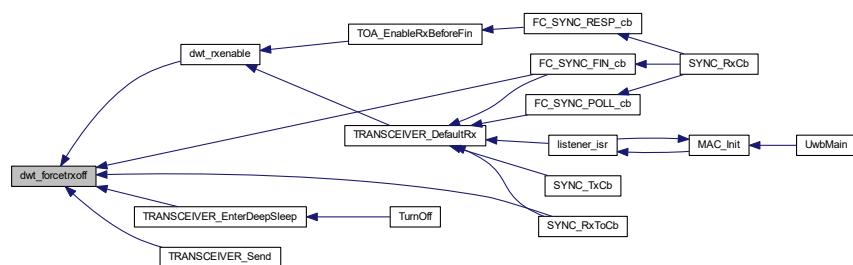
input parameters

output parameters

no return value Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.28 dwt_geteui()

```
void dwt_geteui (
    uint8 * eui64 )
```

Here is the call graph for this function:



6.12.3.29 dwt_getinitxtaltrim()

```
dwt_getinitxtaltrim (
    void )
```

This function returns the value of XTAL trim that has been applied during initialisation (dwt_init). This can be either the value read in OTP memory or a default value.

NOTE: The value returned by this function is the initial value only! It is not updated on dwt_setxtaltrim calls.

input parameters

output parameters

returns the XTAL trim value set upon initialisation

6.12.3.30 dwt_getlotid()

```
dwt_getlotid (
    void )
```

This is used to return the read lot ID of the device.

NOTE: [dwt_initialise\(\)](#) must be called prior to this function so that it can return a relevant value.

input parameters

output parameters

returns the 32 bit lot ID value as programmed in the factory

6.12.3.31 dwt_getpartid()

```
dwt_getpartid (
    void )
```

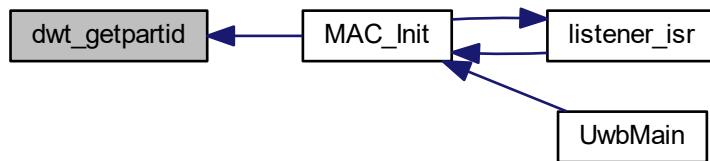
This is used to return the read part ID of the device.

NOTE: [dwt_initialise\(\)](#) must be called prior to this function so that it can return a relevant value.

input parameters

output parameters

returns the 32 bit part ID value as programmed in the factory Here is the caller graph for this function:



6.12.3.32 dwt_initialise()

```
int dwt_initialise (
    uint16 config )
```

6.12.3.33 dwt_isr()

```
dwt_isr (
    void )
```

This is the DW1000's general Interrupt Service Routine. It will process/report the following events:

- RXFCG (through cbRxOk callback)
- TXFRS (through cbTxDone callback)
- RXRFTO/RXPTO (through cbRxTo callback)
- RXPHE/RXFCE/RXRFSL/RXSFDTO/AFFREJ/LDEERR (through cbRxTo cbRxErr)

For all events, corresponding interrupts are cleared and necessary resets are performed. In addition, in the received frame information and frame control are read before calling the callback. If double buffering is active, the ISR will also toggle between reception buffers once the reception callback processing has ended.

/ This version of the ISR supports double buffering but does not support automatic RX re-enabling! */*

NOTE: In PC based system using (Cheetah or ARM) USB to SPI converter there can be no interrupts, however we still need something to take the place of it and operate in a polled way. In an embedded system this function should be configured to be triggered on any of the interrupts described above.

input parameters

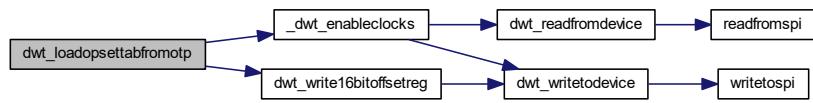
output parameters

no return value

6.12.3.34 dwt_loadopsettabfromotp()

```
void dwt_loadopsettabfromotp (
    uint8 ops_sel )
```

Here is the call graph for this function:



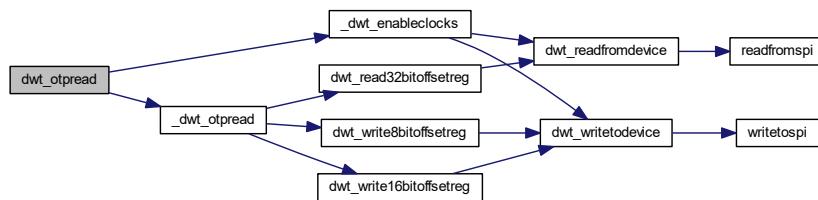
6.12.3.35 dwt_lowpowerlistenISR()

```
void dwt_lowpowerlistenISR (
    void )
```

6.12.3.36 dwt_otpread()

```
void dwt_otpread (
    uint32 address,
    uint32 * array,
    uint8 length )
```

Here is the call graph for this function:



6.12.3.37 dwt_otprevision()

```
dwt_otprevision (
    void )
```

This is used to return the read OTP revision.

NOTE: [dwt_initialise\(\)](#) must be called prior to this function so that it can return a relevant value.

input parameters

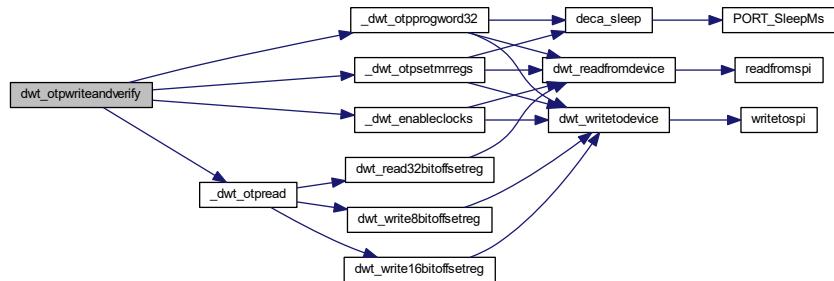
output parameters

returns the read OTP revision value

6.12.3.38 dwt_otpwriteandverify()

```
int dwt_otpwriteandverify (
    uint32 value,
    uint16 address )
```

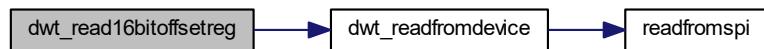
!!!!!!!!!!!!!! NOTE !!!!!!!!!!!!!!! Here is the call graph for this function:



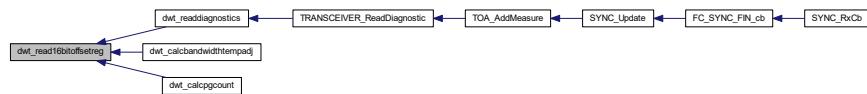
6.12.3.39 dwt_read16bitoffsetreg()

```
uint16 dwt_read16bitoffsetreg (
    int regFileID,
    int regOffset )
```

Here is the call graph for this function:



Here is the caller graph for this function:



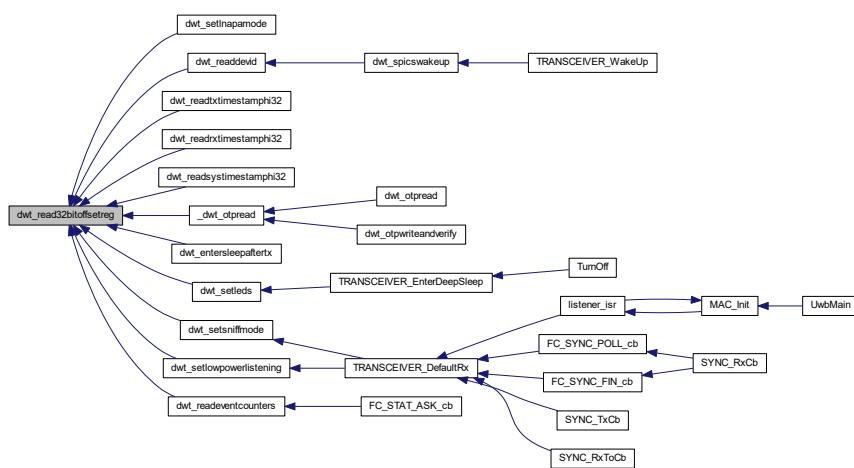
6.12.3.40 dwt_read32bitoffsetreg()

```
uint32 dwt_read32bitoffsetreg (
    int regFileID,
    int regOffset )
```

Here is the call graph for this function:



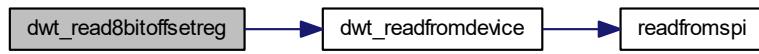
Here is the caller graph for this function:



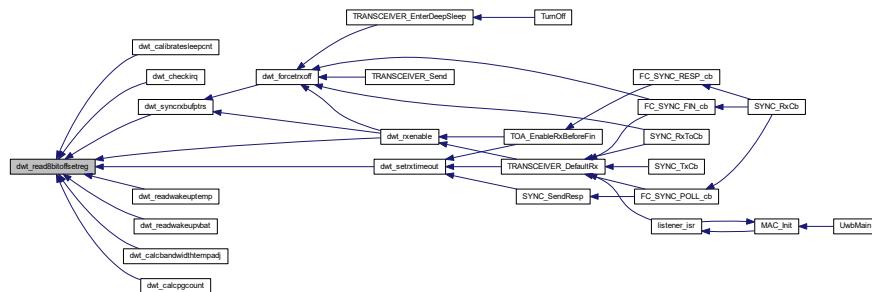
6.12.3.41 dwt_read8bitoffsetreg()

```
uint8 dwt_read8bitoffsetreg (
    int regFileID,
    int regOffset )
```

Here is the call graph for this function:



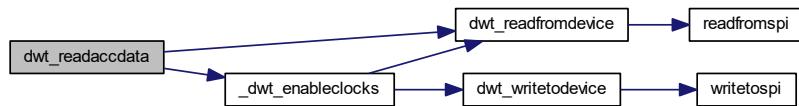
Here is the caller graph for this function:



6.12.3.42 dwt_readaccdata()

```
void dwt_readaccdata (
    uint8 * buffer,
    uint16 len,
    uint16 accOffset )
```

Here is the call graph for this function:



6.12.3.43 dwt_readcarrierintegrator()

```
dwt_readcarrierintegrator (
    void )
```

This is used to read the RX carrier integrator value (relating to the frequency offset of the TX node)

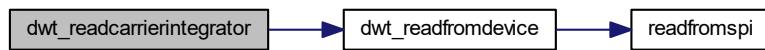
NOTE: This is a 21-bit signed quantity, the function sign extends the most significant bit, which is bit #20 (numbering from bit zero) to return a 32-bit signed integer value.

input parameters - NONE

return value - the (int32) signed carrier integrator value. A positive value means the local RX clock is running faster than the remote TX device.

input parameters - NONE

return value - the (int32) signed carrier integrator value. A positive value means the local RX clock is running faster than the remote TX device. Here is the call graph for this function:



6.12.3.44 dwt_readdevid()

```
dwt_readdevid (
    void )
```

This is used to return the read device type and revision information of the DW1000 device (MP part is 0xDECA0130)

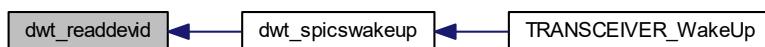
input parameters

output parameters

returns the read value which for DW1000 is 0xDECA0130 Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.45 dwt_readdiagnostics()

```
void dwt_readdiagnostics (
    dwt_rxdiag_t * diagnostics )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.46 dwt_readeventcounters()

```
void dwt_readeventcounters (
    dwt_deviceeventcnts_t * counters )
```

Here is the call graph for this function:



Here is the caller graph for this function:



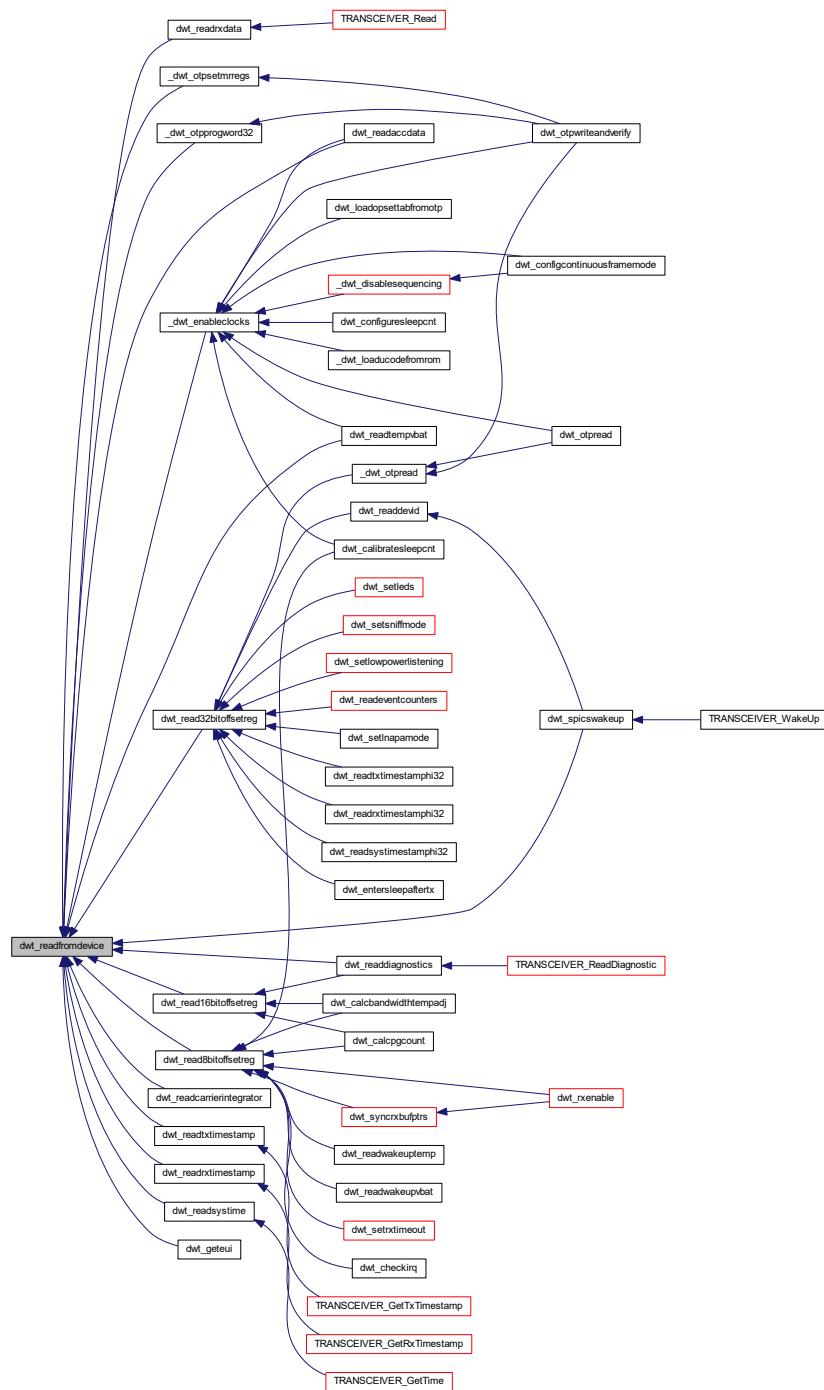
6.12.3.47 dwt_readfromdevice()

```
void dwt_readfromdevice (
    uint16 recordNumber,
    uint16 index,
    uint32 length,
    uint8 * buffer )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.48 `dwt_readrxdata()`

```
void dwt_readrxdata (
    uint8 * buffer,
```

```

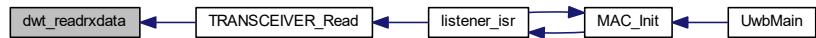
    uint16 length,
    uint16 rxBufferOffset )

```

Here is the call graph for this function:



Here is the caller graph for this function:



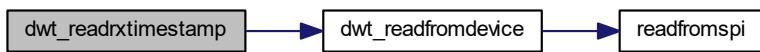
6.12.3.49 dwt_readrxtimestamp()

```

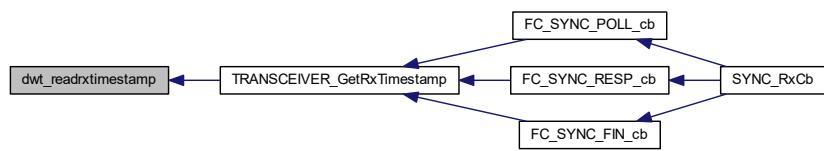
void dwt_readrxtimestamp (
    uint8 * timestamp )

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.50 dwt_readrxtimestamphi32()

```
dwt_readrxtimestamphi32 (
    void )
```

This is used to read the high 32-bits of the RX timestamp (adjusted with the programmed antenna delay)

input parameters

output parameters

returns high 32-bits of RX timestamp Here is the call graph for this function:



6.12.3.51 dwt_readrxtimestamplo32()

```
dwt_readrxtimestamplo32 (
    void )
```

This is used to read the low 32-bits of the RX timestamp (adjusted with the programmed antenna delay)

input parameters

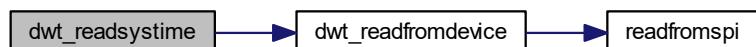
output parameters

returns low 32-bits of RX timestamp

6.12.3.52 dwt_readsystime()

```
void dwt_readsystime (
    uint8 * timestamp )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.53 dwt_readsystimestamphi32()

```
dwt_readsystimestamphi32 (
    void )
```

This is used to read the high 32-bits of the system time.

input parameters

output parameters

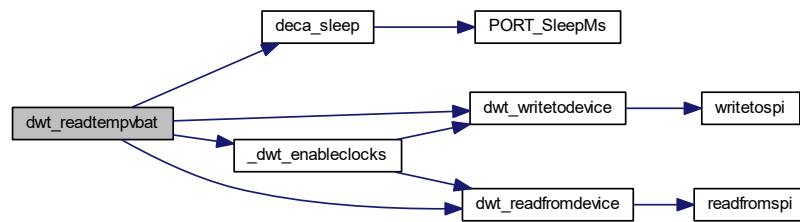
returns high 32-bits of system time timestamp Here is the call graph for this function:



6.12.3.54 dwt_readtempvbat()

```
uint16 dwt_readtempvbat (
    uint8 fastSPI )
```

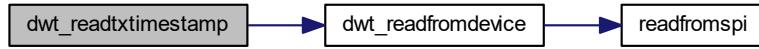
Here is the call graph for this function:



6.12.3.55 dwt_readtxtimestamp()

```
void dwt_readtxtimestamp (
    uint8 * timestamp )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.56 dwt_readtxtimestamphi32()

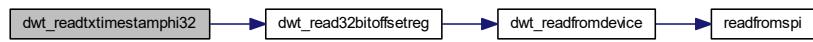
```
dwt_readtxtimestamphi32 (
    void )
```

This is used to read the high 32-bits of the TX timestamp (adjusted with the programmed antenna delay)

input parameters

output parameters

returns high 32-bits of TX timestamp Here is the call graph for this function:



6.12.3.57 dwt_readtxtimestamplo32()

```
dwt_readtxtimestamplo32 (
    void )
```

This is used to read the low 32-bits of the TX timestamp (adjusted with the programmed antenna delay)

input parameters

output parameters

returns low 32-bits of TX timestamp

6.12.3.58 dwt_readwakeuptemp()

```
dwt_readwakeuptemp (
    void )
```

this function reads the temperature of the DW1000 that was sampled on waking from Sleep/Deepsleep. They are not current values, but read on last wakeup if DWT_TANDV bit is set in mode parameter of dwt_configuresleep

input parameters:

output parameters:

returns: 8-bit raw temperature sensor value Here is the call graph for this function:



6.12.3.59 dwt_readwakeupvbat()

```
dwt_readwakeupvbat (
    void )
```

this function reads the battery voltage of the DW1000 that was sampled on waking from Sleep/Deepsleep. They are not current values, but read on last wakeup if DWT_TANDV bit is set in mode parameter of dwt_configuresleep

input parameters:

output parameters:

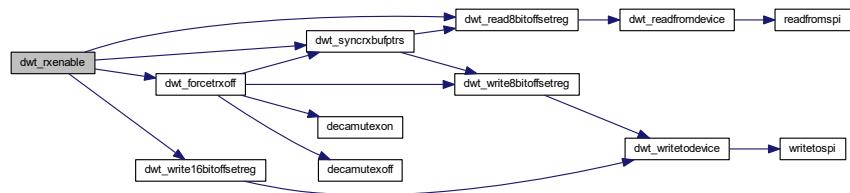
returns: 8-bit raw battery voltage sensor value Here is the call graph for this function:



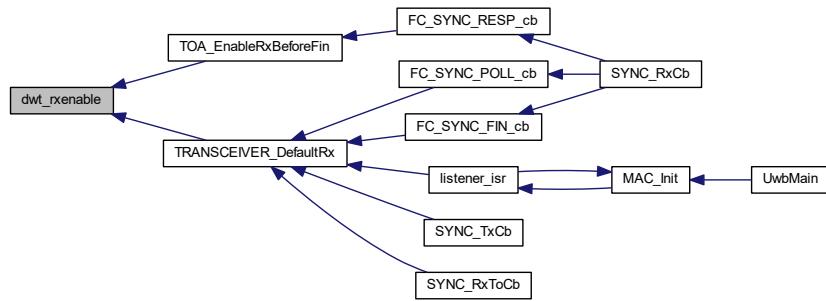
6.12.3.60 dwt_rxenable()

```
int dwt_rxenable (
    int mode )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.61 dwt_rxreset()

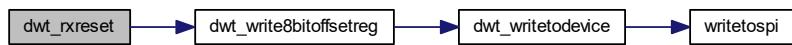
```
dwt_rxreset (
    void )
```

this function resets the receiver of the DW1000

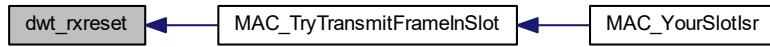
input parameters:

output parameters

no return value Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.62 dwt_setaddress16()

```
void dwt_setaddress16 (
    uint16 shortAddress )
```

Here is the call graph for this function:



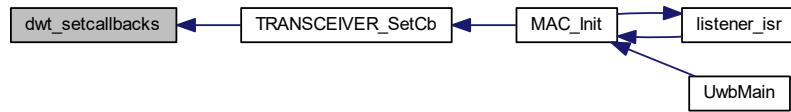
Here is the caller graph for this function:



6.12.3.63 dwt_setcallbacks()

```
void dwt_setcallbacks (
    dwt_cb_t cbTxDone,
    dwt_cb_t cbRxOk,
    dwt_cb_t cbRxTo,
    dwt_cb_t cbRxErr )
```

Here is the caller graph for this function:



6.12.3.64 dwt_setdblrxbuffmode()

```
void dwt_setdblrxbuffmode (
    int enable )
```

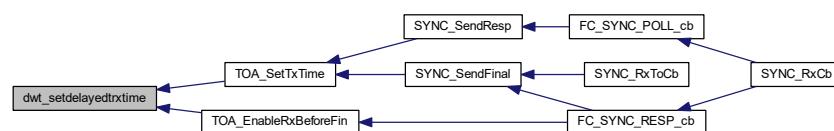
6.12.3.65 dwt_setdelayedtrxtime()

```
void dwt_setdelayedtrxtime (
    uint32 starttime )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.66 dwt_seteui()

```
void dwt_seteui (
    uint8 * eui64 )
```

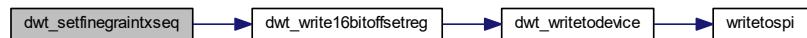
Here is the call graph for this function:



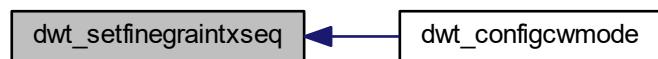
6.12.3.67 dwt_setfinegraintxseq()

```
void dwt_setfinegraintxseq (
    int enable )
```

Here is the call graph for this function:



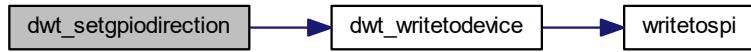
Here is the caller graph for this function:



6.12.3.68 dwt_setgpiodirection()

```
void dwt_setgpiodirection (
    uint32 gpioNum,
    uint32 direction )
```

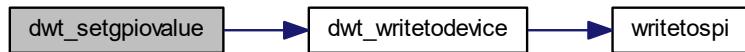
Here is the call graph for this function:



6.12.3.69 dwt_setgpiovalue()

```
void dwt_setgpiovalue (
    uint32 gpioNum,
    uint32 value )
```

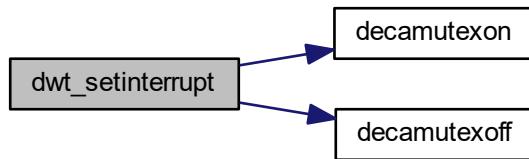
Here is the call graph for this function:



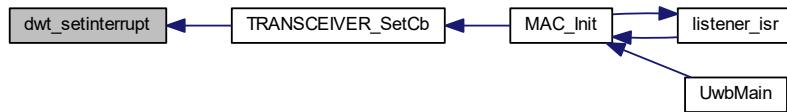
6.12.3.70 dwt_setinterrupt()

```
void dwt_setinterrupt (
    uint32 bitmask,
    uint8 enable )
```

Here is the call graph for this function:



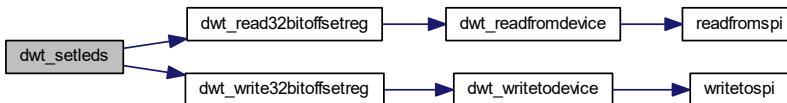
Here is the caller graph for this function:



6.12.3.71 dwt_setleds()

```
void dwt_setleds (
    uint8 mode )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.72 dwt_setlnapemode()

```
void dwt_setlnapemode (
    int lna,
    int pa )
```

Here is the call graph for this function:



6.12.3.73 dwt_setlocaldataptr()

```
int dwt_setlocaldataptr (
    unsigned int index )
```

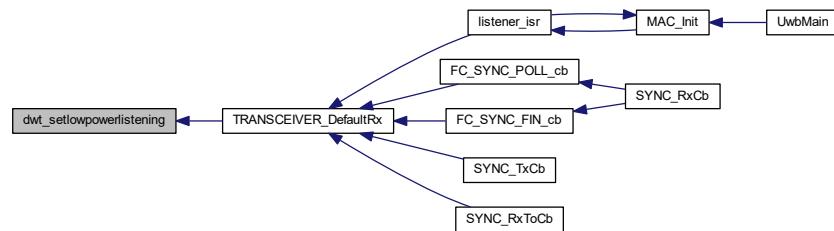
6.12.3.74 dwt_setlowpowerlistening()

```
void dwt_setlowpowerlistening (
    int enable )
```

Here is the call graph for this function:



Here is the caller graph for this function:



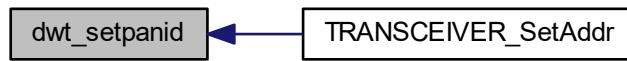
6.12.3.75 dwt_setpanid()

```
void dwt_setpanid (
    uint16 panID )
```

Here is the call graph for this function:



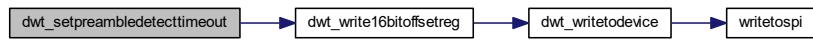
Here is the caller graph for this function:



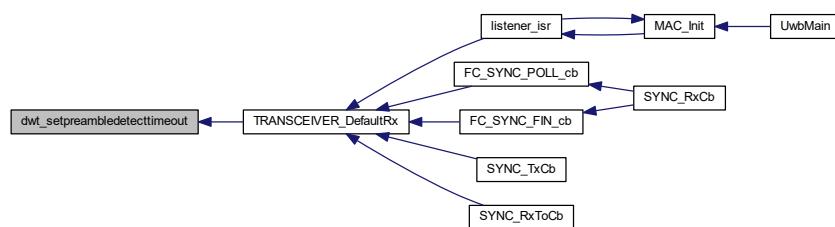
6.12.3.76 dwt_setpreambledetecttimeout()

```
void dwt_setpreambledetecttimeout (
    uint16 timeout )
```

Here is the call graph for this function:



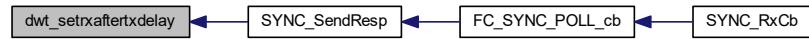
Here is the caller graph for this function:



6.12.3.77 dwt_setrxaftertxdelay()

```
void dwt_setrxaftertxdelay (
    uint32 rxDelayTime )
```

Here is the caller graph for this function:



6.12.3.78 dwt_setrxantennadelay()

```
void dwt_setrxantennadelay (
    uint16 rxDelay )
```

Here is the call graph for this function:



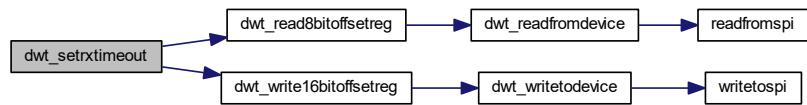
Here is the caller graph for this function:



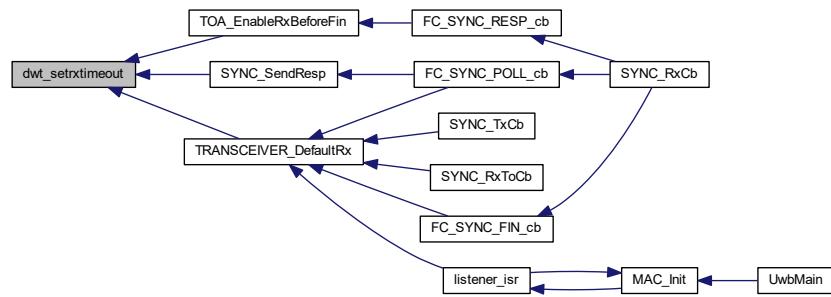
6.12.3.79 dwt_setrxtimeout()

```
void dwt_setrxtimeout (
    uint16 time )
```

Here is the call graph for this function:



Here is the caller graph for this function:



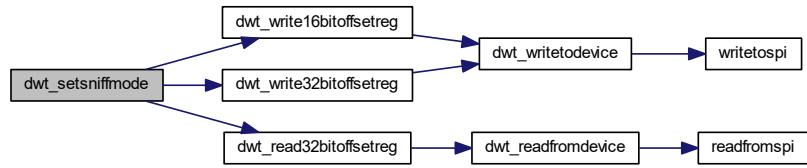
6.12.3.80 dwt_setsmarttxpower()

```
void dwt_setsmarttxpower (
    int enable )
```

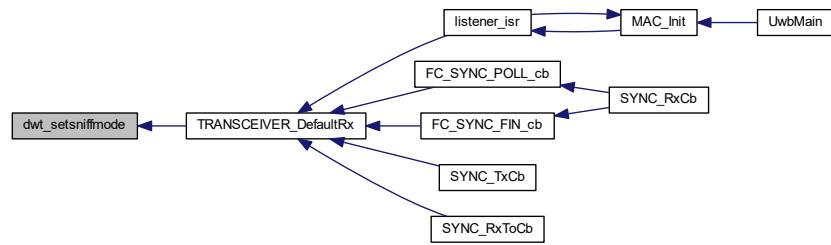
6.12.3.81 dwt_setsniffmode()

```
void dwt_setsniffmode (
    int enable,
    uint8 timeOn,
    uint8 timeOff )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.82 dwt_setsnoozetime()

```
void dwt_setsnoozetime (
    uint8 snooze_time )
```

Here is the call graph for this function:



6.12.3.83 dwt_settxantennadelay()

```
void dwt_settxantennadelay (
    uint16 txDelay )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.84 dwt_setxtaltrim()

```
void dwt_setxtaltrim (
    uint8 value )
```

Here is the call graph for this function:



6.12.3.85 dwt_softreset()

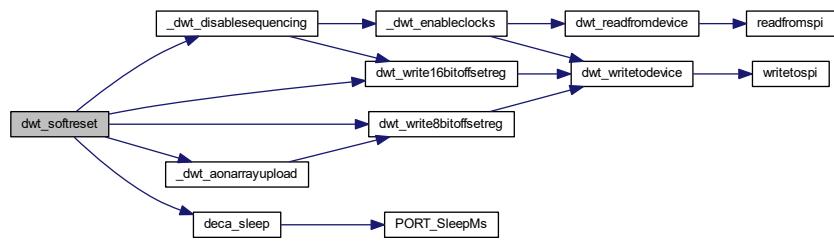
```
dwt_softreset (
    void )
```

this function resets the DW1000

input parameters:

output parameters

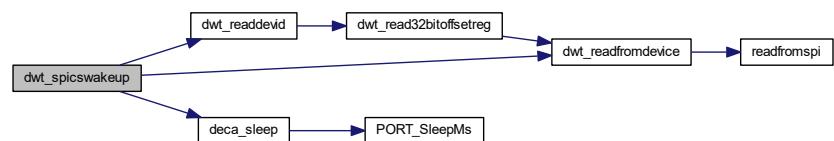
no return value Here is the call graph for this function:



6.12.3.86 dwt_spicswakeup()

```
int dwt_spicswakeup (
    uint8 * buff,
    uint16 length )
```

Here is the call graph for this function:



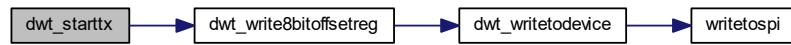
Here is the caller graph for this function:



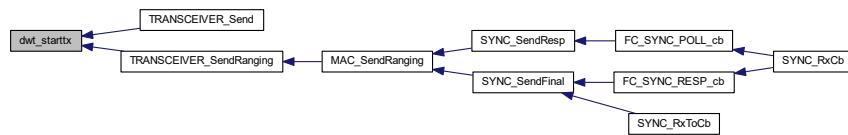
6.12.3.87 dwt_starttx()

```
int dwt_starttx (
    uint8 mode )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.88 dwt_syncrxbufptrs()

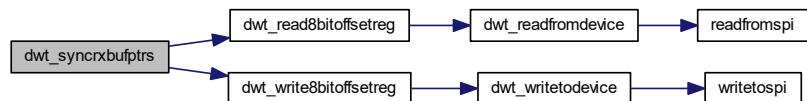
```
dwt_syncrxbufptrs (
    void )
```

this function synchronizes rx buffer pointers need to make sure that the host/IC buffer pointers are aligned before starting RX

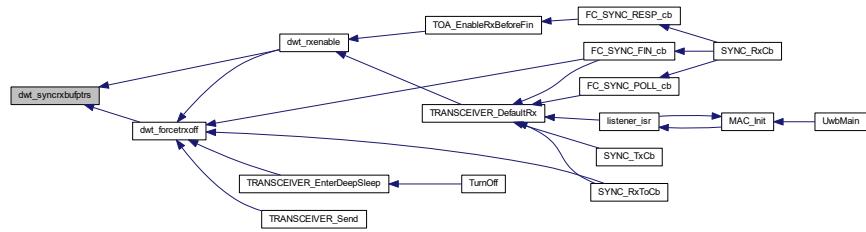
input parameters:

output parameters

no return value Here is the call graph for this function:



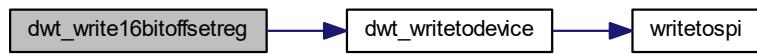
Here is the caller graph for this function:



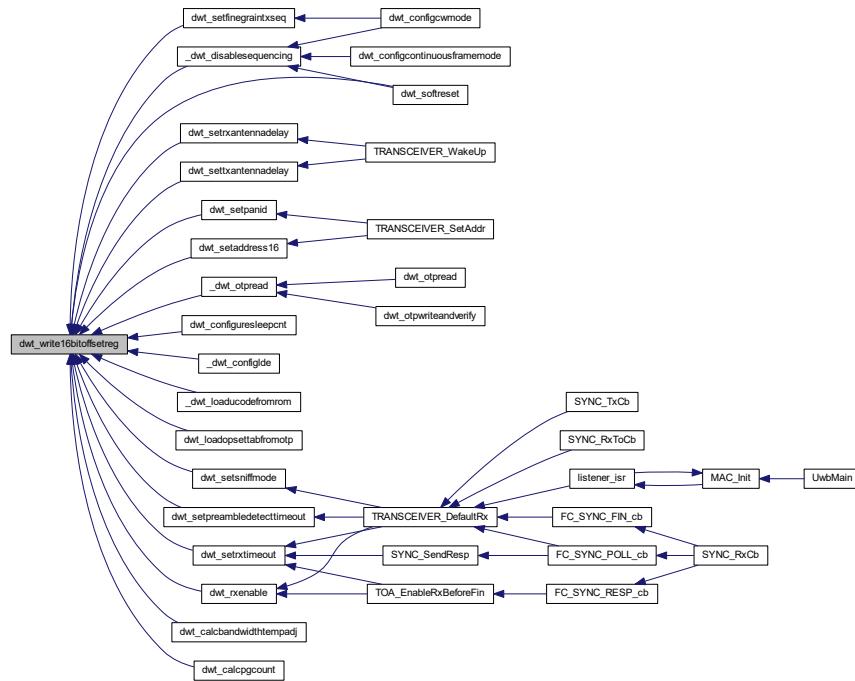
6.12.3.89 dwt_write16bitoffsetreg()

```
void dwt_write16bitoffsetreg (
    int regFileID,
    int regOffset,
    uint16 regval )
```

Here is the call graph for this function:



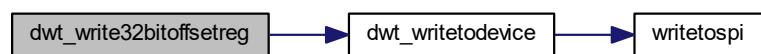
Here is the caller graph for this function:



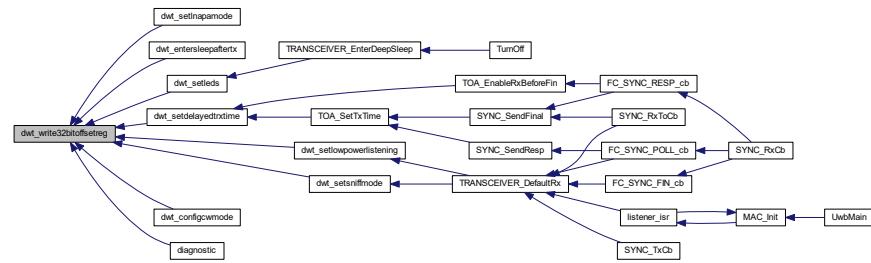
6.12.3.90 `dwt_write32bitoffsetreg()`

```
void dwt_write32bitoffsetreg (
    int regFileID,
    int regOffset,
    uint32 regval )
```

Here is the call graph for this function:



Here is the caller graph for this function:

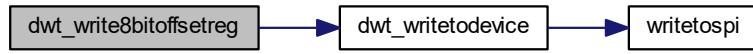


6.12.3.91 dwt_write8bitoffsetreg()

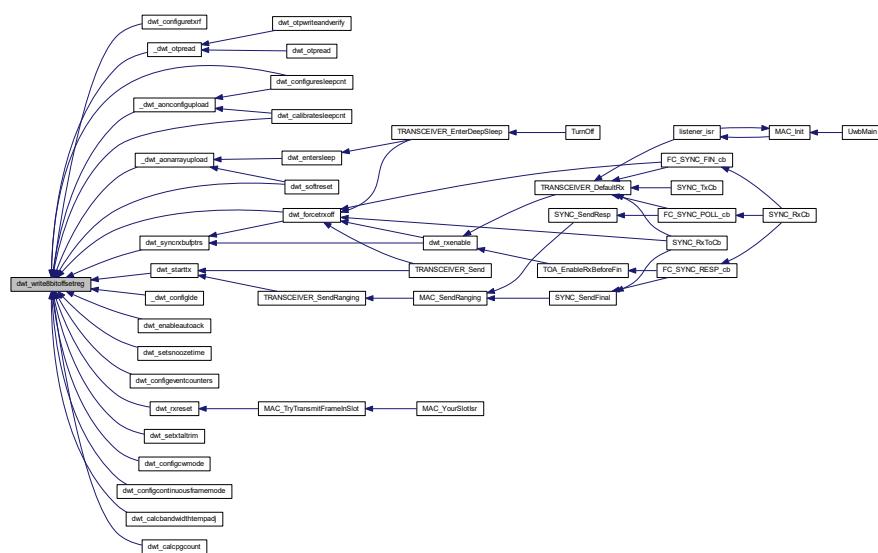
```

void dwt_write8bitoffsetreg (
    int regFileID,
    int regOffset,
    uint8 regval )
  
```

Here is the call graph for this function:



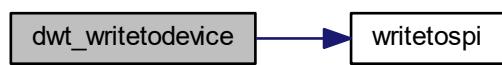
Here is the caller graph for this function:



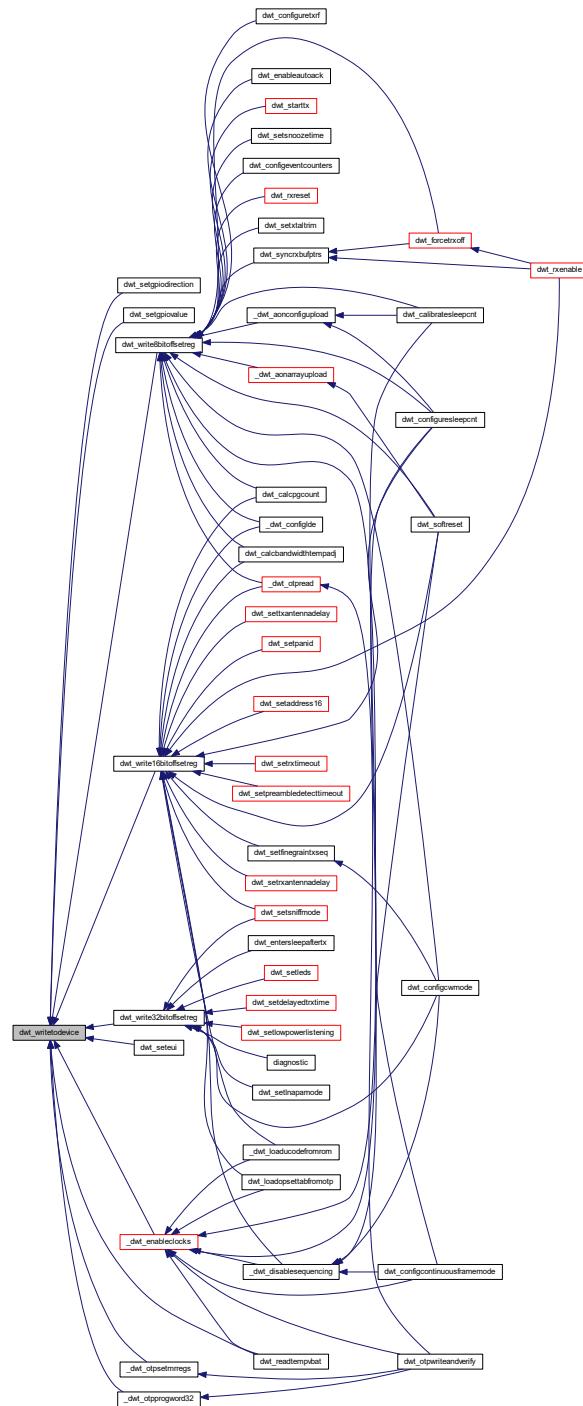
6.12.3.92 dwt_writetodevice()

```
void dwt_writetodevice (
    uint16 recordNumber,
    uint16 index,
    uint32 length,
    const uint8 * buffer )
```

Here is the call graph for this function:



Here is the caller graph for this function:

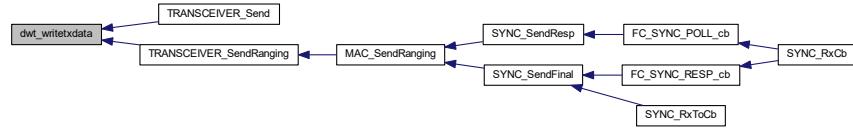


6.12.3.93 dwt_writetxdata()

```
int dwt_writetxdata (
    uint16 txFrameLength,
```

```
uint8 * txFrameBytes,
uint16 txBufferOffset )
```

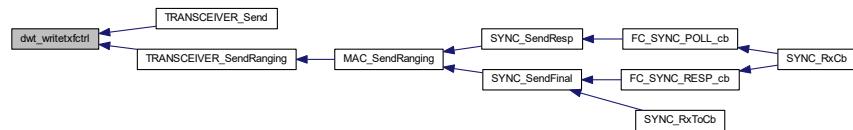
Here is the caller graph for this function:



6.12.3.94 dwt_writetxfctrl()

```
void dwt_writetxfctrl (
    uint16 txFrameLength,
    uint16 txBufferOffset,
    int ranging )
```

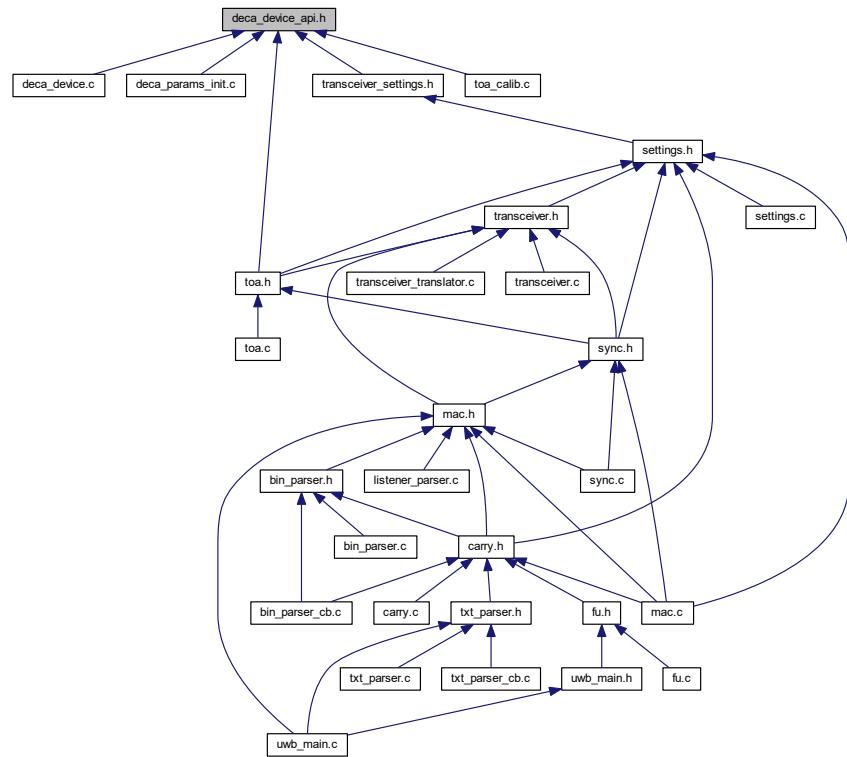
Here is the caller graph for this function:



6.13 deca_device_api.h File Reference

DW1000 API Functions.

This graph shows which files directly or indirectly include this file:



Data Structures

- struct `dwt_cb_data_t`
- struct `dwt_config_t`
- struct `dwt_txconfig_t`
- struct `dwt_rxdiag_t`
- struct `dwt_deviceentcnts_t`

Macros

- `#define _DECA_UINT8_`
- `#define _DECA_UINT16_`
- `#define _DECA_UINT32_`
- `#define _DECA_INT8_`
- `#define _DECA_INT16_`
- `#define _DECA_INT32_`
- `#define DWT_NUM_DW_DEV (1)`
- `#define DWT_SUCCESS (0)`
- `#define DWT_ERROR (-1)`
- `#define DWT_TIME_UNITS (1.0/499.2e6/128.0)`
 $= 15.65e-12 \text{ s}$
- `#define DWT_DEVICE_ID (0xDECA0130)`
DW1000 MP device ID.
- `#define DWT_BR_110K 0`

- #define DWT_BR_850K 1
 - UWB bit rate 850 kbits/s.
- #define DWT_BR_6M8 2
 - UWB bit rate 6.8 Mbits/s.
- #define DWT_PRF_16M 1
 - UWB PRF 16 MHz.
- #define DWT_PRF_64M 2
 - UWB PRF 64 MHz.
- #define DWT_PAC8 0
 - constants for specifying Preamble Acquisition Chunk (PAC) Size in symbols
- #define DWT_PAC16 1
 - PAC 16 (recommended for RX of preamble length 256.
- #define DWT_PAC32 2
 - PAC 32 (recommended for RX of preamble length 512.
- #define DWT_PAC64 3
 - PAC 64 (recommended for RX of preamble length 1024 and up.
- #define DWT_PLEN_4096 0x0C
- #define DWT_PLEN_2048 0x28
- #define DWT_PLEN_1536 0x18
- #define DWT_PLEN_1024 0x08
- #define DWT_PLEN_512 0x34
- #define DWT_PLEN_256 0x24
- #define DWT_PLEN_128 0x14
- #define DWT_PLEN_64 0x04
- #define DWT_SFDTOC_DEF 0x1041
- #define DWT_PHRMODE_STD 0x0
- #define DWT_PHRMODE_EXT 0x3
- #define DWT_START_TX_IMMEDIATE 0
- #define DWT_START_TX_DELAYED 1
- #define DWT_RESPONSE_EXPECTED 2
- #define DWT_START_RX_IMMEDIATE 0
- #define DWT_START_RX_DELAYED 1
- #define DWT_IDLE_ON_DLY_ERR 2
- #define DWT_NO_SYNC_PTRS 4
- #define DWT_LEDS_DISABLE 0x00
- #define DWT_LEDS_ENABLE 0x01
- #define DWT_LEDS_INIT_BLINK 0x02
- #define DWT_FF_NOTYPE_EN 0x000
- #define DWT_FF_COORD_EN 0x002
- #define DWT_FF_BEACON_EN 0x004
- #define DWT_FF_DATA_EN 0x008
- #define DWT_FF_ACK_EN 0x010
- #define DWT_FF_MAC_EN 0x020
- #define DWT_FF_RSVD_EN 0x040
- #define DWT_INT_TFRS 0x000000080
- #define DWT_INT_LDED 0x000000400
- #define DWT_INT_RFCG 0x000004000
- #define DWT_INT_RPHE 0x000001000
- #define DWT_INT_RFCE 0x000008000
- #define DWT_INT_RFSL 0x000010000
- #define DWT_INT_RFTO 0x000020000
- #define DWT_INT_RXOVR 0x00100000

- #define DWT_INT_RXPTO 0x00200000
- #define DWT_INT_SFDT 0x04000000
- #define DWT_INT_ARFE 0x20000000
- #define DWT_PRESRV_SLEEP 0x0100
- #define DWT_LOADOPSET 0x0080
- #define DWT_CONFIG 0x0040
- #define DWT_LOADEUI 0x0008
- #define DWT_RX_EN 0x0002
- #define DWT_TANDV 0x0001
- #define DWT_XTAL_EN 0x10
- #define DWT_WAKE_SLPCNT 0x8
- #define DWT_WAKE_CS 0x4
- #define DWT_WAKE_WK 0x2
- #define DWT_SLP_EN 0x1
- #define DWT_LOADUCODE 0x1
- #define DWT_LOADNONE 0x0
- #define DWT_OPSET_64LEN 0x0
- #define DWT_OPSET_TIGHT 0x1
- #define DWT_OPSET_DEFLT 0x2
- #define DWT_CB_DATA_RX_FLAG RNG 0x1
- #define FREQ_OFFSET_MULTIPLIER (998.4e6/2.0/1024.0/131072.0)
- #define FREQ_OFFSET_MULTIPLIER_110KB (998.4e6/2.0/8192.0/131072.0)
- #define HERTZ_TO_PPM_MULTIPLIER_CHAN_1 (-1.0e6/3494.4e6)
- #define HERTZ_TO_PPM_MULTIPLIER_CHAN_2 (-1.0e6/3993.6e6)
- #define HERTZ_TO_PPM_MULTIPLIER_CHAN_3 (-1.0e6/4492.8e6)
- #define HERTZ_TO_PPM_MULTIPLIER_CHAN_5 (-1.0e6/6489.6e6)
- #define dwt_write32bitreg(x, y) dwt_write32bitoffsetreg(x,0,y)
- #define dwt_read32bitreg(x) dwt_read32bitoffsetreg(x,0)

Typedefs

- typedef unsigned char uint8
- typedef unsigned short uint16
- typedef unsigned long uint32
- typedef signed char int8
- typedef signed short int16
- typedef signed long int32
- typedef void(* dwt_cb_t) (const dwt_cb_data_t *)
- typedef int decalrqStatus_t

Functions

- int dwt_setlocaldataptr (unsigned int index)
 - This is used to return the read part ID of the device.*
- uint32 dwt_getpartid (void)
 - This is used to return the read lot ID of the device.*
- uint32 dwt_readdevid (void)
 - This is used to return the read device type and revision information of the DW1000 device (MP part is 0xDECA0130)*
- uint8 dwt_otprevision (void)
 - This is used to return the read OTP revision.*
- void dwt_setfinegrainTxSeq (int enable)

- void `dwt_setlnapemode` (int lna, int pa)
- void `dwt_setgpiodirection` (uint32 gpioNum, uint32 direction)
- void `dwt_setgpiovalue` (uint32 gpioNum, uint32 value)
- int `dwt_initialise` (uint16 config)
- void `dwt_configure` (dwt_config_t *config)
- void `dwt_configuretxrf` (dwt_txconfig_t *config)
- void `dwt_setrxantennadelay` (uint16 antennaDly)
- void `dwt_settxantennadelay` (uint16 antennaDly)
- void `dwt_setsmarttxpower` (int enable)
- int `dwt_writetxdata` (uint16 txFrameLength, uint8 *txFrameBytes, uint16 txBufferOffset)
- void `dwt_writetxfctrl` (uint16 txFrameLength, uint16 txBufferOffset, int ranging)
- int `dwt_starttx` (uint8 mode)
- void `dwt_setdelayedtrxtime` (uint32 starttime)
- void `dwt_readtxtimestamp` (uint8 *timestamp)
- uint32 `dwt_readtxtimestamphi32` (void)

This is used to read the high 32-bits of the TX timestamp (adjusted with the programmed antenna delay)
- uint32 `dwt_readtxtimestamplo32` (void)

This is used to read the low 32-bits of the TX timestamp (adjusted with the programmed antenna delay)
- void `dwt_readrxtimestamp` (uint8 *timestamp)

This is used to read the high 32-bits of the RX timestamp (adjusted with the programmed antenna delay)
- uint32 `dwt_readrxtimestamplo32` (void)

This is used to read the low 32-bits of the RX timestamp (adjusted with the programmed antenna delay)
- uint32 `dwt_readsystimestamphi32` (void)

This is used to read the high 32-bits of the system time.
- void `dwt_readsystime` (uint8 *timestamp)

This is used to read the low 32-bits of the system time.
- void `dwt_forcetrxoff` (void)

This is used to turn off the transceiver.
- void `dwt_syncrxbufptrs` (void)

this function synchronizes rx buffer pointers need to make sure that the host/IC buffer pointers are aligned before starting RX
- int `dwt_rxenable` (int mode)
- void `dwt_setsniffmode` (int enable, uint8 timeOn, uint8 timeOff)
- void `dwt_setlowpowerlistening` (int enable)
- void `dwt_setsnoozetime` (uint8 snooze_time)
- void `dwt_setdblrxbuffmode` (int enable)
- void `dwt_setrxtimeout` (uint16 time)
- void `dwt_setpreambledetecttimeout` (uint16 timeout)
- uint16 `dwt_calibratesleepcnt` (void)

calibrates the local oscillator as its frequency can vary between 7 and 13kHz depending on temp and voltage
- void `dwt_configuresleepcnt` (uint16 sleepcnt)
- void `dwt_configuresleep` (uint16 mode, uint8 wake)
- void `dwt_entersleep` (void)

This function puts the device into deep sleep or sleep. `dwt_configuresleep()` should be called first to configure the sleep and on-wake/wake-up parameters.
- void `dwt_entersleepaftertx` (int enable)

sets the auto TX to sleep bit. This means that after a frame transmission the device will enter deep sleep mode. The `dwt_configuresleep()` function needs to be called before this to configure the on-wake settings
- int `dwt_spicswakeups` (uint8 *buff, uint16 length)
- void `dwt_setcallbacks` (dwt_cb_t cbTxDone, dwt_cb_t cbRxOk, dwt_cb_t cbRxTo, dwt_cb_t cbRxErr)
- uint8 `dwt_checkirq` (void)

This function checks if the IRQ line is active - this is used instead of interrupt handler.
- void `dwt_isr` (void)

This is the DW1000's general Interrupt Service Routine. It will process/report the following events:

- void `dwt_lowpowerlistenisr` (void)
- void `dwt_setinterrupt` (uint32 bitmask, uint8 enable)
- void `dwt_setpanid` (uint16 panID)
- void `dwt_setaddress16` (uint16 shortAddress)
- void `dwt_seteui` (uint8 *eui64)
- void `dwt_geteui` (uint8 *eui64)
- void `dwt_otpread` (uint32 address, uint32 *array, uint8 length)
- void `dwt_enableframefilter` (uint16 bitmask)
- void `dwt_enableautoack` (uint8 responseDelayTime)
- void `dwt_setrxaftertxdelay` (uint32 rxDelayTime)
- void `dwt_rxreset` (void)

this function resets the receiver of the DW1000
- void `dwt_softreset` (void)

this function resets the DW1000

This is used to read the RX carrier integrator value (relating to the frequency offset of the TX node)

- void `dwt_readdiagnostics` (dwt_rxdiag_t *diagnostics)
- void `dwt_loadopsettabfromotp` (uint8 ops_sel)
- void `dwt_configeventcounters` (int enable)
- void `dwt_readeventcounters` (dwt_deviceentcnts_t *counters)
- int `dwt_otpwriteandverify` (uint32 value, uint16 address)
- void `dwt_setleds` (uint8 mode)
- void `dwt_setxtaltrim` (uint8 value)
- uint8 `dwt_getinitxtaltrim` (void)

This function returns the value of XTAL trim that has been applied during initialisation (dwt_init). This can be either the value read in OTP memory or a default value.

- void `dwt_configcwmode` (uint8 chan)
- void `dwt_configcontinuousframemode` (uint32 framerepetitionrate)
- uint16 `dwt_readtempvbat` (uint8 fastSPI)
- uint8 `dwt_readwakeuptemp` (void)

this function reads the temperature of the DW1000 that was sampled on waking from Sleep/Deepsleep. They are not current values, but read on last wakeup if DWT_TANDV bit is set in mode parameter of dwt_configuresleep

- uint8 `dwt_readwakeupvbat` (void)

this function reads the battery voltage of the DW1000 that was sampled on waking from Sleep/Deepsleep. They are not current values, but read on last wakeup if DWT_TANDV bit is set in mode parameter of dwt_configuresleep

- uint32 `dwt_calcbandwidthtempadj` (uint16 target_count)
- uint32 `dwt_calcpowertempadj` (uint8 channel, uint32 ref_powerreg, double current_temperature, double reference_temperature)
- uint16 `dwt_calcpccount` (uint8 pgdly)
- void `dwt_writetodevice` (uint16 recordNumber, uint16 index, uint32 length, const uint8 *buffer)
- void `dwt_readfromdevice` (uint16 recordNumber, uint16 index, uint32 length, uint8 *buffer)
- uint32 `dwt_read32bitoffsetreg` (int regFileID, int regOffset)
- void `dwt_write32bitoffsetreg` (int regFileID, int regOffset, uint32 regval)
- uint16 `dwt_read16bitoffsetreg` (int regFileID, int regOffset)
- void `dwt_write16bitoffsetreg` (int regFileID, int regOffset, uint16 regval)
- uint8 `dwt_read8bitoffsetreg` (int regFileID, int regOffset)
- void `dwt_write8bitoffsetreg` (int regFileID, int regOffset, uint8 regval)
- int `writetosp1` (uint16 headerLength, const uint8 *headerBuffer, uint32 bodylength, const uint8 *bodyBuffer)
- int `readfromsp1` (uint16 headerLength, const uint8 *headerBuffer, uint32 readlength, uint8 *readBuffer)
- `decalrqStatus_t decamutexon` (void)

This function should disable interrupts. This is called at the start of a critical section It returns the IRQ state before disable, this value is used to re-enable in decamutexoff call.

- void `decamutexoff` (decalrqStatus_t s)
- void `deca_sleep` (unsigned int time_ms)

6.13.1 Detailed Description

DW1000 API Functions.

Attention

Copyright 2013 (c) Decawave Ltd, Dublin, Ireland.

All rights reserved.

6.13.2 Macro Definition Documentation

6.13.2.1 _DECA_INT16_

```
#define _DECA_INT16_
```

6.13.2.2 _DECA_INT32_

```
#define _DECA_INT32_
```

6.13.2.3 _DECA_INT8_

```
#define _DECA_INT8_
```

6.13.2.4 _DECA_UINT16_

```
#define _DECA_UINT16_
```

6.13.2.5 _DECA_UINT32_

```
#define _DECA_UINT32_
```

6.13.2.6 _DECA_UINT8_

```
#define _DECA_UINT8_
```

6.13.2.7 DWT_BR_110K

```
#define DWT_BR_110K 0
```

UWB bit rate 110 kbits/s.

constants for selecting the bit rate for data TX (and RX) These are defined for write (with just a shift) the TX_FCTRL register

6.13.2.8 DWT_BR_6M8

```
#define DWT_BR_6M8 2
```

UWB bit rate 6.8 Mbits/s.

6.13.2.9 DWT_BR_850K

```
#define DWT_BR_850K 1
```

UWB bit rate 850 kbits/s.

6.13.2.10 DWT_CB_DATA_RX_FLAG RNG

```
#define DWT_CB_DATA_RX_FLAG RNG 0x1
```

6.13.2.11 DWT_CONFIG

```
#define DWT_CONFIG 0x0040
```

6.13.2.12 DWT_DEVICE_ID

```
#define DWT_DEVICE_ID (0xDECA0130)
```

DW1000 MP device ID.

6.13.2.13 DWT_ERROR

```
#define DWT_ERROR (-1)
```

6.13.2.14 DWT_FF_ACK_EN

```
#define DWT_FF_ACK_EN 0x010
```

6.13.2.15 DWT_FF_BEACON_EN

```
#define DWT_FF_BEACON_EN 0x004
```

6.13.2.16 DWT_FF_COORD_EN

```
#define DWT_FF_COORD_EN 0x002
```

6.13.2.17 DWT_FF_DATA_EN

```
#define DWT_FF_DATA_EN 0x008
```

6.13.2.18 DWT_FF_MAC_EN

```
#define DWT_FF_MAC_EN 0x020
```

6.13.2.19 DWT_FF_NOTYPE_EN

```
#define DWT_FF_NOTYPE_EN 0x000
```

6.13.2.20 DWT_FF_RSVD_EN

```
#define DWT_FF_RSVD_EN 0x040
```

6.13.2.21 DWT_IDLE_ON_DLY_ERR

```
#define DWT_IDLE_ON_DLY_ERR 2
```

6.13.2.22 DWT_INT_ARFE

```
#define DWT_INT_ARFE 0x20000000
```

6.13.2.23 DWT_INT_LDED

```
#define DWT_INT_LDED 0x00000400
```

6.13.2.24 DWT_INT_RFCE

```
#define DWT_INT_RFCE 0x00008000
```

6.13.2.25 DWT_INT_RFCG

```
#define DWT_INT_RFCG 0x00004000
```

6.13.2.26 DWT_INT_RFSL

```
#define DWT_INT_RFSL 0x00010000
```

6.13.2.27 DWT_INT_RFTO

```
#define DWT_INT_RFTO 0x00020000
```

6.13.2.28 DWT_INT_RPHE

```
#define DWT_INT_RPHE 0x00001000
```

6.13.2.29 DWT_INT_RXOVRR

```
#define DWT_INT_RXOVRR 0x00100000
```

6.13.2.30 DWT_INT_RXPTO

```
#define DWT_INT_RXPTO 0x00200000
```

6.13.2.31 DWT_INT_SFDT

```
#define DWT_INT_SFDT 0x04000000
```

6.13.2.32 DWT_INT_TFRS

```
#define DWT_INT_TFRS 0x00000080
```

6.13.2.33 DWT_LEDS_DISABLE

```
#define DWT_LEDS_DISABLE 0x00
```

6.13.2.34 DWT_LEDS_ENABLE

```
#define DWT_LEDS_ENABLE 0x01
```

6.13.2.35 DWT_LEDS_INIT_BLINK

```
#define DWT_LEDS_INIT_BLINK 0x02
```

6.13.2.36 DWT_LOADEUI

```
#define DWT_LOADEUI 0x0008
```

6.13.2.37 DWT_LOADNONE

```
#define DWT_LOADNONE 0x0
```

6.13.2.38 DWT_LOADOPSET

```
#define DWT_LOADOPSET 0x0080
```

6.13.2.39 DWT_LOADUCODE

```
#define DWT_LOADUCODE 0x1
```

6.13.2.40 DWT_NO_SYNC_PTRS

```
#define DWT_NO_SYNC_PTRS 4
```

6.13.2.41 DWT_NUM_DW_DEV

```
#define DWT_NUM_DW_DEV (1)
```

6.13.2.42 DWT_OPSET_64LEN

```
#define DWT_OPSET_64LEN 0x0
```

6.13.2.43 DWT_OPSET_DEFLT

```
#define DWT_OPSET_DEFLT 0x2
```

6.13.2.44 DWT_OPSET_TIGHT

```
#define DWT_OPSET_TIGHT 0x1
```

6.13.2.45 DWT_PAC16

```
#define DWT_PAC16 1
```

PAC 16 (recommended for RX of preamble length 256.

6.13.2.46 DWT_PAC32

```
#define DWT_PAC32 2
```

PAC 32 (recommended for RX of preamble length 512.

6.13.2.47 DWT_PAC64

```
#define DWT_PAC64 3
```

PAC 64 (recommended for RX of preamble length 1024 and up.

6.13.2.48 DWT_PAC8

```
#define DWT_PAC8 0
```

constants for specifying Preamble Acquisition Chunk (PAC) Size in symbols

PAC 8 (recommended for RX of preamble length 128 and below

6.13.2.49 DWT_PHRMODE_EXT

```
#define DWT_PHRMODE_EXT 0x3
```

6.13.2.50 DWT_PHRMODE_STD

```
#define DWT_PHRMODE_STD 0x0
```

6.13.2.51 DWT_PLEN_1024

```
#define DWT_PLEN_1024 0x08
```

6.13.2.52 DWT_PLEN_128

```
#define DWT_PLEN_128 0x14
```

6.13.2.53 DWT_PLEN_1536

```
#define DWT_PLEN_1536 0x18
```

6.13.2.54 DWT_PLEN_2048

```
#define DWT_PLEN_2048 0x28
```

6.13.2.55 DWT_PLEN_256

```
#define DWT_PLEN_256 0x24
```

6.13.2.56 DWT_PLEN_4096

```
#define DWT_PLEN_4096 0x0C
```

constants for specifying TX Preamble length in symbols These are defined to allow them be directly written into byte 2 of the TX_FCTRL register (i.e. a four bit value destined for bits 20..18 but shifted left by 2 for byte alignment)

6.13.2.57 DWT_PLEN_512

```
#define DWT_PLEN_512 0x34
```

6.13.2.58 DWT_PLEN_64

```
#define DWT_PLEN_64 0x04
```

6.13.2.59 DWT_PRESRV_SLEEP

```
#define DWT_PRESRV_SLEEP 0x0100
```

6.13.2.60 DWT_PRF_16M

```
#define DWT_PRF_16M 1
```

UWB PRF 16 MHz.

constants for specifying the (Nominal) mean Pulse Repetition Frequency These are defined for direct write (with a shift if necessary) to CHAN_CTRL and TX_FCTRL regs

6.13.2.61 DWT_PRF_64M

```
#define DWT_PRF_64M 2
```

UWB PRF 64 MHz.

6.13.2.62 dwt_read32bitreg

```
#define dwt_read32bitreg( x ) dwt_read32bitoffsetreg(x, 0)
```

6.13.2.63 DWT_RESPONSE_EXPECTED

```
#define DWT_RESPONSE_EXPECTED 2
```

6.13.2.64 DWT_RX_EN

```
#define DWT_RX_EN 0x0002
```

6.13.2.65 DWT_SFDTOC_DEF

```
#define DWT_SFDTOC_DEF 0x1041
```

6.13.2.66 DWT_SLP_EN

```
#define DWT_SLP_EN 0x1
```

6.13.2.67 DWT_START_RX_DELAYED

```
#define DWT_START_RX_DELAYED 1
```

6.13.2.68 DWT_START_RX_IMMEDIATE

```
#define DWT_START_RX_IMMEDIATE 0
```

6.13.2.69 DWT_START_TX_DELAYED

```
#define DWT_START_TX_DELAYED 1
```

6.13.2.70 DWT_START_TX_IMMEDIATE

```
#define DWT_START_TX_IMMEDIATE 0
```

6.13.2.71 DWT_SUCCESS

```
#define DWT_SUCCESS (0)
```

6.13.2.72 DWT_TANDV

```
#define DWT_TANDV 0x0001
```

6.13.2.73 DWT_TIME_UNITS

```
#define DWT_TIME_UNITS (1.0/499.2e6/128.0)  
= 15.65e-12 s
```

6.13.2.74 DWT_WAKE_CS

```
#define DWT_WAKE_CS 0x4
```

6.13.2.75 DWT_WAKE_SLPCNT

```
#define DWT_WAKE_SLPCNT 0x8
```

6.13.2.76 DWT_WAKE_WK

```
#define DWT_WAKE_WK 0x2
```

6.13.2.77 dwt_write32bitreg

```
#define dwt_write32bitreg(  
    x,  
    y ) dwt_write32bitoffsetreg(x,0,y)
```

6.13.2.78 DWT_XTAL_EN

```
#define DWT_XTAL_EN 0x10
```

6.13.2.79 FREQ_OFFSET_MULTIPLIER

```
#define FREQ_OFFSET_MULTIPLIER (998.4e6/2.0/1024.0/131072.0)
```

6.13.2.80 FREQ_OFFSET_MULTIPLIER_110KB

```
#define FREQ_OFFSET_MULTIPLIER_110KB (998.4e6/2.0/8192.0/131072.0)
```

6.13.2.81 HERTZ_TO_PPM_MULTIPLIER_CHAN_1

```
#define HERTZ_TO_PPM_MULTIPLIER_CHAN_1 (-1.0e6/3494.4e6)
```

6.13.2.82 HERTZ_TO_PPM_MULTIPLIER_CHAN_2

```
#define HERTZ_TO_PPM_MULTIPLIER_CHAN_2 (-1.0e6/3993.6e6)
```

6.13.2.83 HERTZ_TO_PPM_MULTIPLIER_CHAN_3

```
#define HERTZ_TO_PPM_MULTIPLIER_CHAN_3 (-1.0e6/4492.8e6)
```

6.13.2.84 HERTZ_TO_PPM_MULTIPLIER_CHAN_5

```
#define HERTZ_TO_PPM_MULTIPLIER_CHAN_5 (-1.0e6/6489.6e6)
```

6.13.3 Typedef Documentation**6.13.3.1 decaIrqStatus_t**

```
typedef int decaIrqStatus_t
```

6.13.3.2 dwt_cb_t

```
typedef void(* dwt_cb_t) (const dwt_cb_data_t *)
```

6.13.3.3 int16

```
typedef signed short int16
```

6.13.3.4 int32

```
typedef signed long int32
```

6.13.3.5 int8

```
typedef signed char int8
```

6.13.3.6 uint16

```
typedef unsigned short uint16
```

6.13.3.7 uint32

```
typedef unsigned long uint32
```

6.13.3.8 uint8

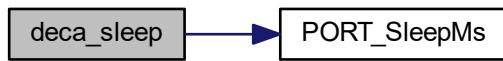
```
typedef unsigned char uint8
```

6.13.4 Function Documentation

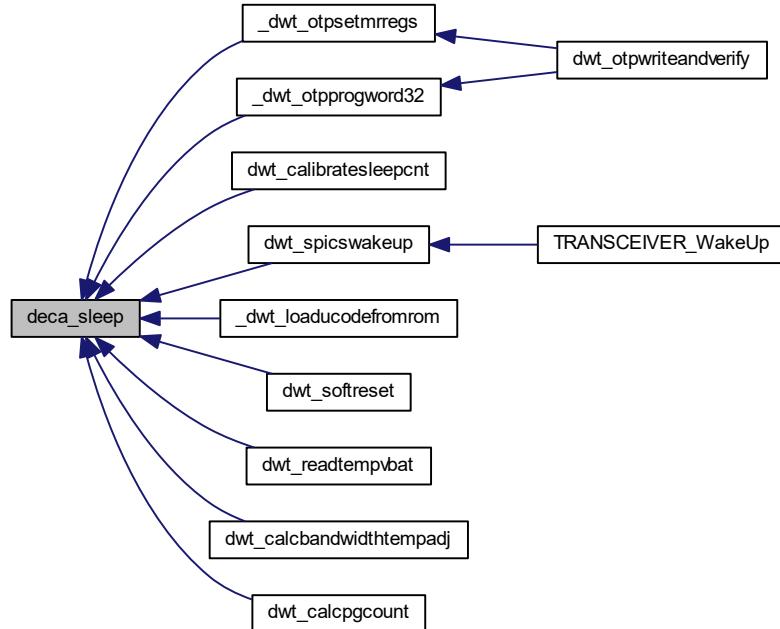
6.13.4.1 deca_sleep()

```
void deca_sleep (
    unsigned int time_ms )
```

Here is the call graph for this function:



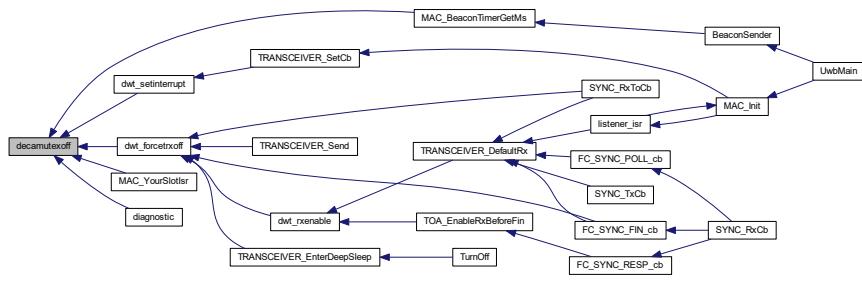
Here is the caller graph for this function:



6.13.4.2 decamutexoff()

```
void decamutexoff (
    decaIrqStatus_t s )
```

Here is the caller graph for this function:



6.13.4.3 decamutexon()

```
decamutexon (
    void )
```

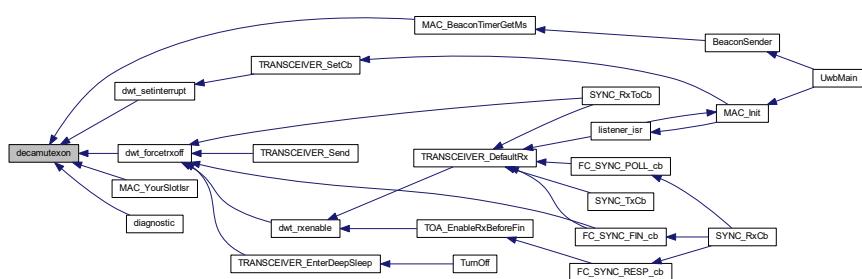
This function should disable interrupts. This is called at the start of a critical section It returns the IRQ state before disable, this value is used to re-enable in decamutexoff call.

Note: The body of this function is defined in deca_mutex.c and is platform specific

input parameters:

output parameters

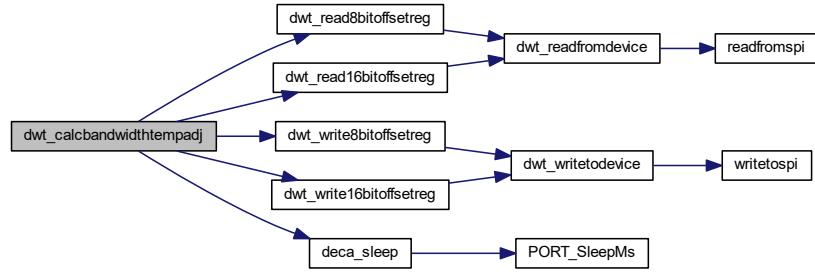
returns the state of the DW1000 interrupt Here is the caller graph for this function:



6.13.4.4 dwt_calcbandwidthtempadj()

```
uint32 dwt_calcbandwidthtempadj (
    uint16 target_count )
```

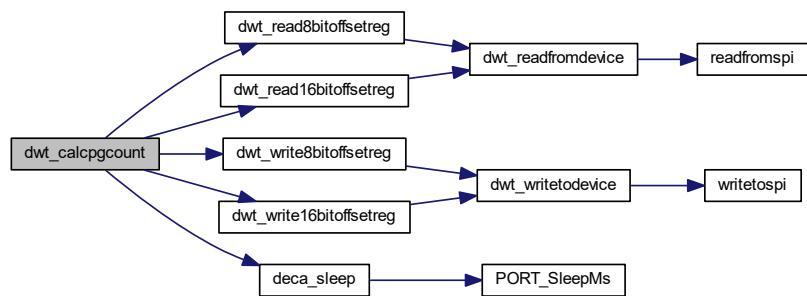
Here is the call graph for this function:



6.13.4.5 dwt_calcpgcount()

```
uint16 dwt_calcpgcount (
    uint8 pgdly )
```

Here is the call graph for this function:



6.13.4.6 dwt_calcpowertempadj()

```
uint32 dwt_calcpowertempadj (
    uint8 channel,
    uint32 ref_powerreg,
    double current_temperature,
    double reference_temperature )
```

Here is the call graph for this function:



6.13.4.7 dwt_calibratesleepcnt()

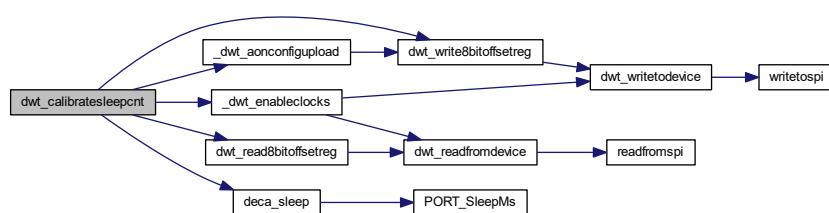
```
uint16 dwt_calibratesleepcnt (
    void )
```

calibrates the local oscillator as its frequency can vary between 7 and 13kHz depending on temp and voltage

NOTE: this function needs to be run before dwt_configuresleepcnt, so that we know what the counter units are
input parameters

output parameters

returns the number of XTAL/2 cycles per low-power oscillator cycle. LP OSC frequency = 19.2 MHz/return value
Here is the call graph for this function:



6.13.4.8 dwt_checkirq()

```
uint8 dwt_checkirq (
    void )
```

This function checks if the IRQ line is active - this is used instead of interrupt handler.

input parameters

output parameters

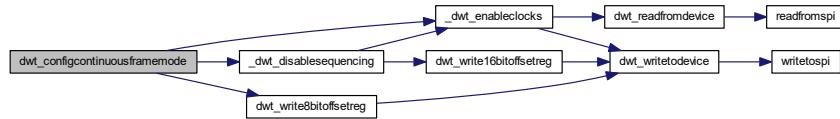
return value is 1 if the IRQS bit is set and 0 otherwise Here is the call graph for this function:



6.13.4.9 dwt_configcontinuousframemode()

```
void dwt_configcontinuousframemode (
    uint32 framerepetitionrate )
```

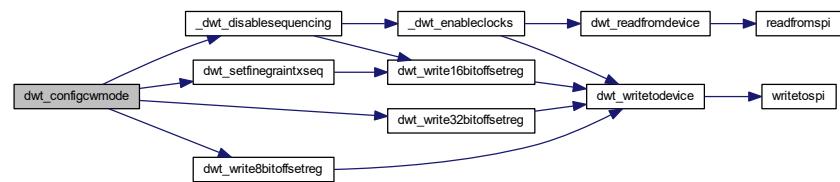
Here is the call graph for this function:



6.13.4.10 dwt_configcwmode()

```
void dwt_configcwmode (
    uint8 chan )
```

Here is the call graph for this function:



6.13.4.11 dwt_configeventcounters()

```
void dwt_configeventcounters (
    int enable )
```

Here is the call graph for this function:



6.13.4.12 dwt_configure()

```
void dwt_configure (
    dwt_config_t * config )
```

6.13.4.13 dwt_configuresleep()

```
void dwt_configuresleep (
    uint16 mode,
    uint8 wake )
```

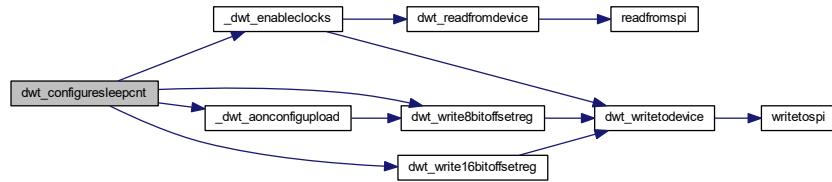
Here is the caller graph for this function:



6.13.4.14 dwt_configuresleepcnt()

```
void dwt_configuresleepcnt (
    uint16 sleepcnt )
```

Here is the call graph for this function:



6.13.4.15 dwt_configuretxrf()

```
void dwt_configuretxrf (
    dwt_txconfig_t * config )
```

Here is the call graph for this function:



6.13.4.16 dwt_enableautoack()

```
void dwt_enableautoack (
    uint8 responseDelayTime )
```

Here is the call graph for this function:



6.13.4.17 dwt_enableframefilter()

```
void dwt_enableframefilter (
    uint16 bitmask )
```

6.13.4.18 dwt_entersleep()

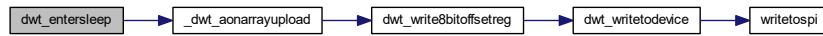
```
void dwt_entersleep (
    void )
```

This function puts the device into deep sleep or sleep. [dwt_configuresleep\(\)](#) should be called first to configure the sleep and on-wake/wake-up parameters.

input parameters

output parameters

no return value Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.19 dwt_entersleepaftertx()

```
void dwt_entersleepaftertx (
    int enable )
```

sets the auto TX to sleep bit. This means that after a frame transmission the device will enter deep sleep mode. The [dwt_configuresleep\(\)](#) function needs to be called before this to configure the on-wake settings

NOTE: the IRQ line has to be low/inactive (i.e. no pending events)

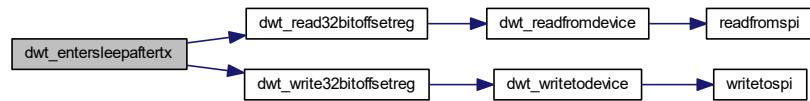
input parameters

Parameters

<code>enable</code>	- 1 to configure the device to enter deep sleep after TX, 0 - disables the configuration
---------------------	--

output parameters

no return value Here is the call graph for this function:

6.13.4.20 `dwt_forcetrxoff()`

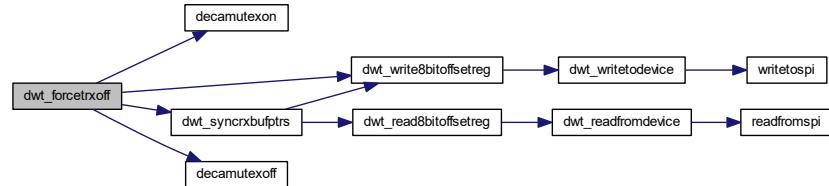
```
void dwt_forcetrxoff (
    void )
```

This is used to turn off the transceiver.

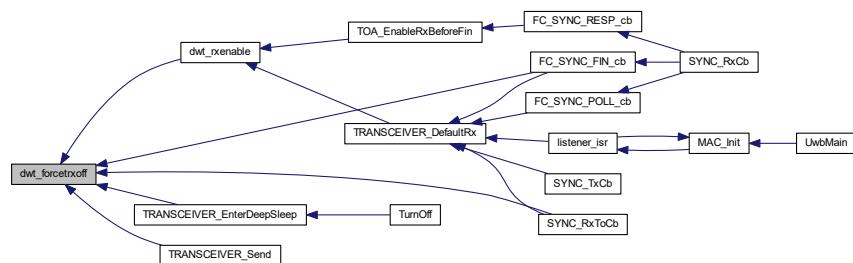
input parameters

output parameters

no return value Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.21 dwt_geteui()

```
void dwt_geteui (
    uint8 * eui64 )
```

Here is the call graph for this function:



6.13.4.22 dwt_getinitxtaltrim()

```
uint8 dwt_getinitxtaltrim (
    void )
```

This function returns the value of XTAL trim that has been applied during initialisation (dwt_init). This can be either the value read in OTP memory or a default value.

NOTE: The value returned by this function is the initial value only! It is not updated on dwt_setxtaltrim calls.

input parameters

output parameters

returns the XTAL trim value set upon initialisation

6.13.4.23 dwt_getlotid()

```
uint32 dwt_getlotid (
    void )
```

This is used to return the read lot ID of the device.

NOTE: [dwt_initialise\(\)](#) must be called prior to this function so that it can return a relevant value.

input parameters

output parameters

returns the 32 bit lot ID value as programmed in the factory

6.13.4.24 dwt_getpartid()

```
uint32 dwt_getpartid (
    void )
```

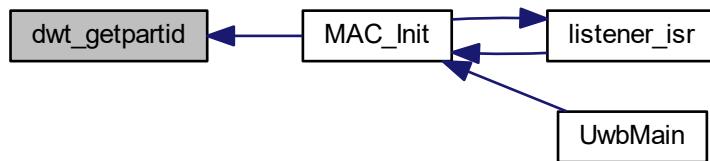
This is used to return the read part ID of the device.

NOTE: [dwt_initialise\(\)](#) must be called prior to this function so that it can return a relevant value.

input parameters

output parameters

returns the 32 bit part ID value as programmed in the factory Here is the caller graph for this function:



6.13.4.25 dwt_initialise()

```
int dwt_initialise (
    uint16 config )
```

6.13.4.26 dwt_isr()

```
void dwt_isr (
    void )
```

This is the DW1000's general Interrupt Service Routine. It will process/report the following events:

- RXFCG (through cbRxOk callback)
- TXFRS (through cbTxDone callback)
- RXRFTO/RXPTO (through cbRxTo callback)
- RXPHE/RXFCE/RXRFSL/RXSFDTO/AFFREJ/LDEERR (through cbRxTo cbRxErr)

For all events, corresponding interrupts are cleared and necessary resets are performed. In addition, in the received frame information and frame control are read before calling the callback. If double buffering is active, the ISR will also toggle between reception buffers once the reception callback processing has ended.

/ This version of the ISR supports double buffering but does not support automatic RX re-enabling! */*

NOTE: In PC based system using (Cheetah or ARM) USB to SPI converter there can be no interrupts, however we still need something to take the place of it and operate in a polled way. In an embedded system this function should be configured to be triggered on any of the interrupts described above.

input parameters

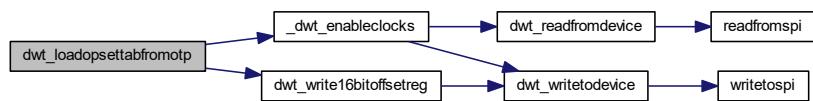
output parameters

no return value

6.13.4.27 dwt_loadopsettabfromotp()

```
void dwt_loadopsettabfromotp (
    uint8 ops_sel )
```

Here is the call graph for this function:



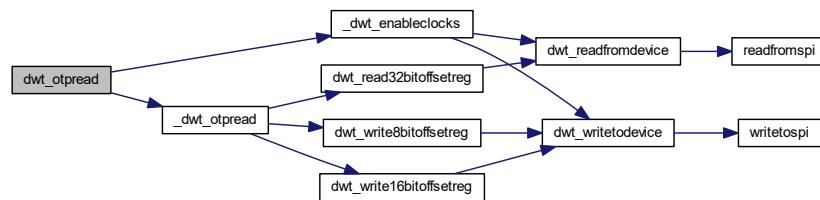
6.13.4.28 dwt_lowpowerlistenISR()

```
void dwt_lowpowerlistenISR (
    void )
```

6.13.4.29 dwt_otpread()

```
void dwt_otpread (
    uint32 address,
    uint32 * array,
    uint8 length )
```

Here is the call graph for this function:



6.13.4.30 dwt_otprevision()

```
uint8 dwt_otprevision (
    void )
```

This is used to return the read OTP revision.

NOTE: [dwt_initialise\(\)](#) must be called prior to this function so that it can return a relevant value.

input parameters

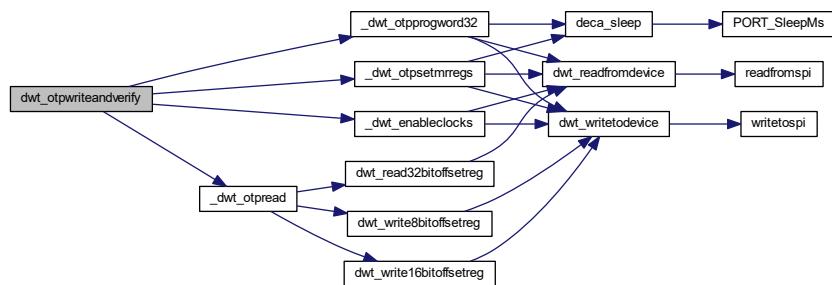
output parameters

returns the read OTP revision value

6.13.4.31 dwt_otpwriteandverify()

```
int dwt_otpwriteandverify (
    uint32 value,
    uint16 address )
```

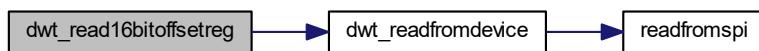
!!!!!!!!!!!!!! NOTE !!!!!!!!!!!!!!! Here is the call graph for this function:



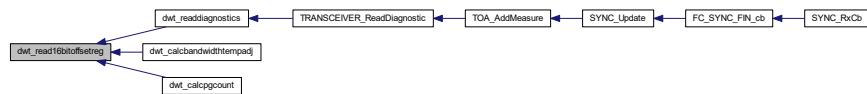
6.13.4.32 dwt_read16bitoffsetreg()

```
uint16 dwt_read16bitoffsetreg (
    int regFileID,
    int regOffset )
```

Here is the call graph for this function:



Here is the caller graph for this function:

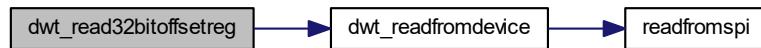


6.13.4.33 dwt_read32bitoffsetreg()

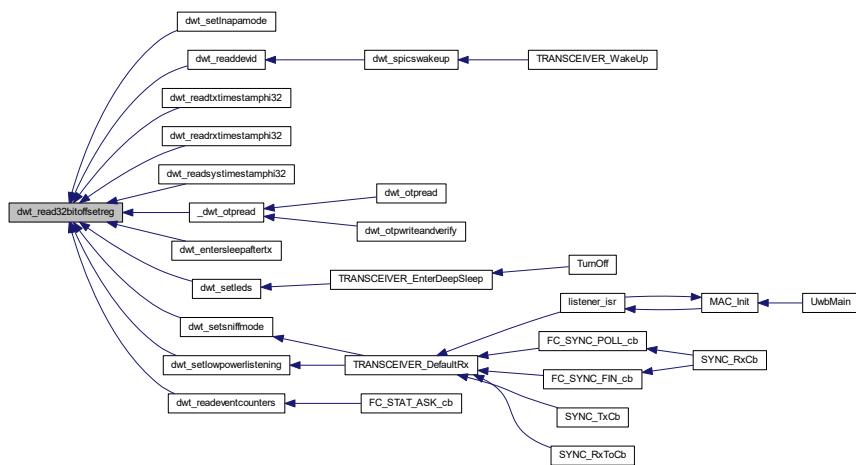
```
uint32 dwt_read32bitoffsetreg (
```

int regFileID,
int regOffset)

Here is the call graph for this function:



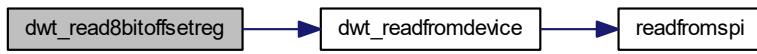
Here is the caller graph for this function:



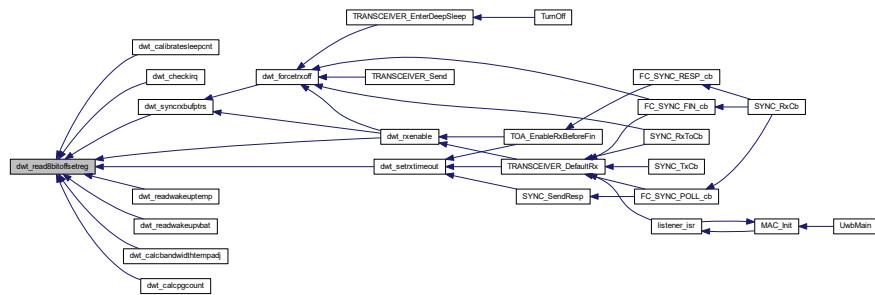
6.13.4.34 dwt_read8bitoffsetreg()

```
uint8 dwt_read8bitoffsetreg (
    int regFileID,
    int regOffset )
```

Here is the call graph for this function:



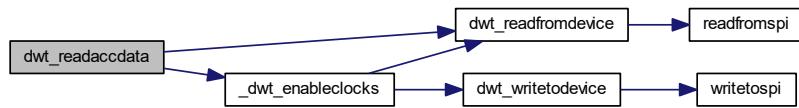
Here is the caller graph for this function:



6.13.4.35 dwt_readaccdata()

```
void dwt_readaccdata (
    uint8 * buffer,
    uint16 length,
    uint16 rxBufferOffset )
```

Here is the call graph for this function:



6.13.4.36 dwt_readcarrierintegrator()

```
int32 dwt_readcarrierintegrator (
    void )
```

This is used to read the RX carrier integrator value (relating to the frequency offset of the TX node)

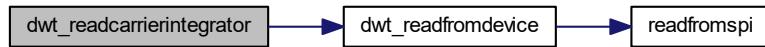
NOTE: This is a 21-bit signed quantity, the function sign extends the most significant bit, which is bit #20 (numbering from bit zero) to return a 32-bit signed integer value.

input parameters - NONE

return value - the (int32) signed carrier integrator value. A positive value means the local RX clock is running faster than the remote TX device.

input parameters - NONE

return value - the (int32) signed carrier integrator value. A positive value means the local RX clock is running faster than the remote TX device. Here is the call graph for this function:



6.13.4.37 dwt_readdevid()

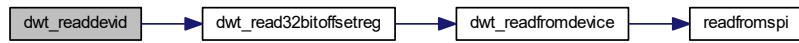
```
uint32 dwt_readdevid (
    void )
```

This is used to return the read device type and revision information of the DW1000 device (MP part is 0xDECA0130)

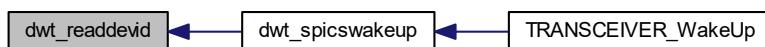
input parameters

output parameters

returns the read value which for DW1000 is 0xDECA0130 Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.38 dwt_readdiagnostics()

```
void dwt_readdiagnostics (
    dwt_rxdiag_t * diagnostics )
```

Here is the call graph for this function:



Here is the caller graph for this function:



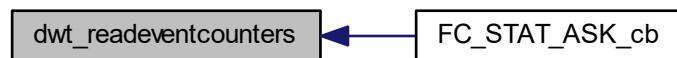
6.13.4.39 dwt_readeventcounters()

```
void dwt_readeventcounters (
    dwt_deviceeventcnts_t * counters )
```

Here is the call graph for this function:



Here is the caller graph for this function:



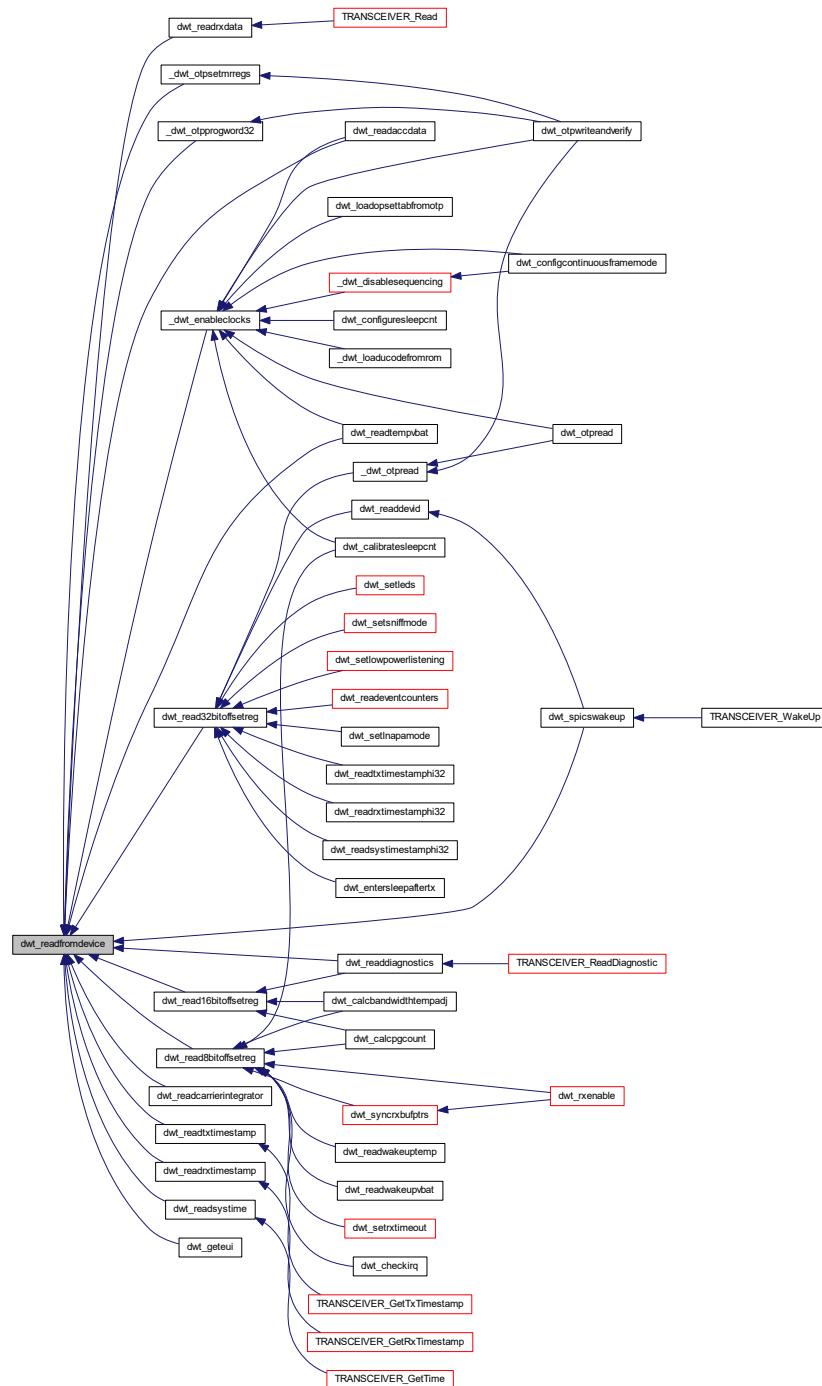
6.13.4.40 dwt_readfromdevice()

```
void dwt_readfromdevice (
    uint16 recordNumber,
    uint16 index,
    uint32 length,
    uint8 * buffer )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.41 dwt_readrxdata()

```
void dwt_readrxdata (
    uint8 * buffer,
```

```

    uint16 length,
    uint16 rxBufferOffset )

```

Here is the call graph for this function:



Here is the caller graph for this function:



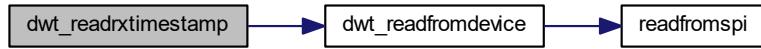
6.13.4.42 dwt_readrxtimestamp()

```

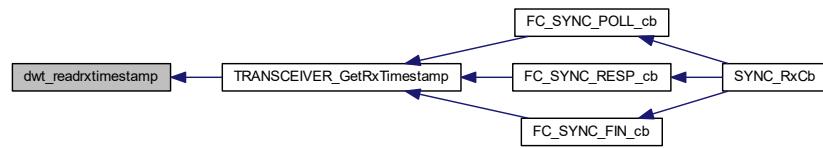
void dwt_readrxtimestamp (
    uint8 * timestamp )

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.43 dwt_readrxtimestamphi32()

```
uint32 dwt_readrxtimestamphi32 (
    void )
```

This is used to read the high 32-bits of the RX timestamp (adjusted with the programmed antenna delay)

input parameters

output parameters

returns high 32-bits of RX timestamp Here is the call graph for this function:



6.13.4.44 dwt_readrxtimestampl032()

```
uint32 dwt_readrxtimestampl032 (
    void )
```

This is used to read the low 32-bits of the RX timestamp (adjusted with the programmed antenna delay)

input parameters

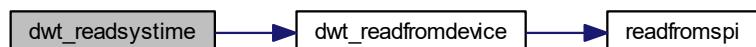
output parameters

returns low 32-bits of RX timestamp

6.13.4.45 dwt_readsystime()

```
void dwt_readsystime (
    uint8 * timestamp )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.46 `dwt_readsystimestamphi32()`

```
uint32 dwt_readsystimestamphi32 (
    void )
```

This is used to read the high 32-bits of the system time.

input parameters

output parameters

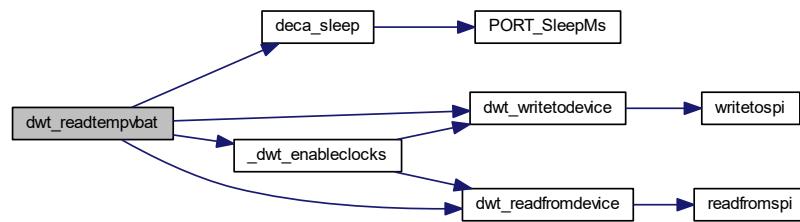
returns high 32-bits of system time timestamp Here is the call graph for this function:



6.13.4.47 `dwt_readtempvbat()`

```
uint16 dwt_readtempvbat (
    uint8 fastSPI )
```

Here is the call graph for this function:



6.13.4.48 dwt_readtxtimestamp()

```
void dwt_readtxtimestamp (
    uint8 * timestamp )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.49 dwt_readtxtimestamphi32()

```
uint32 dwt_readtxtimestamphi32 (
    void )
```

This is used to read the high 32-bits of the TX timestamp (adjusted with the programmed antenna delay)

input parameters

output parameters

returns high 32-bits of TX timestamp Here is the call graph for this function:



6.13.4.50 dwt_readtxtimestamplo32()

```
uint32 dwt_readtxtimestamplo32 (
    void )
```

This is used to read the low 32-bits of the TX timestamp (adjusted with the programmed antenna delay)

input parameters

output parameters

returns low 32-bits of TX timestamp

6.13.4.51 dwt_readwakeuptemp()

```
uint8 dwt_readwakeuptemp (
    void )
```

this function reads the temperature of the DW1000 that was sampled on waking from Sleep/Deepsleep. They are not current values, but read on last wakeup if DWT_TANDV bit is set in mode parameter of dwt_configuresleep

input parameters:

output parameters:

returns: 8-bit raw temperature sensor value Here is the call graph for this function:



6.13.4.52 dwt_readwakeupvbat()

```
uint8 dwt_readwakeupvbat (
    void )
```

this function reads the battery voltage of the DW1000 that was sampled on waking from Sleep/Deepsleep. They are not current values, but read on last wakeup if DWT_TANDV bit is set in mode parameter of dwt_configuresleep

input parameters:

output parameters:

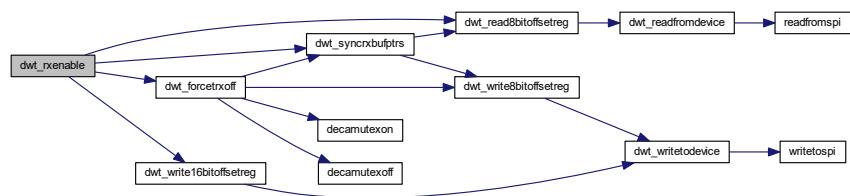
returns: 8-bit raw battery voltage sensor value Here is the call graph for this function:



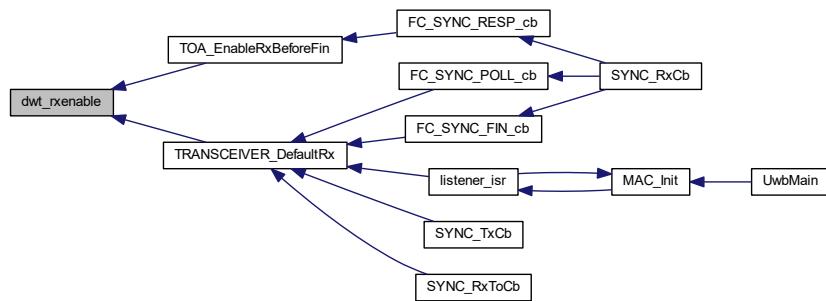
6.13.4.53 dwt_rxenable()

```
int dwt_rxenable (
    int mode )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.54 dwt_rxreset()

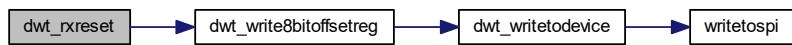
```
void dwt_rxreset (
    void )
```

this function resets the receiver of the DW1000

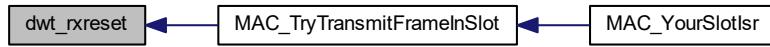
input parameters:

output parameters

no return value Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.55 dwt_setaddress16()

```
void dwt_setaddress16 (
    uint16 shortAddress )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.56 dwt_setcallbacks()

```
void dwt_setcallbacks (
    dwt_cb_t cbTxDone,
    dwt_cb_t cbRxOk,
    dwt_cb_t cbRxTo,
    dwt_cb_t cbRxErr )
```

Here is the caller graph for this function:



6.13.4.57 dwt_setdblrxbuffmode()

```
void dwt_setdblrxbuffmode (
    int enable )
```

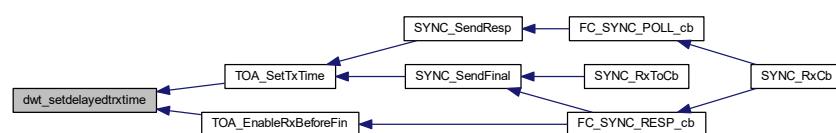
6.13.4.58 dwt_setdelayedtrxtime()

```
void dwt_setdelayedtrxtime (
    uint32 starttime )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.59 dwt_seteui()

```
void dwt_seteui (
    uint8 * eui64 )
```

Here is the call graph for this function:



6.13.4.60 dwt_setfinegraintxseq()

```
void dwt_setfinegraintxseq (
    int enable )
```

Here is the call graph for this function:



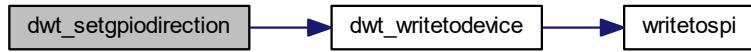
Here is the caller graph for this function:



6.13.4.61 dwt_setgpiodirection()

```
void dwt_setgpiodirection (
    uint32 gpioNum,
    uint32 direction )
```

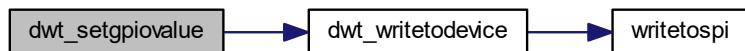
Here is the call graph for this function:



6.13.4.62 dwt_setgpiovalue()

```
void dwt_setgpiovalue (
    uint32 gpioNum,
    uint32 value )
```

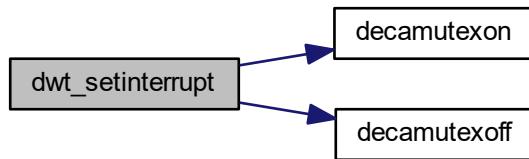
Here is the call graph for this function:



6.13.4.63 dwt_setinterrupt()

```
void dwt_setinterrupt (
    uint32 bitmask,
    uint8 enable )
```

Here is the call graph for this function:



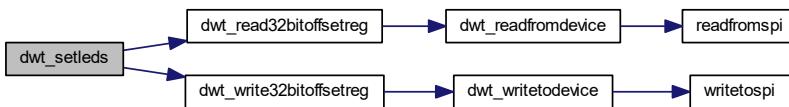
Here is the caller graph for this function:



6.13.4.64 dwt_setleds()

```
void dwt_setleds (
    uint8 mode )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.65 dwt_setlnapemode()

```
void dwt_setlnapemode (
    int lna,
    int pa )
```

Here is the call graph for this function:



6.13.4.66 dwt_setlocaldataptr()

```
int dwt_setlocaldataptr (
    unsigned int index )
```

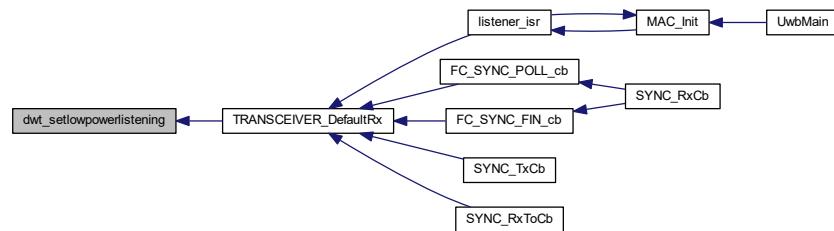
6.13.4.67 dwt_setlowpowerlistening()

```
void dwt_setlowpowerlistening (
    int enable )
```

Here is the call graph for this function:



Here is the caller graph for this function:



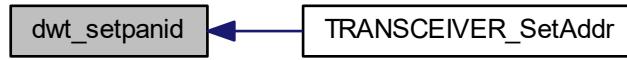
6.13.4.68 dwt_setpanid()

```
void dwt_setpanid (
    uint16 panID )
```

Here is the call graph for this function:



Here is the caller graph for this function:



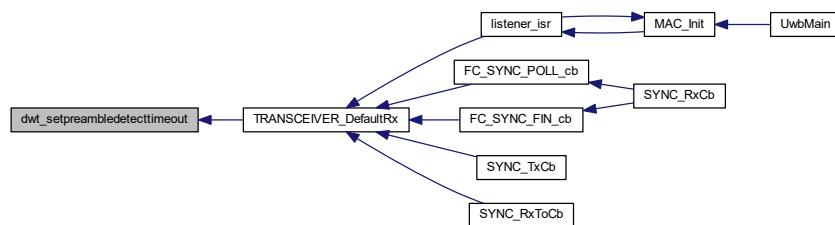
6.13.4.69 dwt_setpreambledetecttimeout()

```
void dwt_setpreambledetecttimeout (
    uint16 timeout )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.70 dwt_setrxaftertxdelay()

```
void dwt_setrxaftertxdelay (
    uint32 rxDelayTime )
```

Here is the caller graph for this function:



6.13.4.71 dwt_setrxantennadelay()

```
void dwt_setrxantennadelay (
    uint16 antennaDly )
```

Here is the call graph for this function:



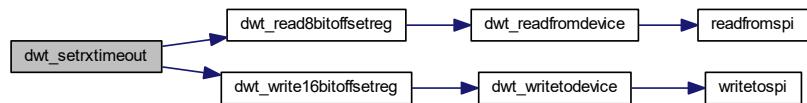
Here is the caller graph for this function:



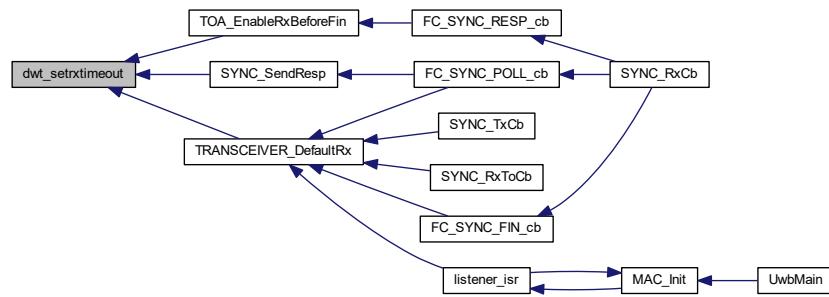
6.13.4.72 dwt_setrxtimeout()

```
void dwt_setrxtimeout (
    uint16 time )
```

Here is the call graph for this function:



Here is the caller graph for this function:



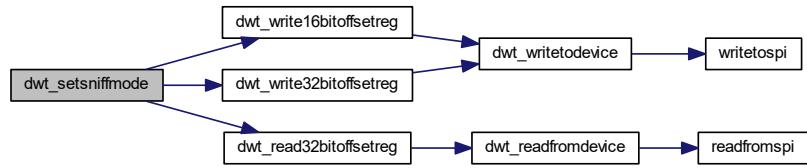
6.13.4.73 dwt_setsmarttxpower()

```
void dwt_setsmarttxpower (
    int enable )
```

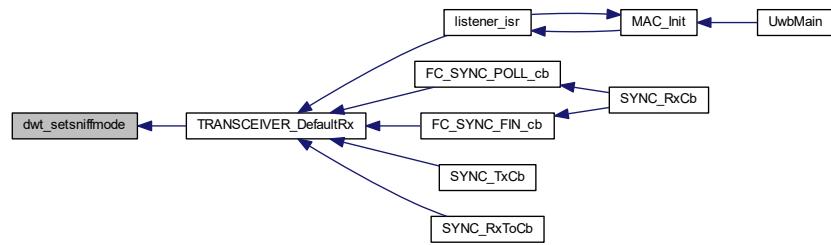
6.13.4.74 dwt_setsniffmode()

```
void dwt_setsniffmode (
    int enable,
    uint8 timeOn,
    uint8 timeOff )
```

Here is the call graph for this function:



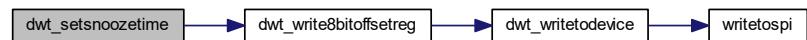
Here is the caller graph for this function:



6.13.4.75 dwt_setsnoozetime()

```
void dwt_setsnoozetime (
    uint8 snooze_time )
```

Here is the call graph for this function:



6.13.4.76 dwt_settxantennadelay()

```
void dwt_settxantennadelay (
    uint16 antennaDly )
```

Here is the call graph for this function:



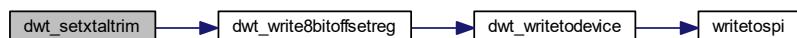
Here is the caller graph for this function:



6.13.4.77 dwt_setxtaltrim()

```
void dwt_setxtaltrim (
    uint8 value )
```

Here is the call graph for this function:



6.13.4.78 dwt_softreset()

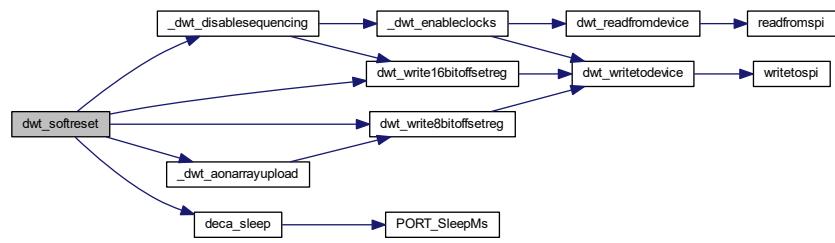
```
void dwt_softreset (
    void )
```

this function resets the DW1000

input parameters:

output parameters

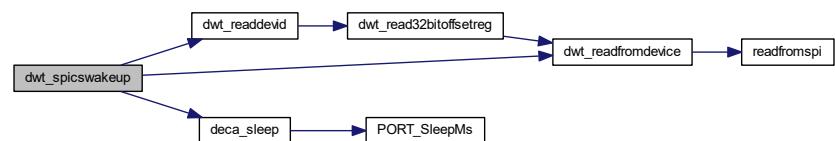
no return value Here is the call graph for this function:



6.13.4.79 dwt_spicswakeup()

```
int dwt_spicswakeup (
    uint8 * buff,
    uint16 length )
```

Here is the call graph for this function:



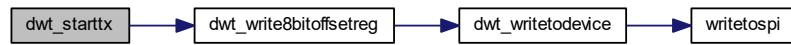
Here is the caller graph for this function:



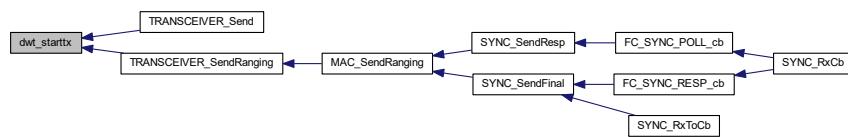
6.13.4.80 dwt_starttx()

```
int dwt_starttx (
    uint8 mode )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.81 dwt_syncrxbufptrs()

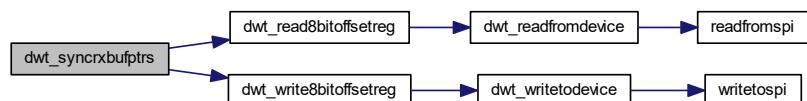
```
void dwt_syncrxbufptrs (
    void )
```

this function synchronizes rx buffer pointers need to make sure that the host/IC buffer pointers are aligned before starting RX

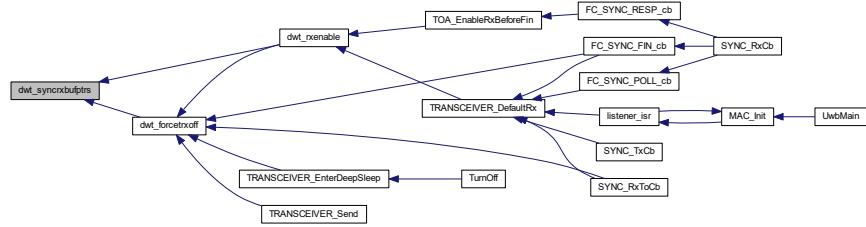
input parameters:

output parameters

no return value Here is the call graph for this function:



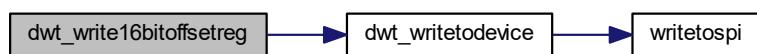
Here is the caller graph for this function:



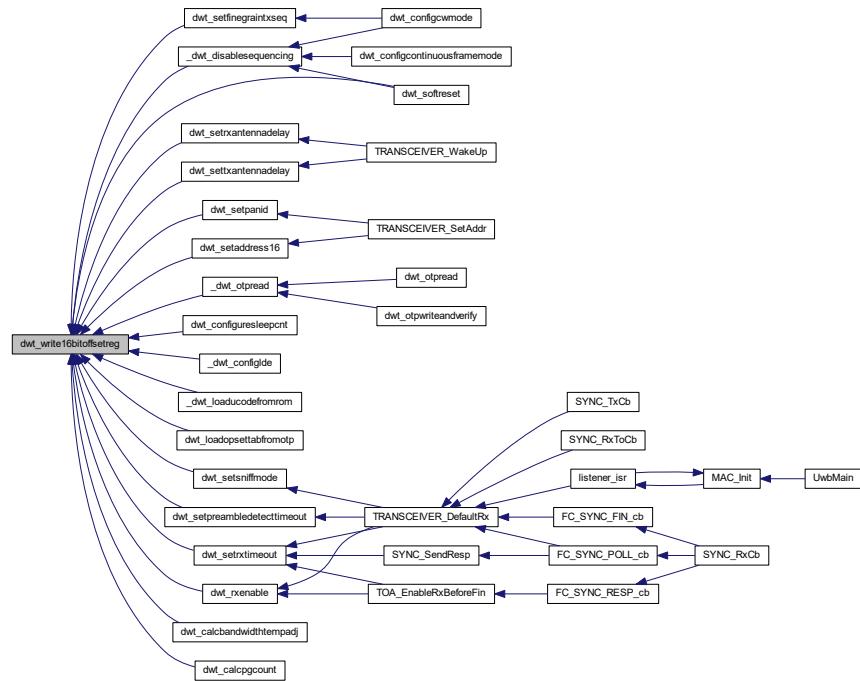
6.13.4.82 `dwt_write16bitoffsetreg()`

```
void dwt_write16bitoffsetreg (
    int regFileID,
    int regOffset,
    uint16 regval )
```

Here is the call graph for this function:



Here is the caller graph for this function:

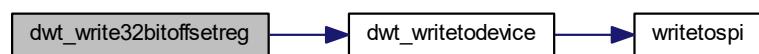


6.13.4.83 dwt_write32bitoffsetreg()

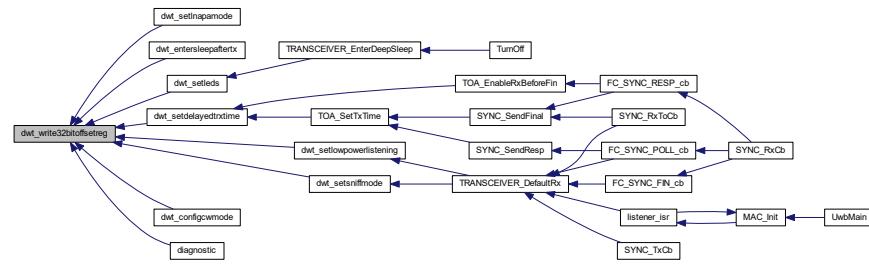
```
void dwt_write32bitoffsetreg (
```

int	<i>regFileID</i> ,
int	<i>regOffset</i> ,
uint32	<i>regval</i>)

Here is the call graph for this function:



Here is the caller graph for this function:



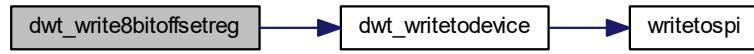
6.13.4.84 dwt_write8bitoffsetreg()

```

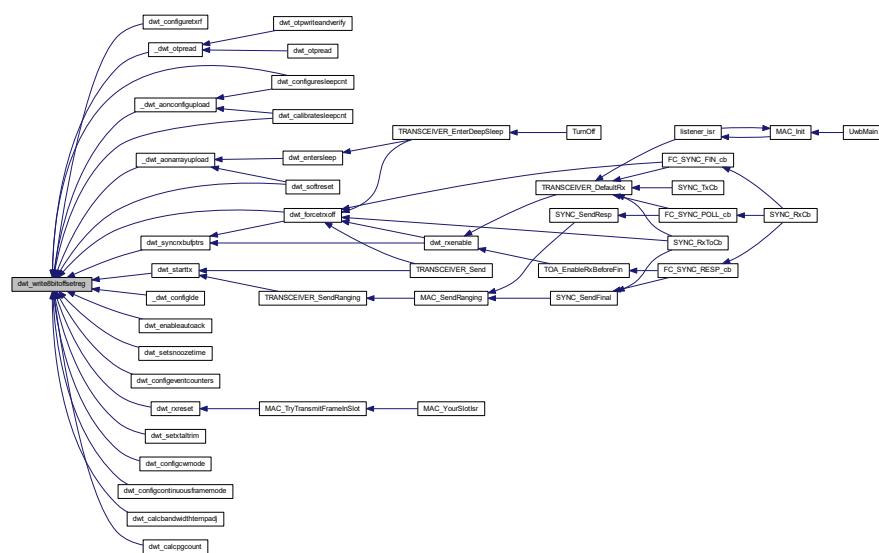
void dwt_write8bitoffsetreg (
    int regFileID,
    int regOffset,
    uint8 regval )

```

Here is the call graph for this function:



Here is the caller graph for this function:



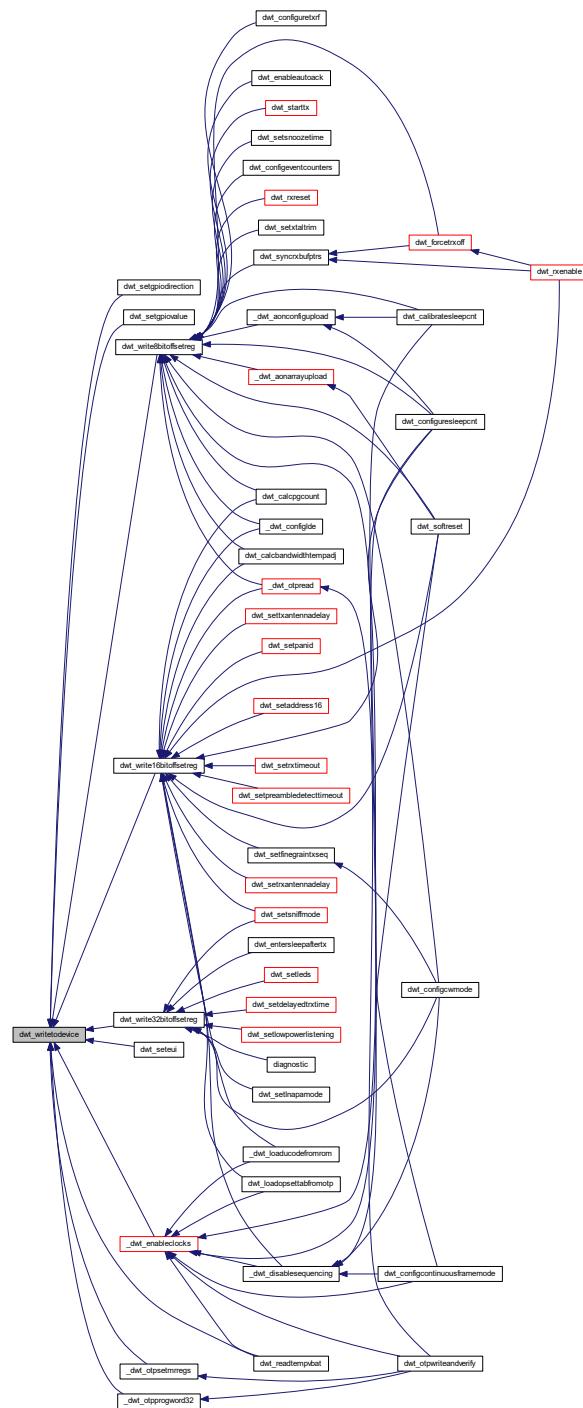
6.13.4.85 dwt_writetodevice()

```
void dwt_writetodevice (
    uint16 recordNumber,
    uint16 index,
    uint32 length,
    const uint8 * buffer )
```

Here is the call graph for this function:



Here is the caller graph for this function:

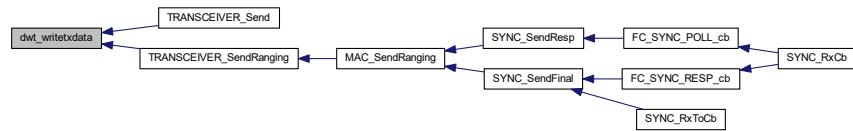


6.13.4.86 `dwt_writetxdata()`

```
int dwt_writetxdata (
    uint16 txFrameLength,
```

```
uint8 * txFrameBytes,
uint16 txBufferOffset )
```

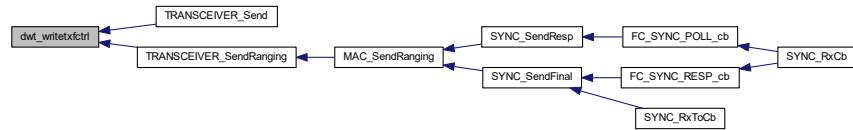
Here is the caller graph for this function:



6.13.4.87 dwt_writetxfctrl()

```
void dwt_writetxfctrl (
    uint16 txFrameLength,
    uint16 txBufferOffset,
    int ranging )
```

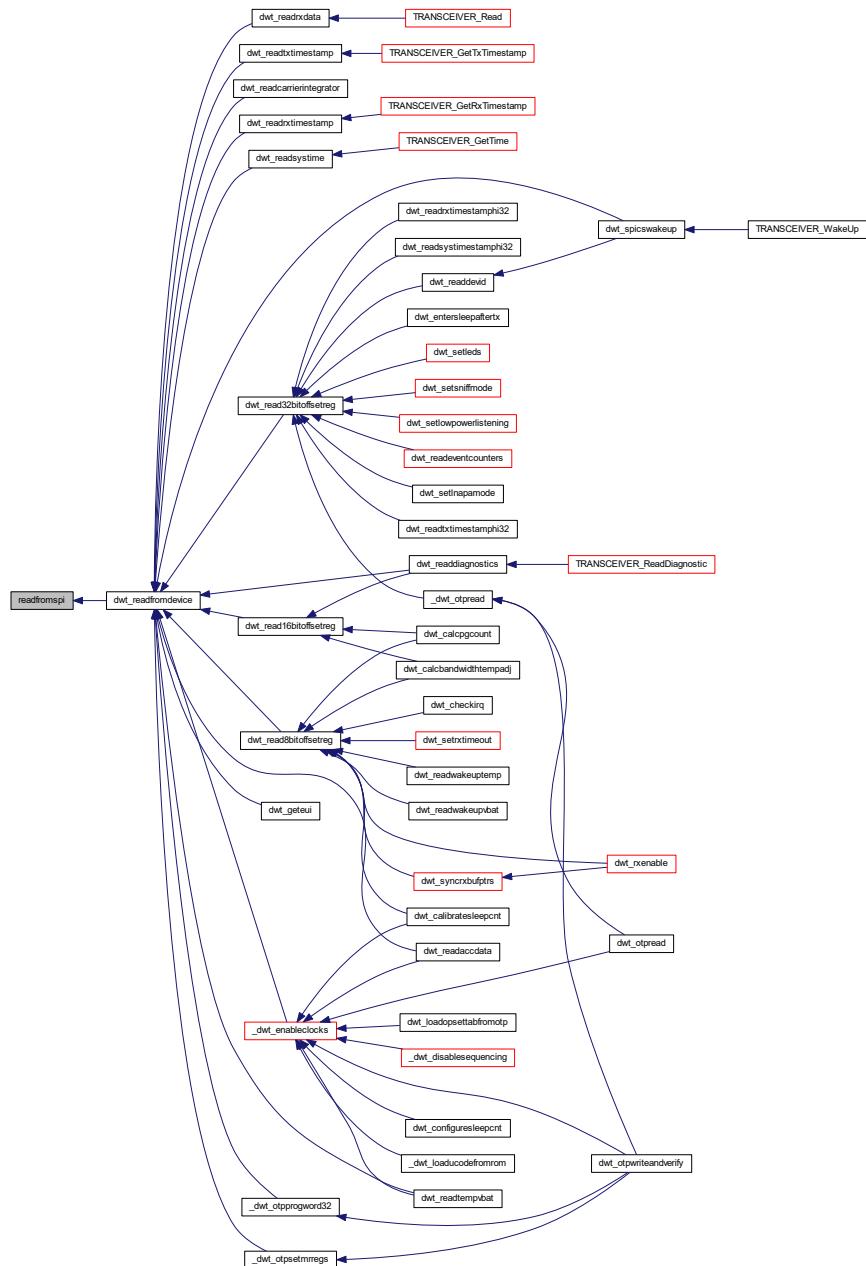
Here is the caller graph for this function:



6.13.4.88 readfromspi()

```
int readfromspi (
    uint16 headerLength,
    const uint8 * headerBuffer,
    uint32 readlength,
    uint8 * readBuffer )
```

Here is the caller graph for this function:



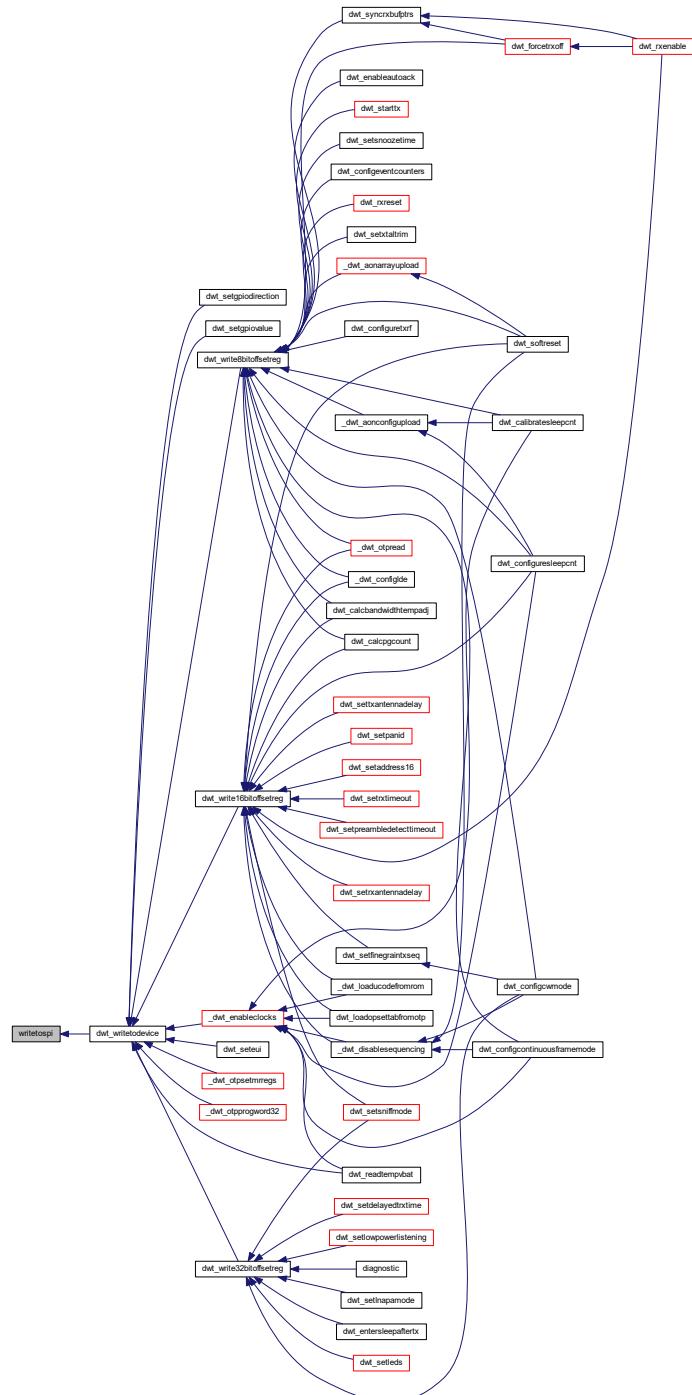
6.13.4.89 writetospi()

```

int writetospi (
    uint16 headerLength,
    const uint8 * headerBuffer,
    uint32 bodylength,
    const uint8 * bodyBuffer )

```

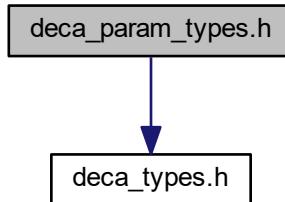
Here is the caller graph for this function:



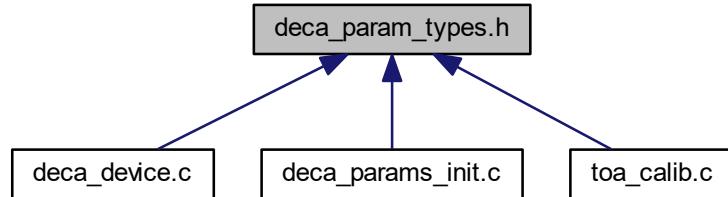
6.14 deca_param_types.h File Reference

Decawave general type definitions for configuration structures.

```
#include "deca_types.h"
Include dependency graph for deca_param_types.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [agc_cfg_struct](#)

Macros

- #define NUM_BR 3
- #define NUM_PRF 2
- #define NUM_PACS 4
- #define NUM_BW 2
- #define NUM_SFD 2
- #define NUM_CH 6
- #define NUM_CH_SUPPORTED 8
- #define PCODES 25
- #define XMLPARAMS_VERSION (1.17f)
- #define PEAK_MULTIPLIER (0x60)
- #define N_STD_FACTOR (13)
- #define LDE_PARAM1 (PEAK_MULTIPLIER | N_STD_FACTOR)
- #define LDE_PARAM3_16 (0x1607)
- #define LDE_PARAM3_64 (0x0607)
- #define MIXER_GAIN_STEP (0.5)
- #define DA_ATTN_STEP (2.5)

Variables

- const `agc_cfg_struct agc_config`
- const `uint16 sftsh [NUM_BR][NUM_SFD]`
- const `uint16 dtune1 [NUM_PRF]`
- const `uint32 fs_pll_cfg [NUM_CH]`
- const `uint8 fs_pll_tune [NUM_CH]`
- const `uint8 rx_config [NUM_BW]`
- const `uint32 tx_config [NUM_CH]`
- const `uint8 dwnsSFDlen [NUM_BR]`
- const `uint32 digital_bb_config [NUM_PRF][NUM_PACS]`
- const `uint8 chan_idx [NUM_CH_SUPPORTED]`
- const double `txpwr_compensation [NUM_CH]`
- const `uint16 lde_replicaCoeff [PCODES]`

6.14.1 Detailed Description

Decawave general type definitions for configuration structures.

Attention

Copyright 2013 (c) Decawave Ltd, Dublin, Ireland.

All rights reserved.

6.14.2 Macro Definition Documentation

6.14.2.1 DA_ATTN_STEP

```
#define DA_ATTN_STEP (2.5)
```

6.14.2.2 LDE_PARAM1

```
#define LDE_PARAM1 (PEAK_MULTIPLIER | N_STD_FACTOR)
```

6.14.2.3 LDE_PARAM3_16

```
#define LDE_PARAM3_16 (0x1607)
```

6.14.2.4 LDE_PARAM3_64

```
#define LDE_PARAM3_64 (0x0607)
```

6.14.2.5 MIXER_GAIN_STEP

```
#define MIXER_GAIN_STEP (0.5)
```

6.14.2.6 N_STD_FACTOR

```
#define N_STD_FACTOR (13)
```

6.14.2.7 NUM_BR

```
#define NUM_BR 3
```

6.14.2.8 NUM_BW

```
#define NUM_BW 2
```

6.14.2.9 NUM_CH

```
#define NUM_CH 6
```

6.14.2.10 NUM_CH_SUPPORTED

```
#define NUM_CH_SUPPORTED 8
```

6.14.2.11 NUM_PACS

```
#define NUM_PACS 4
```

6.14.2.12 NUM_PRF

```
#define NUM_PRF 2
```

6.14.2.13 NUM_SFD

```
#define NUM_SFD 2
```

6.14.2.14 PCODES

```
#define PCODES 25
```

6.14.2.15 PEAK_MULTPLIER

```
#define PEAK_MULTPLIER (0x60)
```

6.14.2.16 XMLPARAMS_VERSION

```
#define XMLPARAMS_VERSION (1.17f)
```

6.14.3 Variable Documentation

6.14.3.1 agc_config

```
const agc_cfg_struct agc_config
```

6.14.3.2 chan_idx

```
const uint8 chan_idx[NUM_CH_SUPPORTED]
```

6.14.3.3 digital_bb_config

```
const uint32 digital_bb_config[NUM_PRF] [NUM_PACS]
```

6.14.3.4 dtune1

```
const uint16 dtune1[NUM_PRF]
```

6.14.3.5 dwnsSFDlen

```
const uint8 dwnsSFDlen[NUM_BR]
```

6.14.3.6 fs_pll_cfg

```
const uint32 fs_pll_cfg[NUM_CH]
```

6.14.3.7 fs_pll_tune

```
const uint8 fs_pll_tune[NUM_CH]
```

6.14.3.8 lde_replicaCoeff

```
const uint16 lde_replicaCoeff[PCODES]
```

6.14.3.9 rx_config

```
const uint8 rx_config[NUM_BW]
```

6.14.3.10 sftsh

```
const uint16 sftsh[NUM_BR] [NUM_SFD]
```

6.14.3.11 tx_config

```
const uint32 tx_config[NUM_CH]
```

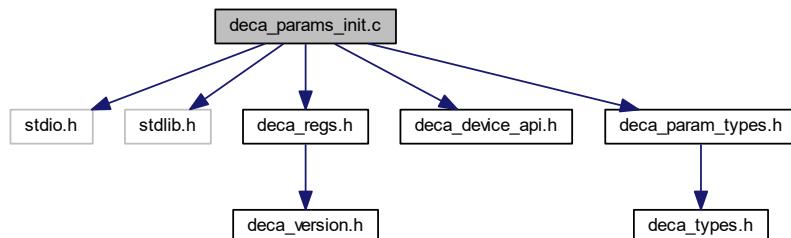
6.14.3.12 txpwr_compensation

```
const double txpwr_compensation[NUM_CH]
```

6.15 deca_params_init.c File Reference

DW1000 configuration parameters.

```
#include <stdio.h>
#include <stdlib.h>
#include "deca_regs.h"
#include "deca_device_api.h"
#include "deca_param_types.h"
Include dependency graph for deca_params_init.c:
```



Variables

- const uint8 chan_idx [NUM_CH_SUPPORTED] = {0, 0, 1, 2, 3, 4, 0, 5}
- const uint32 tx_config [NUM_CH]
- const uint32 fs_pll_cfg [NUM_CH]
- const uint8 fs_pll_tune [NUM_CH]
- const uint8 rx_config [NUM_BW]
- const agc_cfg_struct agc_config
- const uint8 dwnsSFDlen [NUM_BR]
- const uint16 sftsh [NUM_BR][NUM_SFD]
- const uint16 dtune1 [NUM_PRF]
- const uint32 digital_bb_config [NUM_PRF][NUM_PACS]
- const uint16 lde_replicaCoeff [PCODES]
- const double txpwr_compensation [NUM_CH]

6.15.1 Detailed Description

DW1000 configuration parameters.

Attention

Copyright 2013 (c) Decawave Ltd, Dublin, Ireland.

All rights reserved.

6.15.2 Variable Documentation

6.15.2.1 agc_config

```
const agc_cfg_struct agc_config
```

Initial value:

```
=  
{  
    AGC_TUNE2_VAL,  
    { AGC_TUNE1_16M , AGC_TUNE1_64M }  
}
```

6.15.2.2 chan_idx

```
const uint8 chan_idx[NUM_CH_SUPPORTED] = {0, 0, 1, 2, 3, 4, 0, 5}
```

6.15.2.3 digital_bb_config

```
const uint32 digital_bb_config[NUM_PRF][NUM_PACS]
```

Initial value:

```
=  
{  
    {  
        DRX_TUNE2_PRF16_PAC8,  
        DRX_TUNE2_PRF16_PAC16,  
        DRX_TUNE2_PRF16_PAC32,  
        DRX_TUNE2_PRF16_PAC64  
    },  
    {  
        DRX_TUNE2_PRF64_PAC8,  
        DRX_TUNE2_PRF64_PAC16,  
        DRX_TUNE2_PRF64_PAC32,  
        DRX_TUNE2_PRF64_PAC64  
    }  
}
```

6.15.2.4 dtune1

```
const uint16 dtune1[NUM_PRF]
```

Initial value:

```
=
{
    DRX_TUNE1a_PRF16,
    DRX_TUNE1a_PRF64
}
```

6.15.2.5 dwnsSFDlen

```
const uint8 dwnsSFDlen[NUM_BR]
```

Initial value:

```
=
{
    DW_NS_SFD_LEN_110K,
    DW_NS_SFD_LEN_850K,
    DW_NS_SFD_LEN_6M8
}
```

6.15.2.6 fs_pll_cfg

```
const uint32 fs_pll_cfg[NUM_CH]
```

Initial value:

```
=
{
    FS_PLLCFG_CH1,
    FS_PLLCFG_CH2,
    FS_PLLCFG_CH3,
    FS_PLLCFG_CH4,
    FS_PLLCFG_CH5,
    FS_PLLCFG_CH7
}
```

6.15.2.7 fs_pll_tune

```
const uint8 fs_pll_tune[NUM_CH]
```

Initial value:

```
=
{
    FS_PLLTUNE_CH1,
    FS_PLLTUNE_CH2,
    FS_PLLTUNE_CH3,
    FS_PLLTUNE_CH4,
    FS_PLLTUNE_CH5,
    FS_PLLTUNE_CH7
}
```

6.15.2.8 lde_replicaCoeff

```
const uint16 lde_replicaCoeff[PCODES]
```

Initial value:

```
=
{
    0,
    LDE_REPC_PCODE_1,
    LDE_REPC_PCODE_2,
    LDE_REPC_PCODE_3,
    LDE_REPC_PCODE_4,
    LDE_REPC_PCODE_5,
    LDE_REPC_PCODE_6,
    LDE_REPC_PCODE_7,
    LDE_REPC_PCODE_8,
    LDE_REPC_PCODE_9,
    LDE_REPC_PCODE_10,
    LDE_REPC_PCODE_11,
    LDE_REPC_PCODE_12,
    LDE_REPC_PCODE_13,
    LDE_REPC_PCODE_14,
    LDE_REPC_PCODE_15,
    LDE_REPC_PCODE_16,
    LDE_REPC_PCODE_17,
    LDE_REPC_PCODE_18,
    LDE_REPC_PCODE_19,
    LDE_REPC_PCODE_20,
    LDE_REPC_PCODE_21,
    LDE_REPC_PCODE_22,
    LDE_REPC_PCODE_23,
    LDE_REPC_PCODE_24
}
```

6.15.2.9 rx_config

```
const uint8 rx_config[NUM_BW]
```

Initial value:

```
=
{
    RF_RXCTRLH_NBW,
    RF_RXCTRLH_WBW
}
```

6.15.2.10 sftsh

```
const uint16 sftsh[NUM_BR] [NUM_SFD]
```

Initial value:

```
=
{
    {
        DRX_TUNE0b_110K_STD,
        DRX_TUNE0b_110K_NSTD
    },
    {
        DRX_TUNE0b_850K_STD,
        DRX_TUNE0b_850K_NSTD
    },
    {
        DRX_TUNE0b_6M8_STD,
        DRX_TUNE0b_6M8_NSTD
    }
}
```

6.15.2.11 tx_config

```
const uint32 tx_config[NUM_CH]
```

Initial value:

```
=
{
    RF_TXCTRL_CH1,
    RF_TXCTRL_CH2,
    RF_TXCTRL_CH3,
    RF_TXCTRL_CH4,
    RF_TXCTRL_CH5,
    RF_TXCTRL_CH7,
}
```

6.15.2.12 txpwr_compensation

```
const double txpwr_compensation[NUM_CH]
```

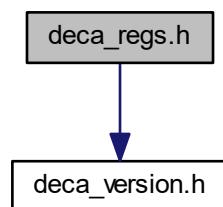
Initial value:

```
= {
    0.0,
    0.035,
    0.0,
    0.0,
    0.065,
    0.0
}
```

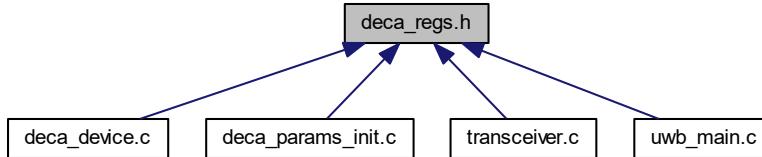
6.16 deca_regs.h File Reference

DW1000 Register Definitions This file supports assembler and C development for DW1000 enabled devices.

```
#include "deca_version.h"
Include dependency graph for deca_regs.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define DEV_ID_ID 0x00 /* Device ID register, includes revision info (0xDECA0130) */`
Bit definitions for register DEV_ID.
- `#define DEV_ID_LEN (4)`
- `#define DEV_ID_REV_MASK 0x0000000FUL /* Revision */`
- `#define DEV_ID_VER_MASK 0x000000F0UL /* Version */`
- `#define DEV_ID_MODEL_MASK 0x0000FF00UL /* The MODEL identifies the device. The DW1000 is device type 0x01 */`
- `#define DEV_ID_RIDTAG_MASK 0xFFFF0000UL /* Register Identification Tag 0XDECA */`
- `#define EUI_64_ID 0x01 /* IEEE Extended Unique Identifier (63:0) */`
Bit definitions for register EUI_64.
- `#define EUI_64_OFFSET 0x00`
- `#define EUI_64_LEN (8)`
- `#define PANADR_ID 0x03 /* PAN ID (31:16) and Short Address (15:0) */`
Bit definitions for register PANADR.
- `#define PANADR_LEN (4)`
- `#define PANADR_SHORT_ADDR_OFFSET 0 /* In bytes */`
- `#define PANADR_SHORT_ADDR_MASK 0x0000FFFFUL /* Short Address */`
- `#define PANADR_PAN_ID_OFFSET 2 /* In bytes */`
- `#define PANADR_PAN_ID_MASK 0xFFFF00F0UL /* PAN Identifier */`
- `#define REG_05_ID_RESERVED 0x05`
Bit definitions for register 0x05.
- `#define SYS_CFG_ID 0x04 /* System Configuration (31:0) */`
Bit definitions for register SYS_CFG.
- `#define SYS_CFG_LEN (4)`
- `#define SYS_CFG_MASK 0xF047FFFFFFUL /* access mask to SYS_CFG_ID */`
- `#define SYS_CFG_FF_ALL_EN 0x0000001FEUL /* Frame filtering options all frames allowed */`
- `#define SYS_CFG_FFE 0x00000001UL /* Frame Filtering Enable. This bit enables the frame filtering functionality */`
- `#define SYS_CFG_FFBC 0x00000002UL /* Frame Filtering Behave as a Co-ordinator */`
- `#define SYS_CFG_FFAB 0x00000004UL /* Frame Filtering Allow Beacon frame reception */`
- `#define SYS_CFG_FFAD 0x00000008UL /* Frame Filtering Allow Data frame reception */`
- `#define SYS_CFG_FFAA 0x00000010UL /* Frame Filtering Allow Acknowledgment frame reception */`
- `#define SYS_CFG_FFAM 0x00000020UL /* Frame Filtering Allow MAC command frame reception */`
- `#define SYS_CFG_FFAR 0x00000040UL /* Frame Filtering Allow Reserved frame types */`
- `#define SYS_CFG_FFA4 0x00000080UL /* Frame Filtering Allow frames with frame type field of 4, (binary 100) */`
- `#define SYS_CFG_FFA5 0x00000100UL /* Frame Filtering Allow frames with frame type field of 5, (binary 101) */`

- #define **SYS_CFG_HIRQ_POL** 0x00000200UL /* Host interrupt polarity */
- #define **SYS_CFG_SPI_EDGE** 0x00000400UL /* SPI data launch edge */
- #define **SYS_CFG_DIS_FCE** 0x00000800UL /* Disable frame check error handling */
- #define **SYS_CFG_DIS_DRXB** 0x00001000UL /* Disable Double RX Buffer */
- #define **SYS_CFG_DIS_PHE** 0x00002000UL /* Disable receiver abort on PHR error */
- #define **SYS_CFG_DIS_RSDE** 0x00004000UL /* Disable Receiver Abort on RSD error */
- #define **SYS_CFG_FCS_INIT2F** 0x00008000UL /* initial seed value for the FCS generation and checking function */
- #define **SYS_CFG_PHR_MODE_SHFT** 16
- #define **SYS_CFG_PHR_MODE_00** 0x00000000UL /* Standard Frame mode */
- #define **SYS_CFG_PHR_MODE_11** 0x00030000UL /* Long Frames mode */
- #define **SYS_CFG_DIS_STXP** 0x00040000UL /* Disable Smart TX Power control */
- #define **SYS_CFG_RXM110K** 0x00400000UL /* Receiver Mode 110 kbps data rate */
- #define **SYS_CFG_RXWTOE** 0x10000000UL /* Receive Wait Timeout Enable. */
- #define **SYS_CFG_RXAUTR** 0x20000000UL /* Receiver Auto-Re-enable. This bit is used to cause the receiver to re-enable automatically */
- #define **SYS_CFG_AUTOACK** 0x40000000UL /* Automatic Acknowledgement Enable */
- #define **SYS_CFG_AACKPEND** 0x80000000UL /* Automatic Acknowledgement Pending bit control */
- #define **SYS_TIME_ID** 0x06 /* System Time Counter (40-bit) */
 - Bit definitions for register SYS_TIME.*
 - #define **SYS_TIME_OFFSET** 0x00
 - #define **SYS_TIME_LEN** (5) /* Note 40 bit register */
 - #define **REG_07_ID_RESERVED** 0x07
- Bit definitions for register 0x07.*
- #define **TX_FCTRL_ID** 0x08 /* Transmit Frame Control */
 - Bit definitions for register TX_FCTRL.*
 - #define **TX_FCTRL_LEN** (5) /* Note 40 bit register */
 - #define **TX_FCTRL_TFLEN_MASK** 0x0000007FUL /* bit mask to access Transmit Frame Length */
 - #define **TX_FCTRL_TFLE_MASK** 0x00000380UL /* bit mask to access Transmit Frame Length Extension */
 - #define **TX_FCTRL_FLE_MASK** 0x000003FFUL /* bit mask to access Frame Length field */
 - #define **TX_FCTRL_TXBR_MASK** 0x00006000UL /* bit mask to access Transmit Bit Rate */
 - #define **TX_FCTRL_TXPRF_MASK** 0x00030000UL /* bit mask to access Transmit Pulse Repetition Frequency */
 - #define **TX_FCTRL_TXPSR_MASK** 0x000C0000UL /* bit mask to access Transmit Preamble Symbol Repetitions (PSR). */
 - #define **TX_FCTRL_PE_MASK** 0x00300000UL /* bit mask to access Preamble Extension */
 - #define **TX_FCTRL_TXPSR_PE_MASK** 0x003C0000UL /* bit mask to access Transmit Preamble Symbol Repetitions (PSR). */
 - #define **TX_FCTRL_SAFE_MASK_32** 0xFFFFE3FFUL /* FSCTRL has fields which should always be written zero */
 - #define **TX_FCTRL_TXBR_110k** 0x00000000UL /* Transmit Bit Rate = 110k */
 - #define **TX_FCTRL_TXBR_850k** 0x00002000UL /* Transmit Bit Rate = 850k */
 - #define **TX_FCTRL_TXBR_6M** 0x00004000UL /* Transmit Bit Rate = 6.8M */
 - #define **TX_FCTRL_TXBR_SHFT** (13) /* shift to access Data Rate field */
 - #define **TX_FCTRL_TR** 0x00008000UL /* Transmit Ranging enable */
 - #define **TX_FCTRL_TR_SHFT** (15) /* shift to access Ranging bit */
 - #define **TX_FCTRL_TXPRF_SHFT** (16) /* shift to access Pulse Repetition Frequency field */
 - #define **TX_FCTRL_TXPRF_4M** 0x00000000UL /* Transmit Pulse Repetition Frequency = 4 Mhz */
 - #define **TX_FCTRL_TXPRF_16M** 0x00010000UL /* Transmit Pulse Repetition Frequency = 16 Mhz */
 - #define **TX_FCTRL_TXPRF_64M** 0x00020000UL /* Transmit Pulse Repetition Frequency = 64 Mhz */
 - #define **TX_FCTRL_TXPSR_SHFT** (18) /* shift to access Preamble Symbol Repetitions field */
 - #define **TX_FCTRL_PE_SHFT** (20) /* shift to access Preamble length Extension to allow specification of non-standard values */
 - #define **TX_FCTRL_TXPSR_PE_16** 0x00000000UL /* bit mask to access Preamble Extension = 16 */

- #define TX_FCTRL_TXPSR_PE_64 0x00040000UL /* bit mask to access Preamble Extension = 64 */
- #define TX_FCTRL_TXPSR_PE_128 0x00140000UL /* bit mask to access Preamble Extension = 128 */
- #define TX_FCTRL_TXPSR_PE_256 0x00240000UL /* bit mask to access Preamble Extension = 256 */
- #define TX_FCTRL_TXPSR_PE_512 0x00340000UL /* bit mask to access Preamble Extension = 512 */
- #define TX_FCTRL_TXPSR_PE_1024 0x00080000UL /* bit mask to access Preamble Extension = 1024 */
- #define TX_FCTRL_TXPSR_PE_1536 0x00180000UL /* bit mask to access Preamble Extension = 1536 */
- #define TX_FCTRL_TXPSR_PE_2048 0x00280000UL /* bit mask to access Preamble Extension = 2048 */
- #define TX_FCTRL_TXPSR_PE_4096 0x000C0000UL /* bit mask to access Preamble Extension = 4096 */
- #define TX_FCTRL_TXBOFFS_SHFT (22) /* Shift to access transmit buffer index offset */
- #define TX_FCTRL_TXBOFFS_MASK 0xFFC00000UL /* bit mask to access Transmit buffer index offset 10-bit field */
- #define TX_FCTRL_IFSDELAY_MASK 0xFF0000000ULL /* bit mask to access Inter-Frame Spacing field */
- #define TX_BUFFER_ID 0x09 /* Transmit Data Buffer */

Bit definitions for register TX_BUFFER.
- #define TX_BUFFER_LEN (1024)
- #define DX_TIME_ID 0xA /* Delayed Send or Receive Time (40-bit) */

Bit definitions for register DX_TIME.
- #define DX_TIME_LEN (5)
- #define REG_0B_ID_RESERVED 0xB

Bit definitions for register 0x08.
- #define RX_FWTO_ID 0xC /* Receive Frame Wait Timeout Period */

Bit definitions for register RX_FWTO.
- #define RX_FWTO_OFFSET 0x00
- #define RX_FWTO_LEN (2)
- #define RX_FWTO_MASK 0xFFFF
- #define SYS_CTRL_ID 0xD /* System Control Register */

Bit definitions for register SYS_CTRL.
- #define SYS_CTRL_OFFSET 0x00
- #define SYS_CTRL_LEN (4)
- #define SYS_CTRL_MASK_32 0x010003CFUL /* System Control Register access mask (all unused fields should always be written as zero) */
- #define SYS_CTRL_SFCST 0x00000001UL /* Suppress Auto-FCS Transmission (on this frame) */
- #define SYS_CTRL_TXSTRT 0x00000002UL /* Start Transmitting Now */
- #define SYS_CTRL_TXDLYS 0x00000004UL /* Transmitter Delayed Sending (initiates sending when SY←S_TIME == TXD_TIME) */
- #define SYS_CTRL_CANSFCS 0x00000008UL /* Cancel Suppression of auto-FCS transmission (on the current frame) */
- #define SYS_CTRL_TRXOFF 0x00000040UL /* Transceiver Off. Force Transciever OFF abort TX or RX immediately */
- #define SYS_CTRL_WAIT4RESP 0x00000080UL /* Wait for Response */
- #define SYS_CTRL_RXENAB 0x00000100UL /* Enable Receiver Now */
- #define SYS_CTRL_RXDLYE 0x00000200UL /* Receiver Delayed Enable (Enables Receiver when SY_T←IME[0x??] == RXD_TIME[0x??] CHECK comment)*/
- #define SYS_CTRL_HSRBToggle 0x01000000UL /* Host side receiver buffer pointer toggle - toggles 0/1 host side data set pointer */
- #define SYS_CTRL_HRBTOGGLE (SYS_CTRL_HSRBToggle)
- #define SYS_CTRL_HRBTOGGLE_OFFSET (3)
- #define SYS_MASK_ID 0xE /* System Event Mask Register */

Bit definitions for register SYS_MASK.
- #define SYS_MASK_LEN (4)
- #define SYS_MASK_MASK_32 0x3FF7FFEUL /* System Event Mask Register access mask (all unused fields should always be written as zero) */
- #define SYS_MASK_MCPLOCK 0x00000002UL /* Mask clock PLL lock event */

- #define **SYS_MASK_MESYNCR** 0x00000004UL /* Mask clock PLL lock event */
- #define **SYS_MASK_MAAT** 0x00000008UL /* Mask automatic acknowledge trigger event */
- #define **SYS_MASK_MTXFRB** 0x00000010UL /* Mask transmit frame begins event */
- #define **SYS_MASK_MTXPRS** 0x00000020UL /* Mask transmit preamble sent event */
- #define **SYS_MASK_MTXPHS** 0x00000040UL /* Mask transmit PHY Header Sent event */
- #define **SYS_MASK_MTXFRS** 0x00000080UL /* Mask transmit frame sent event */
- #define **SYS_MASK_MRXP RD** 0x00000100UL /* Mask receiver preamble detected event */
- #define **SYS_MASK_MRXS FDD** 0x00000200UL /* Mask receiver SFD detected event */
- #define **SYS_MASK_MLD E DONE** 0x00000400UL /* Mask LDE processing done event */
- #define **SYS_MASK_MRXP HD** 0x00000800UL /* Mask receiver PHY header detect event */
- #define **SYS_MASK_MRXP HE** 0x00001000UL /* Mask receiver PHY header error event */
- #define **SYS_MASK_MR XD FR** 0x00002000UL /* Mask receiver data frame ready event */
- #define **SYS_MASK_MR XFCG** 0x00004000UL /* Mask receiver FCS good event */
- #define **SYS_MASK_MR XFCE** 0x00008000UL /* Mask receiver FCS error event */
- #define **SYS_MASK_MR XRFSL** 0x00010000UL /* Mask receiver Reed Solomon Frame Sync Loss event */
- #define **SYS_MASK_MR XRFTO** 0x00020000UL /* Mask Receive Frame Wait Timeout event */
- #define **SYS_MASK_MLD E ERR** 0x00040000UL /* Mask leading edge detection processing error event */
- #define **SYS_MASK_MR XOVRR** 0x0100000UL /* Mask Receiver Overrun event */
- #define **SYS_MASK_MR XPTO** 0x02000000UL /* Mask Preamble detection timeout event */
- #define **SYS_MASK_MG PIOIRQ** 0x04000000UL /* Mask GPIO interrupt event */
- #define **SYS_MASK_MSLP2INIT** 0x08000000UL /* Mask SLEEP to INIT event */
- #define **SYS_MASK_MRFPLL** 0x01000000UL /* Mask RF PLL Loosing Lock warning event */
- #define **SYS_MASK_MCPPLL** 0x02000000UL /* Mask Clock PLL Loosing Lock warning event */
- #define **SYS_MASK_MR XS FDTO** 0x04000000UL /* Mask Receive SFD timeout event */
- #define **SYS_MASK_MHPD WARN** 0x08000000UL /* Mask Half Period Delay Warning event */
- #define **SYS_MASK_MT XBERR** 0x10000000UL /* Mask Transmit Buffer Error event */
- #define **SYS_MASK_MAFFREJ** 0x20000000UL /* Mask Automatic Frame Filtering rejection event */
- #define **SYS_STATUS_ID** 0x0F /* System event Status Register */

Bit definitions for register SYS_STATUS.

- #define **SYS_STATUS_OFFSET** 0x00
- #define **SYS_STATUS_LEN** (5) /* Note 40 bit register */
- #define **SYS_STATUS_MASK_32** 0xFFFF7FFFFUL /* System event Status Register access mask (all unused fields should always be written as zero) */
- #define **SYS_STATUS IRQS** 0x00000001UL /* Interrupt Request Status READ ONLY */
- #define **SYS_STATUS_CPLock** 0x00000002UL /* Clock PLL Lock */
- #define **SYS_STATUS_ESYNCR** 0x00000004UL /* External Sync Clock Reset */
- #define **SYS_STATUS_AAT** 0x00000008UL /* Automatic Acknowledge Trigger */
- #define **SYS_STATUS_TXFRB** 0x00000010UL /* Transmit Frame Begins */
- #define **SYS_STATUS_TXPRS** 0x00000020UL /* Transmit Preamble Sent */
- #define **SYS_STATUS_TXPHS** 0x00000040UL /* Transmit PHY Header Sent */
- #define **SYS_STATUS_TXFRS** 0x00000080UL /* Transmit Frame Sent: This is set when the transmitter has completed the sending of a frame */
- #define **SYS_STATUS_RXPRD** 0x00000100UL /* Receiver Preamble Detected status */
- #define **SYS_STATUS_RXSFDD** 0x00000200UL /* Receiver Start Frame Delimiter Detected. */
- #define **SYS_STATUS_LDEDONE** 0x00000400UL /* LDE processing done */
- #define **SYS_STATUS_RXPHD** 0x00000800UL /* Receiver PHY Header Detect */
- #define **SYS_STATUS_RXPHE** 0x00001000UL /* Receiver PHY Header Error */
- #define **SYS_STATUS_RXDFR** 0x00002000UL /* Receiver Data Frame Ready */
- #define **SYS_STATUS_RXFCG** 0x00004000UL /* Receiver FCS Good */
- #define **SYS_STATUS_RXFCE** 0x00008000UL /* Receiver FCS Error */
- #define **SYS_STATUS_RXRFSL** 0x00010000UL /* Receiver Reed Solomon Frame Sync Loss */
- #define **SYS_STATUS_RXRFTO** 0x00020000UL /* Receive Frame Wait Timeout */
- #define **SYS_STATUS_LDEERR** 0x00040000UL /* Leading edge detection processing error */
- #define **SYS_STATUS_reserved** 0x00080000UL /* bit19 reserved */

- #define **SYS_STATUS_RXOVERR** 0x00100000UL /* Receiver Overrun */
 - #define **SYS_STATUS_RXPTO** 0x00200000UL /* Preamble detection timeout */
 - #define **SYS_STATUS_GPIOIRQ** 0x00400000UL /* GPIO interrupt */
 - #define **SYS_STATUS_SLP2INIT** 0x00800000UL /* SLEEP to INIT */
 - #define **SYS_STATUS_RFPLL_LL** 0x01000000UL /* RF PLL Losing Lock */
 - #define **SYS_STATUS_CLKPLL_LL** 0x02000000UL /* Clock PLL Losing Lock */
 - #define **SYS_STATUS_RXSFDTO** 0x04000000UL /* Receive SFD timeout */
 - #define **SYS_STATUS_HPDWARN** 0x08000000UL /* Half Period Delay Warning */
 - #define **SYS_STATUS_TXBERR** 0x10000000UL /* Transmit Buffer Error */
 - #define **SYS_STATUS_AFFREJ** 0x20000000UL /* Automatic Frame Filtering rejection */
 - #define **SYS_STATUS_HSRBP** 0x40000000UL /* Host Side Receive Buffer Pointer */
 - #define **SYS_STATUS_ICRBP** 0x80000000UL /* IC side Receive Buffer Pointer READ ONLY */
 - #define **SYS_STATUS_RXRSCS** 0x010000000ULL /* Receiver Reed-Solomon Correction Status */
 - #define **SYS_STATUS_RXPREJ** 0x020000000ULL /* Receiver Preamble Rejection */
 - #define **SYS_STATUS_TXPUTE** 0x040000000ULL /* Transmit power up time error */
 - #define **SYS_STATUS_TXERR** (0x0408) /* These bits are the 16 high bits of status register TXPUTE and HPDWARN flags */
 - #define **SYS_STATUS_ALL_RX_GOOD**
 - #define **SYS_STATUS_ALL_DBLBUFF** (**SYS_STATUS_RXDFR** | **SYS_STATUS_RXFCG**)
 - #define **SYS_STATUS_ALL_RX_ERR**
 - #define **SYS_STATUS_ALL_RX_TO** (**SYS_STATUS_RXRFTO** | **SYS_STATUS_RXPTO**)
 - #define **SYS_STATUS_ALL_TX**
 - #define **RX_FINFO_ID** 0x10 /* RX Frame Information (in double buffer set) */
- Bit definitions for register RX_FINFO.*
- #define **RX_FINFO_OFFSET** 0x00
 - #define **RX_FINFO_LEN** (4)
 - #define **RX_FINFO_MASK_32** 0xFFFFFBFFUL /* System event Status Register access mask (all unused fields should always be written as zero) */
 - #define **RX_FINFO_RXFLEN_MASK** 0x0000007FUL /* Receive Frame Length (0 to 127) */
 - #define **RX_FINFO_RXFLE_MASK** 0x00000380UL /* Receive Frame Length Extension (0 to 7)<<7 */
 - #define **RX_FINFO_RXFL_MASK_1023** 0x000003FFUL /* Receive Frame Length Extension (0 to 1023) */
 - #define **RX_FINFO_RXNSPL_MASK** 0x00001800UL /* Receive Non-Standard Preamble Length */
 - #define **RX_FINFO_RXPSR_MASK** 0x000C0000UL /* RX Preamble Repetition. 00 = 16 symbols, 01 = 64 symbols, 10 = 1024 symbols, 11 = 4096 symbols */
 - #define **RX_FINFO_RXPEL_MASK** 0x000C1800UL /* Receive Preamble Length = RXPSR+RXNSPL */
 - #define **RX_FINFO_RXPEL_64** 0x00040000UL /* Receive Preamble length = 64 */
 - #define **RX_FINFO_RXPEL_128** 0x00040800UL /* Receive Preamble length = 128 */
 - #define **RX_FINFO_RXPEL_256** 0x00041000UL /* Receive Preamble length = 256 */
 - #define **RX_FINFO_RXPEL_512** 0x00041800UL /* Receive Preamble length = 512 */
 - #define **RX_FINFO_RXPEL_1024** 0x00080000UL /* Receive Preamble length = 1024 */
 - #define **RX_FINFO_RXPEL_1536** 0x00080800UL /* Receive Preamble length = 1536 */
 - #define **RX_FINFO_RXPEL_2048** 0x00081000UL /* Receive Preamble length = 2048 */
 - #define **RX_FINFO_RXPEL_4096** 0x000C0000UL /* Receive Preamble length = 4096 */
 - #define **RX_FINFO_RXBR_MASK** 0x00006000UL /* Receive Bit Rate report. This field reports the received bit rate */
 - #define **RX_FINFO_RXBR_110k** 0x00000000UL /* Received bit rate = 110 kbps */
 - #define **RX_FINFO_RXBR_850k** 0x00002000UL /* Received bit rate = 850 kbps */
 - #define **RX_FINFO_RXBR_6M** 0x00004000UL /* Received bit rate = 6.8 Mbps */
 - #define **RX_FINFO_RXBR_SHIFT** (13)
 - #define **RX_FINFO_RNG** 0x00008000UL /* Receiver Ranging. Ranging bit in the received PHY header identifying the frame as a ranging packet. */
 - #define **RX_FINFO_RNG_SHIFT** (15)
 - #define **RX_FINFO_RXPRF_MASK** 0x00030000UL /* RX Pulse Repetition Rate report */
 - #define **RX_FINFO_RXPRF_16M** 0x00010000UL /* PRF being employed in the receiver = 16M */
 - #define **RX_FINFO_RXPRF_64M** 0x00020000UL /* PRF being employed in the receiver = 64M */

- #define RX_FINFO_RXPRF_SHIFT (16)
- #define RX_FINFO_RXPACC_MASK 0xFFFF0000UL /* Preamble Accumulation Count */
- #define RX_FINFO_RXPACC_SHIFT (20)
- #define RX_BUFFER_ID 0x11 /* Receive Data Buffer (in double buffer set) */

Bit definitions for register RX_BUFFER.
- #define RX_BUFFER_LEN (1024)
- #define RX_FQUAL_ID 0x12 /* Rx Frame Quality information (in double buffer set) */

Bit definitions for register RX_FQUAL.
- #define RX_FQUAL_LEN (8) /* note 64 bit register*/
- #define RX_EQUAL_STD_NOISE_MASK 0x0000FFFFULL /* Standard Deviation of Noise */
- #define RX_EQUAL_STD_NOISE_SHIFT (0)
- #define STD_NOISE_MASK RX_EQUAL_STD_NOISE_MASK
- #define STD_NOISE_SHIFT RX_EQUAL_STD_NOISE_SHIFT
- #define RX_EQUAL_FP_AMPL2_MASK 0xFFFF0000ULL /* First Path Amplitude point 2 - magnitude of 2nd point after Ceiling(FP_Index) */
- #define RX_EQUAL_FP_AMPL2_SHIFT (16)
- #define FP_AMPL2_MASK RX_EQUAL_FP_AMPL2_MASK
- #define FP_AMPL2_SHIFT RX_EQUAL_FP_AMPL2_SHIFT
- #define RX_EQUAL_FP_AMPL3_MASK 0x0000FFFF00000000ULL /* First Path Amplitude point 3 - magnitude of 1st point after Ceiling(FP_Index) */
- #define RX_EQUAL_FP_AMPL3_SHIFT (32)
- #define FP_AMPL3_MASK RX_EQUAL_FP_AMPL3_MASK
- #define FP_AMPL3_SHIFT RX_EQUAL_FP_AMPL3_SHIFT
- #define RX_EQUAL_CIR_MXG_MASK 0xFFFF000000000000ULL /* Channel Impulse Response Max Growth */
- #define RX_EQUAL_CIR_MXG_SHIFT (48)
- #define CIR_MXG_MASK RX_EQUAL_CIR_MXG_MASK
- #define CIR_MXG_SHIFT RX_EQUAL_CIR_MXG_SHIFT
- #define RX_TTCKI_ID 0x13 /* Receiver Time Tracking Interval (in double buffer set) */

Bit definitions for register RX_TTCKI. The value here is the interval over which the timing offset reported in the R← XTOFS field of Register file: 0x14 RX_TTCKO is measured. The clock offset is calculated by dividing RXTTCKI by RXTOFS. The value in RXTTCKI will take just one of two values depending on the PRF: 0x01F00000 @ 16 MHz PRF, and 0x01FC0000 @ 64 MHz PRF.
- #define RX_TTCKI_LEN (4)
- #define RX_TTCKO_ID 0x14 /* Receiver Time Tracking Offset (in double buffer set) */

Bit definitions for register RX_TTCKO.
- #define RX_TTCKO_LEN (5) /* Note 40 bit register */
- #define RX_TTCKO_MASK_32 0xFF07FFFFUL /* Receiver Time Tracking Offset access mask (all unused fields should always be written as zero) */
- #define RX_TTCKO_RXTOFS_MASK 0x0007FFFFUL /* RX time tracking offset. This RXTOFS value is a 19-bit signed quantity*/
- #define RX_TTCKO_RSMPDEL_MASK 0xFF000000UL /* This 8-bit field reports an internal re-sampler delay value */
- #define RX_TTCKO_RCPHASE_MASK 0x7F000000000ULL /* This 7-bit field reports the receive carrier phase adjustment at time the ranging timestamp is made. */
- #define RX_TIME_ID 0x15 /* Receive Message Time of Arrival (in double buffer set) */

Bit definitions for register RX_TIME.
- #define RX_TIME_LLEN (14)
- #define RX_TIME_RX_STAMP_LEN (5) /* read only 5 bytes (the adjusted timestamp (40:0)) */
- #define RX_STAMP_LEN RX_TIME_RX_STAMP_LEN
- #define RX_TIME_RX_STAMP_OFFSET (0) /* byte 0..4 40 bit Reports the fully adjusted time of reception. */
- #define RX_TIME_FP_INDEX_OFFSET (5) /* byte 5..6 16 bit First path index. */
- #define RX_TIME_FP_AMPL1_OFFSET (7) /* byte 7..8 16 bit First Path Amplitude - magnitude of 3rd point after Ceiling(FP_Index) */

- #define RX_TIME_FP_RAWST_OFFSET (9) /* byte 9..13 40 bit Raw Timestamp for the frame */
 - #define REG_16_ID_RESERVED 0x16

Bit definitions for register.
- #define TX_TIME_ID 0x17 /* Transmit Message Time of Sending */
 - #define TX_TIME_LLLEN (10)
 - #define TX_TIME_TX_STAMP_LEN (5) /* 40-bits = 5 bytes */
 - #define TX_STAMP_LEN TX_TIME_TX_STAMP_LEN
 - #define TX_TIME_TX_STAMP_OFFSET (0) /* byte 0..4 40 bit Reports the fully adjusted time of transmission */
 - #define TX_TIME_TX_RAWST_OFFSET (5) /* byte 5..9 40 bit Raw Timestamp for the frame */
 - #define TX_ANTD_ID 0x18 /* 16-bit Delay from Transmit to Antenna */

Bit definitions for register TX_ANTD.
 - #define SYS_STATE_ID 0x19 /* System State information READ ONLY */
 - #define TX_ANTD_OFFSET 0x00
 - #define TX_ANTD_LEN (2)
 - #define ACK_RESP_T_ID 0x1A /* Acknowledgement Time and Response Time */

Bit definitions for register ACK_RESP_T.

 - #define ACK_RESP_T_LEN (4)
 - #define ACK_RESP_T_MASK 0xFF0FFFFFFUL /* Acknowledgement Time and Response access mask */
 - #define ACK_RESP_T_W4R_TIM_OFFSET 0 /* In bytes */
 - #define ACK_RESP_T_W4R_TIM_MASK 0x000FFFFFFUL /* Wait-for-Response turn-around Time 20 bit field */
 - #define W4R_TIM_MASK ACK_RESP_T_W4R_TIM_MASK
 - #define ACK_RESP_T_ACK_TIM_OFFSET 3 /* In bytes */
 - #define ACK_RESP_T_ACK_TIM_MASK 0xFF000000UL /* Auto-Acknowledgement turn-around Time */
 - #define ACK_TIM_MASK ACK_RESP_T_ACK_TIM_MASK
 - #define REG_1B_ID_RESERVED 0x1B

Bit definitions for register 0x1B 0x1C.

 - #define REG_1C_ID_RESERVED 0x1C
 - #define RX_SNIF_ID 0x1D /* Sniff Mode Configuration */

Bit definitions for register RX_SNIF Sniff Mode Configuration or Pulsed Preamble Reception Configuration.

 - #define RX_SNIF_OFFSET 0x00
 - #define RX_SNIF_LEN (4)
 - #define RX_SNIF_MASK 0x0000FF0FUL /* */
 - #define RX_SNIF_SNIF_ONT_MASK 0x0000000FUL /* SNIFF Mode ON time. Specified in units of PAC */
 - #define SNIF_ONT_MASK RX_SNIF_SNIF_ONT_MASK
 - #define RX_SNIF_SNIF_OFFT_MASK 0x0000FF00UL /* SNIFF Mode OFF time specified in units of approximately 1mkS, or 128 system clock cycles.*/
 - #define SNIF_OFFT_MASK RX_SNIF_SNIF_OFFT_MASK
 - #define TX_POWER_ID 0x1E /* TX Power Control */

Bit definitions for register TX_POWER.

 - #define TX_POWER_LEN (4)
 - #define TX_POWER_BOOSTNORM_MASK 0x00000000UL /* This is the normal power setting used for frames that do not fall */
 - #define BOOSTNORM_MASK TX_POWER_BOOSTNORM_MASK
 - #define TX_POWER_BOOSTNORM_SHIFT (0)
 - #define TX_POWER_BOOSTP500_MASK 0x00000000UL /* This value sets the power applied during transmission at the 6.8 Mbps data rate frames that are less than 0.5 ms duration */
 - #define BOOSTP500_MASK TX_POWER_BOOSTP500_MASK
 - #define TX_POWER_BOOSTP500_SHIFT (8)

- #define TX_POWER_BOOSTP250_MASK 0x00000000UL /* This value sets the power applied during transmission at the 6.8 Mbps data rate frames that are less than 0.25 ms duration */
 - #define BOOSTP250_MASK TX_POWER_BOOSTP250_MASK
 - #define TX_POWER_BOOSTP250_SHIFT (16)
 - #define TX_POWER_BOOSTP125_MASK 0x00000000UL /* This value sets the power applied during transmission at the 6.8 Mbps data rate frames that are less than 0.125 ms */
 - #define BOOSTP125_MASK TX_POWER_BOOSTP125_MASK
 - #define TX_POWER_BOOSTP125_SHIFT (24)
 - #define TX_POWER_MAN_DEFAULT 0x0E080222UL
 - #define TX_POWER_TXPOWPHR_MASK 0x0000FF00UL /* This power setting is applied during the transmission of the PHY header (PHR) portion of the frame. */
 - #define TX_POWER_TXPOWSD_MASK 0x00FF0000UL /* This power setting is applied during the transmission of the synchronisation header (SHR) and data portions of the frame. */
 - #define CHAN_CTRL_ID 0x1F /* Channel Control */
- Bit definitions for register CHAN_CTRL.*
- #define CHAN_CTRL_LEN (4)
 - #define CHAN_CTRL_MASK 0xFFFF00FFUL /* Channel Control Register access mask */
 - #define CHAN_CTRL_TX_CHAN_MASK 0x0000000FUL /* Supported channels are 1, 2, 3, 4, 5, and 7.*/
 - #define CHAN_CTRL_TX_CHAN_SHIFT (0) /* Bits 0..3 TX channel number 0-15 selection */
 - #define CHAN_CTRL_RX_CHAN_MASK 0x0000000F0UL
 - #define CHAN_CTRL_RX_CHAN_SHIFT (4) /* Bits 4..7 RX channel number 0-15 selection */
 - #define CHAN_CTRL_RXFPRF_MASK 0x000C0000UL /* Bits 18..19 Specify (Force) RX Pulse Repetition Rate: 00 = 4 MHz, 01 = 16 MHz, 10 = 64MHz. */
 - #define CHAN_CTRL_RXFPRF_SHIFT (18)
 - #define CHAN_CTRL_RXFPRF_4 0x00000000UL /* Specify (Force) RX Pulse Repetition Rate: 00 = 4 MHz, 01 = 16 MHz, 10 = 64MHz. */
 - #define CHAN_CTRL_RXFPRF_16 0x00040000UL /* Specify (Force) RX Pulse Repetition Rate: 00 = 4 MHz, 01 = 16 MHz, 10 = 64MHz. */
 - #define CHAN_CTRL_RXFPRF_64 0x00080000UL /* Specify (Force) RX Pulse Repetition Rate: 00 = 4 MHz, 01 = 16 MHz, 10 = 64MHz. */
 - #define CHAN_CTRL_TX_PCOD_MASK 0x07C00000UL /* Bits 22..26 TX Preamble Code selection, 1 to 24. */
 - #define CHAN_CTRL_TX_PCOD_SHIFT (22)
 - #define CHAN_CTRL_RX_PCOD_MASK 0xF8000000UL /* Bits 27..31 RX Preamble Code selection, 1 to 24. */
 - #define CHAN_CTRL_RX_PCOD_SHIFT (27)
 - #define CHAN_CTRL_DWSFD 0x00020000UL /* Bit 17 This bit enables a non-standard DecaWave proprietary SFD sequence. */
 - #define CHAN_CTRL_DWSFD_SHIFT (17)
 - #define CHAN_CTRL_TNSSFD 0x00100000UL /* Bit 20 Non-standard SFD in the transmitter */
 - #define CHAN_CTRL_TNSSFD_SHIFT (20)
 - #define CHAN_CTRL_RNSSFD 0x00200000UL /* Bit 21 Non-standard SFD in the receiver */
 - #define CHAN_CTRL_RNSSFD_SHIFT (21)
 - #define REG_20_ID_RESERVED 0x20
- Bit definitions for register 0x20.*
- #define USR_SFD_ID 0x21 /* User-specified short/long TX/RX SFD sequences */
- Bit definitions for register USR_SFD Please read User Manual : User defined SFD sequence.*
- #define USR_SFD_LEN (41)
 - #define DW_NS_SFD_LEN_110K 64 /* Decawave non-standard SFD length for 110 kbps */
 - #define DW_NS_SFD_LEN_850K 16 /* Decawave non-standard SFD length for 850 kbps */
 - #define DW_NS_SFD_LEN_6M8 8 /* Decawave non-standard SFD length for 6.8 Mbps */
 - #define REG_22_ID_RESERVED 0x22
- Bit definitions for register.*
- #define AGC_CTRL_ID 0x23 /* Automatic Gain Control configuration */

Bit definitions for register AGC_CTRL Please take care to write to this register as doing so may cause the DW1000 to malfunction.

- #define AGC_CTRL_LEN (32)
- #define AGC_CFG_STS_ID AGC_CTRL_ID
- #define AGC_CTRL1_OFFSET (0x02)
- #define AGC_CTRL1_LEN (2)
- #define AGC_CTRL1_MASK 0x0001 /* access mask to AGC configuration and control register */
- #define AGC_CTRL1_DIS_AM 0x0001 /* Disable AGC Measurement. The DIS_AM bit is set by default. */
- #define AGC_TUNE1_OFFSET (0x04)
- #define AGC_TUNE1_LEN (2)
- #define AGC_TUNE1_MASK 0xFFFF /* It is a 16-bit tuning register for the AGC. */
- #define AGC_TUNE1_16M 0x8870
- #define AGC_TUNE1_64M 0x889B
- #define AGC_TUNE2_OFFSET (0x0C)
- #define AGC_TUNE2_LEN (4)
- #define AGC_TUNE2_MASK 0xFFFFFFFFUL
- #define AGC_TUNE2_VAL 0X2502A907UL
- #define AGC_TUNE3_OFFSET (0x12)
- #define AGC_TUNE3_LEN (2)
- #define AGC_TUNE3_MASK 0xFFFF
- #define AGC_TUNE3_VAL 0X0035
- #define AGC_STAT1_OFFSET (0x1E)
- #define AGC_STAT1_LEN (3)
- #define AGC_STAT1_MASK 0x0FFF
- #define AGC_STAT1_EDG1_MASK 0x00007C0 /* This 5-bit gain value relates to input noise power measurement. */
- #define AGC_STAT1_EDG2_MASK 0x0FF800 /* This 9-bit value relates to the input noise power measurement. */
- #define EXT_SYNC_ID 0x24 /* External synchronisation control */

Bit definitions for register EXT_SYNC.

- #define EXT_SYNC_LEN (12)
 - #define EC_CTRL_OFFSET (0x00)
 - #define EC_CTRL_LEN (4)
 - #define EC_CTRL_MASK 0x00000FFFUL /* sub-register 0x00 is the External clock synchronisation counter configuration register */
 - #define EC_CTRL_OSTM 0x00000001UL /* External transmit synchronisation mode enable */
 - #define EC_CTRL_OSRSM 0x00000002UL /* External receive synchronisation mode enable */
 - #define EC_CTRL_PLLCK 0x04 /* PLL lock detect enable */
 - #define EC_CTRL_OSTRM 0x00000800UL /* External timebase reset mode enable */
 - #define EC_CTRL_WAIT_MASK 0x000007F8UL /* Wait counter used for external transmit synchronisation and external timebase reset */
 - #define EC_RXTC_OFFSET (0x04)
 - #define EC_RXTC_LEN (4)
 - #define EC_RXTC_MASK 0xFFFFFFFFUL /* External clock synchronisation counter captured on RMAR ↔ KER */
 - #define EC_GOLP (0x08)
 - #define EC_GOLP_LEN (4)
 - #define EC_GOLP_MASK 0x0000003FUL /* sub-register 0x08 is the External clock offset to first path 1 GHz counter, EC_GOLP */
 - #define EC_GOLP_OFFSET_EXT_MASK 0x0000003FUL /* This register contains the 1 GHz count from the arrival of the RMARKER and the next edge of the external clock. */
 - #define ACC_MEM_ID 0x25 /* Read access to accumulator data */
- Bit definitions for register ACC_MEM.*
- #define ACC_MEM_LEN (4064)
 - #define GPIO_CTRL_ID 0x26 /* Peripheral register bus 1 access - GPIO control */

Bit definitions for register GPIO_CTRL.

- #define **GPIO_CTRL_LEN** (44)
- #define **GPIO_MODE_OFFSET** 0x00 /* sub-register 0x00 is the GPIO Mode Control Register */
- #define **GPIO_MODE_LEN** (4)
- #define **GPIO_MODE_MASK** 0x00FFFC0UL
- #define **GPIO_MSGP0_MASK** 0x000000C0UL /* Mode Selection for GPIO0/RXOKLED */
- #define **GPIO_MSGP1_MASK** 0x00000300UL /* Mode Selection for GPIO1/SFDLED */
- #define **GPIO_MSGP2_MASK** 0x00000C00UL /* Mode Selection for GPIO2/RXLED */
- #define **GPIO_MSGP3_MASK** 0x00003000UL /* Mode Selection for GPIO3/TXLED */
- #define **GPIO_MSGP4_MASK** 0x0000C000UL /* Mode Selection for GPIO4/EXTPA */
- #define **GPIO_MSGP5_MASK** 0x00030000UL /* Mode Selection for GPIO5/EXTTXE */
- #define **GPIO_MSGP6_MASK** 0x000C0000UL /* Mode Selection for GPIO6/EXTRXE */
- #define **GPIO_MSGP7_MASK** 0x00300000UL /* Mode Selection for SYNC/GPIO7 */
- #define **GPIO_MSGP8_MASK** 0x00C00000UL /* Mode Selection for IRQ/GPIO8 */
- #define **GPIO_PIN2_RXLED** 0x00000400UL /* The pin operates as the RXLED output */
- #define **GPIO_PIN3_TXLED** 0x00001000UL /* The pin operates as the TXLED output */
- #define **GPIO_PIN4_EXTPA** 0x00004000UL /* The pin operates as the EXTPA output */
- #define **GPIO_PIN5_EXTTXE** 0x00010000UL /* The pin operates as the EXTTXE output */
- #define **GPIO_PIN6_EXTRXE** 0x00040000UL /* The pin operates as the EXTRXE output */
- #define **GPIO_DIR_OFFSET** 0x08 /* sub-register 0x08 is the GPIO Direction Control Register */
- #define **GPIO_DIR_LEN** (3)
- #define **GPIO_DIR_MASK** 0x0011FFFFUL
- #define **GxP0** 0x00000001UL /* GPIO0 Only changed if the GxM0 mask bit has a value of 1 for the write operation*/
- #define **GxP1** 0x00000002UL /* GPIO1. (See GDP0). */
- #define **GxP2** 0x00000004UL /* GPIO2. (See GDP0). */
- #define **GxP3** 0x00000008UL /* GPIO3. (See GDP0). */
- #define **GxP4** 0x00000100UL /* GPIO4. (See GDP0). */
- #define **GxP5** 0x00000200UL /* GPIO5. (See GDP0). */
- #define **GxP6** 0x00000400UL /* GPIO6. (See GDP0). */
- #define **GxP7** 0x00000800UL /* GPIO7. (See GDP0). */
- #define **GxP8** 0x00010000UL /* GPIO8 */
- #define **GxM0** 0x00000010UL /* Mask for GPIO0 */
- #define **GxM1** 0x00000020UL /* Mask for GPIO1. (See GDM0). */
- #define **GxM2** 0x00000040UL /* Mask for GPIO2. (See GDM0). */
- #define **GxM3** 0x00000080UL /* Mask for GPIO3. (See GDM0). */
- #define **GxM4** 0x00001000UL /* Mask for GPIO4. (See GDM0). */
- #define **GxM5** 0x00002000UL /* Mask for GPIO5. (See GDM0). */
- #define **GxM6** 0x00004000UL /* Mask for GPIO6. (See GDM0). */
- #define **GxM7** 0x00008000UL /* Mask for GPIO7. (See GDM0). */
- #define **GxM8** 0x00100000UL /* Mask for GPIO8. (See GDM0). */
- #define **GDP0 GxP0** /* Direction Selection for GPIO0. 1 = input, 0 = output. Only changed if the **GDM0** mask bit has a value of 1 for the write operation*/
- #define **GDP1 GxP1** /* Direction Selection for GPIO1. (See **GDP0**). */
- #define **GDP2 GxP2** /* Direction Selection for GPIO2. (See **GDP0**). */
- #define **GDP3 GxP3** /* Direction Selection for GPIO3. (See **GDP0**). */
- #define **GDP4 GxP4** /* Direction Selection for GPIO4. (See **GDP0**). */
- #define **GDP5 GxP5** /* Direction Selection for GPIO5. (See **GDP0**). */
- #define **GDP6 GxP6** /* Direction Selection for GPIO6. (See **GDP0**). */
- #define **GDP7 GxP7** /* Direction Selection for GPIO7. (See **GDP0**). */
- #define **GDP8 GxP8** /* Direction Selection for GPIO8 */
- #define **GDM0 GxM0** /* Mask for setting the direction of GPIO0 */
- #define **GDM1 GxM1** /* Mask for setting the direction of GPIO1. (See **GDM0**). */
- #define **GDM2 GxM2** /* Mask for setting the direction of GPIO2. (See **GDM0**). */
- #define **GDM3 GxM3** /* Mask for setting the direction of GPIO3. (See **GDM0**). */

- #define GDM4_GxM4 /* Mask for setting the direction of GPIO4. (See [GDM0](#)). */
- #define GDM5_GxM5 /* Mask for setting the direction of GPIO5. (See [GDM0](#)). */
- #define GDM6_GxM6 /* Mask for setting the direction of GPIO6. (See [GDM0](#)). */
- #define GDM7_GxM7 /* Mask for setting the direction of GPIO7. (See [GDM0](#)). */
- #define GDM8_GxM8 /* Mask for setting the direction of GPIO8. (See [GDM0](#)). */
- #define GPIO_DOUT_OFFSET 0x0C /* sub-register 0x0C is the GPIO data output register. */
- #define GPIO_DOUT_LEN (3)
- #define GPIO_DOUT_MASK GPIO_DIR_MASK
- #define GPIO_IRQE_OFFSET 0x10 /* sub-register 0x10 is the GPIO interrupt enable register */
- #define GPIO_IRQE_LEN (4)
- #define GPIO_IRQE_MASK 0x0000001FFUL
- #define GIRQx0 0x00000001UL /* IRQ bit0 */
- #define GIRQx1 0x00000002UL /* IRQ bit1 */
- #define GIRQx2 0x00000004UL /* IRQ bit2 */
- #define GIRQx3 0x00000008UL /* IRQ bit3 */
- #define GIRQx4 0x00000010UL /* IRQ bit4 */
- #define GIRQx5 0x00000020UL /* IRQ bit5 */
- #define GIRQx6 0x00000040UL /* IRQ bit6 */
- #define GIRQx7 0x00000080UL /* IRQ bit7 */
- #define GIRQx8 0x00000100UL /* IRQ bit8 */
- #define GIRQE0_GIRQx0 /* GPIO IRQ Enable for GPIO0 input. Value 1 = enable, 0 = disable*/
- #define GIRQE1_GIRQx1 /* */
- #define GIRQE2_GIRQx2 /* */
- #define GIRQE3_GIRQx3 /* */
- #define GIRQE4_GIRQx4 /* */
- #define GIRQE5_GIRQx5 /* */
- #define GIRQE6_GIRQx6 /* */
- #define GIRQE7_GIRQx7 /* */
- #define GIRQE8_GIRQx8 /* Value 1 = enable, 0 = disable */
- #define GPIO_ISEN_OFFSET 0x14 /* sub-register 0x14 is the GPIO interrupt sense selection register */
- #define GPIO_ISEN_LEN (4)
- #define GPIO_ISEN_MASK GPIO_IRQE_MASK
- #define GISEN0_GIRQx0 /* GPIO IRQ Sense selection GPIO0 input. Value 0 = High or Rising-Edge, 1 = Low or falling-edge.*/
- #define GISEN1_GIRQx1 /* */
- #define GISEN2_GIRQx2 /* */
- #define GISEN3_GIRQx3 /* */
- #define GISEN4_GIRQx4 /* */
- #define GISEN5_GIRQx5 /* */
- #define GISEN6_GIRQx6 /* */
- #define GISEN7_GIRQx7 /* */
- #define GISEN8_GIRQx8 /* Value 0 = High or Rising-Edge, 1 = Low or falling-edge */
- #define GPIO_IMODE_OFFSET 0x18 /* sub-register 0x18 is the GPIO interrupt mode selection register */
- #define GPIO_IMODE_LEN (4)
- #define GPIO_IMODE_MASK GPIO_IRQE_MASK
- #define GIMOD0_GIRQx0 /* GPIO IRQ Mode selection for GPIO0 input. Value 0 = Level sensitive interrupt. Value 1 = Edge triggered interrupt */
- #define GIMOD1_GIRQx1 /* */
- #define GIMOD2_GIRQx2 /* */
- #define GIMOD3_GIRQx3 /* */
- #define GIMOD4_GIRQx4 /* */
- #define GIMOD5_GIRQx5 /* */
- #define GIMOD6_GIRQx6 /* */
- #define GIMOD7_GIRQx7 /* */
- #define GIMOD8_GIRQx8 /* Value 0 = Level, 1 = Edge. */

- #define GPIO_IBES_OFFSET 0x1C /* sub-register 0x1C is the GPIO interrupt Both Edge selection register */
 - #define GPIO_IBES_LEN (4)
 - #define GPIO_IBES_MASK GPIO_IRQE_MASK /* */
 - #define GIBES0 GIRQx0 /* GPIO IRQ Both Edge selection for GPIO0 input. Value 0 = GPIO_IMODE register selects the edge. Value 1 = Both edges trigger the interrupt. */
 - #define GIBES1 GIRQx1 /* */
 - #define GIBES2 GIRQx2 /* */
 - #define GIBES3 GIRQx3 /* */
 - #define GIBES4 GIRQx4 /* */
 - #define GIBES5 GIRQx5 /* */
 - #define GIBES6 GIRQx6 /* */
 - #define GIBES7 GIRQx7 /* */
 - #define GIBES8 GIRQx8 /* Value 0 = use GPIO_IMODE, 1 = Both Edges */
 - #define GPIO_ICLR_OFFSET 0x20 /* sub-register 0x20 is the GPIO interrupt clear register */
 - #define GPIO_ICLR_LEN (4)
 - #define GPIO_ICLR_MASK GPIO_IRQE_MASK /* */
 - #define GICLR0 GIRQx0 /* GPIO IRQ latch clear for GPIO0 input. Write 1 to clear the GPIO0 interrupt latch. Writing 0 has no effect. Reading returns zero */
 - #define GICLR1 GIRQx1 /* */
 - #define GICLR2 GIRQx2 /* */
 - #define GICLR3 GIRQx3 /* */
 - #define GICLR4 GIRQx4 /* */
 - #define GICLR5 GIRQx5 /* */
 - #define GICLR6 GIRQx6 /* */
 - #define GICLR7 GIRQx7 /* */
 - #define GICLR8 GIRQx8 /* Write 1 to clear the interrupt latch */
 - #define GPIO_IDBE_OFFSET 0x24 /* sub-register 0x24 is the GPIO interrupt de-bounce enable register */
 - #define GPIO_IDBE_LEN (4)
 - #define GPIO_IDBE_MASK GPIO_IRQE_MASK
 - #define GIDBE0 GIRQx0 /* GPIO IRQ de-bounce enable for GPIO0. Value 1 = de-bounce enabled. Value 0 = de-bounce disabled */
 - #define GIDBE1 GIRQx1 /* */
 - #define GIDBE2 GIRQx2 /* */
 - #define GIDBE3 GIRQx3 /* */
 - #define GIDBE4 GIRQx4 /* */
 - #define GIDBE5 GIRQx5 /* */
 - #define GIDBE6 GIRQx6 /* */
 - #define GIDBE7 GIRQx7 /* */
 - #define GIDBE8 GIRQx8 /* Value 1 = de-bounce enabled, 0 = de-bounce disabled */
 - #define GPIO_RAW_OFFSET 0x28 /* sub-register 0x28 allows the raw state of the GPIO pin to be read. */
 - #define GPIO_RAW_LEN (4)
 - #define GPIO_RAW_MASK GPIO_IRQE_MASK
 - #define GRAWP0 GIRQx0 /* This bit reflects the raw state of GPIO0 */
 - #define GRAWP1 GIRQx1 /* */
 - #define GRAWP2 GIRQx2 /* */
 - #define GRAWP3 GIRQx3 /* */
 - #define GRAWP4 GIRQx4 /* */
 - #define GRAWP5 GIRQx5 /* */
 - #define GRAWP6 GIRQx6 /* */
 - #define GRAWP7 GIRQx7 /* */
 - #define GRAWP8 GIRQx8 /* This bit reflects the raw state of GPIO8 */
 - #define DRX_CONF_ID 0x27 /* Digital Receiver configuration */
- Bit definitions for register DRX_CONF Digital Receiver configuration block.*
- #define DRX_CONF_LEN (44)

- #define DRX_TUNE0b_OFFSET (0x02) /* sub-register 0x02 is a 16-bit tuning register. */
- #define DRX_TUNE0b_LEN (2)
- #define DRX_TUNE0b_MASK 0xFFFF /* 7.2.40.2 Sub-Register 0x27:02 DRX_TUNE0b */
- #define DRX_TUNE0b_110K_STD 0x000A
- #define DRX_TUNE0b_110K_NSTD 0x0016
- #define DRX_TUNE0b_850K_STD 0x0001
- #define DRX_TUNE0b_850K_NSTD 0x0006
- #define DRX_TUNE0b_6M8_STD 0x0001
- #define DRX_TUNE0b_6M8_NSTD 0x0002
- #define DRX_TUNE1a_OFFSET 0x04 /* 7.2.40.3 Sub-Register 0x27:04 DRX_TUNE1a */
- #define DRX_TUNE1a_LEN (2)
- #define DRX_TUNE1a_MASK 0xFFFF
- #define DRX_TUNE1a_PRF16 0x0087
- #define DRX_TUNE1a_PRF64 0x008D
- #define DRX_TUNE1b_OFFSET 0x06 /* 7.2.40.4 Sub-Register 0x27:06 DRX_TUNE1b */
- #define DRX_TUNE1b_LEN (2)
- #define DRX_TUNE1b_MASK 0xFFFF
- #define DRX_TUNE1b_110K 0x0064
- #define DRX_TUNE1b_850K_6M8 0x0020
- #define DRX_TUNE1b_6M8_PRE64 0x0010
- #define DRX_TUNE2_OFFSET 0x08 /* 7.2.40.5 Sub-Register 0x27:08 DRX_TUNE2 */
- #define DRX_TUNE2_LEN (4)
- #define DRX_TUNE2_MASK 0xFFFFFFFFUL
- #define DRX_TUNE2_PRF16_PAC8 0x311A002DUL
- #define DRX_TUNE2_PRF16_PAC16 0x331A0052UL
- #define DRX_TUNE2_PRF16_PAC32 0x351A009AUL
- #define DRX_TUNE2_PRF16_PAC64 0x371A011DUL
- #define DRX_TUNE2_PRF64_PAC8 0x313B006BUL
- #define DRX_TUNE2_PRF64_PAC16 0x333B00BEUL
- #define DRX_TUNE2_PRF64_PAC32 0x353B015EUL
- #define DRX_TUNE2_PRF64_PAC64 0x373B0296UL
- #define DRX_SFDTOC_OFFSET 0x20 /* 7.2.40.7 Sub-Register 0x27:20 DRX_SFDTOC */
- #define DRX_SFDTOC_LEN (2)
- #define DRX_SFDTOC_MASK 0xFFFF
- #define DRX_PRETOC_OFFSET 0x24 /* 7.2.40.9 Sub-Register 0x27:24 DRX_PRETOC */
- #define DRX_PRETOC_LEN (2)
- #define DRX_PRETOC_MASK 0xFFFF
- #define DRX_TUNE4H_OFFSET 0x26 /* 7.2.40.10 Sub-Register 0x27:26 DRX_TUNE4H */
- #define DRX_TUNE4H_LEN (2)
- #define DRX_TUNE4H_MASK 0xFFFF
- #define DRX_TUNE4H_PRE64 0x0010
- #define DRX_TUNE4H_PRE128PLUS 0x0028
- #define DRX_CARRIER_INT_OFFSET 0x28
- #define DRX_CARRIER_INT_LEN (3)
- #define DRX_CARRIER_INT_MASK 0x001FFFFF
- #define RF_CONF_ID 0x28 /* Analog RF Configuration */

Bit definitions for register RF_CONF Analog RF Configuration block Refer to section 7.2.41 Register file: 0x28 Analog RF configuration block.

- #define RF_CONF_LEN (58)
- #define RF_CONF_TXEN_MASK 0x00400000UL /* TX enable */
- #define RF_CONF_RXEN_MASK 0x00200000UL /* RX enable */
- #define RF_CONF_TXPOW_MASK 0x001F0000UL /* turn on power all LDOs */
- #define RF_CONF_PLLEN_MASK 0x0000E000UL /* enable PLLs */
- #define RF_CONF_PGMIXBIASEN_MASK 0x0000A700UL /* Enable TX mixer bias and pulse gen */

- #define RF_CONF_TXBLOCKSEN_MASK 0x00001F00UL /* enable TX blocks */
- #define RF_CONF_TXPLLPOWEN_MASK (RF_CONF_PLLEN_MASK | RF_CONF_TXPOW_MASK)
- #define RF_CONF_TXALLEN_MASK (RF_CONF_TXEN_MASK | RF_CONF_TXPOW_MASK | RF_CONF_PLLEN_MASK | RF_CONF_TXBLOCKSEN_MASK)
- #define RF_RXCTRLH_OFFSET 0x0B /* Analog RX Control Register */
- #define RF_RXCTRLH_LEN (1)
- #define RF_RXCTRLH_NBW 0xD8 /* RXCTRLH value for narrow bandwidth channels */
- #define RF_RXCTRLH_WBW 0xBC /* RXCTRLH value for wide bandwidth channels */
- #define RF_TXCTRL_OFFSET 0x0C /* Analog TX Control Register */
- #define RF_TXCTRL_LEN (4)
- #define RF_TXCTRL_TXMTUNE_MASK 0x000001E0UL /* Transmit mixer tuning register */
- #define RF_TXCTRL_TXTXMQ_MASK 0x00000E00UL /* Transmit mixer Q-factor tuning register */
- #define RF_TXCTRL_CH1 0x00005C40UL /* 32-bit value to program to Sub-Register 0x28:0C RF_TXCTRL */
- #define RF_TXCTRL_CH2 0x00045CA0UL /* 32-bit value to program to Sub-Register 0x28:0C RF_TXCTRL */
- #define RF_TXCTRL_CH3 0x00086CC0UL /* 32-bit value to program to Sub-Register 0x28:0C RF_TXCTRL */
- #define RF_TXCTRL_CH4 0x00045C80UL /* 32-bit value to program to Sub-Register 0x28:0C RF_TXCTRL */
- #define RF_TXCTRL_CH5 0x001E3FE0UL /* 32-bit value to program to Sub-Register 0x28:0C RF_TXCTRL */
- #define RF_TXCTRL_CH7 0x001E7DE0UL /* 32-bit value to program to Sub-Register 0x28:0C RF_TXCTRL */
- #define RF_STATUS_OFFSET 0x2C
- #define REG_29_ID_RESERVED 0x29

Bit definitions for register.

- #define TX_CAL_ID 0x2A /* Transmitter calibration block */

Bit definitions for register TX_CAL Refer to section 7.2.43 Register file: 0x2A Transmitter Calibration block.

- #define TX_CAL_LEN (52)
- #define TC_SARL_SAR_C (0) /* SAR control */
- #define TC_SARL_SAR_LVBAT_OFFSET (3) /* Latest SAR reading for Voltage level */
- #define TC_SARL_SAR_LTEMP_OFFSET (4) /* Latest SAR reading for Temperature level */
- #define TC_SARW_SAR_WTEMP_OFFSET 0x06 /* SAR reading of Temperature level taken at last wakeup event */
- #define TC_SARW_SAR_WVBAT_OFFSET 0x07 /* SAR reading of Voltage level taken at last wakeup event */
- #define TC_PGCCTRL_OFFSET 0x08 /* Pulse Generator Calibration control */
- #define TC_PGCCTRL_LEN (1)
- #define TC_PGCCTRL_CALSTART 0x01 /* Start PG cal procedure */
- #define TC_PGCCTRL_AUTOCAL 0x02 /* Starts a PG autocalibration loop */
- #define TC_PGCCTRL_TMEAS_MASK 0x3C /* Mask to retrieve number of clock cycles over which to run PG cal counter */
- #define TC_PGCCTRL_ON_TX 0x40 /* Perform autocal on each TX enable */
- #define TC_PGCCTRL_DIR_CONV 0x80 /* Direction (converging) of autocal binary search */
- #define TC_PGCAL_STATUS_OFFSET 0x09 /* Status register from PG calibration block */
- #define TC_PGCAL_STATUS_LEN (1)
- #define TC_PGCAL_STATUS_DELAY_MASK 0xFFFF /* Mask to retrieve PG delay count from calibration */
- #define TC_PGDELAY_OFFSET 0x0B /* Transmitter Calibration Pulse Generator Delay */
- #define TC_PGDELAY_LEN (1)
- #define TC_PGDELAY_CH1 0xC9 /* Recommended value for channel 1 */
- #define TC_PGDELAY_CH2 0xC2 /* Recommended value for channel 2 */
- #define TC_PGDELAY_CH3 0xC5 /* Recommended value for channel 3 */
- #define TC_PGDELAY_CH4 0x95 /* Recommended value for channel 4 */
- #define TC_PGDELAY_CH5 0xC0 /* Recommended value for channel 5 */

- #define **TC_PGDELAY_CH7** 0x93 /* Recommended value for channel 7 */
- #define **TC_PGTTEST_OFFSET** 0x0C /* Transmitter Calibration Pulse Generator Test */
- #define **TC_PGTTEST_LEN** (1)
- #define **TC_PGTTEST_NORMAL** 0x00 /* Normal operation */
- #define **TC_PGTTEST_CW** 0x13 /* Continuous Wave (CW) Test Mode */
- #define **FS_CTRL_ID** 0x2B /* Frequency synthesiser control block */

Bit definitions for register Refer to section 7.2.44 Register file: 0x2B Frequency synthesiser control block.

- #define **FS_CTRL_LEN** (21)
- #define **FS_RES1_OFFSET** 0x00 /* reserved area. Please take care not to write to this area as doing so may cause the DW1000 to malfunction. */
- #define **FS_RES1_LEN** (7)
- #define **FS_PLLCFG_OFFSET** 0x07 /* Frequency synthesiser PLL configuration */
- #define **FS_PLLCFG_LEN** (5)
- #define **FS_PLLCFG_CH1** 0x09000407UL /* Operating Channel 1 */
- #define **FS_PLLCFG_CH2** 0x08400508UL /* Operating Channel 2 */
- #define **FS_PLLCFG_CH3** 0x08401009UL /* Operating Channel 3 */
- #define **FS_PLLCFG_CH4 FS_PLLCFG_CH2** /* Operating Channel 4 (same as 2) */
- #define **FS_PLLCFG_CH5** 0x0800041DUL /* Operating Channel 5 */
- #define **FS_PLLCFG_CH7 FS_PLLCFG_CH5** /* Operating Channel 7 (same as 5) */
- #define **FS_PLLTUNE_OFFSET** 0x0B /* Frequency synthesiser PLL Tuning */
- #define **FS_PLLTUNE_LEN** (1)
- #define **FS_PLLTUNE_CH1** 0x1E /* Operating Channel 1 */
- #define **FS_PLLTUNE_CH2** 0x26 /* Operating Channel 2 */
- #define **FS_PLLTUNE_CH3** 0x56 /* Operating Channel 3 */
- #define **FS_PLLTUNE_CH4 FS_PLLTUNE_CH2** /* Operating Channel 4 (same as 2) */
- #define **FS_PLLTUNE_CH5** 0xBE /* Operating Channel 5 */
- #define **FS_PLLTUNE_CH7 FS_PLLTUNE_CH5** /* Operating Channel 7 (same as 5) */
- #define **FS_RES2_OFFSET** 0x0C /* reserved area. Please take care not to write to this area as doing so may cause the DW1000 to malfunction. */
- #define **FS_RES2_LEN** (2)
- #define **FS_XTALT_OFFSET** 0x0E /* Frequency synthesiser Crystal trim */
- #define **FS_XTALT_LEN** (1)
- #define **FS_XTALT_MASK** 0x1F /* Crystal Trim. Crystals may be trimmed using this register setting to tune out errors, see 8.1 IC Calibration Crystal Oscillator Trim. */
- #define **FS_XTALT_MIDRANGE** 0x10
- #define **FS_RES3_OFFSET** 0x0F /* reserved area. Please take care not to write to this area as doing so may cause the DW1000 to malfunction. */
- #define **FS_RES3_LEN** (6)
- #define **AON_ID** 0x2C /* Always-On register set */

Bit definitions for register.

- #define **AON_LEN** (12)
- #define **AON_WCFG_OFFSET** 0x00 /* used to control what the DW1000 IC does as it wakes up from low-power SLEEP or DEEPSLEEPstates. */
- #define **AON_WCFG_LEN** (2)
- #define **AON_WCFG_MASK** 0x09CB /* access mask to AON_WCFG register*/
- #define **AON_WCFG_ONW_RADC** 0x0001 /* On Wake-up Run the (temperature and voltage) Analog-to-Digital Convertors */
- #define **AON_WCFG_ONW_RX** 0x0002 /* On Wake-up turn on the Receiver */
- #define **AON_WCFG_ONW_LEUI** 0x0008 /* On Wake-up load the EUI from OTP memory into Register file: 0x01 Extended Unique Identifier. */
- #define **AON_WCFG_ONW_LDC** 0x0040 /* On Wake-up load configurations from the AON memory into the host interface register set */
- #define **AON_WCFG_ONW_L64P** 0x0080 /* On Wake-up load the Length64 receiver operating parameter set */

- #define AON_WCFG_PRES_SLEEP 0x0100 /* Preserve Sleep. This bit determines what the DW1000 does with respect to the ARXSLP and ATXSLP sleep controls */
- #define AON_WCFG_ONW_LLDE 0x0800 /* On Wake-up load the LDE microcode. */
- #define AON_WCFG_ONW_LLDO 0x1000 /* On Wake-up load the LDO tune value. */
- #define AON_CTRL_OFFSET 0x02 /* The bits in this register in general cause direct activity within the AON block with respect to the stored AON memory */
- #define AON_CTRL_LEN (1)
- #define AON_CTRL_MASK 0x8F /* access mask to AON_CTRL register */
- #define AON_CTRL_RESTORE 0x01 /* When this bit is set the DW1000 will copy the user configurations from the AON memory to the host interface register set. */
- #define AON_CTRL_SAVE 0x02 /* When this bit is set the DW1000 will copy the user configurations from the host interface register set into the AON memory */
- #define AON_CTRL_UPL_CFG 0x04 /* Upload the AON block configurations to the AON */
- #define AON_CTRL_DCA_READ 0x08 /* Direct AON memory access read */
- #define AON_CTRL_DCA_ENAB 0x80 /* Direct AON memory access enable bit */
- #define AON_RDAT_OFFSET 0x03 /* AON Direct Access Read Data Result */
- #define AON_RDAT_LEN (1)
- #define AON_ADDR_OFFSET 0x04 /* AON Direct Access Address */
- #define AON_ADDR_LEN (1)
- #define AON_ADDR_LPOSC_CAL_0 117 /* Address of low-power oscillator calibration value (lower byte) */
- #define AON_ADDR_LPOSC_CAL_1 118 /* Address of low-power oscillator calibration value (lower byte) */
- #define AON_CFG0_OFFSET 0x06 /* 32-bit configuration register for the always on block. */
- #define AON_CFG0_LEN (4)
- #define AON_CFG0_SLEEP_EN 0x00000001UL /* This is the sleep enable configuration bit */
- #define AON_CFG0_WAKE_PIN 0x00000002UL /* Wake using WAKEUP pin */
- #define AON_CFG0_WAKE_SPI 0x00000004UL /* Wake using SPI access SPICSn */
- #define AON_CFG0_WAKE_CNT 0x00000008UL /* Wake when sleep counter elapses */
- #define AON_CFG0_LPDIV_EN 0x00000010UL /* Low power divider enable configuration */
- #define AON_CFG0_LPCLKDIVA_MASK 0x0000FFE0UL /* divider count for dividing the raw DW1000 X←TAL oscillator frequency to set an LP clock frequency */
- #define AON_CFG0_LPCLKDIVA_SHIFT (5)
- #define AON_CFG0_SLEEP_TIM 0xFFFF0000UL /* Sleep time. This field configures the sleep time count elapse value */
- #define AON_CFG0_SLEEP_SHIFT (16)
- #define AON_CFG0_SLEEP_TIM_OFFSET 2 /* In bytes */
- #define AON_CFG1_OFFSET 0x0A
- #define AON_CFG1_LEN (2)
- #define AON_CFG1_MASK 0x0007 /* aceess mask to AON_CFG1 */
- #define AON_CFG1_SLEEP_CEN 0x0001 /* This bit enables the sleep counter */
- #define AON_CFG1_SMXX 0x0002 /* This bit needs to be set to 0 for correct operation in the SLEEP state within the DW1000 */
- #define AON_CFG1_LPOSC_CAL 0x0004 /* This bit enables the calibration function that measures the period of the ICs internal low powered oscillator */
- #define OTP_IF_ID 0x2D /* One Time Programmable Memory Interface */

Bit definitions for register OTP_IF Refer to section 7.2.46 Register file: 0x2D OTP Memory Interface.

- #define OTP_IF_LEN (18)
- #define OTP_WDAT 0x00 /* 32-bit register. The data value to be programmed into an OTP location */
- #define OTP_WDAT_LEN (4)
- #define OTP_ADDR 0x04 /* 16-bit register used to select the address within the OTP memory block */
- #define OTP_ADDR_LEN (2)
- #define OTP_ADDR_MASK 0x07FF /* This 11-bit field specifies the address within OTP memory that will be accessed read or written. */
- #define OTP_CTRL 0x06 /* used to control the operation of the OTP memory */
- #define OTP_CTRL_LEN (2)
- #define OTP_CTRL_MASK 0x8002

- #define `OTP_CTRL_OTPRDEN` 0x0001 /* This bit forces the OTP into manual read mode */
- #define `OTP_CTRL_OTPREAD` 0x0002 /* This bit commands a read operation from the address specified in the OTP_ADDR register */
- #define `OTP_CTRL_LDELOAD` 0x8000 /* This bit forces a load of LDE microcode */
- #define `OTP_CTRL_OTPPROG` 0x0040 /* Setting this bit will cause the contents of OTP_WDAT to be written to OTP_ADDR. */
- #define `OTP_STAT` 0x08
- #define `OTP_STAT_LEN` (2)
- #define `OTP_STAT_MASK` 0x0003
- #define `OTP_STAT_OTPPRGD` 0x0001 /* OTP Programming Done */
- #define `OTP_STAT_OTPVPOK` 0x0002 /* OTP Programming Voltage OK */
- #define `OTP_RDAT` 0x0A /* 32-bit register. The data value read from an OTP location will appear here */
- #define `OTP_RDAT_LEN` (4)
- #define `OTP_SRDAT` 0x0E /* 32-bit register. The data value stored in the OTP SR (0x400) location will appear here after power up */
- #define `OTP_SRDAT_LEN` (4)
- #define `OTP_SF` 0x12 /*8-bit special function register used to select and load special receiver operational parameter */
- #define `OTP_SF_LEN` (1)
- #define `OTP_SF_MASK` 0x63
- #define `OTP_SF_OPS_KICK` 0x01 /* This bit when set initiates a load of the operating parameter set selected by the OPS_SEL */
- #define `OTP_SF_LDO_KICK` 0x02 /* This bit when set initiates a load of the LDO tune code */
- #define `OTP_SF_OPS_SEL_SHFT` 5
- #define `OTP_SF_OPS_SEL_MASK` 0x60
- #define `OTP_SF_OPS_SEL_L64` 0x00 /* Operating parameter set selection: Length64 */
- #define `OTP_SF_OPS_SEL_TIGHT` 0x20 /* Operating parameter set selection: Tight */
- #define `LDE_IF_ID` 0x2E /* Leading edge detection control block */

*Bit definitions for register LDE_IF Refer to section 7.2.47 Register file: 0x2E Leading Edge Detection Interface P←
LEASE NOTE: Other areas within the address space of Register file: 0x2E Leading Edge Detection Interface are reserved. To ensure proper operation of the LDE algorithm (i.e. to avoid loss of performance or a malfunction), care must be taken not to write to any byte locations other than those defined in the sub-sections below.*

- #define `LDE_IF_LEN` (0)
- #define `LDE_THRESH_OFFSET` 0x0000 /* 16-bit status register reporting the threshold that was used to find the first path */
- #define `LDE_THRESH_LEN` (2)
- #define `LDE_CFG1_OFFSET` 0x0806 /*8-bit configuration register*/
- #define `LDE_CFG1_LEN` (1)
- #define `LDE_CFG1_NSTDEV_MASK` 0x1F /* Number of Standard Deviations mask. */
- #define `LDE_CFG1_PMULT_MASK` 0xE0 /* Peak Multiplier mask. */
- #define `LDE_PPINDEX_OFFSET` 0x1000 /* reporting the position within the accumulator that the LDE algorithm has determined to contain the maximum */
- #define `LDE_PPINDEX_LEN` (2)
- #define `LDE_PPAMPL_OFFSET` 0x1002 /* reporting the magnitude of the peak signal seen in the accumulator data memory */
- #define `LDE_PPAMPL_LEN` (2)
- #define `LDE_RXANTD_OFFSET` 0x1804 /* 16-bit configuration register for setting the receive antenna delay */
- #define `LDE_RXANTD_LEN` (2)
- #define `LDE_CFG2_OFFSET` 0x1806 /* 16-bit LDE configuration tuning register */
- #define `LDE_CFG2_LEN` (2)
- #define `LDE_REPC_OFFSET` 0x2804 /* 16-bit configuration register for setting the replica avoidance coefficient */
- #define `LDE_REPC_LEN` (2)
- #define `LDE_REPC_PCODE_1` 0x5998

- #define LDE_REPCPCODE_2 0x5998
 - #define LDE_REPCPCODE_3 0x51EA
 - #define LDE_REPCPCODE_4 0x428E
 - #define LDE_REPCPCODE_5 0x451E
 - #define LDE_REPCPCODE_6 0x2E14
 - #define LDE_REPCPCODE_7 0x8000
 - #define LDE_REPCPCODE_8 0x51EA
 - #define LDE_REPCPCODE_9 0x28F4
 - #define LDE_REPCPCODE_10 0x3332
 - #define LDE_REPCPCODE_11 0x3AE0
 - #define LDE_REPCPCODE_12 0x3D70
 - #define LDE_REPCPCODE_13 0x3AE0
 - #define LDE_REPCPCODE_14 0x35C2
 - #define LDE_REPCPCODE_15 0x2B84
 - #define LDE_REPCPCODE_16 0x35C2
 - #define LDE_REPCPCODE_17 0x3332
 - #define LDE_REPCPCODE_18 0x35C2
 - #define LDE_REPCPCODE_19 0x35C2
 - #define LDE_REPCPCODE_20 0x47AE
 - #define LDE_REPCPCODE_21 0x3AE0
 - #define LDE_REPCPCODE_22 0x3850
 - #define LDE_REPCPCODE_23 0x30A2
 - #define LDE_REPCPCODE_24 0x3850
 - #define DIG_DIAG_ID 0x2F /* Digital Diagnostics Interface */
- Bit definitions for register DIG_DIAG Digital Diagnostics interface. It contains a number of sub-registers that give diagnostics information.*
- #define DIG_DIAG_LEN (41)
 - #define EVC_CTRL_OFFSET 0x00 /* Event Counter Control */
 - #define EVC_CTRL_LEN (4)
 - #define EVC_CTRL_MASK 0x00000003UL/* access mask to Register for bits should always be set to zero to avoid any malfunction of the device. */
 - #define EVC_EN 0x00000001UL/* Event Counters Enable bit */
 - #define EVC_CLR 0x00000002UL
 - #define EVC_PHE_OFFSET 0x04 /* PHR Error Event Counter */
 - #define EVC_PHE_LEN (2)
 - #define EVC_PHE_MASK 0xFFFF
 - #define EVC_RSE_OFFSET 0x06 /* Reed Solomon decoder (Frame Sync Loss) Error Event Counter */
 - #define EVC_RSE_LEN (2)
 - #define EVC_RSE_MASK 0xFFFF
 - #define EVC_FCG_OFFSET 0x08 /* The EVC_FCG field is a 12-bit counter of the frames received with good CRC/FCS sequence. */
 - #define EVC_FCG_LEN (2)
 - #define EVC_FCG_MASK 0xFFFF
 - #define EVC_FCE_OFFSET 0x0A /* The EVC_FCE field is a 12-bit counter of the frames received with bad CRC/FCS sequence. */
 - #define EVC_FCE_LEN (2)
 - #define EVC_FCE_MASK 0xFFFF
 - #define EVC_FFR_OFFSET 0x0C /* The EVC_FFR field is a 12-bit counter of the frames rejected by the receive frame filtering function. */
 - #define EVC_FFR_LEN (2)
 - #define EVC_FFR_MASK 0xFFFF
 - #define EVC_OVR_OFFSET 0x0E /* The EVC_OVR field is a 12-bit counter of receive overrun events */
 - #define EVC_OVR_LEN (2)
 - #define EVC_OVR_MASK 0xFFFF
 - #define EVC_STO_OFFSET 0x10 /* The EVC_STO field is a 12-bit counter of SFD Timeout Error events */

- #define EVC_OVR_LEN (2)
- #define EVC_OVR_MASK 0x0FFF
- #define EVC_PTO_OFFSET 0x12 /* The EVC_PTO field is a 12-bit counter of Preamble detection Timeout events */
- #define EVC_PTO_LEN (2)
- #define EVC_PTO_MASK 0x0FFF
- #define EVC_FWTO_OFFSET 0x14 /* The EVC_FWTO field is a 12-bit counter of receive frame wait timeout events */
- #define EVC_FWTO_LEN (2)
- #define EVC_FWTO_MASK 0x0FFF
- #define EVC_TXFS_OFFSET 0x16 /* The EVC_TXFS field is a 12-bit counter of transmit frames sent. This is incremented every time a frame is sent */
- #define EVC_TXFS_LEN (2)
- #define EVC_TXFS_MASK 0x0FFF
- #define EVC_HPW_OFFSET 0x18 /* The EVC_HPW field is a 12-bit counter of Half Period Warnings. */
- #define EVC_HPW_LEN (2)
- #define EVC_HPW_MASK 0x0FFF
- #define EVC_TPW_OFFSET 0x1A /* The EVC_TPW field is a 12-bit counter of Transmitter Power-Up Warnings. */
- #define EVC_TPW_LEN (2)
- #define EVC_TPW_MASK 0x0FFF
- #define EVC_RES1_OFFSET 0x1C /* Please take care not to write to this register as doing so may cause the DW1000 to malfunction. */
- #define DIAG_TMC_OFFSET 0x24
- #define DIAG_TMC_LEN (2)
- #define DIAG_TMC_MASK 0x0010
- #define DIAG_TMC_TX_PSTM 0x0010 /* This test mode is provided to help support regulatory approvals spectral testing. When the TX_PSTM bit is set it enables a repeating transmission of the data from the TX_BUFFER */
- #define REG_30_ID_RESERVED 0x30

Bit definitions for register 0x30-0x35 Please take care not to write to these registers as doing so may cause the DW1000 to malfunction.

 - #define REG_31_ID_RESERVED 0x31
 - #define REG_32_ID_RESERVED 0x32
 - #define REG_33_ID_RESERVED 0x33
 - #define REG_34_ID_RESERVED 0x34
 - #define REG_35_ID_RESERVED 0x35
 - #define PMSC_ID 0x36 /* Power Management System Control Block */

Bit definitions for register PMSC.

 - #define PMSC_LEN (48)
 - #define PMSC_CTRL0_OFFSET 0x00
 - #define PMSC_CTRL0_LEN (4)
 - #define PMSC_CTRL0_MASK 0xF18F847FUL /* access mask to register PMSC_CTRL0 */
 - #define PMSC_CTRL0_SYSCLKS_AUTO 0x00000000UL /* The system clock will run off the 19.2 MHz XTI clock until the PLL is calibrated and locked, then it will switch over the 125 MHz PLL clock */
 - #define PMSC_CTRL0_SYSCLKS_19M 0x00000001UL /* Force system clock to be the 19.2 MHz XTI clock. */
 - #define PMSC_CTRL0_SYSCLKS_125M 0x00000002UL /* Force system clock to the 125 MHz PLL clock. */
 - #define PMSC_CTRL0_RXCLKS_AUTO 0x00000000UL /* The RX clock will be disabled until it is required for an RX operation */
 - #define PMSC_CTRL0_RXCLKS_19M 0x00000004UL /* Force RX clock enable and sourced clock from the 19.2 MHz XTI clock */
 - #define PMSC_CTRL0_RXCLKS_125M 0x00000008UL /* Force RX clock enable and sourced from the 125 MHz PLL clock */

- #define PMSC_CTRL0_RXCLKS_OFF 0x0000000CUL /* Force RX clock off. */
- #define PMSC_CTRL0_TXCLKS_AUTO 0x00000000UL /* The TX clock will be disabled until it is required for a TX operation */
- #define PMSC_CTRL0_TXCLKS_19M 0x00000010UL /* Force TX clock enable and sourced clock from the 19.2 MHz XT1 clock */
- #define PMSC_CTRL0_TXCLKS_125M 0x00000020UL /* Force TX clock enable and sourced from the 125 MHz PLL clock */
- #define PMSC_CTRL0_TXCLKS_OFF 0x00000030UL /* Force TX clock off */
- #define PMSC_CTRL0_FACE 0x00000040UL /* Force Accumulator Clock Enable */
- #define PMSC_CTRL0_GPDCE 0x00040000UL /* GPIO De-bounce Clock Enable */
- #define PMSC_CTRL0_KHZCLEN 0x00800000UL /* Kilohertz Clock Enable */
- #define PMSC_CTRL0_PLL2_SEQ_EN 0x01000000UL /* Enable PLL2 on/off sequencing by SNIFF mode */
- #define PMSC_CTRL0_SOFTRESET_OFFSET 3 /* In bytes */
- #define PMSC_CTRL0_RESET_ALL 0x00 /* Assuming only 4th byte of the register is read */
- #define PMSC_CTRL0_RESET_RX 0xE0 /* Assuming only 4th byte of the register is read */
- #define PMSC_CTRL0_RESET_CLEAR 0xF0 /* Assuming only 4th byte of the register is read */
- #define PMSC_CTRL1_OFFSET 0x04
- #define PMSC_CTRL1_LEN (4)
- #define PMSC_CTRL1_MASK 0xFC02F802UL /* access mask to register PMSC_CTRL1 */
- #define PMSC_CTRL1_ARX2INIT 0x00000002UL /* Automatic transition from receive mode into the INIT state */
- #define PMSC_CTRL1_ATXSLP 0x00000800UL /* If this bit is set then the DW1000 will automatically transition into SLEEP or DEEPSLEEP mode after transmission of a frame */
- #define PMSC_CTRL1_ARXSLP 0x00001000UL /* this bit is set then the DW1000 will automatically transition into SLEEP mode after a receive attempt */
- #define PMSC_CTRL1_SNOZE 0x00002000UL /* Snooze Enable */
- #define PMSC_CTRL1_SNOZR 0x00004000UL /* The SNOZR bit is set to allow the snooze timer to repeat twice */
- #define PMSC_CTRL1_PLLSYN 0x00008000UL /* This enables a special 1 GHz clock used for some external SYNC modes */
- #define PMSC_CTRL1_LDERUNE 0x00020000UL /* This bit enables the running of the LDE algorithm */
- #define PMSC_CTRL1_KHZCLKDIV_MASK 0xFC000000UL /* Kilohertz clock divisor */
- #define PMSC_CTRL1_PKTSEQ_DISABLE 0x00 /* writing this to PMSC CONTROL 1 register (bits 10-3) disables PMSC control of analog RF subsystems */
- #define PMSC_CTRL1_PKTSEQ_ENABLE 0xE7 /* writing this to PMSC CONTROL 1 register (bits 10-3) enables PMSC control of analog RF subsystems */
- #define PMSC_RES1_OFFSET 0x08
- #define PMSC_SNOZT_OFFSET 0x0C /* PMSC Snooze Time Register */
- #define PMSC_SNOZT_LEN (1)
- #define PMSC_RES2_OFFSET 0x10
- #define PMSC_RES3_OFFSET 0x24
- #define PMSC_TXFINESEQ_OFFSET 0x26
- #define PMSC_TXFINESEQ_DISABLE 0x0 /* Writing this disables fine grain sequencing in the transmitter */
- #define PMSC_TXFINESEQ_ENABLE 0xB74 /* Writing this enables fine grain sequencing in the transmitter */
- #define PMSC_LED_C_OFFSET 0x28
- #define PMSC_LED_C_LEN (4)
- #define PMSC_LED_C_MASK 0x000001FFUL /* 32-bit LED control register. */
- #define PMSC_LED_C_BLINK_TIM_MASK 0x000000FFUL /* This field determines how long the LEDs remain lit after an event that causes them to be set on. */
- #define PMSC_LED_C_BLNKEN 0x00000100UL /* Blink Enable. When this bit is set to 1 the LED blink feature is enabled. */
- #define PMSC_LED_C_BLINK_TIME_DEF 0x10

- #define PMSC_LED_C_BLINK_NOW_ALL 0x000F0000UL
- #define REG_37_ID_RESERVED 0x37

Bit definitions for register 0x37-0x3F Please take care not to write to these registers as doing so may cause the DW1000 to malfunction.

- #define REG_38_ID_RESERVED 0x38
- #define REG_39_ID_RESERVED 0x39
- #define REG_3A_ID_RESERVED 0x3A
- #define REG_3B_ID_RESERVED 0x3B
- #define REG_3C_ID_RESERVED 0x3C
- #define REG_3D_ID_RESERVED 0x3D
- #define REG_3E_ID_RESERVED 0x3E
- #define REG_3F_ID_RESERVED 0x3F

6.16.1 Detailed Description

DW1000 Register Definitions This file supports assembler and C development for DW1000 enabled devices.

Attention

Copyright 2013 (c) Decawave Ltd, Dublin, Ireland.

All rights reserved.

6.16.2 Macro Definition Documentation

6.16.2.1 ACC_MEM_ID

```
#define ACC_MEM_ID 0x25 /* Read access to accumulator data */
```

Bit definitions for register ACC_MEM.

6.16.2.2 ACC_MEM_LEN

```
#define ACC_MEM_LEN (4064)
```

6.16.2.3 ACK_RESP_T_ACK_TIM_MASK

```
#define ACK_RESP_T_ACK_TIM_MASK 0xFF000000UL /* Auto-Acknowledgement turn-around Time */
```

6.16.2.4 ACK_RESP_T_ACK_TIM_OFFSET

```
#define ACK_RESP_T_ACK_TIM_OFFSET 3 /* In bytes */
```

6.16.2.5 ACK_RESP_T_ID

```
#define ACK_RESP_T_ID 0x1A /* Acknowledgement Time and Response Time */
```

Bit definitions for register ACK_RESP_T.

6.16.2.6 ACK_RESP_T_LEN

```
#define ACK_RESP_T_LEN (4)
```

6.16.2.7 ACK_RESP_T_MASK

```
#define ACK_RESP_T_MASK 0xFF0FFFFFUL /* Acknowledgement Time and Response access mask */
```

6.16.2.8 ACK_RESP_T_W4R_TIM_MASK

```
#define ACK_RESP_T_W4R_TIM_MASK 0x000FFFFFUL /* Wait-for-Response turn-around Time 20 bit field */
```

6.16.2.9 ACK_RESP_T_W4R_TIM_OFFSET

```
#define ACK_RESP_T_W4R_TIM_OFFSET 0 /* In bytes */
```

6.16.2.10 ACK_TIM_MASK

```
#define ACK_TIM_MASK ACK_RESP_T_ACK_TIM_MASK
```

6.16.2.11 AGC_CFG_STS_ID

```
#define AGC_CFG_STS_ID AGC_CTRL_ID
```

6.16.2.12 AGC_CTRL1_DIS_AM

```
#define AGC_CTRL1_DIS_AM 0x0001 /* Disable AGC Measurement. The DIS_AM bit is set by default.  
*/
```

6.16.2.13 AGC_CTRL1_LEN

```
#define AGC_CTRL1_LEN (2)
```

6.16.2.14 AGC_CTRL1_MASK

```
#define AGC_CTRL1_MASK 0x0001 /* access mask to AGC configuration and control register */
```

6.16.2.15 AGC_CTRL1_OFFSET

```
#define AGC_CTRL1_OFFSET (0x02)
```

6.16.2.16 AGC_CTRL_ID

```
#define AGC_CTRL_ID 0x23 /* Automatic Gain Control configuration */
```

Bit definitions for register AGC_CTRL Please take care to write to this register as doing so may cause the DW1000 to malfunction.

6.16.2.17 AGC_CTRL_LEN

```
#define AGC_CTRL_LEN (32)
```

6.16.2.18 AGC_STAT1_EDG1_MASK

```
#define AGC_STAT1_EDG1_MASK 0x0007C0 /* This 5-bit gain value relates to input noise power measurement. */
```

6.16.2.19 AGC_STAT1_EDG2_MASK

```
#define AGC_STAT1_EDG2_MASK 0x0FF800 /* This 9-bit value relates to the input noise power measurement. */
```

6.16.2.20 AGC_STAT1_LEN

```
#define AGC_STAT1_LEN (3)
```

6.16.2.21 AGC_STAT1_MASK

```
#define AGC_STAT1_MASK 0xFFFF
```

6.16.2.22 AGC_STAT1_OFFSET

```
#define AGC_STAT1_OFFSET (0x1E)
```

6.16.2.23 AGC_TUNE1_16M

```
#define AGC_TUNE1_16M 0x8870
```

6.16.2.24 AGC_TUNE1_64M

```
#define AGC_TUNE1_64M 0x889B
```

6.16.2.25 AGC_TUNE1_LEN

```
#define AGC_TUNE1_LEN (2)
```

6.16.2.26 AGC_TUNE1_MASK

```
#define AGC_TUNE1_MASK 0xFFFF /* It is a 16-bit tuning register for the AGC. */
```

6.16.2.27 AGC_TUNE1_OFFSET

```
#define AGC_TUNE1_OFFSET (0x04)
```

6.16.2.28 AGC_TUNE2_LEN

```
#define AGC_TUNE2_LEN (4)
```

6.16.2.29 AGC_TUNE2_MASK

```
#define AGC_TUNE2_MASK 0xFFFFFFFFFUL
```

6.16.2.30 AGC_TUNE2_OFFSET

```
#define AGC_TUNE2_OFFSET (0x0C)
```

6.16.2.31 AGC_TUNE2_VAL

```
#define AGC_TUNE2_VAL 0X2502A907UL
```

6.16.2.32 AGC_TUNE3_LEN

```
#define AGC_TUNE3_LEN (2)
```

6.16.2.33 AGC_TUNE3_MASK

```
#define AGC_TUNE3_MASK 0xFFFF
```

6.16.2.34 AGC_TUNE3_OFFSET

```
#define AGC_TUNE3_OFFSET (0x12)
```

6.16.2.35 AGC_TUNE3_VAL

```
#define AGC_TUNE3_VAL 0X0035
```

6.16.2.36 AON_ADDR_LEN

```
#define AON_ADDR_LEN (1)
```

6.16.2.37 AON_ADDR_LPOSC_CAL_0

```
#define AON_ADDR_LPOSC_CAL_0 117 /* Address of low-power oscillator calibration value (lower byte) */
```

6.16.2.38 AON_ADDR_LPOSC_CAL_1

```
#define AON_ADDR_LPOSC_CAL_1 118 /* Address of low-power oscillator calibration value (lower byte) */
```

6.16.2.39 AON_ADDR_OFFSET

```
#define AON_ADDR_OFFSET 0x04 /* AON Direct Access Address */
```

6.16.2.40 AON_CFG0_LEN

```
#define AON_CFG0_LEN (4)
```

6.16.2.41 AON_CFG0_LPCLKDIVA_MASK

```
#define AON_CFG0_LPCLKDIVA_MASK 0x0000FFE0UL /* divider count for dividing the raw DW1000 XTAL oscillator frequency to set an LP clock frequency */
```

6.16.2.42 AON_CFG0_LPCLKDIVA_SHIFT

```
#define AON_CFG0_LPCLKDIVA_SHIFT (5)
```

6.16.2.43 AON_CFG0_LPDIV_EN

```
#define AON_CFG0_LPDIV_EN 0x00000010UL /* Low power divider enable configuration */
```

6.16.2.44 AON_CFG0_OFFSET

```
#define AON_CFG0_OFFSET 0x06 /* 32-bit configuration register for the always on block. */
```

6.16.2.45 AON_CFG0_SLEEP_EN

```
#define AON_CFG0_SLEEP_EN 0x00000001UL /* This is the sleep enable configuration bit */
```

6.16.2.46 AON_CFG0_SLEEP_SHIFT

```
#define AON_CFG0_SLEEP_SHIFT (16)
```

6.16.2.47 AON_CFG0_SLEEP_TIM

```
#define AON_CFG0_SLEEP_TIM 0xFFFF0000UL /* Sleep time. This field configures the sleep time count elapse value */
```

6.16.2.48 AON_CFG0_SLEEP_TIM_OFFSET

```
#define AON_CFG0_SLEEP_TIM_OFFSET 2 /* In bytes */
```

6.16.2.49 AON_CFG0_WAKE_CNT

```
#define AON_CFG0_WAKE_CNT 0x00000008UL /* Wake when sleep counter elapses */
```

6.16.2.50 AON_CFG0_WAKE_PIN

```
#define AON_CFG0_WAKE_PIN 0x00000002UL /* Wake using WAKEUP pin */
```

6.16.2.51 AON_CFG0_WAKE_SPI

```
#define AON_CFG0_WAKE_SPI 0x00000004UL /* Wake using SPI access SPICSn */
```

6.16.2.52 AON_CFG1_LEN

```
#define AON_CFG1_LEN (2)
```

6.16.2.53 AON_CFG1_LPOSC_CAL

```
#define AON_CFG1_LPOSC_CAL 0x0004 /* This bit enables the calibration function that measures the period of the ICs internal low powered oscillator */
```

6.16.2.54 AON_CFG1_MASK

```
#define AON_CFG1_MASK 0x0007 /* aceess mask to AON_CFG1 */
```

6.16.2.55 AON_CFG1_OFFSET

```
#define AON_CFG1_OFFSET 0x0A
```

6.16.2.56 AON_CFG1_SLEEP_CEN

```
#define AON_CFG1_SLEEP_CEN 0x0001 /* This bit enables the sleep counter */
```

6.16.2.57 AON_CFG1_SMXX

```
#define AON_CFG1_SMXX 0x0002 /* This bit needs to be set to 0 for correct operation in the SL↔EEP state within the DW1000 */
```

6.16.2.58 AON_CTRL_DCA_ENAB

```
#define AON_CTRL_DCA_ENAB 0x80 /* Direct AON memory access enable bit */
```

6.16.2.59 AON_CTRL_DCA_READ

```
#define AON_CTRL_DCA_READ 0x08 /* Direct AON memory access read */
```

6.16.2.60 AON_CTRL_LEN

```
#define AON_CTRL_LEN (1)
```

6.16.2.61 AON_CTRL_MASK

```
#define AON_CTRL_MASK 0x8F /* access mask to AON_CTRL register */
```

6.16.2.62 AON_CTRL_OFFSET

```
#define AON_CTRL_OFFSET 0x02 /* The bits in this register in general cause direct activity  
within the AON block with respect to the stored AON memory */
```

6.16.2.63 AON_CTRL_RESTORE

```
#define AON_CTRL_RESTORE 0x01 /* When this bit is set the DW1000 will copy the user configurations  
from the AON memory to the host interface register set. */
```

6.16.2.64 AON_CTRL_SAVE

```
#define AON_CTRL_SAVE 0x02 /* When this bit is set the DW1000 will copy the user configurations  
from the host interface register set into the AON memory */
```

6.16.2.65 AON_CTRL_UPL_CFG

```
#define AON_CTRL_UPL_CFG 0x04 /* Upload the AON block configurations to the AON */
```

6.16.2.66 AON_ID

```
#define AON_ID 0x2C /* Always-On register set */
```

Bit definitions for register.

6.16.2.67 AON_LEN

```
#define AON_LEN (12)
```

6.16.2.68 AON_RDAT_LEN

```
#define AON_RDAT_LEN (1)
```

6.16.2.69 AON_RDAT_OFFSET

```
#define AON_RDAT_OFFSET 0x03 /* AON Direct Access Read Data Result */
```

6.16.2.70 AON_WCFG_LEN

```
#define AON_WCFG_LEN (2)
```

6.16.2.71 AON_WCFG_MASK

```
#define AON_WCFG_MASK 0x09CB /* access mask to AON_WCFG register*/
```

6.16.2.72 AON_WCFG_OFFSET

```
#define AON_WCFG_OFFSET 0x00 /* used to control what the DW1000 IC does as it wakes up from low-power SLEEP or DEEPSLEEPstates. */
```

6.16.2.73 AON_WCFG_ONW_L64P

```
#define AON_WCFG_ONW_L64P 0x0080 /* On Wake-up load the Length64 receiver operating parameter set */
```

6.16.2.74 AON_WCFG_ONW_LDC

```
#define AON_WCFG_ONW_LDC 0x0040 /* On Wake-up load configurations from the AON memory into the host interface register set */
```

6.16.2.75 AON_WCFG_ONW_LEUI

```
#define AON_WCFG_ONW_LEUI 0x0008 /* On Wake-up load the EUI from OTP memory into Register  
file: 0x01 Extended Unique Identifier. */
```

6.16.2.76 AON_WCFG_ONW_LLDE

```
#define AON_WCFG_ONW_LLDE 0x0800 /* On Wake-up load the LDE microcode. */
```

6.16.2.77 AON_WCFG_ONW_LLDO

```
#define AON_WCFG_ONW_LLDO 0x1000 /* On Wake-up load the LDO tune value. */
```

6.16.2.78 AON_WCFG_ONW_RADC

```
#define AON_WCFG_ONW_RADC 0x0001 /* On Wake-up Run the (temperature and voltage) Analog-to-Digital Convertors */
```

6.16.2.79 AON_WCFG_ONW_RX

```
#define AON_WCFG_ONW_RX 0x0002 /* On Wake-up turn on the Receiver */
```

6.16.2.80 AON_WCFG_PRES_SLEEP

```
#define AON_WCFG_PRES_SLEEP 0x0100 /* Preserve Sleep. This bit determines what the DW1000 does with respect to the ARXSLP and ATXSLP sleep controls */
```

6.16.2.81 BOOSTNORM_MASK

```
#define BOOSTNORM_MASK TX_POWER_BOOSTNORM_MASK
```

6.16.2.82 BOOSTP125_MASK

```
#define BOOSTP125_MASK TX_POWER_BOOSTP125_MASK
```

6.16.2.83 BOOSTP250_MASK

```
#define BOOSTP250_MASK TX_POWER_BOOSTP250_MASK
```

6.16.2.84 BOOSTP500_MASK

```
#define BOOSTP500_MASK TX_POWER_BOOSTP500_MASK
```

6.16.2.85 CHAN_CTRL_DWSFD

```
#define CHAN_CTRL_DWSFD 0x00020000UL /* Bit 17 This bit enables a non-standard DecaWave proprietary SFD sequence. */
```

6.16.2.86 CHAN_CTRL_DWSFD_SHIFT

```
#define CHAN_CTRL_DWSFD_SHIFT (17)
```

6.16.2.87 CHAN_CTRL_ID

```
#define CHAN_CTRL_ID 0x1F /* Channel Control */
```

Bit definitions for register CHAN_CTRL.

6.16.2.88 CHAN_CTRL_LEN

```
#define CHAN_CTRL_LEN (4)
```

6.16.2.89 CHAN_CTRL_MASK

```
#define CHAN_CTRL_MASK 0xFFFF00FFUL /* Channel Control Register access mask */
```

6.16.2.90 CHAN_CTRL_RNSSFD

```
#define CHAN_CTRL_RNSSFD 0x00200000UL /* Bit 21 Non-standard SFD in the receiver */
```

6.16.2.91 CHAN_CTRL_RNSSFD_SHIFT

```
#define CHAN_CTRL_RNSSFD_SHIFT (21)
```

6.16.2.92 CHAN_CTRL_RX_CHAN_MASK

```
#define CHAN_CTRL_RX_CHAN_MASK 0x000000F0UL
```

6.16.2.93 CHAN_CTRL_RX_CHAN_SHIFT

```
#define CHAN_CTRL_RX_CHAN_SHIFT (4) /* Bits 4..7 RX channel number 0-15 selection */
```

6.16.2.94 CHAN_CTRL_RX_PCOD_MASK

```
#define CHAN_CTRL_RX_PCOD_MASK 0xF8000000UL /* Bits 27..31 RX Preamble Code selection, 1 to  
24. */
```

6.16.2.95 CHAN_CTRL_RX_PCOD_SHIFT

```
#define CHAN_CTRL_RX_PCOD_SHIFT (27)
```

6.16.2.96 CHAN_CTRL_RXFPRF_16

```
#define CHAN_CTRL_RXFPRF_16 0x00040000UL /* Specify (Force) RX Pulse Repetition Rate: 00 = 4  
MHz, 01 = 16 MHz, 10 = 64MHz. */
```

6.16.2.97 CHAN_CTRL_RXFPRF_4

```
#define CHAN_CTRL_RXFPRF_4 0x00000000UL /* Specify (Force) RX Pulse Repetition Rate: 00 = 4  
MHz, 01 = 16 MHz, 10 = 64MHz. */
```

6.16.2.98 CHAN_CTRL_RXFPRF_64

```
#define CHAN_CTRL_RXFPRF_64 0x00080000UL /* Specify (Force) RX Pulse Repetition Rate: 00 = 4  
MHz, 01 = 16 MHz, 10 = 64MHz. */
```

6.16.2.99 CHAN_CTRL_RXFPRF_MASK

```
#define CHAN_CTRL_RXFPRF_MASK 0x000C0000UL /* Bits 18..19 Specify (Force) RX Pulse Repetition  
Rate: 00 = 4 MHz, 01 = 16 MHz, 10 = 64MHz. */
```

6.16.2.100 CHAN_CTRL_RXFPRF_SHIFT

```
#define CHAN_CTRL_RXFPRF_SHIFT (18)
```

6.16.2.101 CHAN_CTRL_TNSSFD

```
#define CHAN_CTRL_TNSSFD 0x00100000UL /* Bit 20 Non-standard SFD in the transmitter */
```

6.16.2.102 CHAN_CTRL_TNSSFD_SHIFT

```
#define CHAN_CTRL_TNSSFD_SHIFT (20)
```

6.16.2.103 CHAN_CTRL_TX_CHAN_MASK

```
#define CHAN_CTRL_TX_CHAN_MASK 0x0000000FUL /* Supported channels are 1, 2, 3, 4, 5, and 7.*/
```

6.16.2.104 CHAN_CTRL_TX_CHAN_SHIFT

```
#define CHAN_CTRL_TX_CHAN_SHIFT (0) /* Bits 0..3 TX channel number 0-15 selection */
```

6.16.2.105 CHAN_CTRL_TX_PCOD_MASK

```
#define CHAN_CTRL_TX_PCOD_MASK 0x07C00000UL /* Bits 22..26 TX Preamble Code selection, 1 to 24. */
```

6.16.2.106 CHAN_CTRL_TX_PCOD_SHIFT

```
#define CHAN_CTRL_TX_PCOD_SHIFT (22)
```

6.16.2.107 CIR_MXG_MASK

```
#define CIR_MXG_MASK RX_EQUAL_CIR_MXG_MASK
```

6.16.2.108 CIR_MXG_SHIFT

```
#define CIR_MXG_SHIFT RX_EQUAL_CIR_MXG_SHIFT
```

6.16.2.109 DEV_ID_ID

```
#define DEV_ID_ID 0x00 /* Device ID register, includes revision info (0xDECA0130) */
```

Bit definitions for register DEV_ID.

6.16.2.110 DEV_ID_LEN

```
#define DEV_ID_LEN (4)
```

6.16.2.111 DEV_ID_MODEL_MASK

```
#define DEV_ID_MODEL_MASK 0x0000FF00UL /* The MODEL identifies the device. The DW1000 is  
device type 0x01 */
```

6.16.2.112 DEV_ID_REV_MASK

```
#define DEV_ID_REV_MASK 0x0000000FUL /* Revision */
```

6.16.2.113 DEV_ID_RIDTAG_MASK

```
#define DEV_ID_RIDTAG_MASK 0xFFFF0000UL /* Register Identification Tag 0XDECA */
```

6.16.2.114 DEV_ID_VER_MASK

```
#define DEV_ID_VER_MASK 0x000000F0UL /* Version */
```

6.16.2.115 DIAG_TMC_LEN

```
#define DIAG_TMC_LEN (2)
```

6.16.2.116 DIAG_TMC_MASK

```
#define DIAG_TMC_MASK 0x0010
```

6.16.2.117 DIAG_TMC_OFFSET

```
#define DIAG_TMC_OFFSET 0x24
```

6.16.2.118 DIAG_TMC_TX_PSTM

```
#define DIAG_TMC_TX_PSTM 0x0010 /* This test mode is provided to help support regulatory approvals  
spectral testing. When the TX_PSTM bit is set it enables a repeating transmission of the data  
from the TX_BUFFER */
```

6.16.2.119 DIG_DIAG_ID

```
#define DIG_DIAG_ID 0x2F /* Digital Diagnostics Interface */
```

Bit definitions for register DIG_DIAG Digital Diagnostics interface. It contains a number of sub-registers that give diagnostics information.

6.16.2.120 DIG_DIAG_LEN

```
#define DIG_DIAG_LEN (41)
```

6.16.2.121 DRX_CARRIER_INT_LEN

```
#define DRX_CARRIER_INT_LEN (3)
```

6.16.2.122 DRX_CARRIER_INT_MASK

```
#define DRX_CARRIER_INT_MASK 0x001FFFFF
```

6.16.2.123 DRX_CARRIER_INT_OFFSET

```
#define DRX_CARRIER_INT_OFFSET 0x28
```

6.16.2.124 DRX_CONF_ID

```
#define DRX_CONF_ID 0x27 /* Digital Receiver configuration */
```

Bit definitions for register DRX_CONF Digital Receiver configuration block.

6.16.2.125 DRX_CONF_LEN

```
#define DRX_CONF_LEN (44)
```

6.16.2.126 DRX_PRETOC_LEN

```
#define DRX_PRETOC_LEN (2)
```

6.16.2.127 DRX_PRETOC_MASK

```
#define DRX_PRETOC_MASK 0xFFFF
```

6.16.2.128 DRX_PRETOC_OFFSET

```
#define DRX_PRETOC_OFFSET 0x24 /* 7.2.40.9 Sub-Register 0x27:24 DRX_PRETOC */
```

6.16.2.129 DRX_SFDTOC_LEN

```
#define DRX_SFDTOC_LEN (2)
```

6.16.2.130 DRX_SFDTOC_MASK

```
#define DRX_SFDTOC_MASK 0xFFFF
```

6.16.2.131 DRX_SFDTOC_OFFSET

```
#define DRX_SFDTOC_OFFSET 0x20 /* 7.2.40.7 Sub-Register 0x27:20 DRX_SFDTOC */
```

6.16.2.132 DRX_TUNE0b_110K_NSTD

```
#define DRX_TUNE0b_110K_NSTD 0x0016
```

6.16.2.133 DRX_TUNE0b_110K_STD

```
#define DRX_TUNE0b_110K_STD 0x000A
```

6.16.2.134 DRX_TUNE0b_6M8_NSTD

```
#define DRX_TUNE0b_6M8_NSTD 0x0002
```

6.16.2.135 DRX_TUNE0b_6M8_STD

```
#define DRX_TUNE0b_6M8_STD 0x0001
```

6.16.2.136 DRX_TUNE0b_850K_NSTD

```
#define DRX_TUNE0b_850K_NSTD 0x0006
```

6.16.2.137 DRX_TUNE0b_850K_STD

```
#define DRX_TUNE0b_850K_STD 0x0001
```

6.16.2.138 DRX_TUNE0b_LEN

```
#define DRX_TUNE0b_LEN (2)
```

6.16.2.139 DRX_TUNE0b_MASK

```
#define DRX_TUNE0b_MASK 0xFFFF /* 7.2.40.2 Sub-Register 0x27:02 DRX_TUNE0b */
```

6.16.2.140 DRX_TUNE0b_OFFSET

```
#define DRX_TUNE0b_OFFSET (0x02) /* sub-register 0x02 is a 16-bit tuning register. */
```

6.16.2.141 DRX_TUNE1a_LEN

```
#define DRX_TUNE1a_LEN (2)
```

6.16.2.142 DRX_TUNE1a_MASK

```
#define DRX_TUNE1a_MASK 0xFFFF
```

6.16.2.143 DRX_TUNE1a_OFFSET

```
#define DRX_TUNE1a_OFFSET 0x04 /* 7.2.40.3 Sub-Register 0x27:04 DRX_TUNE1a */
```

6.16.2.144 DRX_TUNE1a_PRF16

```
#define DRX_TUNE1a_PRF16 0x0087
```

6.16.2.145 DRX_TUNE1a_PRF64

```
#define DRX_TUNE1a_PRF64 0x008D
```

6.16.2.146 DRX_TUNE1b_110K

```
#define DRX_TUNE1b_110K 0x0064
```

6.16.2.147 DRX_TUNE1b_6M8_PRE64

```
#define DRX_TUNE1b_6M8_PRE64 0x0010
```

6.16.2.148 DRX_TUNE1b_850K_6M8

```
#define DRX_TUNE1b_850K_6M8 0x0020
```

6.16.2.149 DRX_TUNE1b_LEN

```
#define DRX_TUNE1b_LEN (2)
```

6.16.2.150 DRX_TUNE1b_MASK

```
#define DRX_TUNE1b_MASK 0xFFFF
```

6.16.2.151 DRX_TUNE1b_OFFSET

```
#define DRX_TUNE1b_OFFSET 0x06 /* 7.2.40.4 Sub-Register 0x27:06 DRX_TUNE1b */
```

6.16.2.152 DRX_TUNE2_LEN

```
#define DRX_TUNE2_LEN (4)
```

6.16.2.153 DRX_TUNE2_MASK

```
#define DRX_TUNE2_MASK 0xFFFFFFFFFUL
```

6.16.2.154 DRX_TUNE2_OFFSET

```
#define DRX_TUNE2_OFFSET 0x08 /* 7.2.40.5 Sub-Register 0x27:08 DRX_TUNE2 */
```

6.16.2.155 DRX_TUNE2_PRF16_PAC16

```
#define DRX_TUNE2_PRF16_PAC16 0x331A0052UL
```

6.16.2.156 DRX_TUNE2_PRF16_PAC32

```
#define DRX_TUNE2_PRF16_PAC32 0x351A009AUL
```

6.16.2.157 DRX_TUNE2_PRF16_PAC64

```
#define DRX_TUNE2_PRF16_PAC64 0x371A011DUL
```

6.16.2.158 DRX_TUNE2_PRF16_PAC8

```
#define DRX_TUNE2_PRF16_PAC8 0x311A002DUL
```

6.16.2.159 DRX_TUNE2_PRF64_PAC16

```
#define DRX_TUNE2_PRF64_PAC16 0x333B00BEUL
```

6.16.2.160 DRX_TUNE2_PRF64_PAC32

```
#define DRX_TUNE2_PRF64_PAC32 0x353B015EUL
```

6.16.2.161 DRX_TUNE2_PRF64_PAC64

```
#define DRX_TUNE2_PRF64_PAC64 0x373B0296UL
```

6.16.2.162 DRX_TUNE2_PRF64_PAC8

```
#define DRX_TUNE2_PRF64_PAC8 0x313B006BUL
```

6.16.2.163 DRX_TUNE4H_LEN

```
#define DRX_TUNE4H_LEN (2)
```

6.16.2.164 DRX_TUNE4H_MASK

```
#define DRX_TUNE4H_MASK 0xFFFF
```

6.16.2.165 DRX_TUNE4H_OFFSET

```
#define DRX_TUNE4H_OFFSET 0x26 /* 7.2.40.10 Sub-Register 0x27:26 DRX_TUNE4H */
```

6.16.2.166 DRX_TUNE4H_PRE128PLUS

```
#define DRX_TUNE4H_PRE128PLUS 0x0028
```

6.16.2.167 DRX_TUNE4H_PRE64

```
#define DRX_TUNE4H_PRE64 0x0010
```

6.16.2.168 DW_NS_SFD_LEN_110K

```
#define DW_NS_SFD_LEN_110K 64 /* Decawave non-standard SFD length for 110 kbps */
```

6.16.2.169 DW_NS_SFD_LEN_6M8

```
#define DW_NS_SFD_LEN_6M8 8 /* Decawave non-standard SFD length for 6.8 Mbps */
```

6.16.2.170 DW_NS_SFD_LEN_850K

```
#define DW_NS_SFD_LEN_850K 16 /* Decawave non-standard SFD length for 850 kbps */
```

6.16.2.171 DX_TIME_ID

```
#define DX_TIME_ID 0x0A /* Delayed Send or Receive Time (40-bit) */
```

Bit definitions for register DX_TIME.

6.16.2.172 DX_TIME_LEN

```
#define DX_TIME_LEN (5)
```

6.16.2.173 EC_CTRL_LEN

```
#define EC_CTRL_LEN (4)
```

6.16.2.174 EC_CTRL_MASK

```
#define EC_CTRL_MASK 0x00000FFBUL /* sub-register 0x00 is the External clock synchronisation counter configuration register */
```

6.16.2.175 EC_CTRL_OFFSET

```
#define EC_CTRL_OFFSET (0x00)
```

6.16.2.176 EC_CTRL_OSRSM

```
#define EC_CTRL_OSRSM 0x00000002UL /* External receive synchronisation mode enable */
```

6.16.2.177 EC_CTRL_OSTRM

```
#define EC_CTRL_OSTRM 0x00000800UL /* External timebase reset mode enable */
```

6.16.2.178 EC_CTRL_OSTSM

```
#define EC_CTRL_OSTSM 0x00000001UL /* External transmit synchronisation mode enable */
```

6.16.2.179 EC_CTRL_PLLCK

```
#define EC_CTRL_PLLCK 0x04 /* PLL lock detect enable */
```

6.16.2.180 EC_CTRL_WAIT_MASK

```
#define EC_CTRL_WAIT_MASK 0x000007F8UL /* Wait counter used for external transmit synchronisation  
and external timebase reset */
```

6.16.2.181 EC_GOLP

```
#define EC_GOLP (0x08)
```

6.16.2.182 EC_GOLP_LEN

```
#define EC_GOLP_LEN (4)
```

6.16.2.183 EC_GOLP_MASK

```
#define EC_GOLP_MASK 0x0000003FUL /* sub-register 0x08 is the External clock offset to first  
path 1 GHz counter, EC_GOLP */
```

6.16.2.184 EC_GOLP_OFFSET_EXT_MASK

```
#define EC_GOLP_OFFSET_EXT_MASK 0x0000003FUL /* This register contains the 1 GHz count from  
the arrival of the RMARKER and the next edge of the external clock. */
```

6.16.2.185 EC_RXTC_LEN

```
#define EC_RXTC_LEN (4)
```

6.16.2.186 EC_RXTC_MASK

```
#define EC_RXTC_MASK 0xFFFFFFFFUL /* External clock synchronisation counter captured on RMARK←  
ER */
```

6.16.2.187 EC_RXTC_OFFSET

```
#define EC_RXTC_OFFSET (0x04)
```

6.16.2.188 EUI_64_ID

```
#define EUI_64_ID 0x01 /* IEEE Extended Unique Identifier (63:0) */
```

Bit definitions for register EUI_64.

6.16.2.189 EUI_64_LEN

```
#define EUI_64_LEN (8)
```

6.16.2.190 EUI_64_OFFSET

```
#define EUI_64_OFFSET 0x00
```

6.16.2.191 EVC_CLR

```
#define EVC_CLR 0x00000002UL
```

6.16.2.192 EVC_CTRL_LEN

```
#define EVC_CTRL_LEN (4)
```

6.16.2.193 EVC_CTRL_MASK

```
#define EVC_CTRL_MASK 0x00000003UL/* access mask to Register for bits should always be set to zero to avoid any malfunction of the device. */
```

6.16.2.194 EVC_CTRL_OFFSET

```
#define EVC_CTRL_OFFSET 0x00 /* Event Counter Control */
```

6.16.2.195 EVC_EN

```
#define EVC_EN 0x00000001UL/* Event Counters Enable bit */
```

6.16.2.196 EVC_FCE_LEN

```
#define EVC_FCE_LEN (2)
```

6.16.2.197 EVC_FCE_MASK

```
#define EVC_FCE_MASK 0xFFFF
```

6.16.2.198 EVC_FCE_OFFSET

```
#define EVC_FCE_OFFSET 0x0A /* The EVC_FCE field is a 12-bit counter of the frames received with bad CRC/FCS sequence. */
```

6.16.2.199 EVC_FCG_LEN

```
#define EVC_FCG_LEN (2)
```

6.16.2.200 EVC_FCG_MASK

```
#define EVC_FCG_MASK 0xFFFF
```

6.16.2.201 EVC_FCG_OFFSET

```
#define EVC_FCG_OFFSET 0x08 /* The EVC_FCG field is a 12-bit counter of the frames received  
with good CRC/FCS sequence. */
```

6.16.2.202 EVC_FFR_LEN

```
#define EVC_FFR_LEN (2)
```

6.16.2.203 EVC_FFR_MASK

```
#define EVC_FFR_MASK 0xFFFF
```

6.16.2.204 EVC_FFR_OFFSET

```
#define EVC_FFR_OFFSET 0x0C /* The EVC_FFR field is a 12-bit counter of the frames rejected by  
the receive frame filtering function. */
```

6.16.2.205 EVC_FWTO_LEN

```
#define EVC_FWTO_LEN (2)
```

6.16.2.206 EVC_FWTO_MASK

```
#define EVC_FWTO_MASK 0xFFFF
```

6.16.2.207 EVC_FWTO_OFFSET

```
#define EVC_FWTO_OFFSET 0x14 /* The EVC_FWTO field is a 12-bit counter of receive frame wait
timeout events */
```

6.16.2.208 EVC_HPW_LEN

```
#define EVC_HPW_LEN (2)
```

6.16.2.209 EVC_HPW_MASK

```
#define EVC_HPW_MASK 0xFFFF
```

6.16.2.210 EVC_HPW_OFFSET

```
#define EVC_HPW_OFFSET 0x18 /* The EVC_HPW field is a 12-bit counter of Half Period Warnings.
*/
```

6.16.2.211 EVC_OVR_LEN [1/2]

```
#define EVC_OVR_LEN (2)
```

6.16.2.212 EVC_OVR_LEN [2/2]

```
#define EVC_OVR_LEN (2)
```

6.16.2.213 EVC_OVR_MASK [1/2]

```
#define EVC_OVR_MASK 0x0FFF
```

6.16.2.214 EVC_OVR_MASK [2/2]

```
#define EVC_OVR_MASK 0x0FFF
```

6.16.2.215 EVC_OVR_OFFSET

```
#define EVC_OVR_OFFSET 0x0E /* The EVC_OVR field is a 12-bit counter of receive overrun events */
```

6.16.2.216 EVC_PHE_LEN

```
#define EVC_PHE_LEN (2)
```

6.16.2.217 EVC_PHE_MASK

```
#define EVC_PHE_MASK 0x0FFF
```

6.16.2.218 EVC_PHE_OFFSET

```
#define EVC_PHE_OFFSET 0x04 /* PHR Error Event Counter */
```

6.16.2.219 EVC_PTO_LEN

```
#define EVC_PTO_LEN (2)
```

6.16.2.220 EVC_PTO_MASK

```
#define EVC_PTO_MASK 0x0FFF
```

6.16.2.221 EVC_PTO_OFFSET

```
#define EVC_PTO_OFFSET 0x12 /* The EVC_PTO field is a 12-bit counter of Preamble detection  
Timeout events */
```

6.16.2.222 EVC_RES1_OFFSET

```
#define EVC_RES1_OFFSET 0x1C /* Please take care not to write to this register as doing so may  
cause the DW1000 to malfunction. */
```

6.16.2.223 EVC_RSE_LEN

```
#define EVC_RSE_LEN (2)
```

6.16.2.224 EVC_RSE_MASK

```
#define EVC_RSE_MASK 0x0FFF
```

6.16.2.225 EVC_RSE_OFFSET

```
#define EVC_RSE_OFFSET 0x06 /* Reed Solomon decoder (Frame Sync Loss) Error Event Counter */
```

6.16.2.226 EVC_STO_OFFSET

```
#define EVC_STO_OFFSET 0x10 /* The EVC_STO field is a 12-bit counter of SFD Timeout Error  
events */
```

6.16.2.227 EVC_TPW_LEN

```
#define EVC_TPW_LEN (2)
```

6.16.2.228 EVC_TPW_MASK

```
#define EVC_TPW_MASK 0x0FFF
```

6.16.2.229 EVC_TPW_OFFSET

```
#define EVC_TPW_OFFSET 0x1A /* The EVC_TPW field is a 12-bit counter of Transmitter Power-Up  
Warnings. */
```

6.16.2.230 EVC_TXFS_LEN

```
#define EVC_TXFS_LEN (2)
```

6.16.2.231 EVC_TXFS_MASK

```
#define EVC_TXFS_MASK 0x0FFF
```

6.16.2.232 EVC_TXFS_OFFSET

```
#define EVC_TXFS_OFFSET 0x16 /* The EVC_TXFS field is a 12-bit counter of transmit frames  
sent. This is incremented every time a frame is sent */
```

6.16.2.233 EXT_SYNC_ID

```
#define EXT_SYNC_ID 0x24 /* External synchronisation control */
```

Bit definitions for register EXT_SYNC.

6.16.2.234 EXT_SYNC_LEN

```
#define EXT_SYNC_LEN (12)
```

6.16.2.235 FP_AMPL2_MASK

```
#define FP_AMPL2_MASK RX_EQUAL_FP_AMPL2_MASK
```

6.16.2.236 FP_AMPL2_SHIFT

```
#define FP_AMPL2_SHIFT RX_EQUAL_FP_AMPL2_SHIFT
```

6.16.2.237 FP_AMPL3_MASK

```
#define FP_AMPL3_MASK RX_EQUAL_FP_AMPL3_MASK
```

6.16.2.238 FP_AMPL3_SHIFT

```
#define FP_AMPL3_SHIFT RX_EQUAL_FP_AMPL3_SHIFT
```

6.16.2.239 FS_CTRL_ID

```
#define FS_CTRL_ID 0x2B /* Frequency synthesiser control block */
```

Bit definitions for register Refer to section 7.2.44 Register file: 0x2B Frequency synthesiser control block.

6.16.2.240 FS_CTRL_LEN

```
#define FS_CTRL_LEN (21)
```

6.16.2.241 FS_PLLCFG_CH1

```
#define FS_PLLCFG_CH1 0x09000407UL /* Operating Channel 1 */
```

6.16.2.242 FS_PLLCFG_CH2

```
#define FS_PLLCFG_CH2 0x08400508UL /* Operating Channel 2 */
```

6.16.2.243 FS_PLLCFG_CH3

```
#define FS_PLLCFG_CH3 0x08401009UL /* Operating Channel 3 */
```

6.16.2.244 FS_PLLCFG_CH4

```
#define FS_PLLCFG_CH4 FS_PLLCFG_CH2 /* Operating Channel 4 (same as 2) */
```

6.16.2.245 FS_PLLCFG_CH5

```
#define FS_PLLCFG_CH5 0x0800041DUL /* Operating Channel 5 */
```

6.16.2.246 FS_PLLCFG_CH7

```
#define FS_PLLCFG_CH7 FS_PLLCFG_CH5 /* Operating Channel 7 (same as 5) */
```

6.16.2.247 FS_PLLCFG_LEN

```
#define FS_PLLCFG_LEN (5)
```

6.16.2.248 FS_PLLCFG_OFFSET

```
#define FS_PLLCFG_OFFSET 0x07 /* Frequency synthesiser PLL configuration */
```

6.16.2.249 FS_PLLTUNE_CH1

```
#define FS_PLLTUNE_CH1 0x1E /* Operating Channel 1 */
```

6.16.2.250 FS_PLLTUNE_CH2

```
#define FS_PLLTUNE_CH2 0x26 /* Operating Channel 2 */
```

6.16.2.251 FS_PLLTUNE_CH3

```
#define FS_PLLTUNE_CH3 0x56 /* Operating Channel 3 */
```

6.16.2.252 FS_PLLTUNE_CH4

```
#define FS_PLLTUNE_CH4 FS_PLLTUNE_CH2 /* Operating Channel 4 (same as 2) */
```

6.16.2.253 FS_PLLTUNE_CH5

```
#define FS_PLLTUNE_CH5 0xBE /* Operating Channel 5 */
```

6.16.2.254 FS_PLLTUNE_CH7

```
#define FS_PLLTUNE_CH7 FS_PLLTUNE_CH5 /* Operating Channel 7 (same as 5) */
```

6.16.2.255 FS_PLLTUNE_LEN

```
#define FS_PLLTUNE_LEN (1)
```

6.16.2.256 FS_PLLTUNE_OFFSET

```
#define FS_PLLTUNE_OFFSET 0x0B /* Frequency synthesiser PLL Tuning */
```

6.16.2.257 FS_RES1_LEN

```
#define FS_RES1_LEN (7)
```

6.16.2.258 FS_RES1_OFFSET

```
#define FS_RES1_OFFSET 0x00 /* reserved area. Please take care not to write to this area as  
doing so may cause the DW1000 to malfunction. */
```

6.16.2.259 FS_RES2_LEN

```
#define FS_RES2_LEN (2)
```

6.16.2.260 FS_RES2_OFFSET

```
#define FS_RES2_OFFSET 0x0C /* reserved area. Please take care not to write to this area as  
doing so may cause the DW1000 to malfunction. */
```

6.16.2.261 FS_RES3_LEN

```
#define FS_RES3_LEN (6)
```

6.16.2.262 FS_RES3_OFFSET

```
#define FS_RES3_OFFSET 0x0F /* reserved area. Please take care not to write to this area as  
doing so may cause the DW1000 to malfunction. */
```

6.16.2.263 FS_XTALT_LEN

```
#define FS_XTALT_LEN (1)
```

6.16.2.264 FS_XTALT_MASK

```
#define FS_XTALT_MASK 0x1F /* Crystal Trim. Crystals may be trimmed using this register setting to tune out errors, see 8.1 IC Calibration Crystal Oscillator Trim. */
```

6.16.2.265 FS_XTALT_MIDRANGE

```
#define FS_XTALT_MIDRANGE 0x10
```

6.16.2.266 FS_XTALT_OFFSET

```
#define FS_XTALT_OFFSET 0x0E /* Frequency synthesiser Crystal trim */
```

6.16.2.267 GDM0

```
#define GDM0 GxM0 /* Mask for setting the direction of GPIO0 */
```

6.16.2.268 GDM1

```
#define GDM1 GxM1 /* Mask for setting the direction of GPIO1. (See GDM0). */
```

6.16.2.269 GDM2

```
#define GDM2 GxM2 /* Mask for setting the direction of GPIO2. (See GDM0). */
```

6.16.2.270 GDM3

```
#define GDM3 GxM3 /* Mask for setting the direction of GPIO3. (See GDM0). */
```

6.16.2.271 GDM4

```
#define GDM4 GxM4 /* Mask for setting the direction of GPIO4. (See GDM0). */
```

6.16.2.272 GDM5

```
#define GDM5 GxM5 /* Mask for setting the direction of GPIO5. (See GDM0). */
```

6.16.2.273 GDM6

```
#define GDM6 GxM6 /* Mask for setting the direction of GPIO6. (See GDM0). */
```

6.16.2.274 GDM7

```
#define GDM7 GxM7 /* Mask for setting the direction of GPIO7. (See GDM0). */
```

6.16.2.275 GDM8

```
#define GDM8 GxM8 /* Mask for setting the direction of GPIO8. (See GDM0). */
```

6.16.2.276 GDP0

```
#define GDP0 GxP0 /* Direction Selection for GPIO0. 1 = input, 0 = output. Only changed if  
the GDM0 mask bit has a value of 1 for the write operation*/
```

6.16.2.277 GDP1

```
#define GDP1 GxP1 /* Direction Selection for GPIO1. (See GDP0). */
```

6.16.2.278 GDP2

```
#define GDP2 GxP2 /* Direction Selection for GPIO2. (See GDP0). */
```

6.16.2.279 GDP3

```
#define GDP3 GxP3 /* Direction Selection for GPIO3. (See GDP0). */
```

6.16.2.280 GDP4

```
#define GDP4 GxP4 /* Direction Selection for GPIO4. (See GDP0). */
```

6.16.2.281 GDP5

```
#define GDP5 GxP5 /* Direction Selection for GPIO5. (See GDP0). */
```

6.16.2.282 GDP6

```
#define GDP6 GxP6 /* Direction Selection for GPIO6. (See GDP0). */
```

6.16.2.283 GDP7

```
#define GDP7 GxP7 /* Direction Selection for GPIO7. (See GDP0). */
```

6.16.2.284 GDP8

```
#define GDP8 GxP8 /* Direction Selection for GPIO8 */
```

6.16.2.285 GIBES0

```
#define GIBES0 GIRQx0 /* GPIO IRQ Both Edge selection for GPIO0 input. Value 0 = GPIO_IMODE register selects the edge. Value 1 = Both edges trigger the interrupt. */
```

6.16.2.286 GIBES1

```
#define GIBES1 GPIOx1 /* */
```

6.16.2.287 GIBES2

```
#define GIBES2 GPIOx2 /* */
```

6.16.2.288 GIBES3

```
#define GIBES3 GPIOx3 /* */
```

6.16.2.289 GIBES4

```
#define GIBES4 GPIOx4 /* */
```

6.16.2.290 GIBES5

```
#define GIBES5 GPIOx5 /* */
```

6.16.2.291 GIBES6

```
#define GIBES6 GPIOx6 /* */
```

6.16.2.292 GIBES7

```
#define GIBES7 GPIOx7 /* */
```

6.16.2.293 GIBES8

```
#define GIBES8 GPIOx8 /* Value 0 = use GPIO_IMODE, 1 = Both Edges */
```

6.16.2.294 GICLR0

```
#define GICLR0 GPIOx0 /* GPIO IRQ latch clear for GPIO0 input. Write 1 to clear the GPIO0  
interrupt latch. Writing 0 has no effect. Reading returns zero */
```

6.16.2.295 GICLR1

```
#define GICLR1 GPIOx1 /* */
```

6.16.2.296 GICLR2

```
#define GICLR2 GPIOx2 /* */
```

6.16.2.297 GICLR3

```
#define GICLR3 GPIOx3 /* */
```

6.16.2.298 GICLR4

```
#define GICLR4 GPIOx4 /* */
```

6.16.2.299 GICLR5

```
#define GICLR5 GPIOx5 /* */
```

6.16.2.300 GICLR6

```
#define GICLR6 GPIOx6 /* */
```

6.16.2.301 GICLR7

```
#define GICLR7 GIRQx7 /* */
```

6.16.2.302 GICLR8

```
#define GICLR8 GIRQx8 /* Write 1 to clear the interrupt latch */
```

6.16.2.303 GIDBE0

```
#define GIDBE0 GIRQx0 /* GPIO IRQ de-bounce enable for GPIO0. Value 1 = de-bounce enabled.  
Value 0 = de-bounce disabled */
```

6.16.2.304 GIDBE1

```
#define GIDBE1 GIRQx1 /* */
```

6.16.2.305 GIDBE2

```
#define GIDBE2 GIRQx2 /* */
```

6.16.2.306 GIDBE3

```
#define GIDBE3 GIRQx3 /* */
```

6.16.2.307 GIDBE4

```
#define GIDBE4 GIRQx4 /* */
```

6.16.2.308 GIDBE5

```
#define GIDBE5 GIRQx5 /* */
```

6.16.2.309 GIDBE6

```
#define GIDBE6 GIRQx6 /* */
```

6.16.2.310 GIDBE7

```
#define GIDBE7 GIRQx7 /* */
```

6.16.2.311 GIDBE8

```
#define GIDBE8 GIRQx8 /* Value 1 = de-bounce enabled, 0 = de-bounce disabled */
```

6.16.2.312 GIMOD0

```
#define GIMOD0 GIRQx0 /* GPIO IRQ Mode selection for GPIO0 input. Value 0 = Level sensitive  
interrupt. Value 1 = Edge triggered interrupt */
```

6.16.2.313 GIMOD1

```
#define GIMOD1 GIRQx1 /* */
```

6.16.2.314 GIMOD2

```
#define GIMOD2 GIRQx2 /* */
```

6.16.2.315 GIMOD3

```
#define GIMOD3 GPIOx3 /* */
```

6.16.2.316 GIMOD4

```
#define GIMOD4 GPIOx4 /* */
```

6.16.2.317 GIMOD5

```
#define GIMOD5 GPIOx5 /* */
```

6.16.2.318 GIMOD6

```
#define GIMOD6 GPIOx6 /* */
```

6.16.2.319 GIMOD7

```
#define GIMOD7 GPIOx7 /* */
```

6.16.2.320 GIMOD8

```
#define GIMOD8 GPIOx8 /* Value 0 = Level, 1 = Edge. */
```

6.16.2.321 GIRQE0

```
#define GIRQE0 GPIOx0 /* GPIO IRQ Enable for GPIO0 input. Value 1 = enable, 0 = disable*/
```

6.16.2.322 GIRQE1

```
#define GIRQE1 GPIOx1 /* */
```

6.16.2.323 GIRQE2

```
#define GIRQE2 GIRQx2 /* */
```

6.16.2.324 GIRQE3

```
#define GIRQE3 GIRQx3 /* */
```

6.16.2.325 GIRQE4

```
#define GIRQE4 GIRQx4 /* */
```

6.16.2.326 GIRQE5

```
#define GIRQE5 GIRQx5 /* */
```

6.16.2.327 GIRQE6

```
#define GIRQE6 GIRQx6 /* */
```

6.16.2.328 GIRQE7

```
#define GIRQE7 GIRQx7 /* */
```

6.16.2.329 GIRQE8

```
#define GIRQE8 GIRQx8 /* Value 1 = enable, 0 = disable */
```

6.16.2.330 GIRQx0

```
#define GIRQx0 0x00000001UL /* IRQ bit0 */
```

6.16.2.331 GIRQx1

```
#define GIRQx1 0x00000002UL /* IRQ bit1 */
```

6.16.2.332 GIRQx2

```
#define GIRQx2 0x00000004UL /* IRQ bit2 */
```

6.16.2.333 GIRQx3

```
#define GIRQx3 0x00000008UL /* IRQ bit3 */
```

6.16.2.334 GIRQx4

```
#define GIRQx4 0x00000010UL /* IRQ bit4 */
```

6.16.2.335 GIRQx5

```
#define GIRQx5 0x00000020UL /* IRQ bit5 */
```

6.16.2.336 GIRQx6

```
#define GIRQx6 0x00000040UL /* IRQ bit6 */
```

6.16.2.337 GIRQx7

```
#define GIRQx7 0x00000080UL /* IRQ bit7 */
```

6.16.2.338 GIRQx8

```
#define GIRQx8 0x00000100UL /* IRQ bit8 */
```

6.16.2.339 GISEN0

```
#define GISEN0 GPIOx0 /* GPIO IRQ Sense selection GPIO0 input. Value 0 = High or Rising-Edge,  
1 = Low or falling-edge.*/
```

6.16.2.340 GISEN1

```
#define GISEN1 GPIOx1 /* */
```

6.16.2.341 GISEN2

```
#define GISEN2 GPIOx2 /* */
```

6.16.2.342 GISEN3

```
#define GISEN3 GPIOx3 /* */
```

6.16.2.343 GISEN4

```
#define GISEN4 GPIOx4 /* */
```

6.16.2.344 GISEN5

```
#define GISEN5 GPIOx5 /* */
```

6.16.2.345 GISEN6

```
#define GISEN6 GPIOx6 /* */
```

6.16.2.346 GISEN7

```
#define GISEN7 GIRQx7 /* */
```

6.16.2.347 GISEN8

```
#define GISEN8 GIRQx8 /* Value 0 = High or Rising-Edge, 1 = Low or falling-edge */
```

6.16.2.348 GPIO_CTRL_ID

```
#define GPIO_CTRL_ID 0x26 /* Peripheral register bus 1 access - GPIO control */
```

Bit definitions for register GPIO_CTRL.

6.16.2.349 GPIO_CTRL_LEN

```
#define GPIO_CTRL_LEN (44)
```

6.16.2.350 GPIO_DIR_LEN

```
#define GPIO_DIR_LEN (3)
```

6.16.2.351 GPIO_DIR_MASK

```
#define GPIO_DIR_MASK 0x0011FFFFUL
```

6.16.2.352 GPIO_DIR_OFFSET

```
#define GPIO_DIR_OFFSET 0x08 /* sub-register 0x08 is the GPIO Direction Control Register */
```

6.16.2.353 GPIO_DOUT_LEN

```
#define GPIO_DOUT_LEN (3)
```

6.16.2.354 GPIO_DOUT_MASK

```
#define GPIO_DOUT_MASK GPIO_DIR_MASK
```

6.16.2.355 GPIO_DOUT_OFFSET

```
#define GPIO_DOUT_OFFSET 0x0C /* sub-register 0x0C is the GPIO data output register. */
```

6.16.2.356 GPIO_IBES_LEN

```
#define GPIO_IBES_LEN (4)
```

6.16.2.357 GPIO_IBES_MASK

```
#define GPIO_IBES_MASK GPIO_IRQE_MASK /* */
```

6.16.2.358 GPIO_IBES_OFFSET

```
#define GPIO_IBES_OFFSET 0x1C /* sub-register 0x1C is the GPIO interrupt Both Edge selection register */
```

6.16.2.359 GPIO_ICLR_LEN

```
#define GPIO_ICLR_LEN (4)
```

6.16.2.360 GPIO_ICLR_MASK

```
#define GPIO_ICLR_MASK GPIO_IRQE_MASK /* */
```

6.16.2.361 GPIO_ICLR_OFFSET

```
#define GPIO_ICLR_OFFSET 0x20 /* sub-register 0x20 is the GPIO interrupt clear register */
```

6.16.2.362 GPIO_IDBE_LEN

```
#define GPIO_IDBE_LEN (4)
```

6.16.2.363 GPIO_IDBE_MASK

```
#define GPIO_IDBE_MASK GPIO_IRQE_MASK
```

6.16.2.364 GPIO_IDBE_OFFSET

```
#define GPIO_IDBE_OFFSET 0x24 /* sub-register 0x24 is the GPIO interrupt de-bounce enable register */
```

6.16.2.365 GPIO_IMODE_LEN

```
#define GPIO_IMODE_LEN (4)
```

6.16.2.366 GPIO_IMODE_MASK

```
#define GPIO_IMODE_MASK GPIO_IRQE_MASK
```

6.16.2.367 GPIO_IMODE_OFFSET

```
#define GPIO_IMODE_OFFSET 0x18 /* sub-register 0x18 is the GPIO interrupt mode selection register */
```

6.16.2.368 GPIO_IRQE_LEN

```
#define GPIO_IRQE_LEN (4)
```

6.16.2.369 GPIO_IRQE_MASK

```
#define GPIO_IRQE_MASK 0x0000001FFUL
```

6.16.2.370 GPIO_IRQE_OFFSET

```
#define GPIO_IRQE_OFFSET 0x10 /* sub-register 0x10 is the GPIO interrupt enable register */
```

6.16.2.371 GPIO_ISEN_LEN

```
#define GPIO_ISEN_LEN (4)
```

6.16.2.372 GPIO_ISEN_MASK

```
#define GPIO_ISEN_MASK GPIO_IRQE_MASK
```

6.16.2.373 GPIO_ISEN_OFFSET

```
#define GPIO_ISEN_OFFSET 0x14 /* sub-register 0x14 is the GPIO interrupt sense selection register */
```

6.16.2.374 GPIO_MODE_LEN

```
#define GPIO_MODE_LEN (4)
```

6.16.2.375 GPIO_MODE_MASK

```
#define GPIO_MODE_MASK 0x00FFFFFFUL
```

6.16.2.376 GPIO_MODE_OFFSET

```
#define GPIO_MODE_OFFSET 0x00 /* sub-register 0x00 is the GPIO Mode Control Register */
```

6.16.2.377 GPIO_MSGP0_MASK

```
#define GPIO_MSGP0_MASK 0x00000000UL /* Mode Selection for GPIO0/RXOKLED */
```

6.16.2.378 GPIO_MSGP1_MASK

```
#define GPIO_MSGP1_MASK 0x00000300UL /* Mode Selection for GPIO1/SFDLED */
```

6.16.2.379 GPIO_MSGP2_MASK

```
#define GPIO_MSGP2_MASK 0x00000C00UL /* Mode Selection for GPIO2/RXLED */
```

6.16.2.380 GPIO_MSGP3_MASK

```
#define GPIO_MSGP3_MASK 0x000003000UL /* Mode Selection for GPIO3/TXLED */
```

6.16.2.381 GPIO_MSGP4_MASK

```
#define GPIO_MSGP4_MASK 0x00000C000UL /* Mode Selection for GPIO4/EXTPA */
```

6.16.2.382 GPIO_MSGP5_MASK

```
#define GPIO_MSGP5_MASK 0x00030000UL /* Mode Selection for GPIO5/EXTTXE */
```

6.16.2.383 GPIO_MSGP6_MASK

```
#define GPIO_MSGP6_MASK 0x000C0000UL /* Mode Selection for GPIO6/EXTRXE */
```

6.16.2.384 GPIO_MSGP7_MASK

```
#define GPIO_MSGP7_MASK 0x00300000UL /* Mode Selection for SYNC/GPIO7 */
```

6.16.2.385 GPIO_MSGP8_MASK

```
#define GPIO_MSGP8_MASK 0x00C00000UL /* Mode Selection for IRQ/GPIO8 */
```

6.16.2.386 GPIO_PIN2_RXLED

```
#define GPIO_PIN2_RXLED 0x00000400UL /* The pin operates as the RXLED output */
```

6.16.2.387 GPIO_PIN3_TXLED

```
#define GPIO_PIN3_TXLED 0x00001000UL /* The pin operates as the TXLED output */
```

6.16.2.388 GPIO_PIN4_EXTPA

```
#define GPIO_PIN4_EXTPA 0x00004000UL /* The pin operates as the EXTPA output */
```

6.16.2.389 GPIO_PIN5_EXTTXE

```
#define GPIO_PIN5_EXTTXE 0x00010000UL /* The pin operates as the EXTTXE output */
```

6.16.2.390 GPIO_PIN6_EXTRXE

```
#define GPIO_PIN6_EXTRXE 0x00040000UL /* The pin operates as the EXTRXE output */
```

6.16.2.391 GPIO_RAW_LEN

```
#define GPIO_RAW_LEN (4)
```

6.16.2.392 GPIO_RAW_MASK

```
#define GPIO_RAW_MASK GPIO_IRQE_MASK
```

6.16.2.393 GPIO_RAW_OFFSET

```
#define GPIO_RAW_OFFSET 0x28 /* sub-register 0x28 allows the raw state of the GPIO pin to be  
read. */
```

6.16.2.394 GRAWP0

```
#define GRAWP0 GPIOx0 /* This bit reflects the raw state of GPIO0 */
```

6.16.2.395 GRAWP1

```
#define GRAWP1 GPIOx1 /* */
```

6.16.2.396 GRAWP2

```
#define GRAWP2 GPIOx2 /* */
```

6.16.2.397 GRAWP3

```
#define GRAWP3 GPIOx3 /* */
```

6.16.2.398 GRAWP4

```
#define GRAWP4 GPIOx4 /* */
```

6.16.2.399 GRAWP5

```
#define GRAWP5 GPIOx5 /* */
```

6.16.2.400 GRAWP6

```
#define GRAWP6 GPIOx6 /* */
```

6.16.2.401 GRAWP7

```
#define GRAWP7 GPIOx7 /* */
```

6.16.2.402 GRAWP8

```
#define GRAWP8 GPIOx8 /* This bit reflects the raw state of GPIO8 */
```

6.16.2.403 GxM0

```
#define GxM0 0x000000010UL /* Mask for GPIO0 */
```

6.16.2.404 GxM1

```
#define GxM1 0x000000020UL /* Mask for GPIO1. (See GDM0). */
```

6.16.2.405 GxM2

```
#define GxM2 0x00000040UL /* Mask for GPIO2. (See GDM0). */
```

6.16.2.406 GxM3

```
#define GxM3 0x00000080UL /* Mask for GPIO3. (See GDM0). */
```

6.16.2.407 GxM4

```
#define GxM4 0x00001000UL /* Mask for GPIO4. (See GDM0). */
```

6.16.2.408 GxM5

```
#define GxM5 0x00002000UL /* Mask for GPIO5. (See GDM0). */
```

6.16.2.409 GxM6

```
#define GxM6 0x00004000UL /* Mask for GPIO6. (See GDM0). */
```

6.16.2.410 GxM7

```
#define GxM7 0x00008000UL /* Mask for GPIO7. (See GDM0). */
```

6.16.2.411 GxM8

```
#define GxM8 0x00100000UL /* Mask for GPIO8. (See GDM0). */
```

6.16.2.412 GxP0

```
#define GxP0 0x00000001UL /* GPIO0 Only changed if the GxM0 mask bit has a value of 1 for the write operation*/
```

6.16.2.413 GxP1

```
#define GxP1 0x00000002UL /* GPIO1. (See GDP0). */
```

6.16.2.414 GxP2

```
#define GxP2 0x00000004UL /* GPIO2. (See GDP0). */
```

6.16.2.415 GxP3

```
#define GxP3 0x00000008UL /* GPIO3. (See GDP0). */
```

6.16.2.416 GxP4

```
#define GxP4 0x00000100UL /* GPIO4. (See GDP0). */
```

6.16.2.417 GxP5

```
#define GxP5 0x00000200UL /* GPIO5. (See GDP0). */
```

6.16.2.418 GxP6

```
#define GxP6 0x00000400UL /* GPIO6. (See GDP0). */
```

6.16.2.419 GxP7

```
#define GxP7 0x00000800UL /* GPIO7. (See GDP0). */
```

6.16.2.420 GxP8

```
#define GxP8 0x00010000UL /* GPIO8 */
```

6.16.2.421 LDE_CFG1_LEN

```
#define LDE_CFG1_LEN (1)
```

6.16.2.422 LDE_CFG1_NSTDEV_MASK

```
#define LDE_CFG1_NSTDEV_MASK 0x1F /* Number of Standard Deviations mask. */
```

6.16.2.423 LDE_CFG1_OFFSET

```
#define LDE_CFG1_OFFSET 0x0806 /*8-bit configuration register*/
```

6.16.2.424 LDE_CFG1_PMULT_MASK

```
#define LDE_CFG1_PMULT_MASK 0xE0 /* Peak Multiplier mask. */
```

6.16.2.425 LDE_CFG2_LEN

```
#define LDE_CFG2_LEN (2)
```

6.16.2.426 LDE_CFG2_OFFSET

```
#define LDE_CFG2_OFFSET 0x1806 /* 16-bit LDE configuration tuning register */
```

6.16.2.427 LDE_IF_ID

```
#define LDE_IF_ID 0x2E /* Leading edge detection control block */
```

Bit definitions for register LDE_IF Refer to section 7.2.47 Register file: 0x2E Leading Edge Detection Interface
PLEASE NOTE: Other areas within the address space of Register file: 0x2E Leading Edge Detection Interface are reserved. To ensure proper operation of the LDE algorithm (i.e. to avoid loss of performance or a malfunction), care must be taken not to write to any byte locations other than those defined in the sub-sections below.

6.16.2.428 LDE_IF_LEN

```
#define LDE_IF_LEN (0)
```

6.16.2.429 LDE_PPAMPL_LEN

```
#define LDE_PPAMPL_LEN (2)
```

6.16.2.430 LDE_PPAMPL_OFFSET

```
#define LDE_PPAMPL_OFFSET 0x1002 /* reporting the magnitude of the peak signal seen in the  
accumulator data memory */
```

6.16.2.431 LDE_PPINDEX_LEN

```
#define LDE_PPINDEX_LEN (2)
```

6.16.2.432 LDE_PPINDEX_OFFSET

```
#define LDE_PPINDEX_OFFSET 0x1000 /* reporting the position within the accumulator that the LDE  
algorithm has determined to contain the maximum */
```

6.16.2.433 LDE_REPC_LEN

```
#define LDE_REPC_LEN (2)
```

6.16.2.434 LDE_REPC_OFFSET

```
#define LDE_REPC_OFFSET 0x2804 /* 16-bit configuration register for setting the replica avoidance  
coefficient */
```

6.16.2.435 LDE_REPC_PCODE_1

```
#define LDE_REPC_PCODE_1 0x5998
```

6.16.2.436 LDE_REPC_PCODE_10

```
#define LDE_REPC_PCODE_10 0x3332
```

6.16.2.437 LDE_REPC_PCODE_11

```
#define LDE_REPC_PCODE_11 0x3AE0
```

6.16.2.438 LDE_REPC_PCODE_12

```
#define LDE_REPC_PCODE_12 0x3D70
```

6.16.2.439 LDE_REPC_PCODE_13

```
#define LDE_REPC_PCODE_13 0x3AE0
```

6.16.2.440 LDE_REPC_PCODE_14

```
#define LDE_REPC_PCODE_14 0x35C2
```

6.16.2.441 LDE_REPC_PCODE_15

```
#define LDE_REPC_PCODE_15 0x2B84
```

6.16.2.442 LDE_REPC_PCODE_16

```
#define LDE_REPC_PCODE_16 0x35C2
```

6.16.2.443 LDE_REPC_PCODE_17

```
#define LDE_REPC_PCODE_17 0x3332
```

6.16.2.444 LDE_REPC_PCODE_18

```
#define LDE_REPC_PCODE_18 0x35C2
```

6.16.2.445 LDE_REPC_PCODE_19

```
#define LDE_REPC_PCODE_19 0x35C2
```

6.16.2.446 LDE_REPC_PCODE_2

```
#define LDE_REPC_PCODE_2 0x5998
```

6.16.2.447 LDE_REPC_PCODE_20

```
#define LDE_REPC_PCODE_20 0x47AE
```

6.16.2.448 LDE_REPC_PCODE_21

```
#define LDE_REPC_PCODE_21 0x3AE0
```

6.16.2.449 LDE_REPC_PCODE_22

```
#define LDE_REPC_PCODE_22 0x3850
```

6.16.2.450 LDE_REPC_PCODE_23

```
#define LDE_REPC_PCODE_23 0x30A2
```

6.16.2.451 LDE_REPC_PCODE_24

```
#define LDE_REPC_PCODE_24 0x3850
```

6.16.2.452 LDE_REPC_PCODE_3

```
#define LDE_REPC_PCODE_3 0x51EA
```

6.16.2.453 LDE_REPC_PCODE_4

```
#define LDE_REPC_PCODE_4 0x428E
```

6.16.2.454 LDE_REPC_PCODE_5

```
#define LDE_REPC_PCODE_5 0x451E
```

6.16.2.455 LDE_REPC_PCODE_6

```
#define LDE_REPC_PCODE_6 0x2E14
```

6.16.2.456 LDE_REPC_PCODE_7

```
#define LDE_REPC_PCODE_7 0x8000
```

6.16.2.457 LDE_REPC_PCODE_8

```
#define LDE_REPC_PCODE_8 0x51EA
```

6.16.2.458 LDE_REPC_PCODE_9

```
#define LDE_REPC_PCODE_9 0x28F4
```

6.16.2.459 LDE_RXANTD_LEN

```
#define LDE_RXANTD_LEN (2)
```

6.16.2.460 LDE_RXANTD_OFFSET

```
#define LDE_RXANTD_OFFSET 0x1804 /* 16-bit configuration register for setting the receive antenna delay */
```

6.16.2.461 LDE_THRESH_LEN

```
#define LDE_THRESH_LEN (2)
```

6.16.2.462 LDE_THRESH_OFFSET

```
#define LDE_THRESH_OFFSET 0x0000 /* 16-bit status register reporting the threshold that was used to find the first path */
```

6.16.2.463 OTP_ADDR

```
#define OTP_ADDR 0x04 /* 16-bit register used to select the address within the OTP memory block */
```

6.16.2.464 OTP_ADDR_LEN

```
#define OTP_ADDR_LEN (2)
```

6.16.2.465 OTP_ADDR_MASK

```
#define OTP_ADDR_MASK 0x07FF /* This 11-bit field specifies the address within OTP memory that will be accessed read or written. */
```

6.16.2.466 OTP_CTRL

```
#define OTP_CTRL 0x06 /* used to control the operation of the OTP memory */
```

6.16.2.467 OTP_CTRL_LDELOAD

```
#define OTP_CTRL_LDELOAD 0x8000 /* This bit forces a load of LDE microcode */
```

6.16.2.468 OTP_CTRL_LEN

```
#define OTP_CTRL_LEN (2)
```

6.16.2.469 OTP_CTRL_MASK

```
#define OTP_CTRL_MASK 0x8002
```

6.16.2.470 OTP_CTRL_OTPPROG

```
#define OTP_CTRL_OTPPROG 0x0040 /* Setting this bit will cause the contents of OTP_WDAT to be written to OTP_ADDR. */
```

6.16.2.471 OTP_CTRL_OTPRDEN

```
#define OTP_CTRL_OTPRDEN 0x0001 /* This bit forces the OTP into manual read mode */
```

6.16.2.472 OTP_CTRL_OTPREAD

```
#define OTP_CTRL_OTPREAD 0x0002 /* This bit commands a read operation from the address specified in the OTP_ADDR register */
```

6.16.2.473 OTP_IF_ID

```
#define OTP_IF_ID 0x2D /* One Time Programmable Memory Interface */
```

Bit definitions for register OTP_IF Refer to section 7.2.46 Register file: 0x2D OTP Memory Interface.

6.16.2.474 OTP_IF_LEN

```
#define OTP_IF_LEN (18)
```

6.16.2.475 OTP_RDAT

```
#define OTP_RDAT 0x0A /* 32-bit register. The data value read from an OTP location will appear here */
```

6.16.2.476 OTP_RDAT_LEN

```
#define OTP_RDAT_LEN (4)
```

6.16.2.477 OTP_SF

```
#define OTP_SF 0x12 /*8-bit special function register used to select and load special receiver operational parameter */
```

6.16.2.478 OTP_SF_LDO_KICK

```
#define OTP_SF_LDO_KICK 0x02 /* This bit when set initiates a load of the LDO tune code */
```

6.16.2.479 OTP_SF_LEN

```
#define OTP_SF_LEN (1)
```

6.16.2.480 OTP_SF_MASK

```
#define OTP_SF_MASK 0x63
```

6.16.2.481 OTP_SF_OPS_KICK

```
#define OTP_SF_OPS_KICK 0x01 /* This bit when set initiates a load of the operating parameter  
set selected by the OPS_SEL */
```

6.16.2.482 OTP_SF_OPS_SEL_L64

```
#define OTP_SF_OPS_SEL_L64 0x00 /* Operating parameter set selection: Length64 */
```

6.16.2.483 OTP_SF_OPS_SEL_MASK

```
#define OTP_SF_OPS_SEL_MASK 0x60
```

6.16.2.484 OTP_SF_OPS_SEL_SHFT

```
#define OTP_SF_OPS_SEL_SHFT 5
```

6.16.2.485 OTP_SF_OPS_SEL_TIGHT

```
#define OTP_SF_OPS_SEL_TIGHT 0x20 /* Operating parameter set selection: Tight */
```

6.16.2.486 OTP_SRDAT

```
#define OTP_SRDAT 0x0E /* 32-bit register. The data value stored in the OTP SR (0x400) location  
will appear here after power up */
```

6.16.2.487 OTP_SRDAT_LEN

```
#define OTP_SRDAT_LEN (4)
```

6.16.2.488 OTP_STAT

```
#define OTP_STAT 0x08
```

6.16.2.489 OTP_STAT_LEN

```
#define OTP_STAT_LEN (2)
```

6.16.2.490 OTP_STAT_MASK

```
#define OTP_STAT_MASK 0x0003
```

6.16.2.491 OTP_STAT_OTPPRGD

```
#define OTP_STAT_OTPPRGD 0x0001 /* OTP Programming Done */
```

6.16.2.492 OTP_STAT_OTPVPOK

```
#define OTP_STAT_OTPVPOK 0x0002 /* OTP Programming Voltage OK */
```

6.16.2.493 OTP_WDAT

```
#define OTP_WDAT 0x00 /* 32-bit register. The data value to be programmed into an OTP location */
```

6.16.2.494 OTP_WDAT_LEN

```
#define OTP_WDAT_LEN (4)
```

6.16.2.495 PANADR_ID

```
#define PANADR_ID 0x03 /* PAN ID (31:16) and Short Address (15:0) */
```

Bit definitions for register PANADR.

6.16.2.496 PANADR_LEN

```
#define PANADR_LEN (4)
```

6.16.2.497 PANADR_PAN_ID_MASK

```
#define PANADR_PAN_ID_MASK 0xFFFF00F0UL /* PAN Identifier */
```

6.16.2.498 PANADR_PAN_ID_OFFSET

```
#define PANADR_PAN_ID_OFFSET 2 /* In bytes */
```

6.16.2.499 PANADR_SHORT_ADDR_MASK

```
#define PANADR_SHORT_ADDR_MASK 0x0000FFFFUL /* Short Address */
```

6.16.2.500 PANADR_SHORT_ADDR_OFFSET

```
#define PANADR_SHORT_ADDR_OFFSET 0 /* In bytes */
```

6.16.2.501 PMSC_CTRL0_FACE

```
#define PMSC_CTRL0_FACE 0x00000040UL /* Force Accumulator Clock Enable */
```

6.16.2.502 PMSC_CTRL0_GPDCE

```
#define PMSC_CTRL0_GPDCE 0x00040000UL /* GPIO De-bounce Clock Enable */
```

6.16.2.503 PMSC_CTRL0_KHZCLEN

```
#define PMSC_CTRL0_KHZCLEN 0x00800000UL /* Kilohertz Clock Enable */
```

6.16.2.504 PMSC_CTRL0_LEN

```
#define PMSC_CTRL0_LEN (4)
```

6.16.2.505 PMSC_CTRL0_MASK

```
#define PMSC_CTRL0_MASK 0xF18F847FUL /* access mask to register PMSC_CTRL0 */
```

6.16.2.506 PMSC_CTRL0_OFFSET

```
#define PMSC_CTRL0_OFFSET 0x00
```

6.16.2.507 PMSC_CTRL0_PLL2_SEQ_EN

```
#define PMSC_CTRL0_PLL2_SEQ_EN 0x01000000UL /* Enable PLL2 on/off sequencing by SNIFF mode */
```

6.16.2.508 PMSC_CTRL0_RESET_ALL

```
#define PMSC_CTRL0_RESET_ALL 0x00 /* Assuming only 4th byte of the register is read */
```

6.16.2.509 PMSC_CTRL0_RESET_CLEAR

```
#define PMSC_CTRL0_RESET_CLEAR 0xF0 /* Assuming only 4th byte of the register is read */
```

6.16.2.510 PMSC_CTRL0_RESET_RX

```
#define PMSC_CTRL0_RESET_RX 0xE0 /* Assuming only 4th byte of the register is read */
```

6.16.2.511 PMSC_CTRL0_RXCLKS_125M

```
#define PMSC_CTRL0_RXCLKS_125M 0x00000008UL /* Force RX clock enable and sourced from the 125 MHz PLL clock */
```

6.16.2.512 PMSC_CTRL0_RXCLKS_19M

```
#define PMSC_CTRL0_RXCLKS_19M 0x00000004UL /* Force RX clock enable and sourced clock from the 19.2 MHz XT1 clock */
```

6.16.2.513 PMSC_CTRL0_RXCLKS_AUTO

```
#define PMSC_CTRL0_RXCLKS_AUTO 0x00000000UL /* The RX clock will be disabled until it is required for an RX operation */
```

6.16.2.514 PMSC_CTRL0_RXCLKS_OFF

```
#define PMSC_CTRL0_RXCLKS_OFF 0x0000000CUL /* Force RX clock off. */
```

6.16.2.515 PMSC_CTRL0_SOFTRESET_OFFSET

```
#define PMSC_CTRL0_SOFTRESET_OFFSET 3 /* In bytes */
```

6.16.2.516 PMSC_CTRL0_SYSCLKS_125M

```
#define PMSC_CTRL0_SYSCLKS_125M 0x00000002UL /* Force system clock to the 125 MHz PLL clock.  
*/
```

6.16.2.517 PMSC_CTRL0_SYSCLKS_19M

```
#define PMSC_CTRL0_SYSCLKS_19M 0x00000001UL /* Force system clock to be the 19.2 MHz XT↔  
I clock. */
```

6.16.2.518 PMSC_CTRL0_SYSCLKS_AUTO

```
#define PMSC_CTRL0_SYSCLKS_AUTO 0x00000000UL /* The system clock will run off the 19.2 MHz XT↔  
I clock until the PLL is calibrated and locked, then it will switch over the 125 MHz PLL clock  
*/
```

6.16.2.519 PMSC_CTRL0_TXCLKS_125M

```
#define PMSC_CTRL0_TXCLKS_125M 0x00000020UL /* Force TX clock enable and sourced from the 125  
MHz PLL clock */
```

6.16.2.520 PMSC_CTRL0_TXCLKS_19M

```
#define PMSC_CTRL0_TXCLKS_19M 0x00000010UL /* Force TX clock enable and sourced clock from the  
19.2 MHz XTI clock */
```

6.16.2.521 PMSC_CTRL0_TXCLKS_AUTO

```
#define PMSC_CTRL0_TXCLKS_AUTO 0x00000000UL /* The TX clock will be disabled until it is required  
for a TX operation */
```

6.16.2.522 PMSC_CTRL0_TXCLKS_OFF

```
#define PMSC_CTRL0_TXCLKS_OFF 0x00000030UL /* Force TX clock off */
```

6.16.2.523 PMSC_CTRL1_ARX2INIT

```
#define PMSC_CTRL1_ARX2INIT 0x00000002UL /* Automatic transition from receive mode into the I↔  
NIT state */
```

6.16.2.524 PMSC_CTRL1_ARXSLP

```
#define PMSC_CTRL1_ARXSLP 0x00001000UL /* this bit is set then the DW1000 will automatically  
transition into SLEEP mode after a receive attempt */
```

6.16.2.525 PMSC_CTRL1_ATXSLP

```
#define PMSC_CTRL1_ATXSLP 0x00000800UL /* If this bit is set then the DW1000 will automatically  
transition into SLEEP or DEEPSLEEP mode after transmission of a frame */
```

6.16.2.526 PMSC_CTRL1_KHZCLKDIV_MASK

```
#define PMSC_CTRL1_KHZCLKDIV_MASK 0xFC000000UL /* Kilohertz clock divisor */
```

6.16.2.527 PMSC_CTRL1_LDERUNE

```
#define PMSC_CTRL1_LDERUNE 0x00020000UL /* This bit enables the running of the LDE algorithm */
```

6.16.2.528 PMSC_CTRL1_LEN

```
#define PMSC_CTRL1_LEN (4)
```

6.16.2.529 PMSC_CTRL1_MASK

```
#define PMSC_CTRL1_MASK 0xFC02F802UL /* access mask to register PMSC_CTRL1 */
```

6.16.2.530 PMSC_CTRL1_OFFSET

```
#define PMSC_CTRL1_OFFSET 0x04
```

6.16.2.531 PMSC_CTRL1_PKTSEQ_DISABLE

```
#define PMSC_CTRL1_PKTSEQ_DISABLE 0x00 /* writing this to PMSC CONTROL 1 register (bits 10-3)  
disables PMSC control of analog RF subsystems */
```

6.16.2.532 PMSC_CTRL1_PKTSEQ_ENABLE

```
#define PMSC_CTRL1_PKTSEQ_ENABLE 0xE7 /* writing this to PMSC CONTROL 1 register (bits 10-3)  
enables PMSC control of analog RF subsystems */
```

6.16.2.533 PMSC_CTRL1_PLLSYN

```
#define PMSC_CTRL1_PLLSYN 0x00008000UL /* This enables a special 1 GHz clock used for some  
external SYNC modes */
```

6.16.2.534 PMSC_CTRL1_SNOZE

```
#define PMSC_CTRL1_SNOZE 0x00002000UL /* Snooze Enable */
```

6.16.2.535 PMSC_CTRL1_SNOZR

```
#define PMSC_CTRL1_SNOZR 0x00004000UL /* The SNOZR bit is set to allow the snooze timer to  
repeat twice */
```

6.16.2.536 PMSC_ID

```
#define PMSC_ID 0x36 /* Power Management System Control Block */
```

Bit definitions for register PMSC.

6.16.2.537 PMSC_LED_C_BLINK_NOW_ALL

```
#define PMSC_LED_C_BLINK_NOW_ALL 0x000F0000UL
```

6.16.2.538 PMSC_LED_C_BLINK_TIM_MASK

```
#define PMSC_LED_C_BLINK_TIM_MASK 0x000000FFUL /* This field determines how long the LEDs remain lit after an event that causes them to be set on. */
```

6.16.2.539 PMSC_LED_C_BLINK_TIME_DEF

```
#define PMSC_LED_C_BLINK_TIME_DEF 0x10
```

6.16.2.540 PMSC_LED_C_BLNKEN

```
#define PMSC_LED_C_BLNKEN 0x00000100UL /* Blink Enable. When this bit is set to 1 the LE←D blink feature is enabled. */
```

6.16.2.541 PMSC_LED_C_LEN

```
#define PMSC_LED_C_LEN (4)
```

6.16.2.542 PMSC_LED_C_MASK

```
#define PMSC_LED_C_MASK 0x000001FFUL /* 32-bit LED control register. */
```

6.16.2.543 PMSC_LED_C_OFFSET

```
#define PMSC_LED_C_OFFSET 0x28
```

6.16.2.544 PMSC_LEN

```
#define PMSC_LEN (48)
```

6.16.2.545 PMSC_RES1_OFFSET

```
#define PMSC_RES1_OFFSET 0x08
```

6.16.2.546 PMSC_RES2_OFFSET

```
#define PMSC_RES2_OFFSET 0x10
```

6.16.2.547 PMSC_RES3_OFFSET

```
#define PMSC_RES3_OFFSET 0x24
```

6.16.2.548 PMSC_SNOZT_LEN

```
#define PMSC_SNOZT_LEN (1)
```

6.16.2.549 PMSC_SNOZT_OFFSET

```
#define PMSC_SNOZT_OFFSET 0x0C /* PMSC Snooze Time Register */
```

6.16.2.550 PMSC_TXFINESEQ_DISABLE

```
#define PMSC_TXFINESEQ_DISABLE 0x0 /* Writing this disables fine grain sequencing in the transmitter */
```

6.16.2.551 PMSC_TXFINESEQ_ENABLE

```
#define PMSC_TXFINESEQ_ENABLE 0x0B74 /* Writing this enables fine grain sequencing in the transmitter */
```

6.16.2.552 PMSC_TXFINESEQ_OFFSET

```
#define PMSC_TXFINESEQ_OFFSET 0x26
```

6.16.2.553 REG_05_ID_RESERVED

```
#define REG_05_ID_RESERVED 0x05
```

Bit definitions for register 0x05.

6.16.2.554 REG_07_ID_RESERVED

```
#define REG_07_ID_RESERVED 0x07
```

Bit definitions for register 0x07.

6.16.2.555 REG_0B_ID_RESERVED

```
#define REG_0B_ID_RESERVED 0x0B
```

Bit definitions for register 0x08.

6.16.2.556 REG_16_ID_RESERVED

```
#define REG_16_ID_RESERVED 0x16
```

Bit definitions for register.

6.16.2.557 REG_1B_ID_RESERVED

```
#define REG_1B_ID_RESERVED 0x1B
```

Bit definitions for register 0x1B 0x1C.

6.16.2.558 REG_1C_ID_RESERVED

```
#define REG_1C_ID_RESERVED 0x1C
```

6.16.2.559 REG_20_ID_RESERVED

```
#define REG_20_ID_RESERVED 0x20
```

Bit definitions for register 0x20.

6.16.2.560 REG_22_ID_RESERVED

```
#define REG_22_ID_RESERVED 0x22
```

Bit definitions for register.

6.16.2.561 REG_29_ID_RESERVED

```
#define REG_29_ID_RESERVED 0x29
```

Bit definitions for register.

6.16.2.562 REG_30_ID_RESERVED

```
#define REG_30_ID_RESERVED 0x30
```

Bit definitions for register 0x30-0x35 Please take care not to write to these registers as doing so may cause the DW1000 to malfunction.

6.16.2.563 REG_31_ID_RESERVED

```
#define REG_31_ID_RESERVED 0x31
```

6.16.2.564 REG_32_ID_RESERVED

```
#define REG_32_ID_RESERVED 0x32
```

6.16.2.565 REG_33_ID_RESERVED

```
#define REG_33_ID_RESERVED 0x33
```

6.16.2.566 REG_34_ID_RESERVED

```
#define REG_34_ID_RESERVED 0x34
```

6.16.2.567 REG_35_ID_RESERVED

```
#define REG_35_ID_RESERVED 0x35
```

6.16.2.568 REG_37_ID_RESERVED

```
#define REG_37_ID_RESERVED 0x37
```

Bit definitions for register 0x37-0x3F Please take care not to write to these registers as doing so may cause the DW1000 to malfunction.

6.16.2.569 REG_38_ID_RESERVED

```
#define REG_38_ID_RESERVED 0x38
```

6.16.2.570 REG_39_ID_RESERVED

```
#define REG_39_ID_RESERVED 0x39
```

6.16.2.571 REG_3A_ID_RESERVED

```
#define REG_3A_ID_RESERVED 0x3A
```

6.16.2.572 REG_3B_ID_RESERVED

```
#define REG_3B_ID_RESERVED 0x3B
```

6.16.2.573 REG_3C_ID_RESERVED

```
#define REG_3C_ID_RESERVED 0x3C
```

6.16.2.574 REG_3D_ID_RESERVED

```
#define REG_3D_ID_RESERVED 0x3D
```

6.16.2.575 REG_3E_ID_RESERVED

```
#define REG_3E_ID_RESERVED 0x3E
```

6.16.2.576 REG_3F_ID_RESERVED

```
#define REG_3F_ID_RESERVED 0x3F
```

6.16.2.577 RF_CONF_ID

```
#define RF_CONF_ID 0x28 /* Analog RF Configuration */
```

Bit definitions for register RF_CONF Analog RF Configuration block Refer to section 7.2.41 Register file: 0x28 Analog RF configuration block.

6.16.2.578 RF_CONF_LEN

```
#define RF_CONF_LEN (58)
```

6.16.2.579 RF_CONF_PGMIXBIASEN_MASK

```
#define RF_CONF_PGMIXBIASEN_MASK 0x0000A700UL /* Enable TX mixer bias and pulse gen */
```

6.16.2.580 RF_CONF_PLLEN_MASK

```
#define RF_CONF_PLLEN_MASK 0x0000E000UL /* enable PLLs */
```

6.16.2.581 RF_CONF_RXEN_MASK

```
#define RF_CONF_RXEN_MASK 0x00200000UL /* RX enable */
```

6.16.2.582 RF_CONF_TXALLEN_MASK

```
#define RF_CONF_TXALLEN_MASK (RF_CONF_TXEN_MASK | RF_CONF_TXPOW_MASK | RF_CONF_PLLEN_MASK |  
RF_CONF_TXBLOCKSEN_MASK)
```

6.16.2.583 RF_CONF_TXBLOCKSEN_MASK

```
#define RF_CONF_TXBLOCKSEN_MASK 0x00001F00UL /* enable TX blocks */
```

6.16.2.584 RF_CONF_TXEN_MASK

```
#define RF_CONF_TXEN_MASK 0x00400000UL /* TX enable */
```

6.16.2.585 RF_CONF_TXPLLPOWEN_MASK

```
#define RF_CONF_TXPLLPOWEN_MASK (RF_CONF_PLEN_MASK | RF_CONF_TXPOW_MASK)
```

6.16.2.586 RF_CONF_TXPOW_MASK

```
#define RF_CONF_TXPOW_MASK 0x001F0000UL /* turn on power all LDOS */
```

6.16.2.587 RF_RXCTRLH_LEN

```
#define RF_RXCTRLH_LEN (1)
```

6.16.2.588 RF_RXCTRLH_NBW

```
#define RF_RXCTRLH_NBW 0xD8 /* RXCTRLH value for narrow bandwidth channels */
```

6.16.2.589 RF_RXCTRLH_OFFSET

```
#define RF_RXCTRLH_OFFSET 0x0B /* Analog RX Control Register */
```

6.16.2.590 RF_RXCTRLH_WBW

```
#define RF_RXCTRLH_WBW 0xBC /* RXCTRLH value for wide bandwidth channels */
```

6.16.2.591 RF_STATUS_OFFSET

```
#define RF_STATUS_OFFSET 0x2C
```

6.16.2.592 RF_TXCTRL_CH1

```
#define RF_TXCTRL_CH1 0x00005C40UL /* 32-bit value to program to Sub-Register 0x28:0C RF_TXC↔TRL */
```

6.16.2.593 RF_TXCTRL_CH2

```
#define RF_TXCTRL_CH2 0x00045CA0UL /* 32-bit value to program to Sub-Register 0x28:0C RF_TXC↔TRL */
```

6.16.2.594 RF_TXCTRL_CH3

```
#define RF_TXCTRL_CH3 0x00086CC0UL /* 32-bit value to program to Sub-Register 0x28:0C RF_TXC↔TRL */
```

6.16.2.595 RF_TXCTRL_CH4

```
#define RF_TXCTRL_CH4 0x00045C80UL /* 32-bit value to program to Sub-Register 0x28:0C RF_TXC↔TRL */
```

6.16.2.596 RF_TXCTRL_CH5

```
#define RF_TXCTRL_CH5 0x001E3FE0UL /* 32-bit value to program to Sub-Register 0x28:0C RF_TXC↔TRL */
```

6.16.2.597 RF_TXCTRL_CH7

```
#define RF_TXCTRL_CH7 0x001E7DE0UL /* 32-bit value to program to Sub-Register 0x28:0C RF_TXC↔TRL */
```

6.16.2.598 RF_TXCTRL_LEN

```
#define RF_TXCTRL_LEN (4)
```

6.16.2.599 RF_TXCTRL_OFFSET

```
#define RF_TXCTRL_OFFSET 0x0C /* Analog TX Control Register */
```

6.16.2.600 RF_TXCTRL_TXMTUNE_MASK

```
#define RF_TXCTRL_TXMTUNE_MASK 0x0000001E0UL /* Transmit mixer tuning register */
```

6.16.2.601 RF_TXCTRL_TXTXMQ_MASK

```
#define RF_TXCTRL_TXTXMQ_MASK 0x000000E00UL /* Transmit mixer Q-factor tuning register */
```

6.16.2.602 RX_BUFFER_ID

```
#define RX_BUFFER_ID 0x11 /* Receive Data Buffer (in double buffer set) */
```

Bit definitions for register RX_BUFFER.

6.16.2.603 RX_BUFFER_LEN

```
#define RX_BUFFER_LEN (1024)
```

6.16.2.604 RX_EQUAL_CIR_MXG_MASK

```
#define RX_EQUAL_CIR_MXG_MASK 0xFFFF0000000000ULL /* Channel Impulse Response Max Growth */
```

6.16.2.605 RX_EQUAL_CIR_MXG_SHIFT

```
#define RX_EQUAL_CIR_MXG_SHIFT (48)
```

6.16.2.606 RX_EQUAL_FP_AMPL2_MASK

```
#define RX_EQUAL_FP_AMPL2_MASK 0xFFFF0000ULL /* First Path Amplitude point 2 - magnitude of  
2nd point after Ceiling(FP_Index) */
```

6.16.2.607 RX_EQUAL_FP_AMPL2_SHIFT

```
#define RX_EQUAL_FP_AMPL2_SHIFT (16)
```

6.16.2.608 RX_EQUAL_FP_AMPL3_MASK

```
#define RX_EQUAL_FP_AMPL3_MASK 0x0000FFFF00000000ULL /* First Path Amplitude point 3 - magnitude  
of 1st point after Ceiling(FP_Index) */
```

6.16.2.609 RX_EQUAL_FP_AMPL3_SHIFT

```
#define RX_EQUAL_FP_AMPL3_SHIFT (32)
```

6.16.2.610 RX_EQUAL_STD_NOISE_MASK

```
#define RX_EQUAL_STD_NOISE_MASK 0x0000FFFFULL /* Standard Deviation of Noise */
```

6.16.2.611 RX_EQUAL_STD_NOISE_SHIFT

```
#define RX_EQUAL_STD_NOISE_SHIFT (0)
```

6.16.2.612 RX_FINFO_ID

```
#define RX_FINFO_ID 0x10 /* RX Frame Information (in double buffer set) */
```

Bit definitions for register RX_FINFO.

6.16.2.613 RX_INFO_LEN

```
#define RX_INFO_LEN (4)
```

6.16.2.614 RX_INFO_MASK_32

```
#define RX_INFO_MASK_32 0xFFFFFBFFUL /* System event Status Register access mask (all unused fields should always be written as zero) */
```

6.16.2.615 RX_INFO_OFFSET

```
#define RX_INFO_OFFSET 0x00
```

6.16.2.616 RX_INFO_RNG

```
#define RX_INFO_RNG 0x00008000UL /* Receiver Ranging. Ranging bit in the received PHY header identifying the frame as a ranging packet. */
```

6.16.2.617 RX_INFO_RNG_SHIFT

```
#define RX_INFO_RNG_SHIFT (15)
```

6.16.2.618 RX_INFO_RXBR_110k

```
#define RX_INFO_RXBR_110k 0x00000000UL /* Received bit rate = 110 kbps */
```

6.16.2.619 RX_INFO_RXBR_6M

```
#define RX_INFO_RXBR_6M 0x00004000UL /* Received bit rate = 6.8 Mbps */
```

6.16.2.620 RX_FINFO_RXBR_850k

```
#define RX_FINFO_RXBR_850k 0x00002000UL /* Received bit rate = 850 kbps */
```

6.16.2.621 RX_FINFO_RXBR_MASK

```
#define RX_FINFO_RXBR_MASK 0x00006000UL /* Receive Bit Rate report. This field reports the received bit rate */
```

6.16.2.622 RX_FINFO_RXBR_SHIFT

```
#define RX_FINFO_RXBR_SHIFT (13)
```

6.16.2.623 RX_FINFO_RXFL_MASK_1023

```
#define RX_FINFO_RXFL_MASK_1023 0x000003FFUL /* Receive Frame Length Extension (0 to 1023) */
```

6.16.2.624 RX_FINFO_RXFLE_MASK

```
#define RX_FINFO_RXFLE_MASK 0x00000380UL /* Receive Frame Length Extension (0 to 7)<<7 */
```

6.16.2.625 RX_FINFO_RXFLEN_MASK

```
#define RX_FINFO_RXFLEN_MASK 0x0000007FUL /* Receive Frame Length (0 to 127) */
```

6.16.2.626 RX_FINFO_RXNSPL_MASK

```
#define RX_FINFO_RXNSPL_MASK 0x00001800UL /* Receive Non-Standard Preamble Length */
```

6.16.2.627 RX_FINFO_RXPACC_MASK

```
#define RX_FINFO_RXPACC_MASK 0xFFFF0000UL /* Preamble Accumulation Count */
```

6.16.2.628 RX_FINFO_RXPACC_SHIFT

```
#define RX_FINFO_RXPACC_SHIFT (20)
```

6.16.2.629 RX_FINFO_RXPEL_1024

```
#define RX_FINFO_RXPEL_1024 0x00080000UL /* Receive Preamble length = 1024 */
```

6.16.2.630 RX_FINFO_RXPEL_128

```
#define RX_FINFO_RXPEL_128 0x00040800UL /* Receive Preamble length = 128 */
```

6.16.2.631 RX_FINFO_RXPEL_1536

```
#define RX_FINFO_RXPEL_1536 0x00080800UL /* Receive Preamble length = 1536 */
```

6.16.2.632 RX_FINFO_RXPEL_2048

```
#define RX_FINFO_RXPEL_2048 0x00081000UL /* Receive Preamble length = 2048 */
```

6.16.2.633 RX_FINFO_RXPEL_256

```
#define RX_FINFO_RXPEL_256 0x00041000UL /* Receive Preamble length = 256 */
```

6.16.2.634 RX_FINFO_RXPEL_4096

```
#define RX_FINFO_RXPEL_4096 0x000C0000UL /* Receive Preamble length = 4096 */
```

6.16.2.635 RX_INFO_RXPEL_512

```
#define RX_INFO_RXPEL_512 0x00041800UL /* Receive Preamble length = 512 */
```

6.16.2.636 RX_INFO_RXPEL_64

```
#define RX_INFO_RXPEL_64 0x00040000UL /* Receive Preamble length = 64 */
```

6.16.2.637 RX_INFO_RXPEL_MASK

```
#define RX_INFO_RXPEL_MASK 0x000C1800UL /* Receive Preamble Length = RXPSR+RXNSPL */
```

6.16.2.638 RX_INFO_RXPRF_16M

```
#define RX_INFO_RXPRF_16M 0x00010000UL /* PRF being employed in the receiver = 16M */
```

6.16.2.639 RX_INFO_RXPRF_64M

```
#define RX_INFO_RXPRF_64M 0x00020000UL /* PRF being employed in the receiver = 64M */
```

6.16.2.640 RX_INFO_RXPRF_MASK

```
#define RX_INFO_RXPRF_MASK 0x00030000UL /* RX Pulse Repetition Rate report */
```

6.16.2.641 RX_INFO_RXPRF_SHIFT

```
#define RX_INFO_RXPRF_SHIFT (16)
```

6.16.2.642 RX_INFO_RXPSR_MASK

```
#define RX_INFO_RXPSR_MASK 0x000C0000UL /* RX Preamble Repetition. 00 = 16 symbols, 01 = 64 symbols, 10 = 1024 symbols, 11 = 4096 symbols */
```

6.16.2.643 RX_FQUAL_ID

```
#define RX_FQUAL_ID 0x12 /* Rx Frame Quality information (in double buffer set) */
```

Bit definitions for register RX_FQUAL.

6.16.2.644 RX_FQUAL_LEN

```
#define RX_FQUAL_LEN (8) /* note 64 bit register*/
```

6.16.2.645 RX_FWTO_ID

```
#define RX_FWTO_ID 0x0C /* Receive Frame Wait Timeout Period */
```

Bit definitions for register RX_FWTO.

6.16.2.646 RX_FWTO_LEN

```
#define RX_FWTO_LEN (2)
```

6.16.2.647 RX_FWTO_MASK

```
#define RX_FWTO_MASK 0xFFFF
```

6.16.2.648 RX_FWTO_OFFSET

```
#define RX_FWTO_OFFSET 0x00
```

6.16.2.649 RX_SNIFF_ID

```
#define RX_SNIFF_ID 0x1D /* Sniff Mode Configuration */
```

Bit definitions for register RX_SNIFF Sniff Mode Configuration or Pulsed Preamble Reception Configuration.

6.16.2.650 RX_SNIFF_LEN

```
#define RX_SNIFF_LEN (4)
```

6.16.2.651 RX_SNIFF_MASK

```
#define RX_SNIFF_MASK 0x0000FF0FUL /* */
```

6.16.2.652 RX_SNIFF_OFFSET

```
#define RX_SNIFF_OFFSET 0x00
```

6.16.2.653 RX_SNIFF_SNIFF_OFFT_MASK

```
#define RX_SNIFF_SNIFF_OFFT_MASK 0x0000FF00UL /* SNIFF Mode OFF time specified in units of  
approximately 1mks, or 128 system clock cycles.*/
```

6.16.2.654 RX_SNIFF_SNIFF_ONT_MASK

```
#define RX_SNIFF_SNIFF_ONT_MASK 0x0000000FUL /* SNIFF Mode ON time. Specified in units of PAC  
*/
```

6.16.2.655 RX_STAMP_LEN

```
#define RX_STAMP_LEN RX_TIME_RX_STAMP_LEN
```

6.16.2.656 RX_TIME_FP_AMPL1_OFFSET

```
#define RX_TIME_FP_AMPL1_OFFSET (7) /* byte 7..8 16 bit First Path Amplitude - magnitude of  
3rd point after Ceiling(FP_Index) */
```

6.16.2.657 RX_TIME_FP_INDEX_OFFSET

```
#define RX_TIME_FP_INDEX_OFFSET (5) /* byte 5..6 16 bit First path index. */
```

6.16.2.658 RX_TIME_FP_RAWST_OFFSET

```
#define RX_TIME_FP_RAWST_OFFSET (9) /* byte 9..13 40 bit Raw Timestamp for the frame */
```

6.16.2.659 RX_TIME_ID

```
#define RX_TIME_ID 0x15 /* Receive Message Time of Arrival (in double buffer set) */
```

Bit definitions for register RX_TIME.

6.16.2.660 RX_TIME_LLEN

```
#define RX_TIME_LLEN (14)
```

6.16.2.661 RX_TIME_RX_STAMP_LEN

```
#define RX_TIME_RX_STAMP_LEN (5) /* read only 5 bytes (the adjusted timestamp (40:0)) */
```

6.16.2.662 RX_TIME_RX_STAMP_OFFSET

```
#define RX_TIME_RX_STAMP_OFFSET (0) /* byte 0..4 40 bit Reports the fully adjusted time of reception. */
```

6.16.2.663 RX_TTCKI_ID

```
#define RX_TTCKI_ID 0x13 /* Receiver Time Tracking Interval (in double buffer set) */
```

Bit definitions for register RX_TTCKI The value here is the interval over which the timing offset reported in the RXTOFS field of Register file: 0x14 RX_TTCKO is measured. The clock offset is calculated by dividing RXTTCKI by RXTOFS. The value in RXTTCKI will take just one of two values depending on the PRF: 0x01F00000 @ 16 MHz PRF, and 0x01FC0000 @ 64 MHz PRF.

6.16.2.664 RX_TTCKI_LEN

```
#define RX_TTCKI_LEN (4)
```

6.16.2.665 RX_TTCKO_ID

```
#define RX_TTCKO_ID 0x14 /* Receiver Time Tracking Offset (in double buffer set) */
```

Bit definitions for register RX_TTCKO.

6.16.2.666 RX_TTCKO_LEN

```
#define RX_TTCKO_LEN (5) /* Note 40 bit register */
```

6.16.2.667 RX_TTCKO_MASK_32

```
#define RX_TTCKO_MASK_32 0xFF07FFFFUL /* Receiver Time Tracking Offset access mask (all unused fields should always be written as zero) */
```

6.16.2.668 RX_TTCKO_RCPHASE_MASK

```
#define RX_TTCKO_RCPHASE_MASK 0x7F000000000ULL /* This 7-bit field reports the receive carrier phase adjustment at time the ranging timestamp is made. */
```

6.16.2.669 RX_TTCKO_RSMPDEL_MASK

```
#define RX_TTCKO_RSMPDEL_MASK 0xFF000000UL /* This 8-bit field reports an internal re-sampler delay value */
```

6.16.2.670 RX_TTCKO_RXTOFS_MASK

```
#define RX_TTCKO_RXTOFS_MASK 0x0007FFFFUL /* RX time tracking offset. This RXTOFS value is a 19-bit signed quantity*/
```

6.16.2.671 SNIFF_OFFT_MASK

```
#define SNIFF_OFFT_MASK RX_SNIFF_SNIFFT_MASK
```

6.16.2.672 SNIFF_ONT_MASK

```
#define SNIFF_ONT_MASK RX_SNIFF_SNIFF_ONT_MASK
```

6.16.2.673 STD_NOISE_MASK

```
#define STD_NOISE_MASK RX_EQUAL_STD_NOISE_MASK
```

6.16.2.674 STD_NOISE_SHIFT

```
#define STD_NOISE_SHIFT RX_EQUAL_STD_NOISE_SHIFT
```

6.16.2.675 SYS_CFG_AACKPEND

```
#define SYS_CFG_AACKPEND 0x80000000UL /* Automatic Acknowledgement Pending bit control */
```

6.16.2.676 SYS_CFG_AUTOACK

```
#define SYS_CFG_AUTOACK 0x40000000UL /* Automatic Acknowledgement Enable */
```

6.16.2.677 SYS_CFG_DIS_DRXB

```
#define SYS_CFG_DIS_DRXB 0x00001000UL /* Disable Double RX Buffer */
```

6.16.2.678 SYS_CFG_DIS_FCE

```
#define SYS_CFG_DIS_FCE 0x00000800UL /* Disable frame check error handling */
```

6.16.2.679 SYS_CFG_DIS_PHE

```
#define SYS_CFG_DIS_PHE 0x00002000UL /* Disable receiver abort on PHR error */
```

6.16.2.680 SYS_CFG_DIS_RSDE

```
#define SYS_CFG_DIS_RSDE 0x00004000UL /* Disable Receiver Abort on RSD error */
```

6.16.2.681 SYS_CFG_DIS_STXP

```
#define SYS_CFG_DIS_STXP 0x00040000UL /* Disable Smart TX Power control */
```

6.16.2.682 SYS_CFG_FCS_INIT2F

```
#define SYS_CFG_FCS_INIT2F 0x00008000UL /* initial seed value for the FCS generation and checking function */
```

6.16.2.683 SYS_CFG_FF_ALL_EN

```
#define SYS_CFG_FF_ALL_EN 0x000001FEUL /* Frame filtering options all frames allowed */
```

6.16.2.684 SYS_CFG_FFA4

```
#define SYS_CFG_FFA4 0x00000080UL /* Frame Filtering Allow frames with frame type field of 4, (binary 100) */
```

6.16.2.685 SYS_CFG_FFA5

```
#define SYS_CFG_FFA5 0x00000100UL /* Frame Filtering Allow frames with frame type field of 5, (binary 101) */
```

6.16.2.686 SYS_CFG_FFAA

```
#define SYS_CFG_FFAA 0x00000010UL /* Frame Filtering Allow Acknowledgment frame reception */
```

6.16.2.687 SYS_CFG_FFAB

```
#define SYS_CFG_FFAB 0x00000004UL /* Frame Filtering Allow Beacon frame reception */
```

6.16.2.688 SYS_CFG_FFAD

```
#define SYS_CFG_FFAD 0x00000008UL /* Frame Filtering Allow Data frame reception */
```

6.16.2.689 SYS_CFG_FFAM

```
#define SYS_CFG_FFAM 0x00000020UL /* Frame Filtering Allow MAC command frame reception */
```

6.16.2.690 SYS_CFG_FFAR

```
#define SYS_CFG_FFAR 0x00000040UL /* Frame Filtering Allow Reserved frame types */
```

6.16.2.691 SYS_CFG_FFBC

```
#define SYS_CFG_FFBC 0x00000002UL /* Frame Filtering Behave as a Co-ordinator */
```

6.16.2.692 SYS_CFG_FFE

```
#define SYS_CFG_FFE 0x00000001UL /* Frame Filtering Enable. This bit enables the frame filtering functionality */
```

6.16.2.693 SYS_CFG_HIRQ_POL

```
#define SYS_CFG_HIRQ_POL 0x00000200UL /* Host interrupt polarity */
```

6.16.2.694 SYS_CFG_ID

```
#define SYS_CFG_ID 0x04 /* System Configuration (31:0) */
```

Bit definitions for register SYS_CFG.

6.16.2.695 SYS_CFG_LEN

```
#define SYS_CFG_LEN (4)
```

6.16.2.696 SYS_CFG_MASK

```
#define SYS_CFG_MASK 0xF047FFFFUL /* access mask to SYS_CFG_ID */
```

6.16.2.697 SYS_CFG_PHR_MODE_00

```
#define SYS_CFG_PHR_MODE_00 0x00000000UL /* Standard Frame mode */
```

6.16.2.698 SYS_CFG_PHR_MODE_11

```
#define SYS_CFG_PHR_MODE_11 0x00030000UL /* Long Frames mode */
```

6.16.2.699 SYS_CFG_PHR_MODE_SHFT

```
#define SYS_CFG_PHR_MODE_SHFT 16
```

6.16.2.700 **SYS_CFG_RXAUTR**

```
#define SYS_CFG_RXAUTR 0x20000000UL /* Receiver Auto-Re-enable. This bit is used to cause the receiver to re-enable automatically */
```

6.16.2.701 **SYS_CFG_RXM110K**

```
#define SYS_CFG_RXM110K 0x00400000UL /* Receiver Mode 110 kbps data rate */
```

6.16.2.702 **SYS_CFG_RXWTOE**

```
#define SYS_CFG_RXWTOE 0x10000000UL /* Receive Wait Timeout Enable. */
```

6.16.2.703 **SYS_CFG_SPI_EDGE**

```
#define SYS_CFG_SPI_EDGE 0x00000400UL /* SPI data launch edge */
```

6.16.2.704 **SYS_CTRL_CANSFCS**

```
#define SYS_CTRL_CANSFCS 0x00000008UL /* Cancel Suppression of auto-FCS transmission (on the current frame) */
```

6.16.2.705 **SYS_CTRL_HRBT**

```
#define SYS_CTRL_HRBT (SYS\_CTRL\_HSRBT\_TOGGLE)
```

6.16.2.706 **SYS_CTRL_HRBT_OFFSET**

```
#define SYS_CTRL_HRBT_OFFSET (3)
```

6.16.2.707 SYS_CTRL_HSRBToggle

```
#define SYS_CTRL_HSRBToggle 0x01000000UL /* Host side receiver buffer pointer toggle - toggles  
0/1 host side data set pointer */
```

6.16.2.708 SYS_CTRL_ID

```
#define SYS_CTRL_ID 0x0D /* System Control Register */
```

Bit definitions for register SYS_CTRL.

6.16.2.709 SYS_CTRL_LEN

```
#define SYS_CTRL_LEN (4)
```

6.16.2.710 SYS_CTRL_MASK_32

```
#define SYS_CTRL_MASK_32 0x010003CFUL /* System Control Register access mask (all unused fields  
should always be written as zero) */
```

6.16.2.711 SYS_CTRL_OFFSET

```
#define SYS_CTRL_OFFSET 0x00
```

6.16.2.712 SYS_CTRL_RXDLYE

```
#define SYS_CTRL_RXDLYE 0x00000200UL /* Receiver Delayed Enable (Enables Receiver when SY_TIM←  
E[0x??] == RXD_TIME[0x??] CHECK comment*)
```

6.16.2.713 SYS_CTRL_RXENAB

```
#define SYS_CTRL_RXENAB 0x00000100UL /* Enable Receiver Now */
```

6.16.2.714 **SYS_CTRL_SFCST**

```
#define SYS_CTRL_SFCST 0x00000001UL /* Suppress Auto-FCS Transmission (on this frame) */
```

6.16.2.715 **SYS_CTRL_TRXOFF**

```
#define SYS_CTRL_TRXOFF 0x00000040UL /* Transceiver Off. Force Transciever OFF abort TX or RX immediately */
```

6.16.2.716 **SYS_CTRL_TXDLYS**

```
#define SYS_CTRL_TXDLYS 0x00000004UL /* Transmitter Delayed Sending (initiates sending when S←YS_TIME == TXD_TIME) */
```

6.16.2.717 **SYS_CTRL_TXSTRT**

```
#define SYS_CTRL_TXSTRT 0x00000002UL /* Start Transmitting Now */
```

6.16.2.718 **SYS_CTRL_WAIT4RESP**

```
#define SYS_CTRL_WAIT4RESP 0x00000080UL /* Wait for Response */
```

6.16.2.719 **SYS_MASK_ID**

```
#define SYS_MASK_ID 0x0E /* System Event Mask Register */
```

Bit definitions for register SYS_MASK.

6.16.2.720 **SYS_MASK_LEN**

```
#define SYS_MASK_LEN (4)
```

6.16.2.721 SYS_MASK_MAAT

```
#define SYS_MASK_MAAT 0x00000008UL /* Mask automatic acknowledge trigger event */
```

6.16.2.722 SYS_MASK_MAFFREJ

```
#define SYS_MASK_MAFFREJ 0x20000000UL /* Mask Automatic Frame Filtering rejection event */
```

6.16.2.723 SYS_MASK_MASK_32

```
#define SYS_MASK_MASK_32 0x3FF7FFEUL /* System Event Mask Register access mask (all unused fields should always be written as zero) */
```

6.16.2.724 SYS_MASK_MCPLLLL

```
#define SYS_MASK_MCPLLLL 0x02000000UL /* Mask Clock PLL Loosing Lock warning event */
```

6.16.2.725 SYS_MASK_MCPLOCK

```
#define SYS_MASK_MCPLOCK 0x00000002UL /* Mask clock PLL lock event */
```

6.16.2.726 SYS_MASK_MESYNCR

```
#define SYS_MASK_MESYNCR 0x00000004UL /* Mask clock PLL lock event */
```

6.16.2.727 SYS_MASK_MGPIOIRQ

```
#define SYS_MASK_MGPIOIRQ 0x00400000UL /* Mask GPIO interrupt event */
```

6.16.2.728 SYS_MASK_MHPDWARN

```
#define SYS_MASK_MHPDWARN 0x08000000UL /* Mask Half Period Delay Warning event */
```

6.16.2.729 SYS_MASK_MLDDEDONE

```
#define SYS_MASK_MLDDEDONE 0x00000400UL /* Mask LDE processing done event */
```

6.16.2.730 SYS_MASK_MLDEERR

```
#define SYS_MASK_MLDEERR 0x00040000UL /* Mask leading edge detection processing error event */
```

6.16.2.731 SYS_MASK_MRFPLLLL

```
#define SYS_MASK_MRFPLLLL 0x01000000UL /* Mask RF PLL Loosing Lock warning event */
```

6.16.2.732 SYS_MASK_MRXdFR

```
#define SYS_MASK_MRXdFR 0x00002000UL /* Mask receiver data frame ready event */
```

6.16.2.733 SYS_MASK_MRxFCE

```
#define SYS_MASK_MRxFCE 0x00008000UL /* Mask receiver FCS error event */
```

6.16.2.734 SYS_MASK_MRxFCG

```
#define SYS_MASK_MRxFCG 0x00004000UL /* Mask receiver FCS good event */
```

6.16.2.735 SYS_MASK_MRxoVRR

```
#define SYS_MASK_MRxoVRR 0x00100000UL /* Mask Receiver Overrun event */
```

6.16.2.736 SYS_MASK_MRXPHD

```
#define SYS_MASK_MRXPHD 0x00000800UL /* Mask receiver PHY header detect event */
```

6.16.2.737 SYS_MASK_MRXPHE

```
#define SYS_MASK_MRXPHE 0x00001000UL /* Mask receiver PHY header error event */
```

6.16.2.738 SYS_MASK_MRXP RD

```
#define SYS_MASK_MRXP RD 0x00000100UL /* Mask receiver preamble detected event */
```

6.16.2.739 SYS_MASK_MRXP TO

```
#define SYS_MASK_MRXP TO 0x00200000UL /* Mask Preamble detection timeout event */
```

6.16.2.740 SYS_MASK_MRXRFSL

```
#define SYS_MASK_MRXRFSL 0x00010000UL /* Mask receiver Reed Solomon Frame Sync Loss event */
```

6.16.2.741 SYS_MASK_MRXRFTO

```
#define SYS_MASK_MRXRFTO 0x00020000UL /* Mask Receive Frame Wait Timeout event */
```

6.16.2.742 SYS_MASK_MRXS FDD

```
#define SYS_MASK_MRXS FDD 0x00000200UL /* Mask receiver SFD detected event */
```

6.16.2.743 SYS_MASK_MRXS FDTO

```
#define SYS_MASK_MRXS FDTO 0x04000000UL /* Mask Receive SFD timeout event */
```

6.16.2.744 SYS_MASK_MSLP2INIT

```
#define SYS_MASK_MSLP2INIT 0x00800000UL /* Mask SLEEP to INIT event */
```

6.16.2.745 SYS_MASK_MTXBERR

```
#define SYS_MASK_MTXBERR 0x10000000UL /* Mask Transmit Buffer Error event */
```

6.16.2.746 SYS_MASK_MTXFRB

```
#define SYS_MASK_MTXFRB 0x00000010UL /* Mask transmit frame begins event */
```

6.16.2.747 SYS_MASK_MTXFRS

```
#define SYS_MASK_MTXFRS 0x00000080UL /* Mask transmit frame sent event */
```

6.16.2.748 SYS_MASK_MTXPHS

```
#define SYS_MASK_MTXPHS 0x00000040UL /* Mask transmit PHY Header Sent event */
```

6.16.2.749 SYS_MASK_MTXPRS

```
#define SYS_MASK_MTXPRS 0x00000020UL /* Mask transmit preamble sent event */
```

6.16.2.750 SYS_STATE_ID

```
#define SYS_STATE_ID 0x19 /* System State information READ ONLY */
```

Bit definitions for register SYS_STATES Register map register file 0x19 is reserved.

6.16.2.751 SYS_STATE_LEN

```
#define SYS_STATE_LEN (5)
```

6.16.2.752 SYS_STATUS_AAT

```
#define SYS_STATUS_AAT 0x00000008UL /* Automatic Acknowledge Trigger */
```

6.16.2.753 SYS_STATUS_AFFREJ

```
#define SYS_STATUS_AFFREJ 0x20000000UL /* Automatic Frame Filtering rejection */
```

6.16.2.754 SYS_STATUS_ALL_DBLBUFF

```
#define SYS_STATUS_ALL_DBLBUFF (SYS_STATUS_RXDFR | SYS_STATUS_RXFCG)
```

6.16.2.755 SYS_STATUS_ALL_RX_ERR

```
#define SYS_STATUS_ALL_RX_ERR
```

Value:

```
(SYS_STATUS_RXPHE | SYS_STATUS_RXFCE |  
SYS_STATUS_RXRFLS | SYS_STATUS_RXSFDTO \  
| SYS_STATUS_AFFREJ |  
SYS_STATUS_LDEERR)
```

6.16.2.756 SYS_STATUS_ALL_RX_GOOD

```
#define SYS_STATUS_ALL_RX_GOOD
```

Value:

```
(SYS_STATUS_RXDFR | SYS_STATUS_RXFCG |  
SYS_STATUS_RXPRD | \  
SYS_STATUS_RXSFDD | SYS_STATUS_RXPHD |  
SYS_STATUS_LDEDONE)
```

6.16.2.757 SYS_STATUS_ALL_RX_TO

```
#define SYS_STATUS_ALL_RX_TO (SYS_STATUS_RXRFTO | SYS_STATUS_RXPTO)
```

6.16.2.758 SYS_STATUS_ALL_TX

```
#define SYS_STATUS_ALL_TX
```

Value:

```
(SYS_STATUS_AAT | SYS_STATUS_TXFRB |
    SYS_STATUS_TXPRS | \
        SYS_STATUS_TXPHS | SYS_STATUS_TXFRS )
```

6.16.2.759 SYS_STATUS_CLKPLL_LL

```
#define SYS_STATUS_CLKPLL_LL 0x02000000UL /* Clock PLL Losing Lock */
```

6.16.2.760 SYS_STATUS_CPLock

```
#define SYS_STATUS_CPLock 0x00000002UL /* Clock PLL Lock */
```

6.16.2.761 SYS_STATUS_ESYNCR

```
#define SYS_STATUS_ESYNCR 0x00000004UL /* External Sync Clock Reset */
```

6.16.2.762 SYS_STATUS_GPIOIRQ

```
#define SYS_STATUS_GPIOIRQ 0x00400000UL /* GPIO interrupt */
```

6.16.2.763 SYS_STATUS_HPDWARN

```
#define SYS_STATUS_HPDWARN 0x08000000UL /* Half Period Delay Warning */
```

6.16.2.764 SYS_STATUS_HSRBP

```
#define SYS_STATUS_HSRBP 0x40000000UL /* Host Side Receive Buffer Pointer */
```

6.16.2.765 SYS_STATUS_ICRBP

```
#define SYS_STATUS_ICRBP 0x80000000UL /* IC side Receive Buffer Pointer READ ONLY */
```

6.16.2.766 SYS_STATUS_ID

```
#define SYS_STATUS_ID 0x0F /* System event Status Register */
```

Bit definitions for register SYS_STATUS.

6.16.2.767 SYS_STATUS IRQS

```
#define SYS_STATUS_IRQS 0x00000001UL /* Interrupt Request Status READ ONLY */
```

6.16.2.768 SYS_STATUS_LDEDONE

```
#define SYS_STATUS_LDEDONE 0x00000400UL /* LDE processing done */
```

6.16.2.769 SYS_STATUS_LDEERR

```
#define SYS_STATUS_LDEERR 0x00040000UL /* Leading edge detection processing error */
```

6.16.2.770 SYS_STATUS_LEN

```
#define SYS_STATUS_LEN (5) /* Note 40 bit register */
```

6.16.2.771 SYS_STATUS_MASK_32

```
#define SYS_STATUS_MASK_32 0xFFFF7FFFFUL /* System event Status Register access mask (all unused fields should always be written as zero) */
```

6.16.2.772 SYS_STATUS_OFFSET

```
#define SYS_STATUS_OFFSET 0x00
```

6.16.2.773 SYS_STATUS_reserved

```
#define SYS_STATUS_reserved 0x00080000UL /* bit19 reserved */
```

6.16.2.774 SYS_STATUS_RFPLL_LL

```
#define SYS_STATUS_RFPLL_LL 0x01000000UL /* RF PLL Losing Lock */
```

6.16.2.775 SYS_STATUS_RXDFR

```
#define SYS_STATUS_RXDFR 0x00002000UL /* Receiver Data Frame Ready */
```

6.16.2.776 SYS_STATUS_RXFCE

```
#define SYS_STATUS_RXFCE 0x00008000UL /* Receiver FCS Error */
```

6.16.2.777 SYS_STATUS_RXFCG

```
#define SYS_STATUS_RXFCG 0x00004000UL /* Receiver FCS Good */
```

6.16.2.778 SYS_STATUS_RXOVRR

```
#define SYS_STATUS_RXOVRR 0x00100000UL /* Receiver Overrun */
```

6.16.2.779 SYS_STATUS_RXPHD

```
#define SYS_STATUS_RXPHD 0x00000800UL /* Receiver PHY Header Detect */
```

6.16.2.780 SYS_STATUS_RXPHE

```
#define SYS_STATUS_RXPHE 0x00001000UL /* Receiver PHY Header Error */
```

6.16.2.781 SYS_STATUS_RXPRD

```
#define SYS_STATUS_RXPRD 0x00000100UL /* Receiver Preamble Detected status */
```

6.16.2.782 SYS_STATUS_RXPREJ

```
#define SYS_STATUS_RXPREJ 0x020000000ULL /* Receiver Preamble Rejection */
```

6.16.2.783 SYS_STATUS_RXPTO

```
#define SYS_STATUS_RXPTO 0x00200000UL /* Preamble detection timeout */
```

6.16.2.784 SYS_STATUS_RXRFLS

```
#define SYS_STATUS_RXRFLS 0x00010000UL /* Receiver Reed Solomon Frame Sync Loss */
```

6.16.2.785 SYS_STATUS_RXRFTO

```
#define SYS_STATUS_RXRFTO 0x00020000UL /* Receive Frame Wait Timeout */
```

6.16.2.786 SYS_STATUS_RXRSCS

```
#define SYS_STATUS_RXRSCS 0x010000000ULL /* Receiver Reed-Solomon Correction Status */
```

6.16.2.787 SYS_STATUS_RXSFDD

```
#define SYS_STATUS_RXSFDD 0x00000200UL /* Receiver Start Frame Delimiter Detected. */
```

6.16.2.788 SYS_STATUS_RXSFDTO

```
#define SYS_STATUS_RXSFDTO 0x04000000UL /* Receive SFD timeout */
```

6.16.2.789 SYS_STATUS_SLP2INIT

```
#define SYS_STATUS_SLP2INIT 0x00800000UL /* SLEEP to INIT */
```

6.16.2.790 SYS_STATUS_TXBERR

```
#define SYS_STATUS_TXBERR 0x10000000UL /* Transmit Buffer Error */
```

6.16.2.791 SYS_STATUS_TXERR

```
#define SYS_STATUS_TXERR (0x0408) /* These bits are the 16 high bits of status register TXPUTE  
and HPDWARN flags */
```

6.16.2.792 SYS_STATUS_TXFRB

```
#define SYS_STATUS_TXFRB 0x00000010UL /* Transmit Frame Begins */
```

6.16.2.793 SYS_STATUS_TXFRS

```
#define SYS_STATUS_TXFRS 0x00000080UL /* Transmit Frame Sent: This is set when the transmitter has completed the sending of a frame */
```

6.16.2.794 SYS_STATUS_TXPHS

```
#define SYS_STATUS_TXPHS 0x00000040UL /* Transmit PHY Header Sent */
```

6.16.2.795 SYS_STATUS_TXPRS

```
#define SYS_STATUS_TXPRS 0x00000020UL /* Transmit Preamble Sent */
```

6.16.2.796 SYS_STATUS_TXPUTE

```
#define SYS_STATUS_TXPUTE 0x0400000000ULL /* Transmit power up time error */
```

6.16.2.797 SYS_TIME_ID

```
#define SYS_TIME_ID 0x06 /* System Time Counter (40-bit) */
```

Bit definitions for register SYS_TIME.

6.16.2.798 SYS_TIME_LEN

```
#define SYS_TIME_LEN (5) /* Note 40 bit register */
```

6.16.2.799 SYS_TIME_OFFSET

```
#define SYS_TIME_OFFSET 0x00
```

6.16.2.800 TC_PGCAL_STATUS_DELAY_MASK

```
#define TC_PGCAL_STATUS_DELAY_MASK 0xFFFF /* Mask to retrieve PG delay count from calibration */
```

6.16.2.801 TC_PGCAL_STATUS_LEN

```
#define TC_PGCAL_STATUS_LEN (1)
```

6.16.2.802 TC_PGCAL_STATUS_OFFSET

```
#define TC_PGCAL_STATUS_OFFSET 0x09 /* Status register from PG calibration block */
```

6.16.2.803 TC_PGCCTRL_AUTOCAL

```
#define TC_PGCCTRL_AUTOCAL 0x02 /* Starts a PG autocalibration loop */
```

6.16.2.804 TC_PGCCTRL_CALSTART

```
#define TC_PGCCTRL_CALSTART 0x01 /* Start PG cal procedure */
```

6.16.2.805 TC_PGCCTRL_DIR_CONV

```
#define TC_PGCCTRL_DIR_CONV 0x80 /* Direction (converging) of autocal binary search */
```

6.16.2.806 TC_PGCCTRL_LEN

```
#define TC_PGCCTRL_LEN (1)
```

6.16.2.807 TC_PGCCTRL_OFFSET

```
#define TC_PGCCTRL_OFFSET 0x08 /* Pulse Generator Calibration control */
```

6.16.2.808 TC_PGCCTRL_ON_TX

```
#define TC_PGCCTRL_ON_TX 0x40 /* Perform autocal on each TX enable */
```

6.16.2.809 TC_PGCCTRL_TMEAS_MASK

```
#define TC_PGCCTRL_TMEAS_MASK 0x3C /* Mask to retrieve number of clock cycles over which to run PG cal counter */
```

6.16.2.810 TC_PGDELAY_CH1

```
#define TC_PGDELAY_CH1 0xC9 /* Recommended value for channel 1 */
```

6.16.2.811 TC_PGDELAY_CH2

```
#define TC_PGDELAY_CH2 0xC2 /* Recommended value for channel 2 */
```

6.16.2.812 TC_PGDELAY_CH3

```
#define TC_PGDELAY_CH3 0xC5 /* Recommended value for channel 3 */
```

6.16.2.813 TC_PGDELAY_CH4

```
#define TC_PGDELAY_CH4 0x95 /* Recommended value for channel 4 */
```

6.16.2.814 TC_PGDELAY_CH5

```
#define TC_PGDELAY_CH5 0xC0 /* Recommended value for channel 5 */
```

6.16.2.815 TC_PGDELAY_CH7

```
#define TC_PGDELAY_CH7 0x93 /* Recommended value for channel 7 */
```

6.16.2.816 TC_PGDELAY_LEN

```
#define TC_PGDELAY_LEN (1)
```

6.16.2.817 TC_PGDELAY_OFFSET

```
#define TC_PGDELAY_OFFSET 0x0B /* Transmitter Calibration Pulse Generator Delay */
```

6.16.2.818 TC_PGTTEST_CW

```
#define TC_PGTTEST_CW 0x13 /* Continuous Wave (CW) Test Mode */
```

6.16.2.819 TC_PGTTEST_LEN

```
#define TC_PGTTEST_LEN (1)
```

6.16.2.820 TC_PGTTEST_NORMAL

```
#define TC_PGTTEST_NORMAL 0x00 /* Normal operation */
```

6.16.2.821 TC_PGTTEST_OFFSET

```
#define TC_PGTTEST_OFFSET 0x0C /* Transmitter Calibration Pulse Generator Test */
```

6.16.2.822 TC_SARL_SAR_C

```
#define TC_SARL_SAR_C (0) /* SAR control */
```

6.16.2.823 TC_SARL_SAR_LTEMP_OFFSET

```
#define TC_SARL_SAR_LTEMP_OFFSET (4) /* Latest SAR reading for Temperature level */
```

6.16.2.824 TC_SARL_SAR_LVBAT_OFFSET

```
#define TC_SARL_SAR_LVBAT_OFFSET (3) /* Latest SAR reading for Voltage level */
```

6.16.2.825 TC_SARW_SAR_WTEMP_OFFSET

```
#define TC_SARW_SAR_WTEMP_OFFSET 0x06 /* SAR reading of Temperature level taken at last wakeup event */
```

6.16.2.826 TC_SARW_SAR_WVBAT_OFFSET

```
#define TC_SARW_SAR_WVBAT_OFFSET 0x07 /* SAR reading of Voltage level taken at last wakeup event */
```

6.16.2.827 TX_ANTD_ID

```
#define TX_ANTD_ID 0x18 /* 16-bit Delay from Transmit to Antenna */
```

Bit definitions for register TX_ANTD.

6.16.2.828 TX_ANTD_LEN

```
#define TX_ANTD_LEN (2)
```

6.16.2.829 TX_ANTD_OFFSET

```
#define TX_ANTD_OFFSET 0x00
```

6.16.2.830 TX_BUFFER_ID

```
#define TX_BUFFER_ID 0x09 /* Transmit Data Buffer */
```

Bit definitions for register TX_BUFFER.

6.16.2.831 TX_BUFFER_LEN

```
#define TX_BUFFER_LEN (1024)
```

6.16.2.832 TX_CAL_ID

```
#define TX_CAL_ID 0x2A /* Transmitter calibration block */
```

Bit definitions for register TX_CAL Refer to section 7.2.43 Register file: 0x2A Transmitter Calibration block.

6.16.2.833 TX_CAL_LEN

```
#define TX_CAL_LEN (52)
```

6.16.2.834 TX_FCTRL_FLE_MASK

```
#define TX_FCTRL_FLE_MASK 0x000003FFUL /* bit mask to access Frame Length field */
```

6.16.2.835 TX_FCTRL_ID

```
#define TX_FCTRL_ID 0x08 /* Transmit Frame Control */
```

Bit definitions for register TX_FCTRL.

6.16.2.836 TX_FCTRL_IFSDELAY_MASK

```
#define TX_FCTRL_IFSDELAY_MASK 0xFF0000000ULL /* bit mask to access Inter-Frame Spacing field */
```

6.16.2.837 TX_FCTRL_LEN

```
#define TX_FCTRL_LEN (5) /* Note 40 bit register */
```

6.16.2.838 TX_FCTRL_PE_MASK

```
#define TX_FCTRL_PE_MASK 0x00300000UL /* bit mask to access Preamble Extension */
```

6.16.2.839 TX_FCTRL_PE_SHFT

```
#define TX_FCTRL_PE_SHFT (20) /* shift to access Preamble length Extension to allow specification  
of non-standard values */
```

6.16.2.840 TX_FCTRL_SAFE_MASK_32

```
#define TX_FCTRL_SAFE_MASK_32 0xFFFFE3FFUL /* FSCTRL has fields which should always be written  
zero */
```

6.16.2.841 TX_FCTRL_TFLE_MASK

```
#define TX_FCTRL_TFLE_MASK 0x00000380UL /* bit mask to access Transmit Frame Length Extension  
*/
```

6.16.2.842 TX_FCTRL_TFLEN_MASK

```
#define TX_FCTRL_TFLEN_MASK 0x0000007FUL /* bit mask to access Transmit Frame Length */
```

6.16.2.843 TX_FCTRL_TR

```
#define TX_FCTRL_TR 0x00008000UL /* Transmit Ranging enable */
```

6.16.2.844 TX_FCTRL_TR_SHFT

```
#define TX_FCTRL_TR_SHFT (15) /* shift to access Ranging bit */
```

6.16.2.845 TX_FCTRL_TXBOFFS_MASK

```
#define TX_FCTRL_TXBOFFS_MASK 0xFFC00000UL /* bit mask to access Transmit buffer index offset  
10-bit field */
```

6.16.2.846 TX_FCTRL_TXBOFFS_SHFT

```
#define TX_FCTRL_TXBOFFS_SHFT (22) /* Shift to access transmit buffer index offset */
```

6.16.2.847 TX_FCTRL_TXBR_110k

```
#define TX_FCTRL_TXBR_110k 0x00000000UL /* Transmit Bit Rate = 110k */
```

6.16.2.848 TX_FCTRL_TXBR_6M

```
#define TX_FCTRL_TXBR_6M 0x00004000UL /* Transmit Bit Rate = 6.8M */
```

6.16.2.849 TX_FCTRL_TXBR_850k

```
#define TX_FCTRL_TXBR_850k 0x00002000UL /* Transmit Bit Rate = 850k */
```

6.16.2.850 TX_FCTRL_TXBR_MASK

```
#define TX_FCTRL_TXBR_MASK 0x00006000UL /* bit mask to access Transmit Bit Rate */
```

6.16.2.851 TX_FCTRL_TXBR_SHFT

```
#define TX_FCTRL_TXBR_SHFT (13) /* shift to access Data Rate field */
```

6.16.2.852 TX_FCTRL_TXPRF_16M

```
#define TX_FCTRL_TXPRF_16M 0x00010000UL /* Transmit Pulse Repetition Frequency = 16 Mhz */
```

6.16.2.853 TX_FCTRL_TXPRF_4M

```
#define TX_FCTRL_TXPRF_4M 0x00000000UL /* Transmit Pulse Repetition Frequency = 4 Mhz */
```

6.16.2.854 TX_FCTRL_TXPRF_64M

```
#define TX_FCTRL_TXPRF_64M 0x00020000UL /* Transmit Pulse Repetition Frequency = 64 Mhz */
```

6.16.2.855 TX_FCTRL_TXPRF_MASK

```
#define TX_FCTRL_TXPRF_MASK 0x00030000UL /* bit mask to access Transmit Pulse Repetition Frequency */
```

6.16.2.856 TX_FCTRL_TXPRF_SHFT

```
#define TX_FCTRL_TXPRF_SHFT (16) /* shift to access Pulse Repetition Frequency field */
```

6.16.2.857 TX_FCTRL_TXPSR_MASK

```
#define TX_FCTRL_TXPSR_MASK 0x000C0000UL /* bit mask to access Transmit Preamble Symbol Repetitions (PSR). */
```

6.16.2.858 TX_FCTRL_TXPSR_PE_1024

```
#define TX_FCTRL_TXPSR_PE_1024 0x00080000UL /* bit mask to access Preamble Extension = 1024 */
```

6.16.2.859 TX_FCTRL_TXPSR_PE_128

```
#define TX_FCTRL_TXPSR_PE_128 0x00140000UL /* bit mask to access Preamble Extension = 128 */
```

6.16.2.860 TX_FCTRL_TXPSR_PE_1536

```
#define TX_FCTRL_TXPSR_PE_1536 0x00180000UL /* bit mask to access Preamble Extension = 1536 */
```

6.16.2.861 TX_FCTRL_TXPSR_PE_16

```
#define TX_FCTRL_TXPSR_PE_16 0x00000000UL /* bit mask to access Preamble Extension = 16 */
```

6.16.2.862 TX_FCTRL_TXPSR_PE_2048

```
#define TX_FCTRL_TXPSR_PE_2048 0x00280000UL /* bit mask to access Preamble Extension = 2048 */
```

6.16.2.863 TX_FCTRL_TXPSR_PE_256

```
#define TX_FCTRL_TXPSR_PE_256 0x00240000UL /* bit mask to access Preamble Extension = 256 */
```

6.16.2.864 TX_FCTRL_TXPSR_PE_4096

```
#define TX_FCTRL_TXPSR_PE_4096 0x000C0000UL /* bit mask to access Preamble Extension = 4096 */
```

6.16.2.865 TX_FCTRL_TXPSR_PE_512

```
#define TX_FCTRL_TXPSR_PE_512 0x00340000UL /* bit mask to access Preamble Extension = 512 */
```

6.16.2.866 TX_FCTRL_TXPSR_PE_64

```
#define TX_FCTRL_TXPSR_PE_64 0x00040000UL /* bit mask to access Preamble Extension = 64 */
```

6.16.2.867 TX_FCTRL_TXPSR_PE_MASK

```
#define TX_FCTRL_TXPSR_PE_MASK 0x003C0000UL /* bit mask to access Transmit Preamble Symbol Repetitions (PSR). */
```

6.16.2.868 TX_FCTRL_TXPSR_SHFT

```
#define TX_FCTRL_TXPSR_SHFT (18) /* shift to access Preamble Symbol Repetitions field */
```

6.16.2.869 TX_POWER_BOOSTNORM_MASK

```
#define TX_POWER_BOOSTNORM_MASK 0x00000000UL /* This is the normal power setting used for frames that do not fall */
```

6.16.2.870 TX_POWER_BOOSTNORM_SHIFT

```
#define TX_POWER_BOOSTNORM_SHIFT (0)
```

6.16.2.871 TX_POWER_BOOSTP125_MASK

```
#define TX_POWER_BOOSTP125_MASK 0x00000000UL /* This value sets the power applied during transmission at the 6.8 Mbps data rate frames that are less than 0.125 ms */
```

6.16.2.872 TX_POWER_BOOSTP125_SHIFT

```
#define TX_POWER_BOOSTP125_SHIFT (24)
```

6.16.2.873 TX_POWER_BOOSTP250_MASK

```
#define TX_POWER_BOOSTP250_MASK 0x00000000UL /* This value sets the power applied during transmission  
at the 6.8 Mbps data rate frames that are less than 0.25 ms duration */
```

6.16.2.874 TX_POWER_BOOSTP250_SHIFT

```
#define TX_POWER_BOOSTP250_SHIFT (16)
```

6.16.2.875 TX_POWER_BOOSTP500_MASK

```
#define TX_POWER_BOOSTP500_MASK 0x00000000UL /* This value sets the power applied during transmission  
at the 6.8 Mbps data rate frames that are less than 0.5 ms duration */
```

6.16.2.876 TX_POWER_BOOSTP500_SHIFT

```
#define TX_POWER_BOOSTP500_SHIFT (8)
```

6.16.2.877 TX_POWER_ID

```
#define TX_POWER_ID 0x1E /* TX Power Control */
```

Bit definitions for register TX_POWER.

6.16.2.878 TX_POWER_LEN

```
#define TX_POWER_LEN (4)
```

6.16.2.879 TX_POWER_MAN_DEFAULT

```
#define TX_POWER_MAN_DEFAULT 0x0E080222UL
```

6.16.2.880 TX_POWER_TXPOWPHR_MASK

```
#define TX_POWER_TXPOWPHR_MASK 0x0000FF00UL /* This power setting is applied during the transmission  
of the PHY header (PHR) portion of the frame. */
```

6.16.2.881 TX_POWER_TXPOWSD_MASK

```
#define TX_POWER_TXPOWSD_MASK 0x00FF0000UL /* This power setting is applied during the transmission  
of the synchronisation header (SHR) and data portions of the frame. */
```

6.16.2.882 TX_STAMP_LEN

```
#define TX_STAMP_LEN TX_TIME_TX_STAMP_LEN
```

6.16.2.883 TX_TIME_ID

```
#define TX_TIME_ID 0x17 /* Transmit Message Time of Sending */
```

Bit definitions for register.

6.16.2.884 TX_TIME_LLLEN

```
#define TX_TIME_LLLEN (10)
```

6.16.2.885 TX_TIME_TX_RAWST_OFFSET

```
#define TX_TIME_TX_RAWST_OFFSET (5) /* byte 5..9 40 bit Raw Timestamp for the frame */
```

6.16.2.886 TX_TIME_TX_STAMP_LEN

```
#define TX_TIME_TX_STAMP_LEN (5) /* 40-bits = 5 bytes */
```

6.16.2.887 TX_TIME_TX_STAMP_OFFSET

```
#define TX_TIME_TX_STAMP_OFFSET (0) /* byte 0..4 40 bit Reports the fully adjusted time of transmission */
```

6.16.2.888 USR_SFD_ID

```
#define USR_SFD_ID 0x21 /* User-specified short/long TX/RX SFD sequences */
```

Bit definitions for register USR_SFD Please read User Manual : User defined SFD sequence.

6.16.2.889 USR_SFD_LEN

```
#define USR_SFD_LEN (41)
```

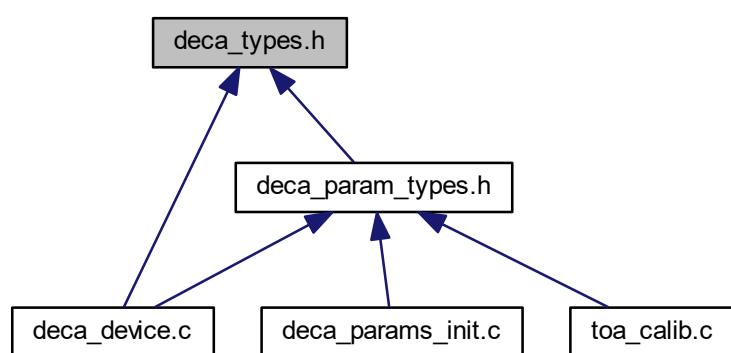
6.16.2.890 W4R_TIM_MASK

```
#define W4R_TIM_MASK ACK_RESP_T_W4R_TIM_MASK
```

6.17 deca_types.h File Reference

Decawave general type definitions.

This graph shows which files directly or indirectly include this file:



Macros

- #define _DECA_UINT8_
- #define _DECA_UINT16_
- #define _DECA_UINT32_
- #define _DECA_INT8_
- #define _DECA_INT16_
- #define _DECA_INT32_
- #define NULL ((void *)0UL)

Typedefs

- typedef unsigned char uint8
- typedef unsigned short uint16
- typedef unsigned long uint32
- typedef signed char int8
- typedef signed short int16
- typedef signed long int32

6.17.1 Detailed Description

Decawave general type definitions.

Attention

Copyright 2013 (c) Decawave Ltd, Dublin, Ireland.

All rights reserved.

6.17.2 Macro Definition Documentation

6.17.2.1 _DECA_INT16_

```
#define _DECA_INT16_
```

6.17.2.2 _DECA_INT32_

```
#define _DECA_INT32_
```

6.17.2.3 _DECA_INT8_

```
#define _DECA_INT8_
```

6.17.2.4 _DECA_UINT16_

```
#define _DECA_UINT16_
```

6.17.2.5 _DECA_UINT32_

```
#define _DECA_UINT32_
```

6.17.2.6 _DECA_UINT8_

```
#define _DECA_UINT8_
```

6.17.2.7 NULL

```
#define NULL ((void *)0UL)
```

6.17.3 Typedef Documentation

6.17.3.1 int16

```
typedef signed short int16
```

6.17.3.2 int32

```
typedef signed long int32
```

6.17.3.3 int8

```
typedef signed char int8
```

6.17.3.4 uint16

```
typedef unsigned short uint16
```

6.17.3.5 uint32

```
typedef unsigned long uint32
```

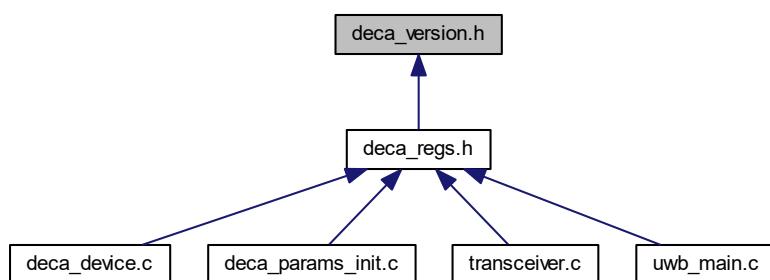
6.17.3.6 uint8

```
typedef unsigned char uint8
```

6.18 deca_version.h File Reference

Defines the version info for the DW1000 device driver including its API.

This graph shows which files directly or indirectly include this file:



Macros

- #define DW1000_DRIVER_VERSION 0x040006
- #define DW1000_DEVICE_DRIVER_VER_STRING "DW1000 Device Driver Version 04.00.06"

6.18.1 Detailed Description

Defines the version info for the DW1000 device driver including its API.

Attention

Copyright 2013 (c) Decawave Ltd, Dublin, Ireland.

All rights reserved.

6.18.2 Macro Definition Documentation

6.18.2.1 DW1000_DEVICE_DRIVER_VER_STRING

```
#define DW1000_DEVICE_DRIVER_VER_STRING "DW1000 Device Driver Version 04.00.06"
```

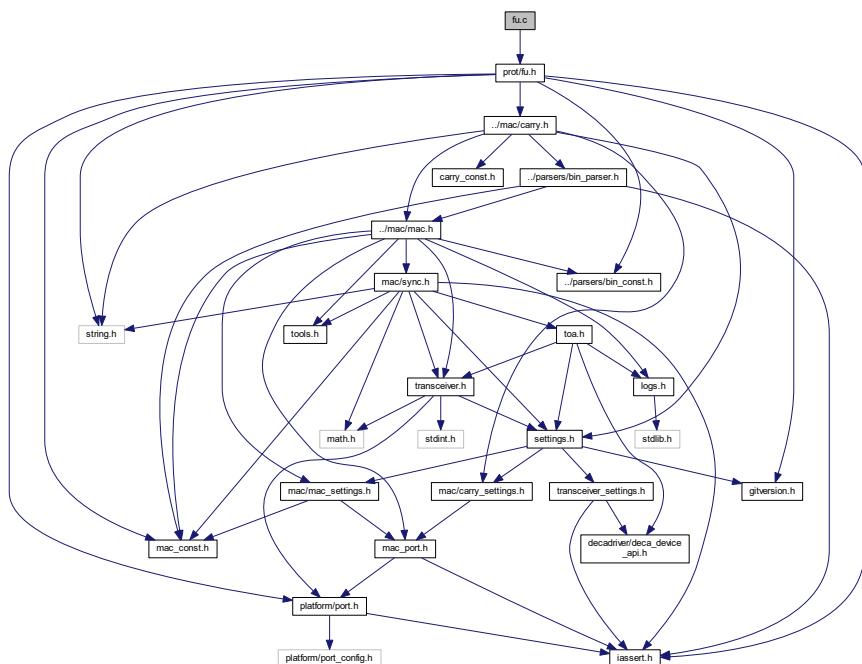
6.18.2.2 DW1000_DRIVER_VERSION

```
#define DW1000_DRIVER_VERSION 0x040006
```

6.19 fu.c File Reference

```
#include "prot/fu.h"
```

Include dependency graph for fu.c:



Data Structures

- struct `FU_instance_t`

Functions

- `uint8_t * FU_GetCurrentFlashBase ()`
Compare address of this function with FU_DESTINATION_x and
- `int FU_AcceptFirmwareVersion (int Ver)`
- `int FU_AcceptHardwareVersion (int Ver)`
- `void FU_HandleAsDevice (const FU_prot *fup, const prot_packet_info_t *info)`
process new message as device
- `void FU_AcceptFirmware ()`
Mark current version as fully functional.
- `void FU_Init (bool forceNoFirmwareCheck)`
Initialize module.

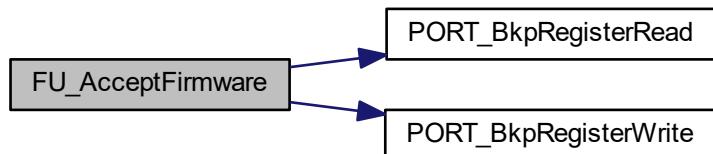
6.19.1 Function Documentation

6.19.1.1 FU_AcceptFirmware()

```
void FU_AcceptFirmware ( )
```

Mark current version as fully functional.

Here is the call graph for this function:



6.19.1.2 FU_AcceptFirmwareVersion()

```
int FU_AcceptFirmwareVersion (
    int Ver )
```

6.19.1.3 FU_AcceptHardwareVersion()

```
int FU_AcceptHardwareVersion (
    int Ver )
```

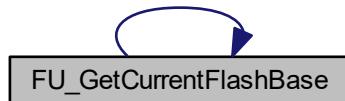
6.19.1.4 FU_GetCurrentFlashBase()

```
uint8_t* FU_GetCurrentFlashBase ( )
compare address of this function with FU_DESTINATION_x and
return base address of current working firmware
```

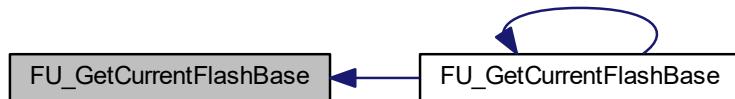
Returns

uint8_t* address of start a currently used flash program space

Here is the call graph for this function:



Here is the caller graph for this function:



6.19.1.5 FU_HandleAsDevice()

```
void FU_HandleAsDevice (
    const FU_prot * fup,
    const prot_packet_info_t * info )
```

process new message as device

Parameters

<i>fup</i>	message to process
<i>info</i>	extra informations about message

6.19.1.6 FU_Init()

```
void FU_Init (
    bool forceNoFirmwareCheck )
```

Initialize module.

Parameters

<i>forceNoFirmwareCheck</i>	set true for sink when firmware verification should be forced to succeed
-----------------------------	--

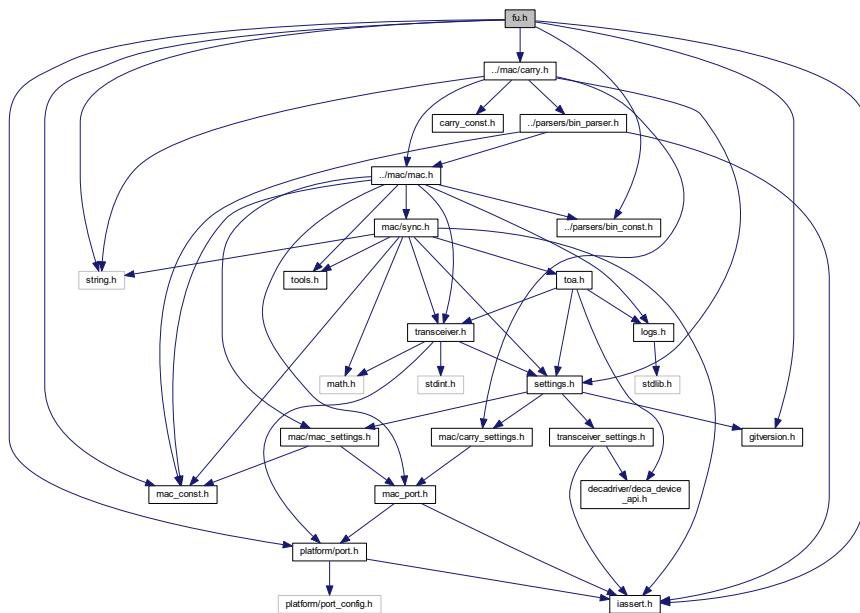
Here is the caller graph for this function:

**6.20 fu.h File Reference**

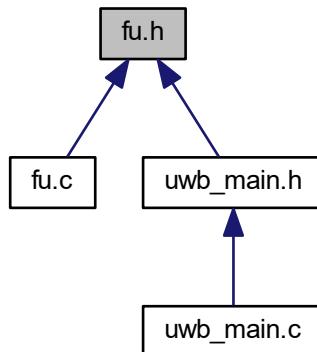
Firmware upgrade protocol.

```
#include <string.h>
#include "../mac/carry.h"
#include "../mac/mac_const.h"
#include "../parsers/bin_const.h"
#include "gitversion.h"
#include "iassert.h"
#include "platform/port.h"
```

Include dependency graph for fu.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [FU_prot](#)
raw firmware upgrade protocol struct
- struct [FU_SOT_prot](#)
start of frame struct

Macros

- `#define FU_ASSERT(expr) IASSERT(expr)`
- `#define FU_DESTINATION_1 ((void *)(&PROG_DESTINATION1))`
- `#define FU_DESTINATION_2 ((void *)(&PROG_DESTINATION2))`
- `#define FU_MAX_PROGRAM_SIZE ((int)(FU_DESTINATION_2 - FU_DESTINATION_1))`
- `#define FU_VERSION_LOC (FLASH_PAGE_SIZE + 4)`
place in address where current firmware version is stored offset from FLASH_BASE
- `#define FU_MAX_DATA_SIZE (8 + 2)`
max tx data in target device is 8 bytes (+2 for CRC) but for source device it depends on data size in block
- `#define FU_BLOCK_SIZE 16`
size of one minimal piece of data in bytes
- `#define FU_OPCODE_SOT 2`
Start Of Transmission.
- `#define FU_OPCODE_DATA 3`
Data.
- `#define FU_OPCODE_ACK 4`
Data or SOT ACK.
- `#define FU_OPCODE_ABORT 5`
Error occurrence.
- `#define FU_OPCODE_EOT 6`
End Of Transmission.
- `#define FU_ERR_BAD_FRAME_CRC 1`
CRC in frame didn't match.
- `#define FU_ERR_FLASH_WRITING 2`
error during writing to flash
- `#define FU_ERR_FLASH_ERASING 3`
error during erasing flash
- `#define FU_ERR_BAD_FRAME_LEN 4`
frame length didn't match
- `#define FU_ERR_BAD_FRAME_HASH 5`
incompatible frame hash
- `#define FU_ERR_BAD_OFFSET 6`
bad offset value in data frame
- `#define FU_ERR_BAD_OPCODE_SET 7`
bad FU opcode in frame
- `#define FU_ERR_FIR_NOT_ACCEPTED_YET 8`
firmware upgrade
- `#define FU_ERR_BAD_FLASH_CRC 10`
- `#define FU_ERR_BAD_F_VERSION 11`
firmware version is for other partition
- `#define FU_ERR_BAD_H_VERSION 12`
hardware major version didn't match
- `#define FU_ERR_BAD_FILE_SIZE 13`
or device has been reset during firmware upgrade
- `#define FU_ERR_BAD_PROT_VER 14`
incompatible firmware protocol version
- `#define FU_ERR_VERSION_IN_PACKAGE 15`
part (should be EOT packet)
- `#define FU_PROT_HEAD_SIZE (1 + 1 + 1 + 1 + 2)`
FU protocol head size.
- `#define FU_PROT_VERSION 1`
protocol version

Functions

- void `FU_Init` (bool forceNoFirmwareCheck)
Initialize module.
- void `FU_AcceptFirmware` ()
Mark current version as fully functional.
- uint8_t * `FU_GetCurrentFlashBase` ()
return base address of current working firmware
- void `FU_HandleAsDevice` (const `FU_prot` *fup, const `prot_packet_info_t` *info)
process new message as device

Variables

- char `PROG_DESTINATION1`
from linker script, address of partition A
- char `PROG_DESTINATION2`
from linker script, address of partition B
- char `PROG_START1`
from linker script, isr vector location (part A)
- char `PROG_SETTINGS1`
from linker script, settings location (part A)

6.20.1 Detailed Description

Firmware upgrade protocol.

Author

Karol Trzcinski

Date

2018-06-29

Firmware upgrade protocol

Goal

It is used to upgrade firmware over the air (OTA), USB or any other media. Transfer layer is provided by carry module and carry implementation is separated from this module.

Details

Overview

To work fine whole available flash memory should be divided into partitions. Current implementation use separate partition for:

- bootloader
- firmware A
- firmware B

Where bootloader should start after each system reset and detect correct firmware to use. Firmware A and B is a place for target application code and when one version is in use then second one is a place for the new one.

Bootloader

After startup bootloader should be first application to run. Then work algorithm looks like:

1. Check reset source
 - power up or bootloader watchdog reset - go to next point
 - others - run last application
1. Check firmware versions compatibility, if change detected then start verification process
1. Save current status if changed
1. Choose better firmware to start

Verification process start bootloader watchdog (with 8s period) and start new firmware execution. After Watchdog reset, value of special register to check if this version run without problems. If this register value is correct then mark firmware as verified, save this as last running app and start this application without bootloader watchdog. When application doesn't set correct value to special register then downscore it and set app to run in a standard way.

Firmware upgrade routine

At the begin SOT frame should be transmitted from Host to Device and Device should response ACK frame. Then is series of Data/ACK frames. ACK is required after each data frame On last data frame, Device response with EOT frame instead of ACK. It's extra information that firmware upgrade was fully successful. To start firmware upgrade

SOT – Start Of Transmission – Host->Device

This frame should be sent at the beginning before frames with raw firmware data. After receiving this frame, whole flash area for new firmware should be erased.

Construction:

[FC:1][frameLen:1][opcode:1][version:4][firmwareCRC:2][size:4][frameCRC:2]

- frameLen - length of while frame, from address of frameLen to the end of frame CRC, equal to 12.
- opcode - set to FU_OPCODE_SOT
- version - [lowest::older] byte of new firmware version
- firmwareCRC - CRC calculated only from complete raw firmware data calculated in this same way as for each frame
- size - big endian, size in bytes of complete raw firmware data
- frameCRC - 16 bit of CRC calculated on data from frameLen to the end of offset, based on FU_CRC_POLY

Default/Data frame – Host->Device

Frame with firmware file data

Construction:

[frameLen:1][opcode:1][version:1][offset:2][data:n][frameCRC:2]

- frameLen - length of while frame, from address of frameLen to the end of frame CRC
- opcode - set to FU_OPCODE_DATA
- version - version of new firmware
- offset - offset in memory for transmitted data. Offset is expressed in FU_BLOCK_SIZE unit
- data - raw byte data, number of them can be readed as frameLen-FU_PROT_HEAD_SIZE-2 (for CRC), number of data should be multiple of FU_BLOCK_SIZE
- frameCRC - 16 bit of CRC calculated on data from frameLen to the end of offset, based on FU_CRC_POLY

ACK – Device->Host

Previously sent packet was correct and device ready to receive new frame

Construction:

[frameLen:1][opcode:1][version:1][extra:2][frameCRC:2]

- frameLen - length of while frame, from address of frameLen to the end of frame CRC. Equal to 6
- opcode - set to FU_OPCODE_ACK
- version - lowest byte of current firmware version
- extra - this field shouldn't be modified in ACK frame

EOT – End Of Transmission – Host->Device

Last data frame. After reception of this frame data compatibility should be checked. When everything is ok, then ACK response should be send and device should reboot to start new firmware verification procedure.

Construction:

```
[frameLen:1][opcode:1][version:1][extra:2][frameCRC:2]
```

- frameLen - length of while frame, from address of frameLen to the end of frame CRC. Equal to 6
- opcode - set to FU_OPCODE_EOT
- version - lowest byte of current firmware version
- extra - this field shouldn't be modified in EOT frame

ABORT – Device->Host

Error info. Firmware upgrade process can't be continued.

Construction:

```
[frameLen:1][opcode:1][version:1][errorCode:2][frameCRC:2]
```

- frameLen - length of while frame, from address of frameLen to the end of frame CRC. Equal to 8
- opcode - set to FU_OPCODE_ABORT
- version - lowest byte of current firmware version
- errorCode - FU_ERR_code enumerate

ASK_VER – Host->Device

Ask about response ACK frame with current version of software

Construction:

```
[frameLen:1][opcode:1][version:4][frameCRC:2]
```

- frameLen - length of while frame, from address of frameLen to the end of frame CRC. Equal to 6
- opcode - set to FU_OPCODE_ASK_VER
- version - BigEndian, lowest byte of current firmware version

6.20.2 Macro Definition Documentation

6.20.2.1 FU_ASSERT

```
#define FU_ASSERT(  
    expre ) IASSERT(expre)
```

6.20.2.2 FU_BLOCK_SIZE

```
#define FU_BLOCK_SIZE 16  
  
size of one minimal piece of data in bytes
```

6.20.2.3 FU_DESTINATION_1

```
#define FU_DESTINATION_1 ((void *)(&PROG_DESTINATION1))
```

6.20.2.4 FU_DESTINATION_2

```
#define FU_DESTINATION_2 ((void *)(&PROG_DESTINATION2))
```

6.20.2.5 FU_ERR_BAD_F_VERSION

```
#define FU_ERR_BAD_F_VERSION 11
```

firmware version is for other partition

6.20.2.6 FU_ERR_BAD_FILE_SIZE

```
#define FU_ERR_BAD_FILE_SIZE 13
```

or device has been reset during firmware upgrade

firmware size exceed partition capacity

6.20.2.7 FU_ERR_BAD_FLASH_CRC

```
#define FU_ERR_BAD_FLASH_CRC 10
```

6.20.2.8 FU_ERR_BAD_FRAME_CRC

```
#define FU_ERR_BAD_FRAME_CRC 1
```

CRC in frame didn't match.

6.20.2.9 FU_ERR_BAD_FRAME_HASH

```
#define FU_ERR_BAD_FRAME_HASH 5
```

incompatible frame hash

6.20.2.10 FU_ERR_BAD_FRAME_LEN

```
#define FU_ERR_BAD_FRAME_LEN 4
```

frame length didn't match

6.20.2.11 FU_ERR_BAD_H_VERSION

```
#define FU_ERR_BAD_H_VERSION 12
```

hardware major version didn't match

6.20.2.12 FU_ERR_BAD_OFFSET

```
#define FU_ERR_BAD_OFFSET 6
```

bad offset value in data frame

6.20.2.13 FU_ERR_BAD_OPCODE_SET

```
#define FU_ERR_BAD_OPCODE_SET 7
```

bad FU opcode in frame

6.20.2.14 FU_ERR_BAD_PROT_VER

```
#define FU_ERR_BAD_PROT_VER 14
```

incompatible firmware protocol version

6.20.2.15 FU_ERR_FIR_NOT_ACCEPTED_YET

```
#define FU_ERR_FIR_NOT_ACCEPTED_YET 8
```

firmware upgrade

firmware need to be accepted before

See also

[FU_AcceptFirmware\(\)](#)

6.20.2.16 FU_ERR_FLASH_ERASING

```
#define FU_ERR_FLASH_ERASING 3
```

error during erasing flash

6.20.2.17 FU_ERR_FLASH_WRITING

```
#define FU_ERR_FLASH_WRITING 2
```

error during writing to flash

6.20.2.18 FU_ERR_VERSION_IN_PACKAGE

```
#define FU_ERR_VERSION_IN_PACKAGE 15
```

part (should be EOT packet)

received data package with version

6.20.2.19 FU_MAX_DATA_SIZE

```
#define FU_MAX_DATA_SIZE (8 + 2)
```

max tx data in target device is 8 bytes (+2 for CRC) but for source device it depends on data size in block

6.20.2.20 FU_MAX_PROGRAM_SIZE

```
#define FU_MAX_PROGRAM_SIZE ((int)(FU_DESTINATION_2 - FU_DESTINATION_1))
```

6.20.2.21 FU_OPCODE_ABORT

```
#define FU_OPCODE_ABORT 5
```

Error occurrence.

6.20.2.22 FU_OPCODE_ACK

```
#define FU_OPCODE_ACK 4
```

Data or SOT ACK.

6.20.2.23 FU_OPCODE_DATA

```
#define FU_OPCODE_DATA 3
```

Data.

6.20.2.24 FU_OPCODE_EOT

```
#define FU_OPCODE_EOT 6
```

End Of Transmission.

6.20.2.25 FU_OPCODE_SOT

```
#define FU_OPCODE_SOT 2
```

Start Of Transmission.

6.20.2.26 FU_PROT_HEAD_SIZE

```
#define FU_PROT_HEAD_SIZE (1 + 1 + 1 + 1 + 2)
```

FU protocol head size.

6.20.2.27 FU_PROT_VERSION

```
#define FU_PROT_VERSION 1
```

protocol version

maximal value is 16

6.20.2.28 FU_VERSION_LOC

```
#define FU_VERSION_LOC (FLASH_PAGE_SIZE + 4)
```

place in address where current firmware version is stored offsed from FLASH_BASE

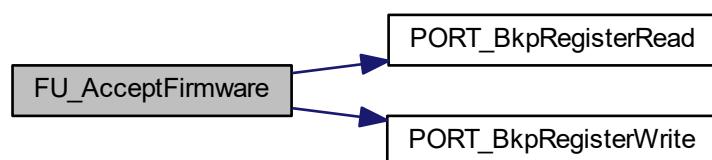
6.20.3 Function Documentation

6.20.3.1 FU_AcceptFirmware()

```
void FU_AcceptFirmware ( )
```

Mark current version as fully functional.

Here is the call graph for this function:



6.20.3.2 FU_GetCurrentFlashBase()

```
uint8_t* FU_GetCurrentFlashBase ( )  
return base address of current working firmware
```

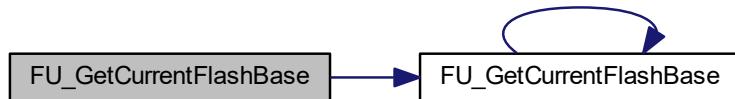
Returns

```
uint8_t* base address of current working firmware  
  
return base address of current working firmware
```

Returns

```
uint8_t* address of start a currently used flash program space
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.20.3.3 FU_HandleAsDevice()

```
void FU_HandleAsDevice (   
    const FU_prot * fup,  
    const prot_packet_info_t * info )  
  
process new message as device
```

Parameters

<i>fup</i>	message to process
<i>info</i>	extra informations about message

6.20.3.4 FU_Init()

```
void FU_Init (
    bool forceNoFirmwareCheck )
```

Initialize module.

Parameters

<i>forceNoFirmwareCheck</i>	set true for sink when firmware verification should be forced to succeed
-----------------------------	--

Here is the caller graph for this function:

**6.20.4 Variable Documentation****6.20.4.1 PROG_DESTINATION1**

```
char PROG_DESTINATION1
from linker script, address of partition A
```

6.20.4.2 PROG_DESTINATION2

```
char PROG_DESTINATION2
from linker script, address of partition B
```

6.20.4.3 PROG_SETTINGS1

char PROG_SETTINGS1

from linker script, settings location (part A)

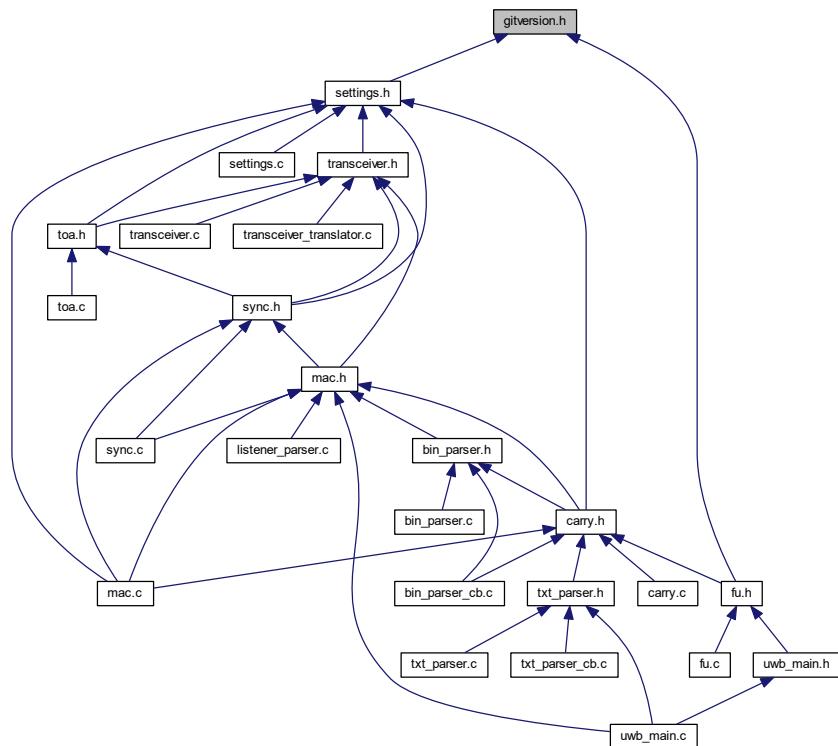
6.20.4.4 PROG_START1

char PROG_START1

from linker script, isr vector location (part A)

6.21 gitversion.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define __F_MAJOR__ 0
- #define __F_MINOR__ 2
- #define __F_HASH__ 0x3978caa7

6.21.1 Macro Definition Documentation

6.21.1.1 __F_HASH__

```
#define __F_HASH__ 0x3978caa7
```

6.21.1.2 __F_MAJOR__

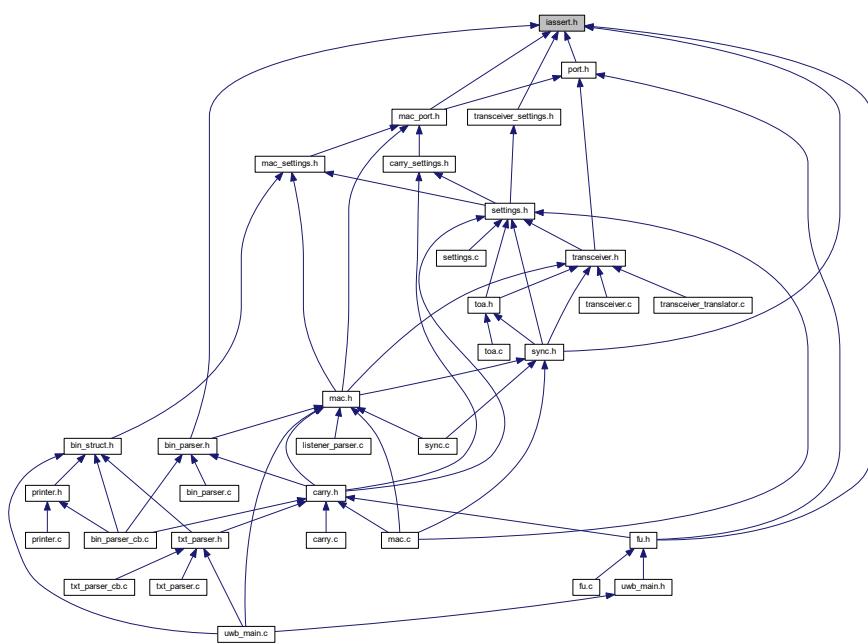
```
#define __F_MAJOR__ 0
```

6.21.1.3 __F_MINOR__

```
#define __F_MINOR__ 2
```

6.22 iassert.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define IASSERT(expr)

Functions

- void PORT_iassert_fun (const char *msg, int line)
described in port.h

6.22.1 Macro Definition Documentation

6.22.1.1 IASSERT

```
#define IASSERT(  
    expr )
```

Value:

```
do {  
    if (!(expr)) {  
        PORT_iassert_fun(__FUNCTION__, __LINE__); \  
    } \  
} while (0) \ \
```

6.22.2 Function Documentation

6.22.2.1 PORT_iassert_fun()

```
void PORT_iassert_fun (  
    const char * msg,  
    int line )
```

described in [port.h](#)

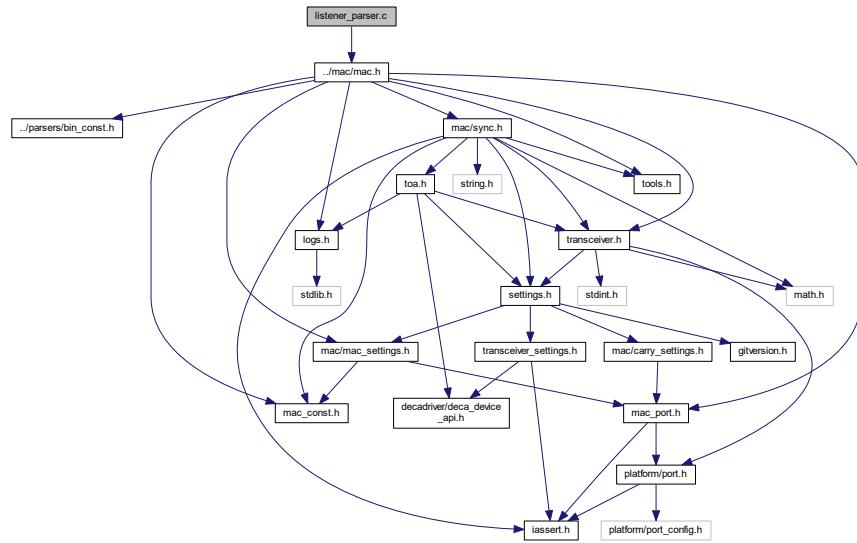
Parameters

msg	
line	

6.23 listener_parser.c File Reference

```
#include "../mac/mac.h"
```

Include dependency graph for listener_parser.c:



Macros

- `#define LISTENER_CASE(str, FC) case FC: str = #FC; break;`

Functions

- `void listener_parse (mac_buf_t *buf)`
- `void listener_isr (const dwt_cb_data_t *data)`
used by mac, externally implemented in platform folder

6.23.1 Macro Definition Documentation

6.23.1.1 LISTENER_CASE

```
#define LISTENER_CASE(
    str,
    FC ) case FC: str = #FC; break;
```

6.23.2 Function Documentation

6.23.2.1 listener_isr()

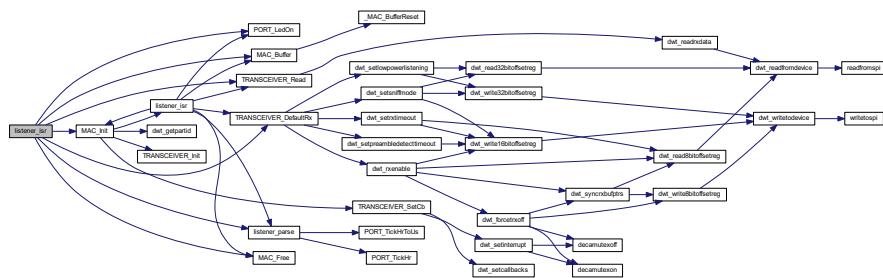
```
void listener_isr (
    const dwt_cb_data_t * data )
```

used by mac, externally implemented in platform folder

Parameters

in	<i>data</i>	full callback data
----	-------------	--------------------

Here is the call graph for this function:



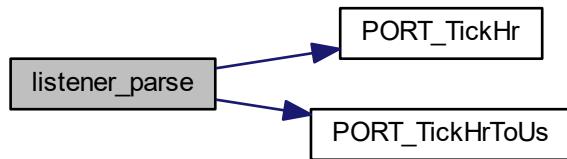
Here is the caller graph for this function:



6.23.2.2 listener_parse()

```
void listener_parse (
    mac_buf_t * buf )
```

Here is the call graph for this function:



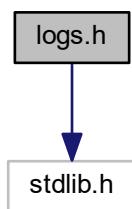
Here is the caller graph for this function:



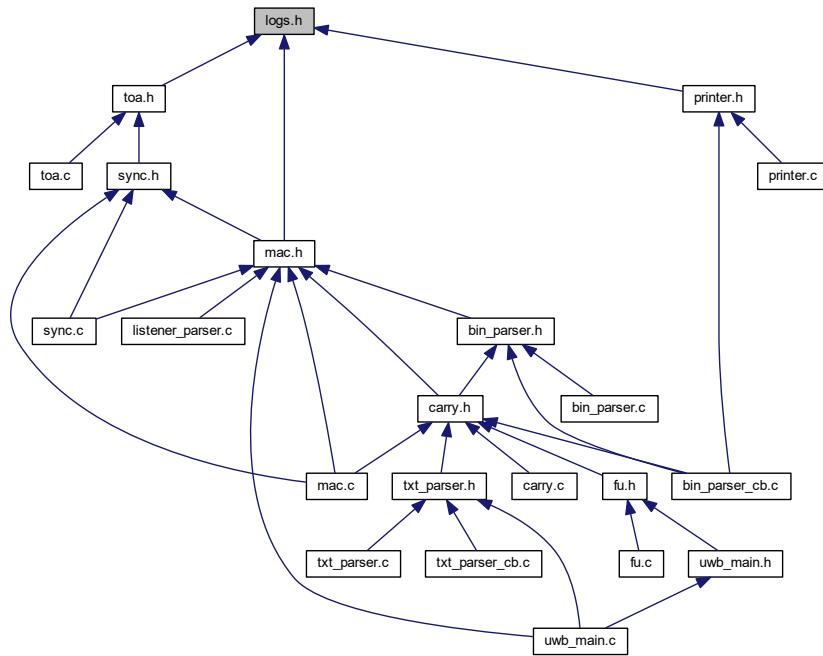
6.24 logs.h File Reference

Interactive assertion module.

```
#include <stdlib.h>
Include dependency graph for logs.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define LOG_CRIT(...)` `LOG_Text('C', __VA_ARGS__)`
critical log
- `#define LOG_ERR(...)` `LOG_Text('E', __VA_ARGS__)`
log error
- `#define LOG_WRN(...)` `LOG_Text('W', __VA_ARGS__)`
log warning
- `#define LOG_INF(...)` `LOG_Text('I', __VA_ARGS__)`
log info
- `#define LOG_DBG(...)` `LOG_Text('D', __VA_ARGS__)`
log debug
- `#define LOG_TEST(...)` `LOG_Text('T', __VA_ARGS__)`
log text
- `#define LOG_BASE64(BUF, LEN)` `log_bin(BUF, LEN)`
log binary data (in base64)

Functions

- `int LOG_Text (char type, const char *frm,...)`
log text data
- `int LOG_Bin (const void *bin, int size)`
log binary data

6.24.1 Detailed Description

Interactive assertion module.

Author

Karol Trzcinski

Date

2018-06-29

6.24.2 Macro Definition Documentation

6.24.2.1 LOG_BASE64

```
#define LOG_BASE64(  
    BUF,  
    LEN ) log_bin(BUF, LEN)
```

log binary data (in base64)

6.24.2.2 LOG_CRIT

```
#define LOG_CRIT(  
    ... ) LOG_Text('C', __VA_ARGS__)
```

critical log

6.24.2.3 LOG_DBG

```
#define LOG_DBG(  
    ... ) LOG_Text('D', __VA_ARGS__)
```

log debug

6.24.2.4 LOG_ERR

```
#define LOG_ERR(  
    ... ) LOG_Text('E', __VA_ARGS__)
```

log error

6.24.2.5 LOG_INF

```
#define LOG_INF( ... ) LOG_Text('I', __VA_ARGS__)
```

log info

6.24.2.6 LOG_TEST

```
#define LOG_TEST( ... ) LOG_Text('T', __VA_ARGS__)
```

log text

6.24.2.7 LOG_WRN

```
#define LOG_WRN( ... ) LOG_Text('W', __VA_ARGS__)
```

log warning

6.24.3 Function Documentation

6.24.3.1 LOG_Bin()

```
int LOG_Bin ( const void * bin, int size )
```

log binary data

implemented in platform folder

Parameters

in	<i>bin</i>	pointer to binary data
in	<i>size</i>	of binary data

Returns

int

Here is the caller graph for this function:



6.24.3.2 LOG_Text()

```
int LOG_Text (
    char type,
    const char * frm,
    ...
)
```

log text data

implemented in platform folder

Parameters

<i>type</i>	log type
<i>frm</i>	formating string
...	extra arguments

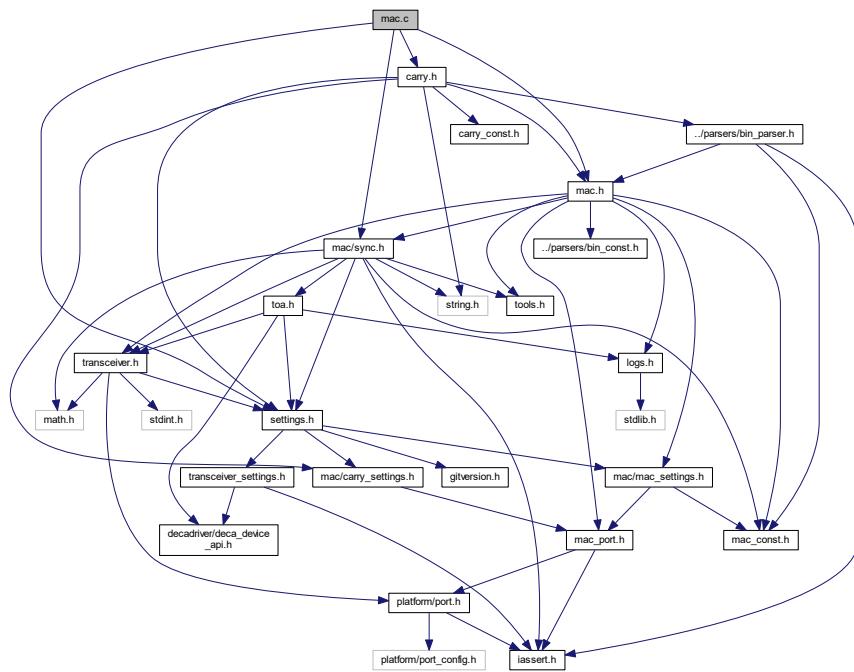
Returns

int

6.25 mac.c File Reference

```
#include "mac.h"
#include "../settings.h"
#include "carry.h"
#include "sync.h"
```

Include dependency graph for mac.c:



Functions

- `mac_buf_t * _MAC_BufGetOldestToTx ()`
- `void _MAC_BufferReset (mac_buf_t *buf)`
- `int MAC_TryTransmitFrameInSlot (int64_t glob_time)`
- `void MAC_Init ()`
 - initialize mac and transceiver*
- `unsigned int MAC_BeaconTimerGetMs ()`
 - return ms from last BeaconTimerReset or received unicast message*
- `void MAC_BeaconTimerReset ()`
 - reset beacon timer after sending a beacon message*
- `int MAC_ToSlotsTimeUs (int64_t glob_time)`
- `void MAC_UpdateSlotTimer (int32_t loc_slot_time_us, int64_t local_time)`
- `void MAC_YourSlotIsr ()`
 - should be called at the beginning of your slot time*
- `void MAC_AckFramesIsr (uint8_t seq_num)`
 - release buffer waiting for this ack*
- `mac_buf_t * MAC_Buffer ()`
 - reserve buffer*
- `void MAC_FillFrameTo (mac_buf_t *buf, dev_addr_t target)`
 - low level function, used only by carry module*
- `void MAC_SetFrameType (mac_buf_t *buf, uint8_t FR_CR_type)`
 - set frame type in 802.15.4 protocol*
- `mac_buf_t * MAC_BufferPrepare (dev_addr_t target, bool can_append)`
 - reserve buffer and fill mac protocol fields*
- `int MAC_BufLen (const mac_buf_t *buf)`

- `MAC_Free (mac_buf_t *buff)`
release buffer
- `MAC_Send (mac_buf_t *buf, bool ack_require)`
add frame to the transmit queue
- `int MAC_SendRanging (mac_buf_t *buf, uint8_t transceiver_flags)`
send packet as a ranging frame
- `unsigned int MAC_UsFromRx ()`
return time in ms from last received packed
- `unsigned char MAC_Read8 (mac_buf_t *frame)`
read one byte from frame and move rw pointer
- `void MAC_Write8 (mac_buf_t *frame, unsigned char value)`
write one byte from frame and move rw pointer
- `void MAC_Read (mac_buf_t *frame, void *destination, unsigned int len)`
read data chunk from frame and move rw pointer
- `void MAC_Write (mac_buf_t *frame, const void *source, unsigned int len)`
write chunk of bytes from frame and move rw pointer

Variables

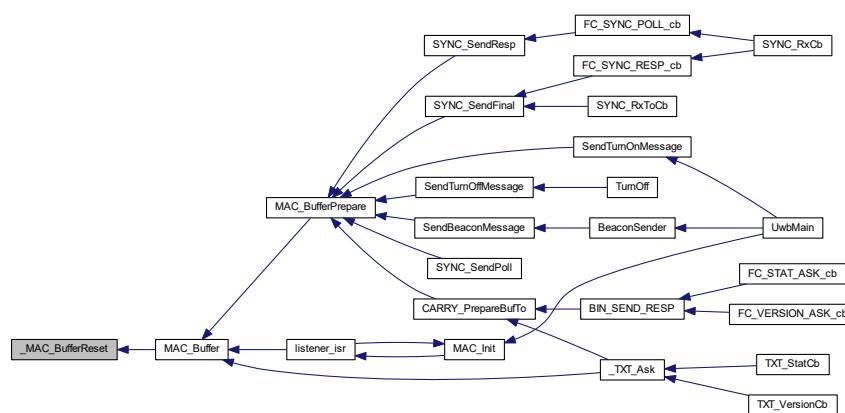
- `mac_instance_t mac`

6.25.1 Function Documentation

6.25.1.1 `_MAC_BufferReset()`

```
void _MAC_BufferReset (
    mac_buf_t * buf )
```

Here is the caller graph for this function:



6.25.1.2 _MAC_BufGetOldestToTx()

```
static mac_buf_t * _MAC_BufGetOldestToTx ( )
```

Here is the caller graph for this function:



6.25.1.3 MAC_AckFrameIsr()

```
void MAC_AckFrameIsr (
    uint8_t seq_num )
```

release buffer waiting for this ack

Parameters

<i>seq_num</i>	frame sequence number
----------------	-----------------------

6.25.1.4 MAC_BeaconTimerGetMs()

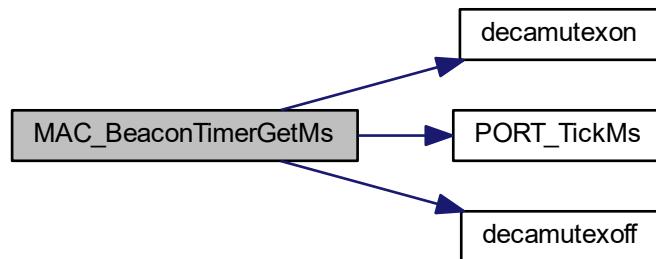
```
unsigned int MAC_BeaconTimerGetMs ( )
```

return ms from last BeaconTimerReset or received unicast message

Returns

unsigned int ms from last BeaconTimerReset or received unicast message

Here is the call graph for this function:



Here is the caller graph for this function:



6.25.1.5 `MAC_BeaconTimerReset()`

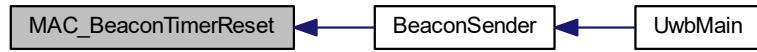
```
void MAC_BeaconTimerReset ( )
```

reset beacon timer after sending a beacon message

Here is the call graph for this function:



Here is the caller graph for this function:



6.25.1.6 MAC_Buffer()

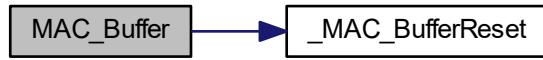
```
mac_buf_t* MAC_Buffer( )
```

reserve buffer

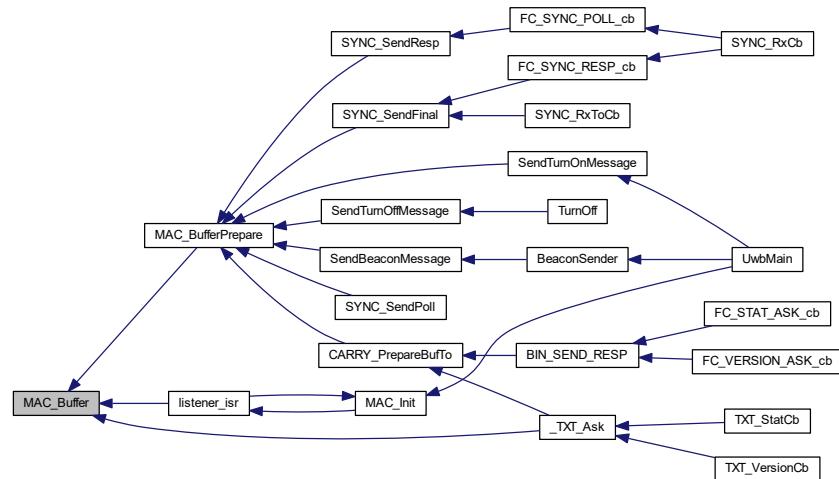
Returns

mac_buf_t* result buffer pointer or zero

Here is the call graph for this function:



Here is the caller graph for this function:



6.25.1.7 MAC_BufferPrepare()

```
mac_buf_t* MAC_BufferPrepare (
    dev_addr_t target,
    bool can_append )
```

reserve buffer and fill mac protocol fields

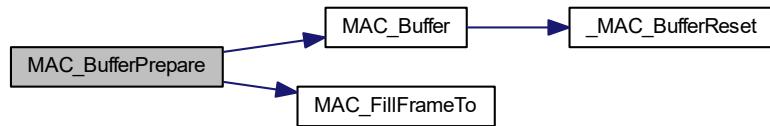
Parameters

in	<i>target</i>	address to target device in range of radio (without hops)
in	<i>can_append</i>	true if can appended to other packet to this target

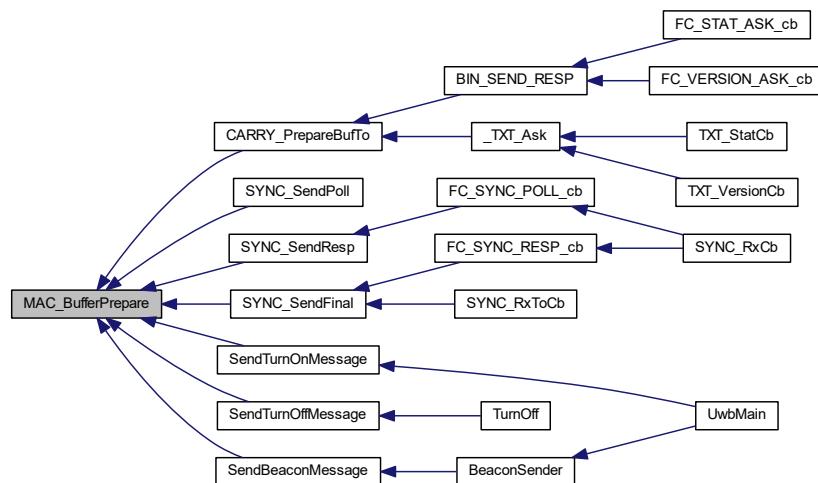
Returns

mac_buf_t* result buffer with filled fields or zero

Here is the call graph for this function:



Here is the caller graph for this function:



6.25.1.8 MAC_BufLen()

```
int MAC_BufLen (
    const mac_buf_t * buf )
```

return length of data already written in frame

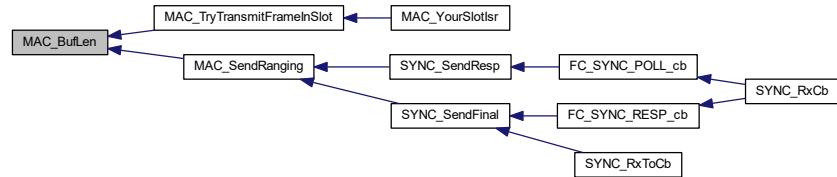
Parameters

in	<i>buf</i>	buffer to analyse
----	------------	-------------------

Returns

int return length of data already written in frame

Here is the caller graph for this function:



6.25.1.9 MAC_FillFrameTo()

```
void MAC_FillFrameTo (
    mac_buf_t * buf,
    dev_addr_t target )
```

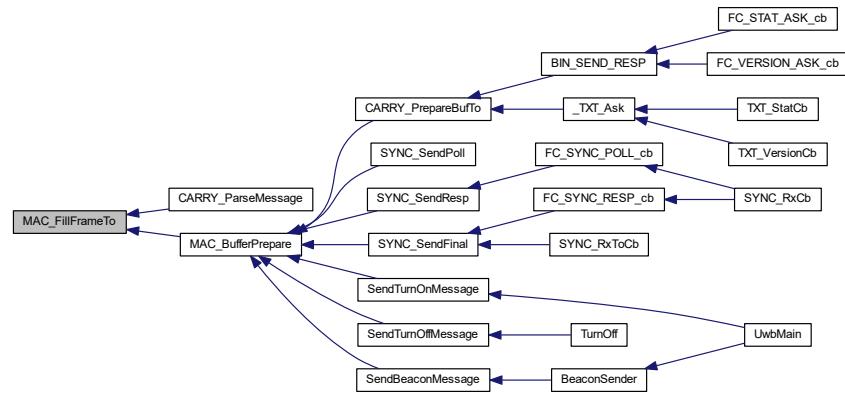
low level function, used only by carry module

fill 802.15.4 fields

Parameters

in, out	<i>buf</i>	
in	<i>target</i>	device address

Here is the caller graph for this function:



6.25.1.10 MAC_Free()

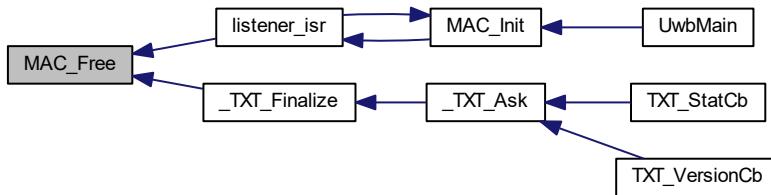
```
void MAC_Free (
    mac_buf_t * buf )
```

release buffer

Parameters

in	buf	buffer to release
----	-----	-------------------

Here is the caller graph for this function:

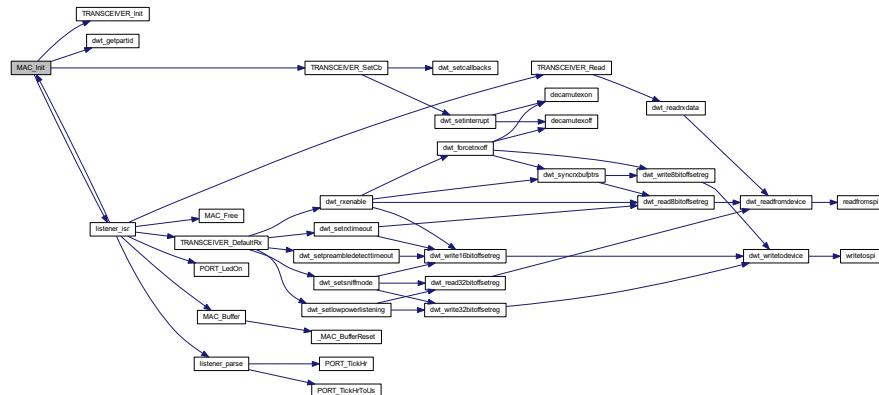


6.25.1.11 MAC_Init()

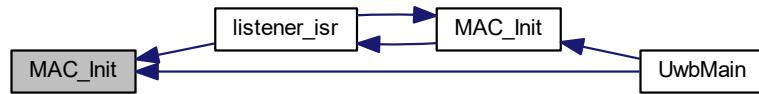
```
void MAC_Init ( )
```

initialize mac and transceiver

Here is the call graph for this function:



Here is the caller graph for this function:



6.25.1.12 MAC_Read()

```
void MAC_Read (mac_buf_t * frame,  
                void * destination,  
                unsigned int len )
```

read data chunk from frame and move rw pointer

Parameters

in	<i>frame</i>	to read
out	<i>destination</i>	address
in	<i>len</i>	number of bytes to write

6.25.1.13 MAC_Read8()

```
unsigned char MAC_Read8 (
    mac_buf_t * frame )
```

read one byte from frame and move rw pointer

Parameters

in	<i>frame</i>	to read
----	--------------	---------

Returns

unsigned char

6.25.1.14 MAC_Send()

```
void MAC_Send (
    mac_buf_t * buf,
    bool ack_require )
```

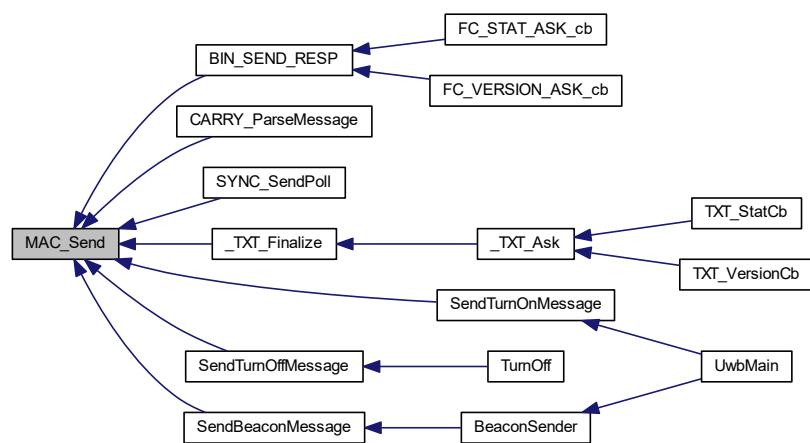
add frame to the transmit queue

It is default way to send data packets. Buf will be released after transmission

Parameters

in	<i>buf</i>	with frame
in	<i>ack_require</i>	

Here is the caller graph for this function:



6.25.1.15 MAC_SendRanging()

```
int MAC_SendRanging (
    mac_buf_t * buf,
    uint8_t transceiver_flags )
```

send packet as a ranging frame

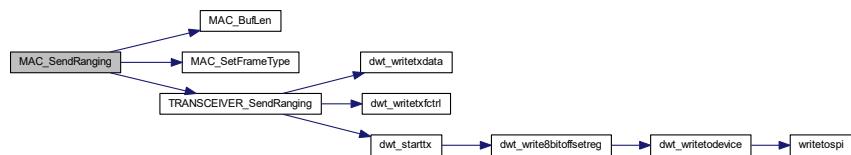
Parameters

<i>buf</i>	to transmit
<i>transceiver_flags</i>	<ul style="list-style-type: none"> • DWT_START_TX_IMMEDIATE • DWT_START_TX_DELAYED • DWT_RESPONSE_EXPECTED

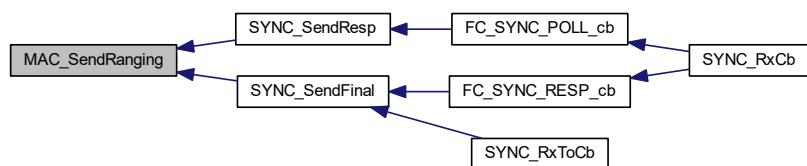
Returns

int 0 if succeed or error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.25.1.16 MAC_SetFrameType()

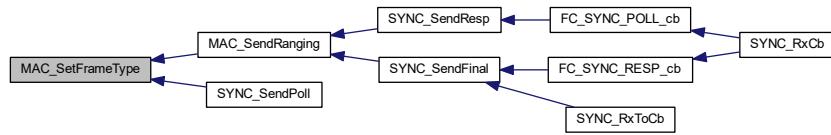
```
void MAC_SetFrameType (
    mac_buf_t * buf,
    uint8_t FR_CR_type )
```

set frame type in 802.15.4 protocol

Parameters

in,out	<i>buf</i>	target buffer
in	<i>FR_CR_type</i>	FC_CR_xx macro

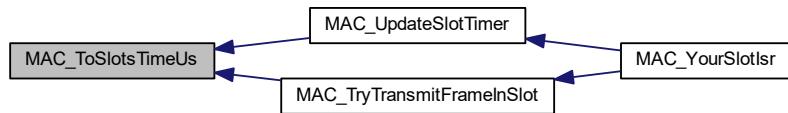
Here is the caller graph for this function:



6.25.1.17 MAC_ToSlotsTimeUs()

```
int MAC_ToSlotsTimeUs (
    int64_t glob_time )
```

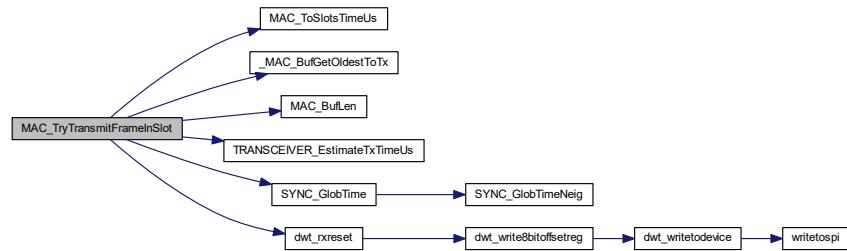
Here is the caller graph for this function:



6.25.1.18 MAC_TryTransmitFrameInSlot()

```
int MAC_TryTransmitFrameInSlot (
    int64_t glob_time )
```

Here is the call graph for this function:



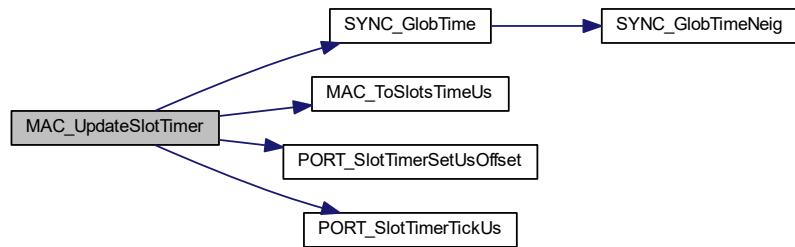
Here is the caller graph for this function:



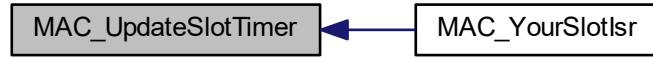
6.25.1.19 MAC_UpdateSlotTimer()

```
void MAC_UpdateSlotTimer (
    int32_t loc_slot_time_us,
    int64_t local_time )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.25.1.20 MAC_UsFromRx()

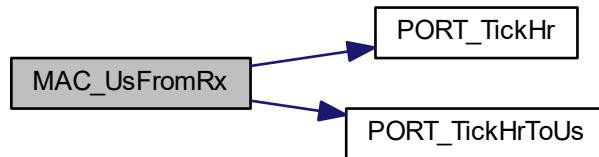
```
unsigned int MAC_UsFromRx ( )
```

return time in ms from last received packed

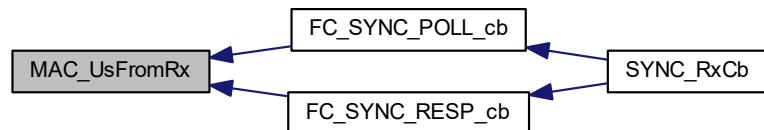
Returns

unsigned int time in ms from last received packed

Here is the call graph for this function:



Here is the caller graph for this function:



6.25.1.21 MAC_Write()

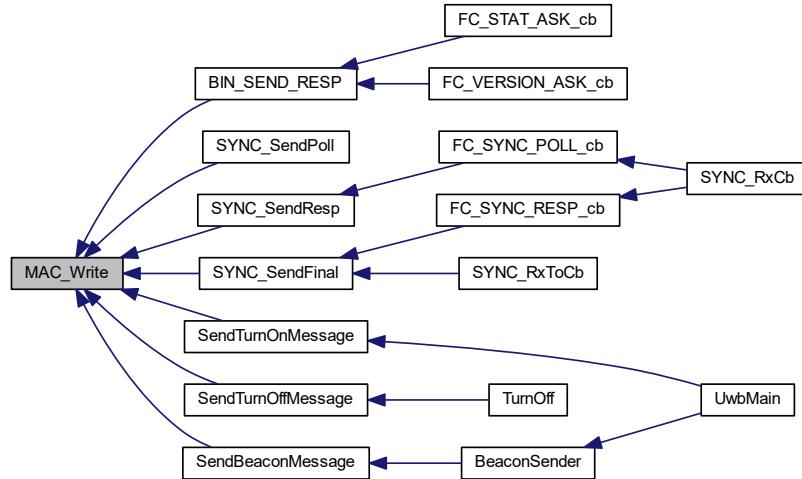
```
void MAC_Write (
    mac_buf_t * frame,
    const void * src,
    unsigned int len )
```

write chunk of bytes from frame and move rw pointer

Parameters

in, out	<i>frame</i>	to write
in	<i>src</i>	address of data to write
in	<i>len</i>	number of bytes to write

Here is the caller graph for this function:



6.25.1.22 MAC_Write8()

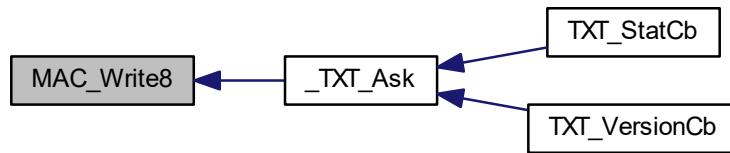
```
void MAC_Write8 (
    mac_buf_t * frame,
    unsigned char value )
```

write one byte from frame and move rw pointer

Parameters

in, out	<i>frame</i>	to write
	<i>value</i>	

Here is the caller graph for this function:

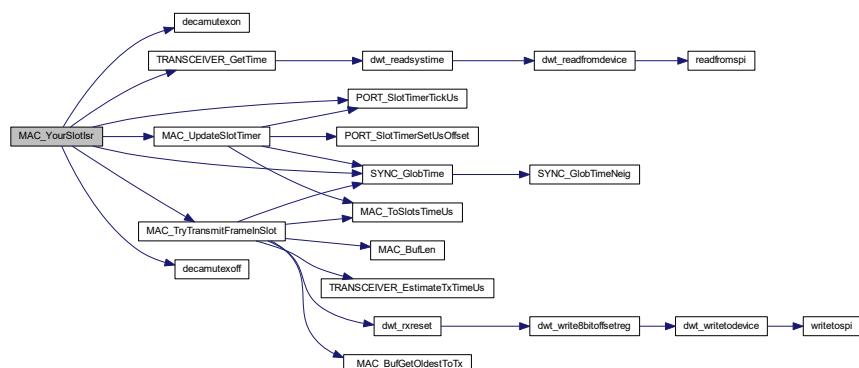


6.25.1.23 MAC_YourSlotIsr()

```
void MAC_YourSlotIsr( )
```

should be called at the beginning of your slot time

Here is the call graph for this function:



6.25.2 Variable Documentation

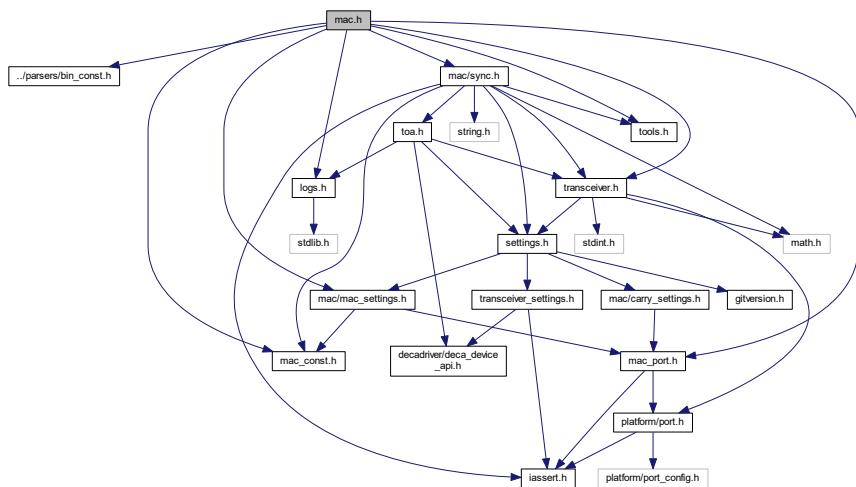
6.25.2.1 mac

```
mac_instance_t mac
```

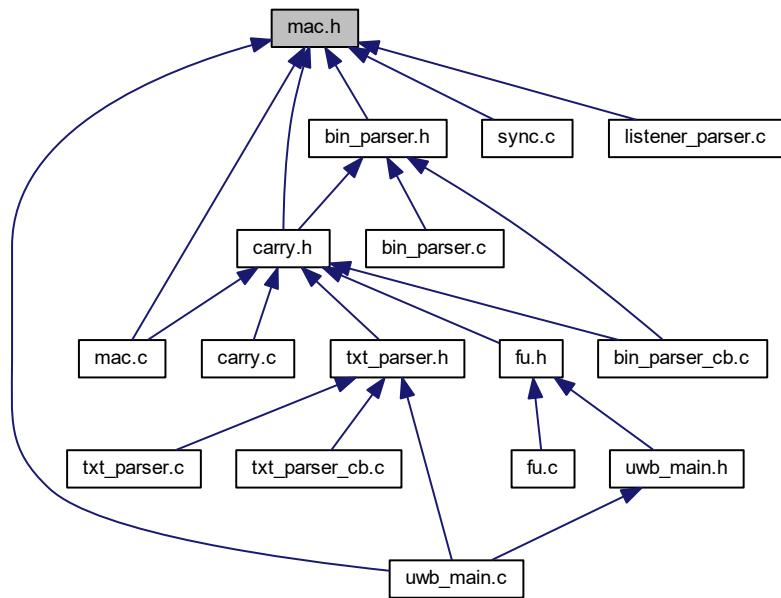
6.26 mac.h File Reference

Medium Access Control layer 802.15.4.

```
#include "../parsers/bin_const.h"
#include "logs.h"
#include "transceiver.h"
#include "mac/mac_const.h"
#include "mac/mac_port.h"
#include "mac/mac_settings.h"
#include "mac/sync.h"
#include "tools.h"
Include dependency graph for mac.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [mac_buf_t](#)
- struct [mac_instance_t](#)

Macros

- #define MAC_TRACE_ENABLED 0
- #define MAC_TRACE(... ALL_UNUSED(__VA_ARGS__))
- #define MAC_USAGE_BUF_START(name)
- #define MAC_USAGE_BUF_STOP(name)
- #define MAC_HEAD_LENGTH (2 + 1 + sizeof([pan_dev_addr_t](#)) + 2 * sizeof([dev_addr_t](#)))
mac protocol minimum head size

Functions

- void [listener_isr](#) (const [dwt_cb_data_t](#) *data)
used by mac, externally implemented in platform folder
- void [MAC_Init](#) ()
initialize mac and transceiver
- unsigned int [MAC_BeaconTimerGetMs](#) ()
return ms from last BeaconTimerReset or received unicast message
- void [MAC_BeaconTimerReset](#) ()
reset beacon timer after sending a beacon message
- int [MAC_ToSlotsTime](#) (int64_t glob_time)

- time converter*
- void **MAC_YourSlotIsr** ()
should be called at the beginning of your slot time
 - void **MAC_AckFrameIsr** (uint8_t seq_num)
release buffer waiting for this ack
 - **mac_buf_t * MAC_Buffer** ()
reserve buffer
 - int **MAC_BufLen** (const **mac_buf_t** *buf)
return length of data already written in frame
 - void **MAC_FillFrameTo** (**mac_buf_t** *buf, **dev_addr_t** target)
low level function, used only by carry module
 - void **MAC_SetFrameType** (**mac_buf_t** *buf, uint8_t FR_CR_type)
set frame type in 802.15.4 protocol
 - **mac_buf_t * MAC_BufferPrepare** (**dev_addr_t** target, bool can_append)
reserve buffer and fill mac protocol fields
 - void **MAC_Free** (**mac_buf_t** *buf)
release buffer
 - void **MAC_Send** (**mac_buf_t** *buf, bool ack_require)
add frame to the transmit queue
 - int **MAC_SendRanging** (**mac_buf_t** *buf, uint8_t transceiver_flags)
send packet as a ranging frame
 - unsigned int **MAC_UsFromRx** ()
return time in ms from last received packed
 - unsigned char **MAC_Read8** (**mac_buf_t** *frame)
read one byte from frame and move rw pointer
 - void **MAC_Write8** (**mac_buf_t** *frame, unsigned char value)
write one byte from frame and move rw pointer
 - void **MAC_Read** (**mac_buf_t** *frame, void *destination, unsigned int len)
read data chunk from frame and move rw pointer
 - void **MAC_Write** (**mac_buf_t** *frame, const void *src, unsigned int len)
write chunk of bytes from frame and move rw pointer

6.26.1 Detailed Description

Medium Access Control layer 802.15.4.

Author

Karol Trzcinski

Date

2018-07-02

This module is responsible for packets collision avoidance, retransmissions and ack response sending.

There should be timer enabled to tick every slot period to start transmission routine in correct time (). For local time synchronization to global one is responsible SYNC file.

6.26.2 Macro Definition Documentation

6.26.2.1 MAC_HEAD_LENGTH

```
#define MAC_HEAD_LENGTH (2 + 1 + sizeof(pan_dev_addr_t) + 2 * sizeof(dev_addr_t))
mac protocol minimum head size
```

6.26.2.2 MAC_TRACE

```
#define MAC_TRACE(
    ... ) ALL_UNUSED(__VA_ARGS__)
```

6.26.2.3 MAC_TRACE_ENABLED

```
#define MAC_TRACE_ENABLED 0
```

6.26.2.4 MAC_USAGE_BUF_START

```
#define MAC_USAGE_BUF_START(
    name )
```

Value:

```
mac_buf_t *#name = MAC_Buffer(); \
if (#name != 0) {
```

6.26.2.5 MAC_USAGE_BUF_STOP

```
#define MAC_USAGE_BUF_STOP(
    name )
```

Value:

```
MAC_Free(#name); \
}
```

6.26.3 Function Documentation

6.26.3.1 listener_isr()

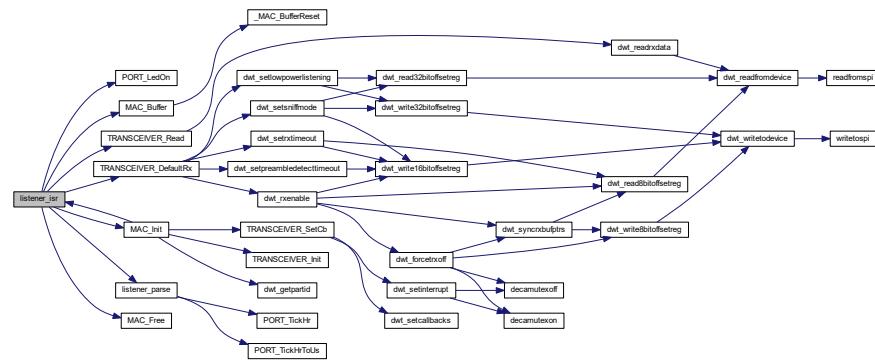
```
void listener_isr (
    const dwt_cb_data_t * data )
```

used by mac, externally implemented in platform folder

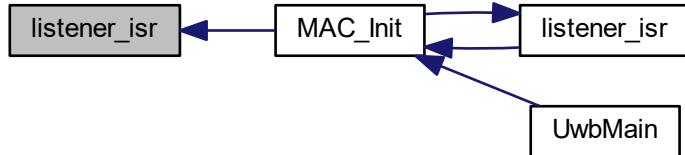
Parameters

in	<i>data</i>	full callback data
----	-------------	--------------------

Here is the call graph for this function:



Here is the caller graph for this function:



6.26.3.2 MAC_AckFrameIsr()

```
void MAC_AckFrameIsr (
    uint8_t seq_num )
```

release buffer waiting for this ack

Parameters

<i>seq_num</i>	frame sequence number
----------------	-----------------------

6.26.3.3 MAC_BeaconTimerGetMs()

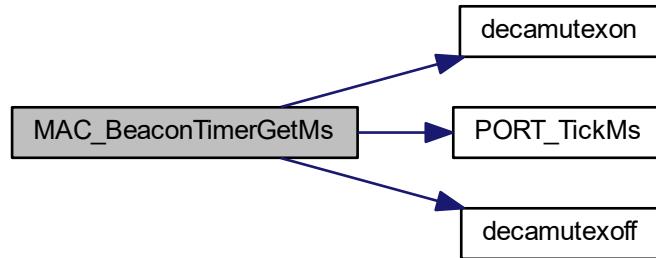
```
unsigned int MAC_BeaconTimerGetMs ( )
```

return ms from last BeaconTimerReset or received unicast message

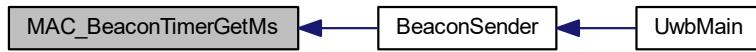
Returns

```
unsigned int ms from last BeaconTimerReset or received unicast message
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.26.3.4 MAC_BeaconTimerReset()

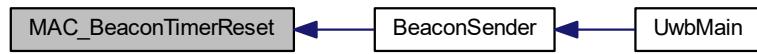
```
void MAC_BeaconTimerReset ( )
```

reset beacon timer after sending a beacon message

Here is the call graph for this function:



Here is the caller graph for this function:



6.26.3.5 MAC_Buffer()

```
mac_buf_t* MAC_Buffer( )
```

reserve buffer

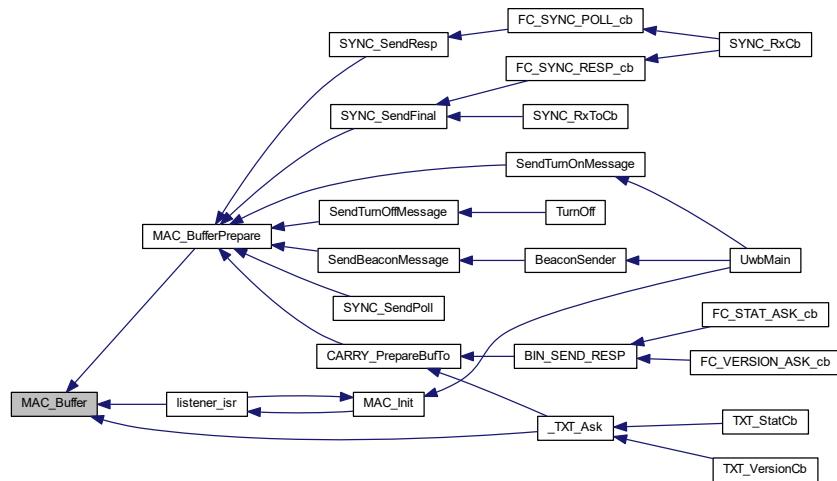
Returns

mac_buf_t* result buffer pointer or zero

Here is the call graph for this function:



Here is the caller graph for this function:



6.26.3.6 MAC_BufferPrepare()

```
mac_buf_t* MAC_BufferPrepare (
    dev_addr_t target,
    bool can_append )
```

reserve buffer and fill mac protocol fields

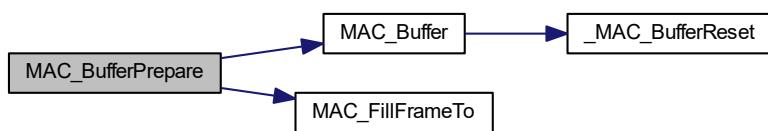
Parameters

in	<i>target</i>	address to target device in range of radio (without hops)
in	<i>can_append</i>	true if can appended to other packet to this target

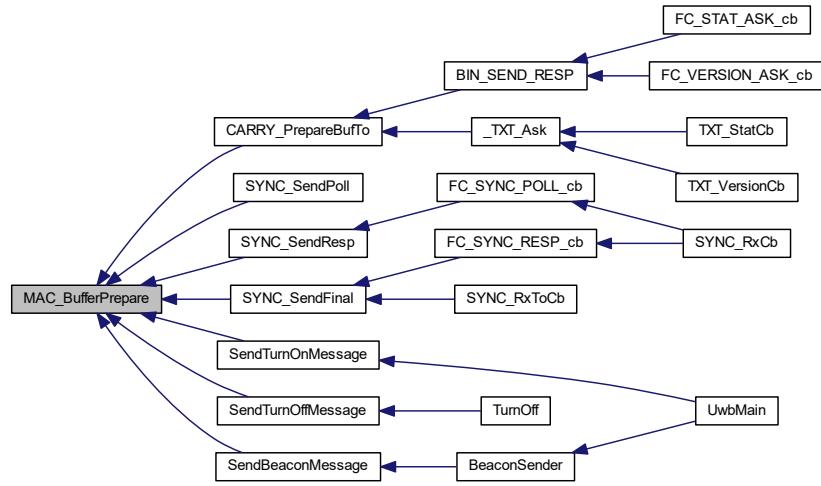
Returns

mac_buf_t* result buffer with filled fields or zero

Here is the call graph for this function:



Here is the caller graph for this function:



6.26.3.7 MAC_BufLen()

```
int MAC_BufLen (
    const mac_buf_t * buf )
```

return length of data already written in frame

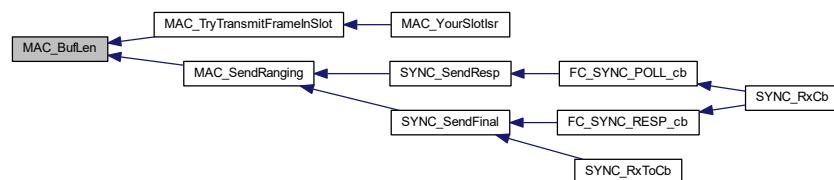
Parameters

in	<i>buf</i>	buffer to analyse
----	------------	-------------------

Returns

int return length of data already written in frame

Here is the caller graph for this function:



6.26.3.8 MAC_FillFrameTo()

```
void MAC_FillFrameTo (
    mac_buf_t * buf,
    dev_addr_t target )
```

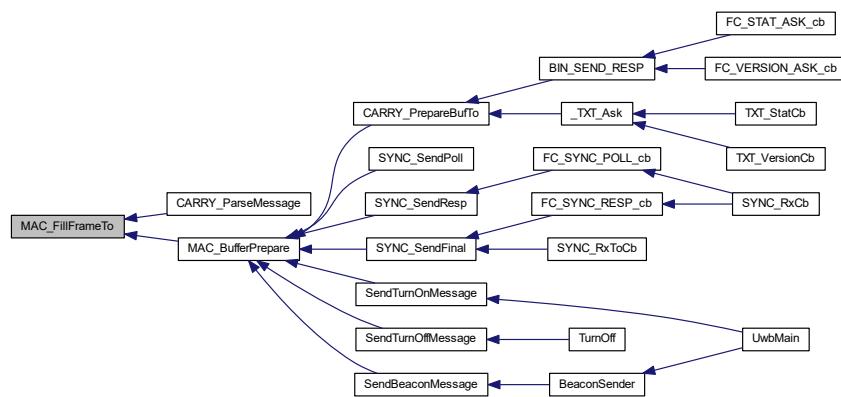
low level function, used only by carry module

fill 802.15.4 fields

Parameters

in, out	<i>buf</i>	
in	<i>target</i>	device address

Here is the caller graph for this function:



6.26.3.9 MAC_Free()

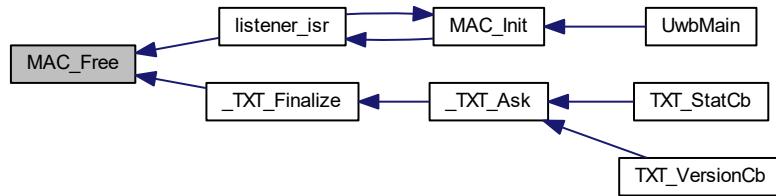
```
void MAC_Free (
    mac_buf_t * buf )
```

release buffer

Parameters

in	<i>buf</i>	buffer to release
----	------------	-------------------

Here is the caller graph for this function:

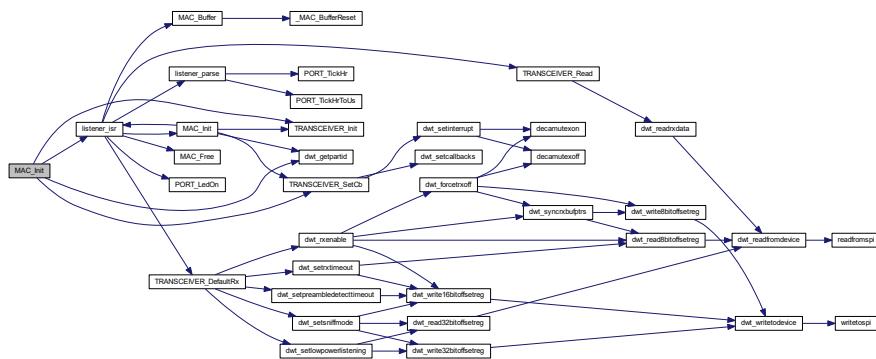


6.26.3.10 MAC_Init()

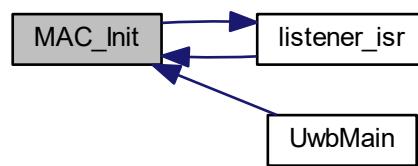
```
void MAC_Init( )
```

initialize mac and transceiver

Here is the call graph for this function:



Here is the caller graph for this function:



6.26.3.11 MAC_Read()

```
void MAC_Read (
    mac_buf_t * frame,
    void * destination,
    unsigned int len )
```

read data chunk from frame and move rw pointer

Parameters

in	<i>frame</i>	to read
out	<i>destination</i>	address
in	<i>len</i>	number of bytes to write

6.26.3.12 MAC_Read8()

```
unsigned char MAC_Read8 (
    mac_buf_t * frame )
```

read one byte from frame and move rw pointer

Parameters

in	<i>frame</i>	to read
----	--------------	---------

Returns

unsigned char

6.26.3.13 MAC_Send()

```
void MAC_Send (
    mac_buf_t * buf,
    bool ack_require )
```

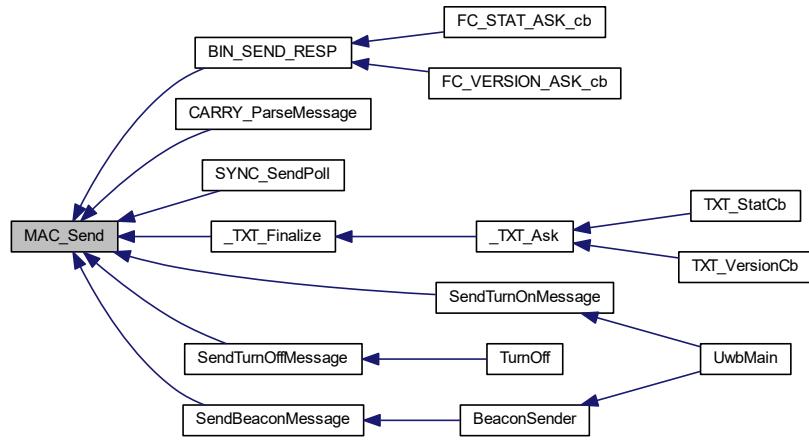
add frame to the transmit queue

It is default way to send data packets. Buf will be released after transmission

Parameters

in	<i>buf</i>	with frame
in	<i>ack_require</i>	

Here is the caller graph for this function:



6.26.3.14 MAC_SendRanging()

```
int MAC_SendRanging (
    mac_buf_t * buf,
    uint8_t transceiver_flags )
```

send packet as a ranging frame

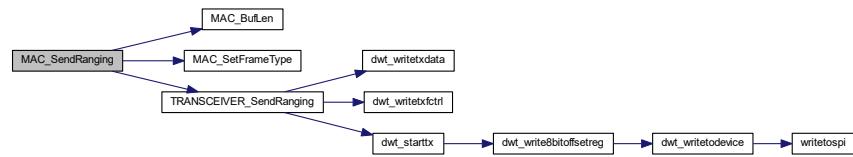
Parameters

<code>buf</code>	to transmit
<code>transceiver_flags</code>	<ul style="list-style-type: none"> • DWT_START_TX_IMMEDIATE • DWT_START_TX_DELAYED • DWT_RESPONSE_EXPECTED

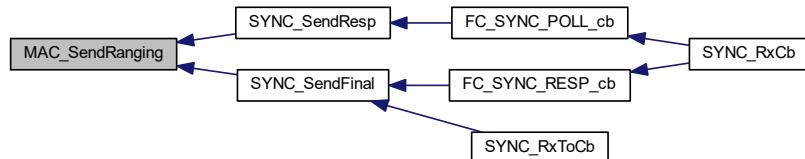
Returns

int 0 if succeed or error code

Here is the call graph for this function:



Here is the caller graph for this function:

**6.26.3.15 MAC_SetFrameType()**

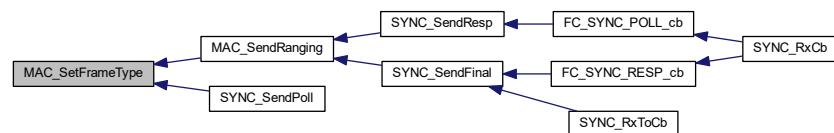
```
void MAC_SetFrameType (
    mac_buf_t * buf,
    uint8_t FR_CR_type )
```

set frame type in 802.15.4 protocol

Parameters

in, out	<i>buf</i>	target buffer
in	<i>FR_CR_type</i>	FC_CR_xx macro

Here is the caller graph for this function:



6.26.3.16 MAC_ToSlotsTime()

```
int MAC_ToSlotsTime (
    int64_t glob_time )
```

time converter

Parameters

<i>glob_time</i>	in Dwt time units
------------------	-------------------

Returns

int slot time in us

6.26.3.17 MAC_UsFromRx()

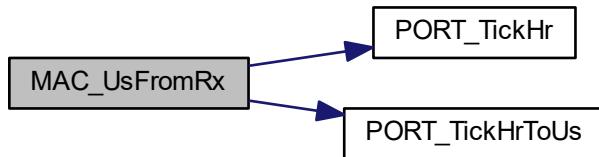
```
unsigned int MAC_UsFromRx ( )
```

return time in ms from last received packed

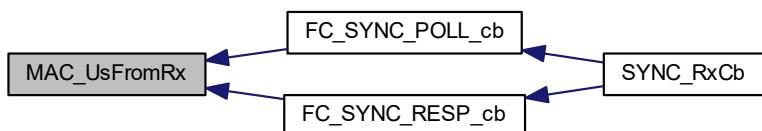
Returns

unsigned int time in ms from last received packed

Here is the call graph for this function:



Here is the caller graph for this function:



6.26.3.18 MAC_Write()

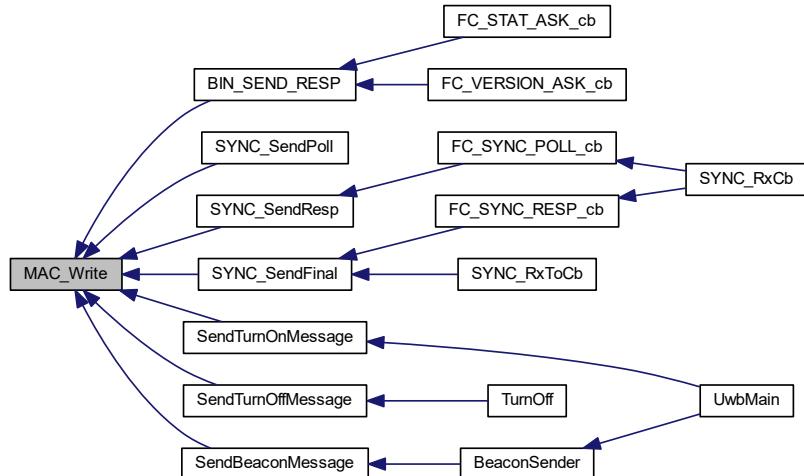
```
void MAC_Write (
    mac_buf_t * frame,
    const void * src,
    unsigned int len )
```

write chunk of bytes from frame and move rw pointer

Parameters

in, out	<i>frame</i>	to write
in	<i>src</i>	address of data to write
in	<i>len</i>	number of bytes to write

Here is the caller graph for this function:



6.26.3.19 MAC_Write8()

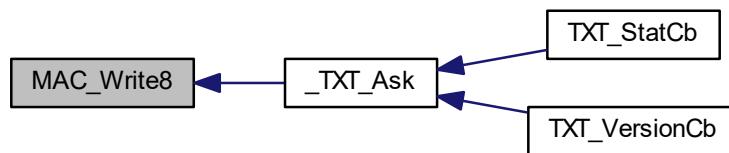
```
void MAC_Write8 (
    mac_buf_t * frame,
    unsigned char value )
```

write one byte from frame and move rw pointer

Parameters

in, out	<i>frame</i>	to write
	<i>value</i>	

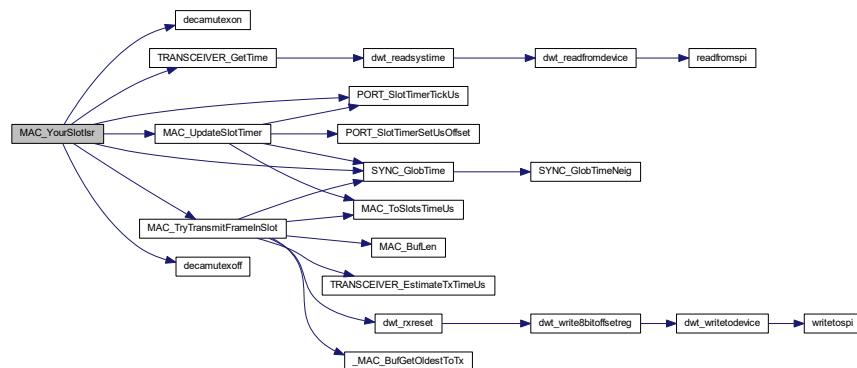
Here is the caller graph for this function:

6.26.3.20 `MAC_YourSlotIsr()`

```
void MAC_YourSlotIsr( )
```

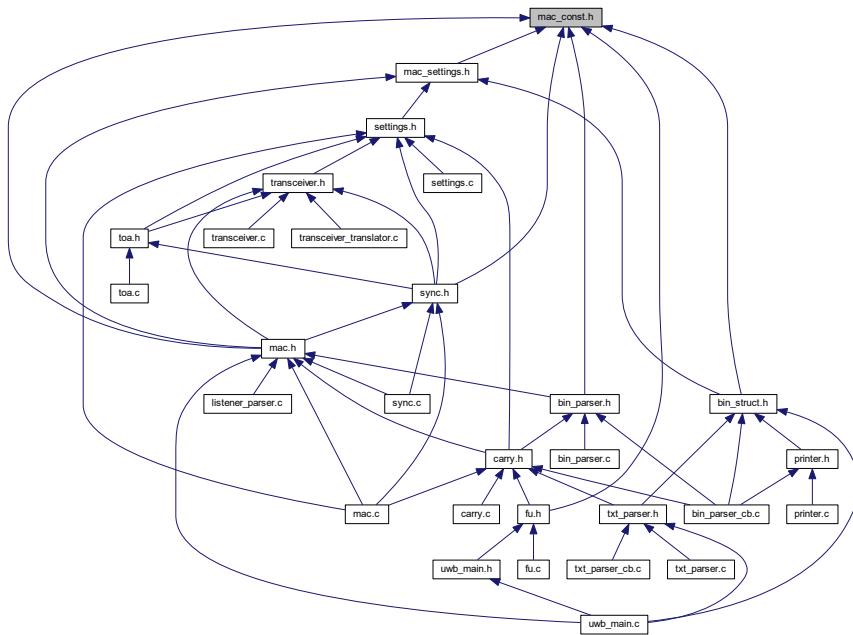
should be called at the beginning of your slot time

Here is the call graph for this function:



6.27 mac_const.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct [prot_packet_info_t](#)
packet extra information struct

Macros

- #define [FR_CR_BEACON](#) 0x00
- #define [FR_CR_DATA](#) 0x01
- #define [FR_CR_ACK](#) 0x02
- #define [FR_CR_MAC](#) 0x03
- #define [FR_CR_TYPE_MASK](#) 0x07
- #define [FR_CR_SECURE](#) 0x08
- #define [FR_CR_PENDING](#) 0x10
- #define [FR_CR_RACK](#) 0x20
- #define [FR_CR_PAN](#) 0x40
- #define [FR_CRH_DA_NONE](#) 0x00
- #define [FR_CRH_DA_SHORT](#) 0x08
- #define [FR_CRH_DA_LONG](#) 0x0C
- #define [FR_CRH_FVER0](#) 0x00
- #define [FR_CRH_FVER1](#) 0x01
- #define [FR_CRH_SA_NONE](#) 0x00
- #define [FR_CRH_SA_SHORT](#) 0x80
- #define [FR_CRH_SA_LONG](#) 0xC0
- #define [ADDR_BROADCAST](#) 0xffff
- #define [ADDR_ANCHOR_FLAG](#) 0x8000

Typedefs

- `typedef unsigned short dev_addr_t`
- `typedef unsigned short pan_dev_addr_t`

Enumerations

- `enum mac_buf_state {
 FREE, BUSY, WAIT_FOR_TX, WAIT_FOR_TX_ACK,
 WAIT_FOR_ACK }`

6.27.1 Macro Definition Documentation

6.27.1.1 ADDR_ANCHOR_FLAG

```
#define ADDR_ANCHOR_FLAG 0x8000
```

6.27.1.2 ADDR_BROADCAST

```
#define ADDR_BROADCAST 0xfffff
```

6.27.1.3 FR_CR_ACK

```
#define FR_CR_ACK 0x02
```

6.27.1.4 FR_CR_BEACON

```
#define FR_CR_BEACON 0x00
```

6.27.1.5 FR_CR_DATA

```
#define FR_CR_DATA 0x01
```

6.27.1.6 FR_CR_MAC

```
#define FR_CR_MAC 0x03
```

6.27.1.7 FR_CR_PAN

```
#define FR_CR_PAN 0x40
```

6.27.1.8 FR_CR_PENDING

```
#define FR_CR_PENDING 0x10
```

6.27.1.9 FR_CR_RACK

```
#define FR_CR_RACK 0x20
```

6.27.1.10 FR_CR_SECURE

```
#define FR_CR_SECURE 0x08
```

6.27.1.11 FR_CR_TYPE_MASK

```
#define FR_CR_TYPE_MASK 0x07
```

6.27.1.12 FR_CRH_DA_LONG

```
#define FR_CRH_DA_LONG 0x0C
```

6.27.1.13 FR_CRH_DA_NONE

```
#define FR_CRH_DA_NONE 0x00
```

6.27.1.14 FR_CRH_DA_SHORT

```
#define FR_CRH_DA_SHORT 0x08
```

6.27.1.15 FR_CRH_FVER0

```
#define FR_CRH_FVER0 0x00
```

6.27.1.16 FR_CRH_FVER1

```
#define FR_CRH_FVER1 0x01
```

6.27.1.17 FR_CRH_SA_LONG

```
#define FR_CRH_SA_LONG 0xC0
```

6.27.1.18 FR_CRH_SA_NONE

```
#define FR_CRH_SA_NONE 0x00
```

6.27.1.19 FR_CRH_SA_SHORT

```
#define FR_CRH_SA_SHORT 0x80
```

6.27.2 Typedef Documentation

6.27.2.1 dev_addr_t

```
typedef unsigned short dev_addr_t
```

6.27.2.2 pan_dev_addr_t

```
typedef unsigned short pan_dev_addr_t
```

6.27.3 Enumeration Type Documentation

6.27.3.1 mac_buf_state

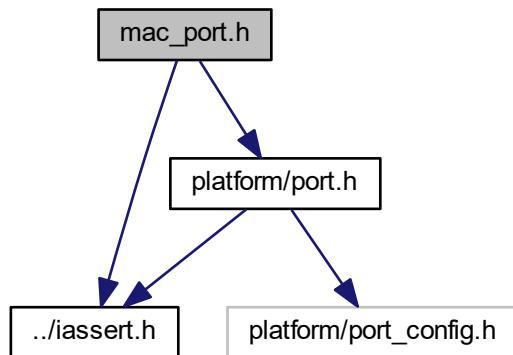
```
enum mac_buf_state
```

Enumerator

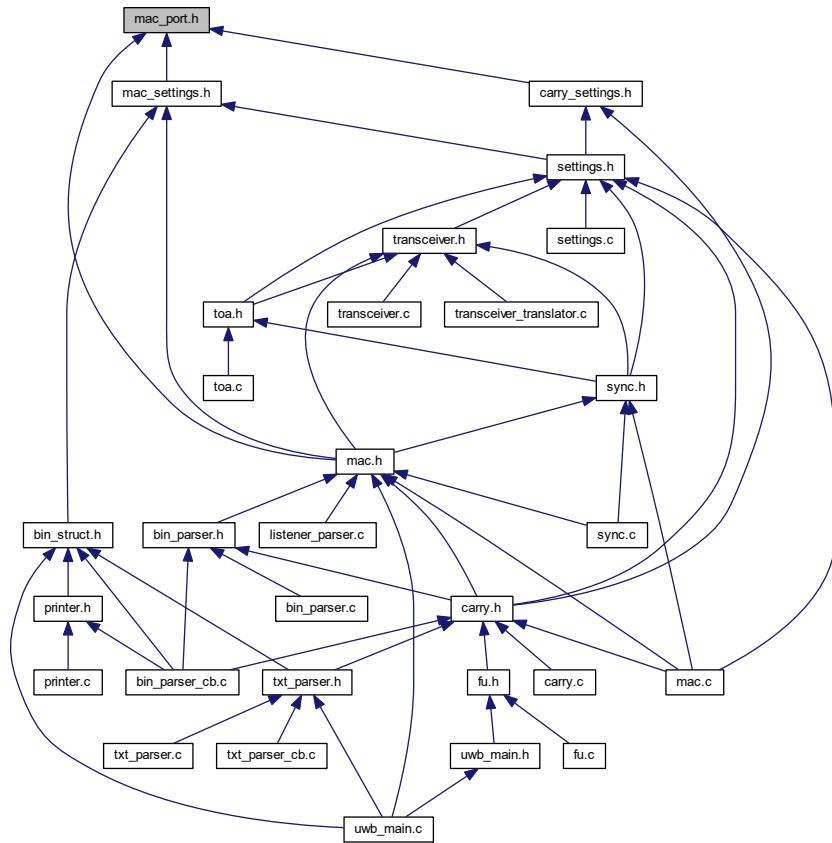
FREE	packet is free
BUSY	packet under edition
WAIT_FOR_TX	when packet wait on tx queue
WAIT_FOR_TX_ACK	when packet wait on tx queue and need ack
WAIT_FOR_ACK	wait for receive ack after transmission

6.28 mac_port.h File Reference

```
#include "../iassert.h"
#include "platform/port.h"
Include dependency graph for mac_port.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- #define MAC_ASSERT(expr) IASSERT(expr)
- #define MAC_LOG_ERR(msg) LOG_ERR(msg)

Typedefs

- typedef unsigned int mac_buff_time_t

6.28.1 Macro Definition Documentation

6.28.1.1 MAC_ASSERT

```
#define MAC_ASSERT(
    expr ) IASSERT(expr)
```

6.28.1.2 MAC_LOG_ERR

```
#define MAC_LOG_ERR(  
    msg ) LOG_ERR(msg)
```

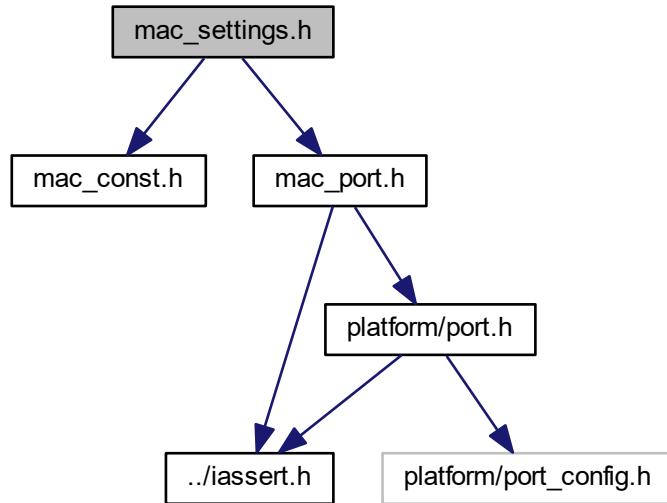
6.28.2 Typedef Documentation

6.28.2.1 mac_buff_time_t

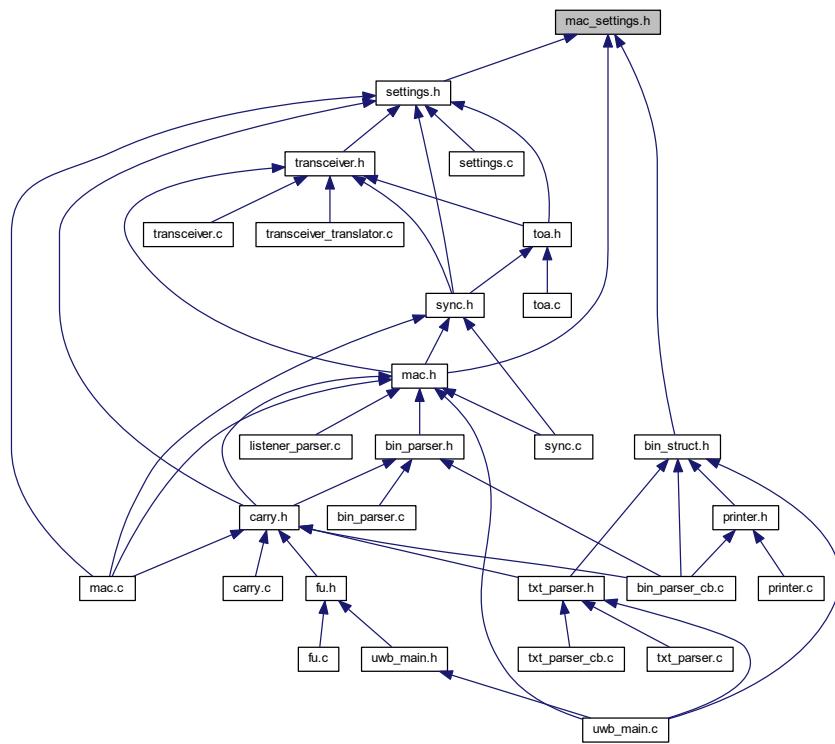
```
typedef unsigned int mac_buff_time_t
```

6.29 mac_settings.h File Reference

```
#include "mac_const.h"  
#include "mac_port.h"  
Include dependency graph for mac_settings.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [toa_settings_t](#)
- struct [mac_settings_t](#)

Macros

- #define MAC_BUF_CNT 5
- #define MAC_BUF_LEN 128
- #define TOA_MAX_DEV_IN_POLL 4
- #define SYNC_MAC_NEIGHBOURS 5
- #define _DEF_SLOT_TIME 10 * 1000
- #define _DEF_SLOT_CNT 10
- #define _DEF_SLOT_SUM_TIME (_DEF_SLOT_CNT * _DEF_SLOT_TIME)
- #define MAC_SETTINGS_DEF

Enumerations

- enum [rtls_role](#) {
 RTLS_TAG = 'T', RTLS_ANCHOR = 'A', RTLS_SINK = 'S', RTLS_LISTENER = 'L',
 RTLS_DEFAULT = 'D' }

6.29.1 Macro Definition Documentation

6.29.1.1 _DEF_SLOT_CNT

```
#define _DEF_SLOT_CNT 10
```

6.29.1.2 _DEF_SLOT_SUM_TIME

```
#define _DEF_SLOT_SUM_TIME (_DEF_SLOT_CNT * _DEF_SLOT_TIME)
```

6.29.1.3 _DEF_SLOT_TIME

```
#define _DEF_SLOT_TIME 10 * 1000
```

6.29.1.4 MAC_BUF_CNT

```
#define MAC_BUF_CNT 5
```

6.29.1.5 MAC_BUF_LEN

```
#define MAC_BUF_LEN 128
```

6.29.1.6 MAC_SETTINGS_DEF

```
#define MAC_SETTINGS_DEF
```

Value:

```
{
    \
    .addr = ADDR_BROADCAST, .pan = 0xDECA, .slot_time_us =
        _DEF_SLOT_TIME,
    .slot_guard_time_us = 200, .slots_sum_time_us = _DEF_SLOT_SUM_TIME, .
        \
        max_frame_fail_cnt = 3,
    .max_buf_inactive_time = 2 * _DEF_SLOT_SUM_TIME, .role =
        RTLS_DEFAULT, .raport_anchor_anchor_distance = false, \
}
```

6.29.1.7 SYNC_MAC_NEIGHBOURS

```
#define SYNC_MAC_NEIGHBOURS 5
```

6.29.1.8 TOA_MAX_DEV_IN_POLL

```
#define TOA_MAX_DEV_IN_POLL 4
```

6.29.2 Enumeration Type Documentation

6.29.2.1 rtls_role

```
enum rtls_role
```

Enumerator

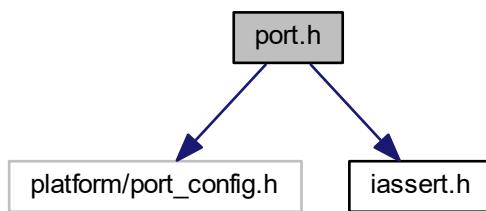
RTLS_TAG	
RTLS_ANCHOR	
RTLS_SINK	
RTLS_LISTENER	
RTLS_DEFAULT	

6.30 port.h File Reference

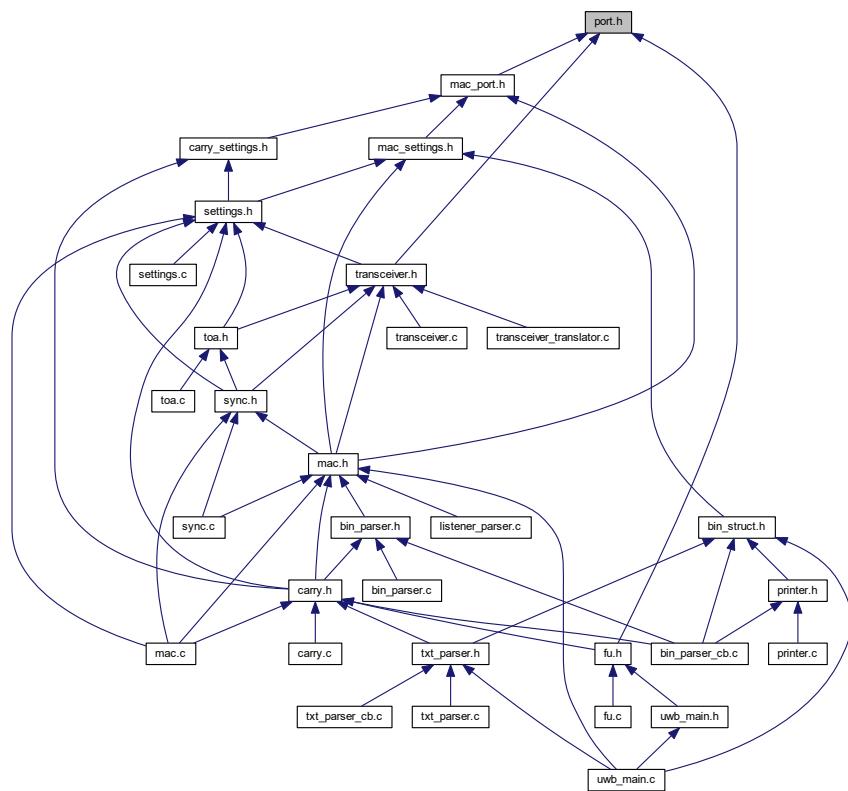
Portable module, strictly hardware dependent.

```
#include "platform/port_config.h"  
#include "iassert.h"
```

Include dependency graph for port.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define PORT_ASSERT(expr) IASSERT(expr)
- #define DBG 1

This is used to set or reset debug mode.

Functions

- void **PORT_Init ()**
Initialization for port modules.
- void **PORT_lassert_fun** (const char *msg, int line)
assert routine function.
- void **PORT_LedOn** (int LED_X)
Turn led on.
- void **PORT_LedOff** (int LED_X)
Turn led off.
- void **PORT_ResetTransceiver ()**
Hard reset DW1000 device by RST pin.
- void **PORT_WakeupTransceiver** (void)
Wake up DW1000 device.
- void **PORT_Reboot ()**
Reset host microcontroller.

- void **PORT_EnterStopMode** ()
turn on low power or stop mode.
- void **PORT_WatchdogInit** ()
Start watchdog work.
- void **PORT_WatchdogRefresh** ()
Refresh watchdog timer.
- void **PORT_BatteryMeasure** ()
Start battery measurement process.
- int **PORT_BatteryVoltage** ()
Return last battery voltage in [mV].
- void **PORT_TimeStartTimers** ()
run timers when device is fully initialized.
- void **PORT_SleepMs** (unsigned int time_ms)
Sleep and refresh watchdog.
- unsigned int **PORT_TickMs** ()
Get current milliseconds timer counter value.
- unsigned int **PORT_TickHr** ()
Get high resolution clock counter value.
- unsigned int **PORT_TickHrToUs** (unsigned int delta)
convert high resolution clock time units to us
- uint32_t **PORT_SlotTimerTickUs** ()
return current slot timer tick counter in milliseconds
- void **PORT_SlotTimerSetUsOffset** (int32 delta_us)
return current slot timer tick counter in milliseconds
- void **PORT_SetSlotTimerPeriodUs** (uint32 us)
set new slot timer period time
- void **PORT_CrcReset** ()
set initial value to the crc calculator
- uint16_t **PORT_CrcFeed** (const void *data, int size)
feed crc calculator with new data and return result
- **decalrqStatus_t decamutexon** (void)
enable deca mutex
- void **decamutexoff** (**decalrqStatus_t** s)
disable deca mutex
- void **PORT_SpiSpeedSlow** (bool slow)
change decawave SPI master clock speed
- int **readfromspi** (uint16_t headerLength, const uint8_t *headerBuffer, uint32_t readlength, uint8_t *readBuffer)
send header and read response from DW1000 device
- int **writetospi** (uint16_t headerLength, const uint8_t *headerBuffer, uint32_t bodylength, const uint8_t *bodyBuffer)
send header and write data to DW1000 device
- void **PORT_BkpRegisterWrite** (uint32_t *reg, uint32_t value)
Save value in reset-safe backup register.
- uint32_t **PORT_BkpRegisterRead** (uint32_t *reg)
Read value in reset-safe backup register.
- int **PORT_FlashErase** (void *flash_addr, uint32_t length)
Erase memory in flash.
- int **PORT_FlashSave** (void *destination, const void *p_source, uint32_t length)
Save data in flash.

6.30.1 Detailed Description

Portable module, strictly hardware dependent.

Author

Karol Trzcimski

Date

06-2018

Each function from this module has to be implemented in current hardware project workspace. To fully port project you need to implement each function from this header and connect interrupts routines, especially:

- [dwt_isr](#)
- [MAC_YourSlotIsr](#)

6.30.2 Macro Definition Documentation

6.30.2.1 DBG

```
#define DBG 1
```

This is used to set or reset debug mode.

Especially difference is during assertion. In debug mode assert lead to IC hang and in release mode to reset.

6.30.2.2 PORT_ASSERT

```
#define PORT_ASSERT( expr ) IASSERT(expr)
```

6.30.3 Function Documentation

6.30.3.1 decamutexoff()

```
void decamutexoff (
    decaIrqStatus_t s )
```

disable deca mutex

Enable [dwt_isr](#) interrupt in host processor

Parameters

in	s	is return value from decamutexon() function
----	---	---

Returns

void

6.30.3.2 decamutexon()

```
decaIrqStatus_t decamutexon (
    void )
```

enable deca mutex

Disable [dwt_isr](#) interrupt in host processor

Returns

0 if previous status of [dwt_isr](#) was disabled, 1 otherwise

6.30.3.3 PORT_BatteryMeasure()

```
void PORT_BatteryMeasure ( )
```

Start battery measurement process.

Returns

void

Here is the caller graph for this function:



6.30.3.4 PORT_BatteryVoltage()

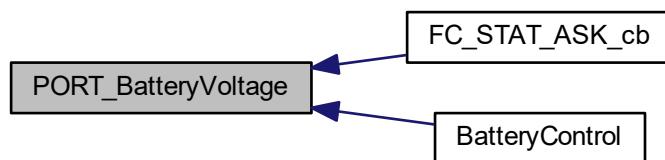
```
int PORT_BatteryVoltage ( )
```

Return last battery voltage in [mV].

Returns

last battery voltage in [mV]

Here is the caller graph for this function:



6.30.3.5 PORT_BkpRegisterRead()

```
uint32_t PORT_BkpRegisterRead (   
    uint32_t * reg )
```

Read value in reset-safe backup register.

Note

it doesn't need to save after power down

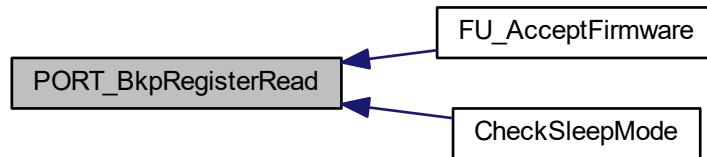
Parameters

in	reg	pointer to memory address
----	-----	---------------------------

Returns

current value

Here is the caller graph for this function:



6.30.3.6 PORT_BkpRegisterWrite()

```
void PORT_BkpRegisterWrite (
    uint32_t * reg,
    uint32_t value )
```

Save value in reset-safe backup register.

Note

it doesn't need to save after power down

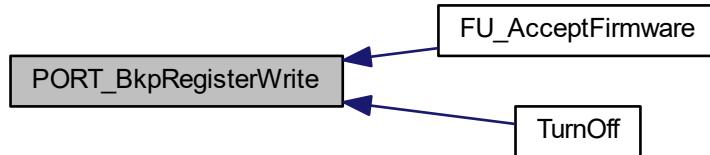
Parameters

in	<i>reg</i>	pointer to memory address
in	<i>value</i>	to write into register

Returns

void

Here is the caller graph for this function:



6.30.3.7 PORT_CrcFeed()

```
uint16_t PORT_CrcFeed (
    const void * data,
    int size )
```

feed crc calculator with new data and return result

Note

return value should be automatically set as initial value during next iteration.

6.30.3.8 PORT_CrcReset()

```
void PORT_CrcReset ( )
```

set initial value to the crc calculator

Note

initial value for IndoorNavi is 0xFFFF

Returns

void

6.30.3.9 PORT_EnterStopMode()

```
void PORT_EnterStopMode( )
```

turn on low power or stop mode.

Used especially after successful firmware upgrade to change working firmware.

Returns

```
void
```

Here is the caller graph for this function:



6.30.3.10 PORT_FlashErase()

```
int PORT_FlashErase (
    void * flash_addr,
    uint32_t length )
```

Erase memory in flash.

It is used before save new settings and during firmware upgrade

Note

watch about watchdog timeout

Parameters

in	<i>flash_addr</i>	start address of region to erase
in	<i>length</i>	minimal length of memory region to erase

Returns

0 if success, error code otherwise

6.30.3.11 PORT_FlashSave()

```
int PORT_FlashSave (
    void * destination,
    const void * p_source,
    uint32_t length )
```

Save data in flash.

It is used to save new settings and during firmware upgrade

Note

watch about watchdog timeout

Parameters

in	<i>destination</i>	start address of region to save in flash region
in	<i>p_source</i>	is pointer to input data buffer
in	<i>length</i>	minimal length of memory region to erase

Returns

0 if success, error code otherwise

6.30.3.12 PORT_Iassert_fun()

```
void PORT_Iassert_fun (
    const char * msg,
    int line )
```

assert routine function.

It is recommended to make implementation of this function depended from DBG flag. By default it should reset device in release mode (to improve system reliability) and hang device in debug mode (to highlight the issue). Moreover in both mode it is recommended to log assert function name and line.

Parameters

in	<i>msg</i>	is pointer to text message with assert function name
in	<i>line</i>	of assert in code

Returns

void

6.30.3.13 PORT_Init()

```
void PORT_Init ( )
```

Initialization for port modules.

Especially difference is during assertion. In debug mode assert lead to IC hang and in release mode to reset.

Returns

```
void
```

Here is the caller graph for this function:



6.30.3.14 PORT_LedOff()

```
void PORT_LedOff (   
    int LED_x )
```

Turn led off.

From UWB_Core there is used LED_STAT and LED_ERR macros (defined in port_config.h file) to indicate current program state.

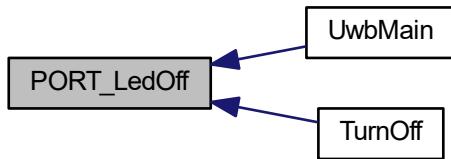
Parameters

in	LED_x	is hardware dependent numerical description of led.
----	---------	---

Returns

```
void
```

Here is the caller graph for this function:

**6.30.3.15 PORT_LedOn()**

```
void PORT_LedOn (
    int LED_x )
```

Turn led on.

From UWB_Core there is used LED_STAT and LED_ERR macros (defined in port_config.h file) to indicate current program state.

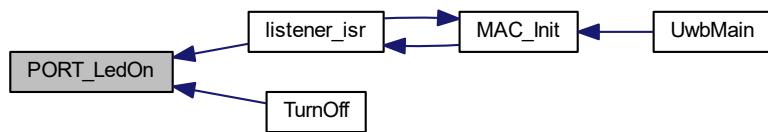
Parameters

in	LED_{\leftarrow} $_x$	is hardware dependent numerical description of led.
----	----------------------------	---

Returns

```
void
```

Here is the caller graph for this function:



6.30.3.16 PORT_Reboot()

```
void PORT_Reboot ( )
```

Reset host microcontroller.

Used especially after successful firmware upgrade to change working firmware. Also assert function can use it.

Returns

```
void
```

Here is the caller graph for this function:



6.30.3.17 PORT_ResetTransceiver()

```
void PORT_ResetTransceiver ( )
```

Hard reset DW1000 device by RST pin.

Note

Keep RST pin down for at least 500 microseconds.

Returns

```
void
```

6.30.3.18 PORT_SetSlotTimerPeriodUs()

```
void PORT_SetSlotTimerPeriodUs (
    uint32 us )
```

set new slot timer period time

Note

watch out when new period is shorter than previous one.

Parameters

in	<i>us</i>	new period duration in microseconds
----	-----------	-------------------------------------

6.30.3.19 PORT_SleepMs()

```
void PORT_SleepMs (
    unsigned int time_ms )
```

Sleep and refresh watchdog.

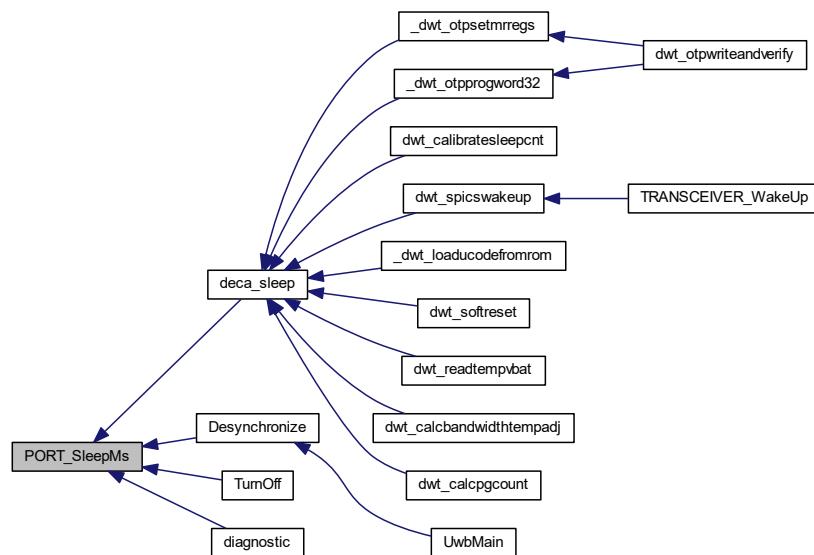
Parameters

in	<i>time_ms</i>	time to sleep in milliseconds
----	----------------	-------------------------------

Returns

void

Here is the caller graph for this function:



6.30.3.20 PORT_SlotTimerSetUsOffset()

```
void PORT_SlotTimerSetUsOffset (
    int32 delta_us )
```

return current slot timer tick counter in milliseconds

It is used during slot timer calibration. In systems when it is impossible to read slot timer counter value it is possible to return always zero in function [PORT_SlotTimerTickUs](#) and in function and treat input parameters as an absolute value, but in this form there will be degradation of precision especially during heavy load.

local slot timer value counter should be updated in form: local_cnt += delta_us In this function it should be checked if this offset is possible to realize and if it doesn't impact on system reliability. Watch out about timer overflow and underflow.

Parameters

in	<i>delta_us</i>	time difference between global and local slot timer value in microseconds
----	-----------------	---

Returns

void

Here is the caller graph for this function:



6.30.3.21 PORT_SlotTimerTickUs()

```
uint32_t PORT_SlotTimerTickUs ( )
```

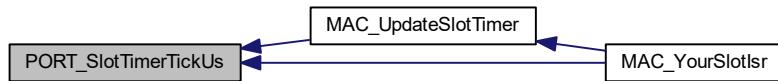
return current slot timer tick counter in milliseconds

It is used during slot timer calibration to save timestamp. In systems when it is impossible to read slot timer counter value it is possible to return always zero and in function [PORT_SlotTimerSetUsOffset](#) treat input parameters as an absolute value, but in this form there will be degradation of precision especially during heavy load.

Returns

current slot timer tick counter in milliseconds

Here is the caller graph for this function:

**6.30.3.22 PORT_SpiSpeedSlow()**

```
void PORT_SpiSpeedSlow (
    bool slow )
```

change decawave SPI master clock speed

Set SPI speed below 20MHz if `slow` is false or below 3 MHz if `slow` is true

Parameters

in	<code>slow</code>	boolean value
----	-------------------	---------------

Here is the caller graph for this function:

**6.30.3.23 PORT_TickHr()**

```
unsigned int PORT_TickHr ( )
```

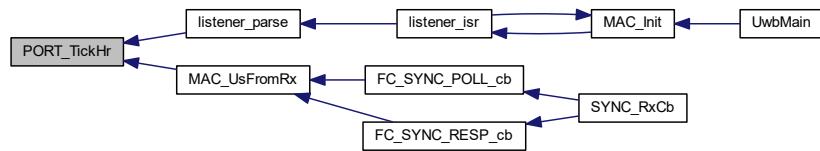
Get high resolution clock counter value.

In Cortex DWT_CYCCNT counter can be used. This function is used only for generate user messages.

Returns

high resolution clock counter value

Here is the caller graph for this function:

**6.30.3.24 PORT_TickHrToUs()**

```
unsigned int PORT_TickHrToUs (
    unsigned int delta )
```

convert high resolution clock time units to us

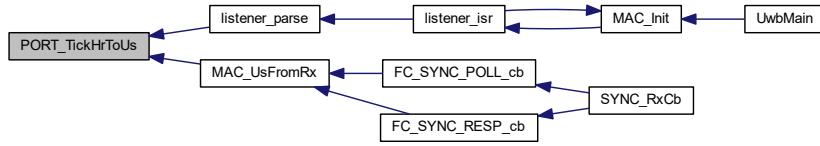
Parameters

in	<i>delta</i>	time difference in high resolution clock units
----	--------------	--

Returns

time difference in microseconds

Here is the caller graph for this function:

**6.30.3.25 PORT_TickMs()**

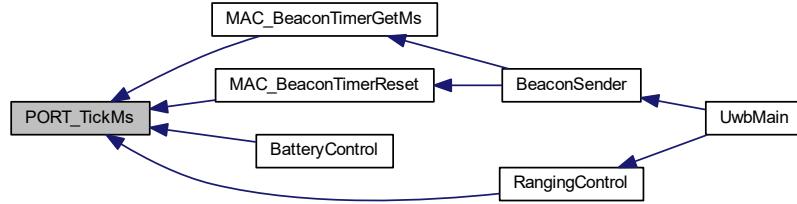
```
unsigned int PORT_TickMs ( )
```

Get current milliseconds timer counter value.

Returns

current time in milliseconds

Here is the caller graph for this function:

**6.30.3.26 PORT_TimeStartTimers()**

```
void PORT_TimeStartTimers ( )
```

run timers when device is fully initialized.

Especially slot and sleep timer

Returns

void

Here is the caller graph for this function:

**6.30.3.27 PORT_WakeupTransceiver()**

```
void PORT_WakeupTransceiver (
    void )
```

Wake up DW1000 device.

It can be done by setting DW1000 CS pin low for at least 500 microseconds.

Returns

void

6.30.3.28 PORT_WatchdogInit()

```
void PORT_WatchdogInit ( )
```

Start watchdog work.

Watchdog is refreshed in each iteration of main loop. Also in PORT_Sleep and PORT_FlashErase and PORT_FlashSave this function should be used.

Returns

```
void
```

6.30.3.29 PORT_WatchdogRefresh()

```
void PORT_WatchdogRefresh ( )
```

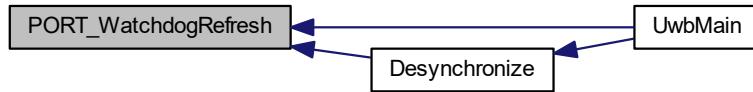
Refresh watchdog timer.

Watchdog is refreshed in each iteration of main loop. Also in PORT_Sleep and PORT_FlashErase and PORT_FlashSave this function should be used.

Returns

```
void
```

Here is the caller graph for this function:



6.30.3.30 readfromspi()

```
int readfromspi (
    uint16_t headerLength,
    const uint8_t * headerBuffer,
    uint32_t readlength,
    uint8_t * readBuffer )
```

send header and read response from DW1000 device

Note

this function is used very frequently and should be optimized for time
 SPI should work in half-duplex mode, so receiving is realized after transmitting header

Parameters

in	<i>headerLength</i>	to transmit in bytes
in	<i>headerBuffer</i>	pointer to header data
in	<i>readlength</i>	length of response in bytes
out	<i>readBuffer</i>	pointer to response buffer

Returns

0 if success error code otherwise

6.30.3.31 writetospi()

```
int writetospi (
    uint16_t headerLength,
    const uint8_t * headerBuffer,
    uint32_t bodylength,
    const uint8_t * bodyBuffer )
```

send header and write data to DW1000 device

Note

this function is used very frequently and should be optimized for time

Parameters

in	<i>headerLength</i>	to transmit in bytes
in	<i>headerBuffer</i>	pointer to header data
in	<i>bodylength</i>	length of data in bytes
in	<i>bodyBuffer</i>	pointer to data buffer

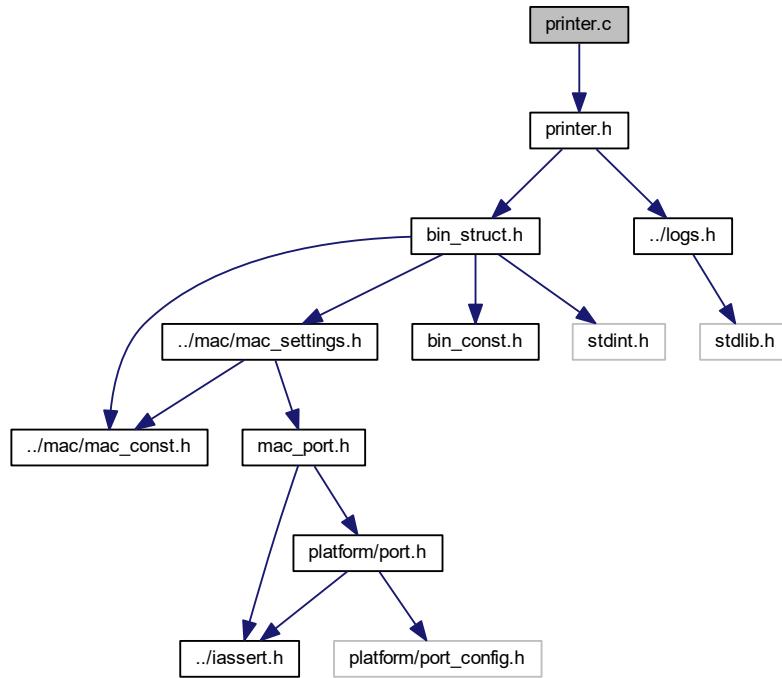
Returns

0 if success error code otherwise

6.31 printer.c File Reference

```
#include "printer.h"
```

Include dependency graph for printer.c:



Functions

- void **PRINT_Version** (const **FC_VERSION_s** *data, **dev_addr_t** did)
print version message
- void **PRINT_Stat** (const **FC_STAT_s** *data, **dev_addr_t** did)
print status message

6.31.1 Function Documentation

6.31.1.1 PRINT_Stat()

```
void PRINT_Stat (
    const FC_STAT_s * data,
    dev_addr_t did )
```

print status message

Parameters

data	pointer to structure with data to print
did	identifier of device

Here is the caller graph for this function:



6.31.1.2 PRINT_Version()

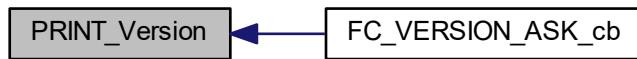
```
void PRINT_Version (
    const FC_VERSION_s * data,
    dev_addr_t did )
```

print version message

Parameters

<i>data</i>	pointer to structure with data to print
<i>did</i>	identifier of device

Here is the caller graph for this function:

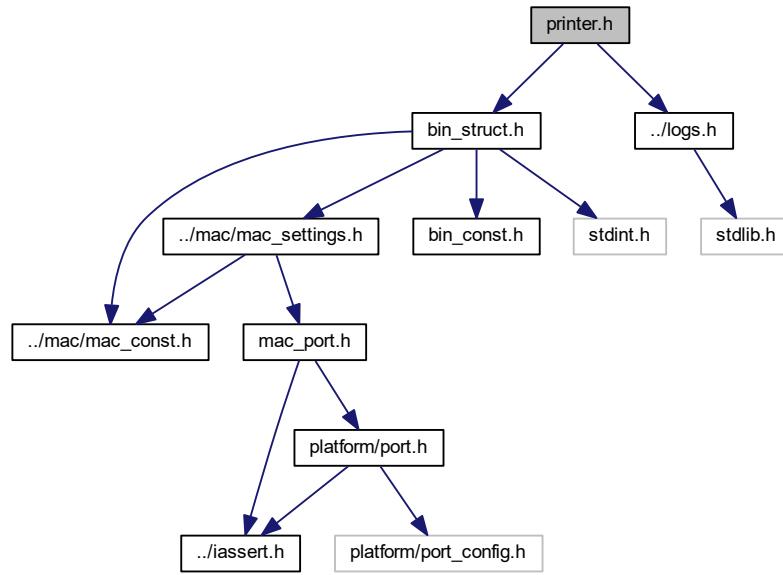


6.32 printer.h File Reference

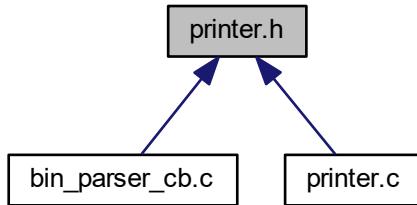
Common messages printers.

```
#include "bin_struct.h"
#include "../logs.h"
```

Include dependency graph for printer.h:



This graph shows which files directly or indirectly include this file:



Functions

- void `PRINT_Version` (const `FC_VERSION_s` *data, `dev_addr_t` did)
print version message
- void `PRINT_Stat` (const `FC_STAT_s` *data, `dev_addr_t` did)
print status message

6.32.1 Detailed Description

Common messages printers.

Printers has been developed to help keep printed communicates consistent. Moreover printing some struct data from any part of code will look this same.

Author

Karol Trzcinski

Date

2018-06-28

6.32.2 Function Documentation

6.32.2.1 PRINT_Status()

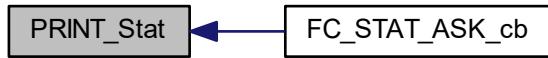
```
void PRINT_Status (
    const FC_STAT_s * data,
    dev_addr_t did )
```

print status message

Parameters

<i>data</i>	pointer to structure with data to print
<i>did</i>	identifier of device

Here is the caller graph for this function:



6.32.2.2 PRINT_Version()

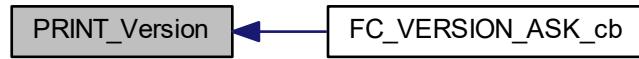
```
void PRINT_Version (
    const FC_VERSION_s * data,
    dev_addr_t did )
```

print version message

Parameters

<i>data</i>	pointer to structure with data to print
<i>did</i>	identifier of device

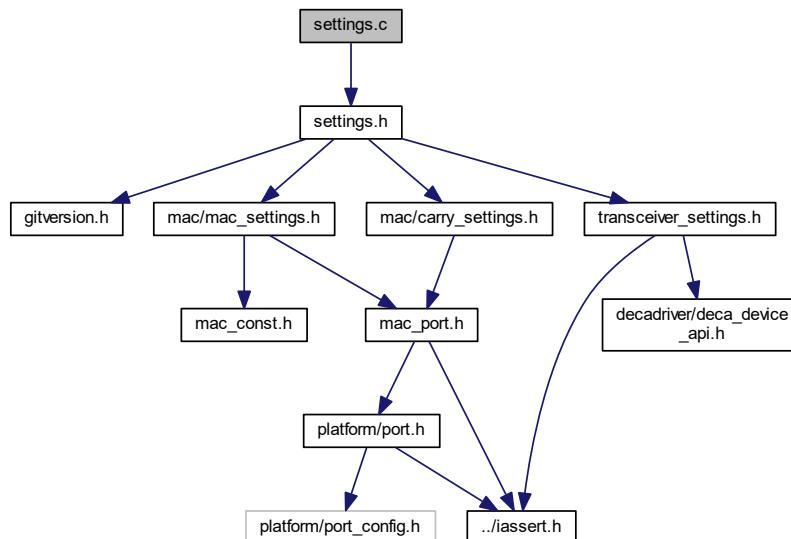
Here is the caller graph for this function:



6.33 README.md File Reference

6.34 settings.c File Reference

```
#include "settings.h"
Include dependency graph for settings.c:
```



Functions

- bool `settings_is_otp_erased ()`
- void `SETTINGS_INIT ()`

Initialize settings with default values.

Variables

- `settings_otp_t _settings_otp`
- `settings_t settings`
 - global instance of settings object*
- const `settings_otp_t * settings_otp`
 - global instance of settings_otp_t object*

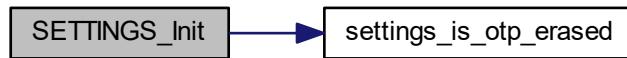
6.34.1 Function Documentation

6.34.1.1 SETTINGS_Init()

```
void SETTINGS_Init ( )
```

Initialize settings with default values.

OTP settings can be initialized conditionally to current value of OTP address. Address from OTP will be used only if it is not filled with 0xFF. Here is the call graph for this function:



Here is the caller graph for this function:



6.34.1.2 settings_is_otp_erased()

```
bool settings_is_otp_erased ( )
```

Here is the caller graph for this function:



6.34.2 Variable Documentation

6.34.2.1 _settings_otp

```
settings_otp_t _settings_otp
```

Initial value:

```
= {  
    .h_major = __H_MAJOR__,  
    .h_minor = __H_MINOR__,  
    .serial = 0,  
}
```

6.34.2.2 settings

```
settings_t settings
```

global instance of settings object

In [SETTINGS_Init\(\)](#) or during startup values should be copied from non-volatile memory to this instance.

6.34.2.3 settings_otp

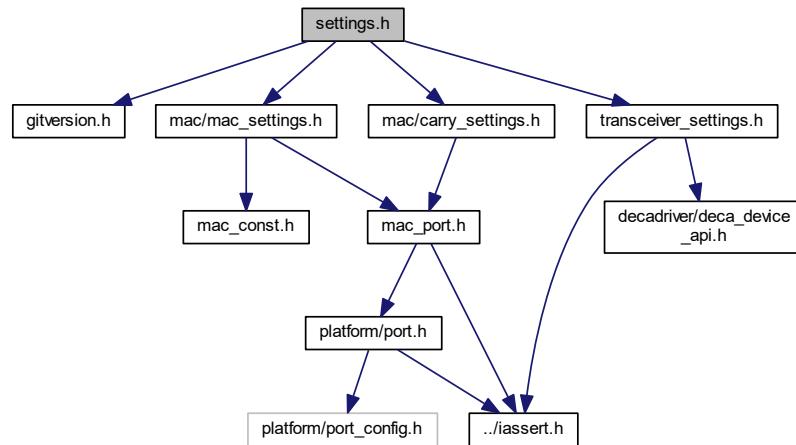
```
const settings_otp_t* settings_otp
```

global instance of [settings_otp_t](#) object

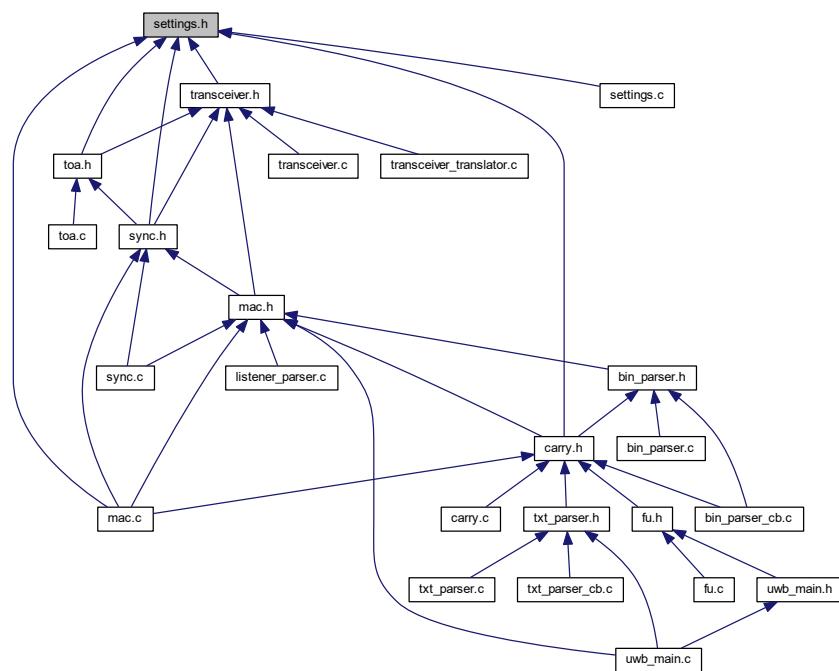
Pointed struct may depends on OTP memory status. When OTP is originally erased then pointed should be default otp struct, generated basing on device identifier.

6.35 settings.h File Reference

```
#include "gitversion.h"
#include "mac/mac_settings.h"
#include "mac/carry_settings.h"
#include "transceiver_settings.h"
Include dependency graph for settings.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `settings_otp_t`
Internal One Time Programmable settings structure.
- struct `settings_version_t`
current firmware and hardware description structure
- struct `settings_t`
main setting container.

Macros

- `#define H_VERSION_CALC(major, minor) ((major << 3) | (minor & 0x07))`
generate hardware version description byte from major and minor
- `#define H_MAJOR_CALC(hversion) ((hversion&0xFF)>>3)`
calculate hardware major version from description byte
- `#define __H_VERSION__ H_VERSION_CALC(__H_MAJOR__, __H_MINOR__)`
generate current version number
- `#define VERSION_SETTINGS_DEF`
default firmware and hardware version settings
- `#define DEF_SETTINGS`
define default settings values

Functions

- void `SETTINGS_Init ()`
Initialize settings with default values.

Variables

- `settings_t settings`
global instance of settings object
- `settings_otp_t const * settings_otp`
global instance of `settings_otp_t` object

6.35.1 Macro Definition Documentation

6.35.1.1 `__H_VERSION__`

```
#define __H_VERSION__ H_VERSION_CALC(__H_MAJOR__, __H_MINOR__)
```

generate current version number

6.35.1.2 DEF_SETTINGS

```
#define DEF_SETTINGS
```

Value:

```
{
    .version = VERSION_SETTINGS_DEF, .transceiver =
        TRANSCIEVER_SETTINGS_DEF, \
    .mac = MAC_SETTINGS_DEF, .carry = CARRY_SETTINGS_DEF,
}
```

define default settings values

6.35.1.3 H_MAJOR_CALC

```
#define H_MAJOR_CALC(
    hversion ) ((hversion&0xFF)>>3)
```

calculate hardware major version from description byte

It is possible to define up to 32 major firmware version and 8 minor.

Note

Minor versions should be pin to pin compatible and run with same firmware.

6.35.1.4 H_VERSION_CALC

```
#define H_VERSION_CALC(
    major,
    minor ) ((major << 3) | (minor & 0x07))
```

generate hardware version description byte from major and minor

It is possible to define up to 32 major firmware version and 8 minor.

Note

Minor versions should be pin to pin compatible and run with same firmware.

6.35.1.5 VERSION_SETTINGS_DEF

```
#define VERSION_SETTINGS_DEF
```

Value:

```
{\n    .boot_reserved = 0,\n    .h_version = __H_VERSION__,\n    .f_major = __F_MAJOR__,\n    .f_minor = __F_MINOR__,\n    .f_hash = __F_HASH__,\n}
```

default firmware and hardware version settings

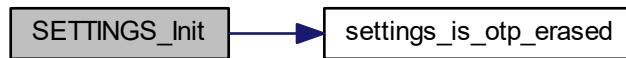
6.35.2 Function Documentation

6.35.2.1 SETTINGS_Init()

```
void SETTINGS_Init ( )
```

Initialize settings with default values.

OTP settings can be initialized conditionally to current value of OTP address. Address from OTP will be used only if it is not filled with 0xFF. Here is the call graph for this function:



Here is the caller graph for this function:



6.35.3 Variable Documentation

6.35.3.1 settings

`settings_t settings`

global instance of `settings` object

In `SETTINGS_Init()` or during startup values should be copied from non-volatile memory to this instance.

6.35.3.2 settings_otp

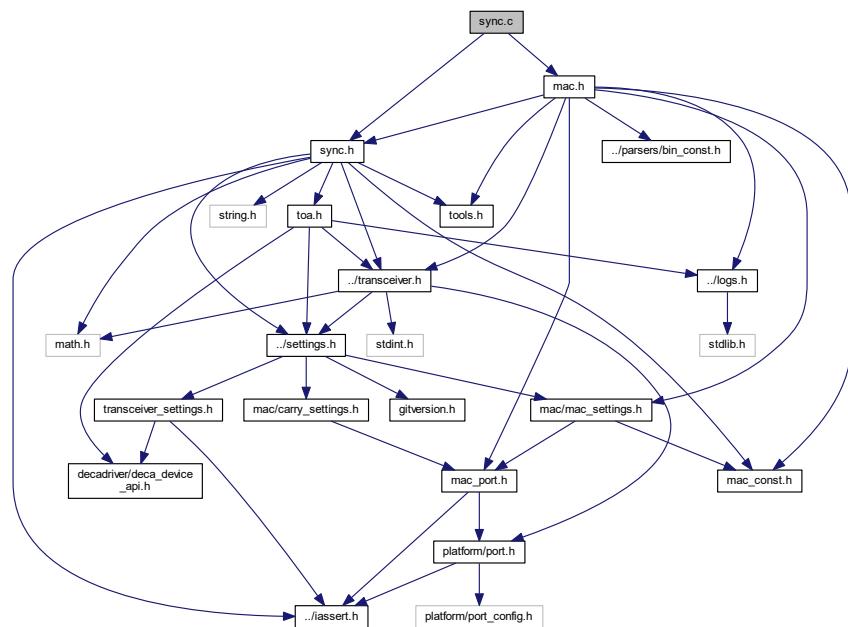
`settings_otp_t const* settings_otp`

global instance of `settings_otp_t` object

Pointed struct may depends on OTP memory status. When OTP is originally erased then pointed should be default otp struct, generated basing on device identifier.

6.36 sync.c File Reference

```
#include "sync.h"
#include "mac.h"
Include dependency graph for sync.c:
```



Macros

- `#define PROT_CHECK_LEN(FC, len, expected)`
- `#define MOVE_ARRAY_ELEM(ARR_EL, N) ARR_EL[N] = ARR_EL[N - 1]`

Functions

- `void SYNC_Init ()`
Initialize Sync module.
- `int64_t SYNC_GlobTimeNeig (sync_neighbour_t *neig, int64_t dw_ts)`
- `int64_t SYNC_GlobTime (int64_t dw_ts)`
return global time
- `void SYNC_InitNeighbour (sync_neighbour_t *neig, dev_addr_t addr, int tree_level)`
- `sync_neighbour_t * SYNC_FindOrCreateNeighbour (dev_addr_t addr, int tree_level)`
Search for neighbour with a given addr.
- `int64_t SYNC_TrimDrift (int64_t drift)`
- `float SYNC_Smooth (float x, float range)`
- `float SYNC_CalcTimeCoeff (sync_neighbour_t *neig)`
- `void SYNC_UpdateNeighbour (sync_neighbour_t *neig, int64_t ext_time, int64_t loc_time, int tof_dw)`
- `void SYNC_UpdateLocalTimeParams (uint64_t transceiver_raw_time)`
- `void SYNC_Update (sync_neighbour_t *neig, int64_t ext_time, int64_t loc_time, int tof_dw)`
- `int SYNC_SendPoll (dev_addr_t dst, dev_addr_t anchors[], int anc_cnt)`
Send Poll message to start SYNC process with a given anchor.
- `int SYNC_SendResp (int64_t PollDwRxTs)`
- `int SYNC_SendFinal ()`
- `int FC_SYNC_POLL_cb (const void *data, const prot_packet_info_t *info)`
- `int FC_SYNC_RESP_cb (const void *data, const prot_packet_info_t *info)`
- `int FC_SYNC_FIN_cb (const void *data, const prot_packet_info_t *info)`
- `int SYNC_RxCb (const void *data, const prot_packet_info_t *info)`
sync rx callback
- `int SYNC_TxCb (int64_t TsDwTx)`
sync tx callback
- `int SYNC_RxToCb ()`
sync rx timeout routine

Variables

- `sync_instance_t sync`
- `mac_instance_t mac`
- `const char bad_len_msg [] = "%s bad len %d!=%d"`

6.36.1 Macro Definition Documentation

6.36.1.1 MOVE_ARRAY_ELEM

```
#define MOVE_ARRAY_ELEM(
    ARR_EL,
    N ) ARR_EL[N] = ARR_EL[N - 1]
```

6.36.1.2 PROT_CHECK_LEN

```
#define PROT_CHECK_LEN(
    FC,
    len,
    expected )
```

Value:

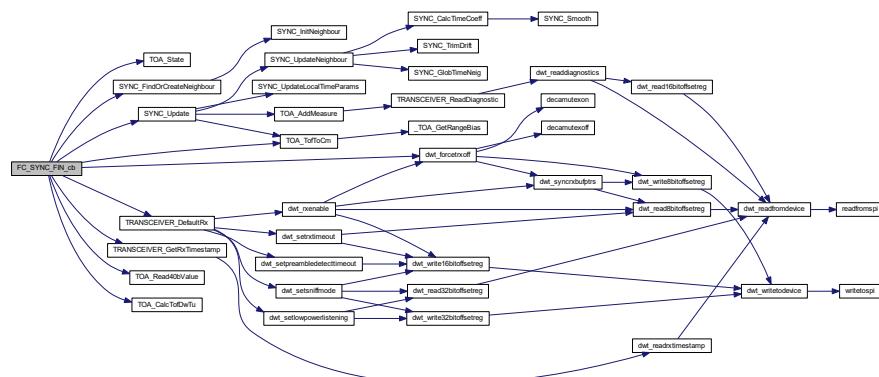
```
\do \
{ \
    if ((len) < (expected)) { \
        LOG_ERR(bad_len_msg, #FC, (len), (expected)); \
        return -1; \
    } \
} \
while(0)
```

6.36.2 Function Documentation

6.36.2.1 FC_SYNC_FIN_cb()

```
int FC_SYNC_FIN_cb (
    const void * data,
    const prot_packet_info_t * info )
```

Here is the call graph for this function:



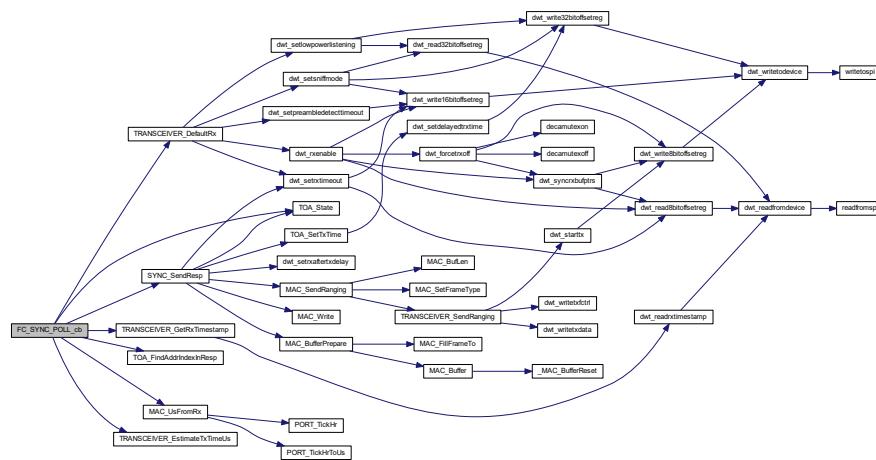
Here is the caller graph for this function:



6.36.2.2 FC_SYNC_POLL_cb()

```
int FC_SYNC_POLL_cb (
    const void * data,
    const prot_packet_info_t * info )
```

Here is the call graph for this function:



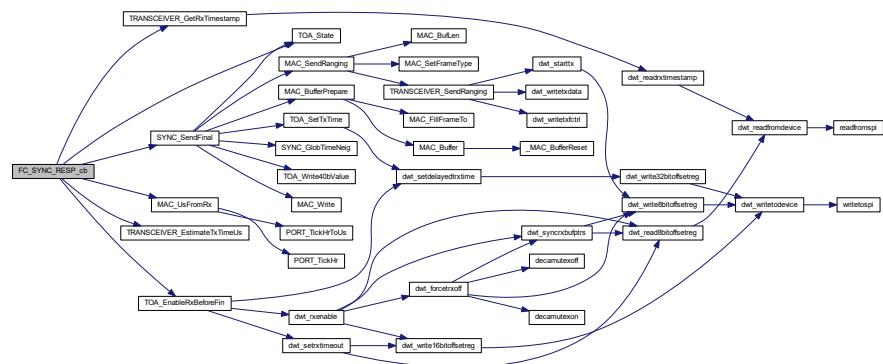
Here is the caller graph for this function:



6.36.2.3 FC_SYNC_RESP_cb()

```
int FC_SYNC_RESP_cb (
    const void * data,
    const prot_packet_info_t * info )
```

Here is the call graph for this function:



Here is the caller graph for this function:



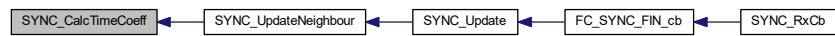
6.36.2.4 SYNC_CalcTimeCoeff()

```
float SYNC_CalcTimeCoeff ( sync_neighbour_t * neig )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.36.2.5 SYNC_FindOrCreateNeighbour()

```
sync_neighbour_t* SYNC_FindOrCreateNeighbour (
    dev_addr_t addr,
    int tree_level )
```

Search for neighbour with a given addr.

When such device isn't in table then initialize new if there is enough memory for it.

Parameters

in	<i>addr</i>	neighbour address
in	<i>tree_level</i>	neighbour tree level

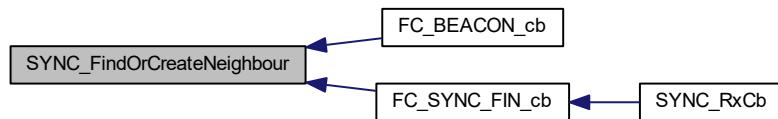
Returns

`sync_neighbour_t*` pointer to neighbour or zero

Here is the call graph for this function:



Here is the caller graph for this function:



6.36.2.6 SYNC_GlobTime()

```
int64_t SYNC_GlobTime (
    int64_t dw_ts )
```

return global time

Parameters

in	dw_{\leftarrow} $_ts$	local dwt time
----	----------------------------	----------------

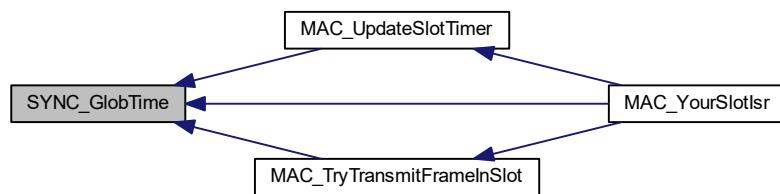
Returns

int64_t global dwt time

Here is the call graph for this function:



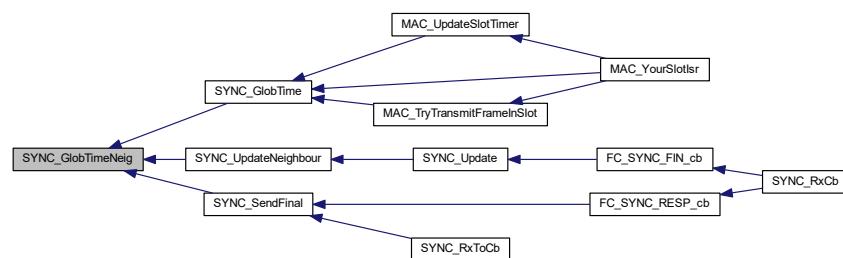
Here is the caller graph for this function:

**6.36.2.7 SYNC_GlobTimeNeig()**

```

int64_t SYNC_GlobTimeNeig (
    sync_neighbour_t *neig,
    int64_t dw_ts )
  
```

Here is the caller graph for this function:



6.36.2.8 SYNC_Init()

```
void SYNC_Init ( )
```

Initialize Sync module.

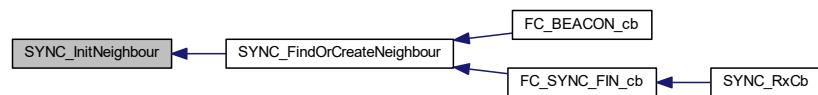
Here is the call graph for this function:



6.36.2.9 SYNC_InitNeighbour()

```
void SYNC_InitNeighbour (
    sync_neighbour_t * neig,
    dev_addr_t addr,
    int tree_level )
```

Here is the caller graph for this function:



6.36.2.10 SYNC_RxCb()

```
int SYNC_RxCb (
    const void * data,
    const prot_packet_info_t * info )
```

sync rx callback

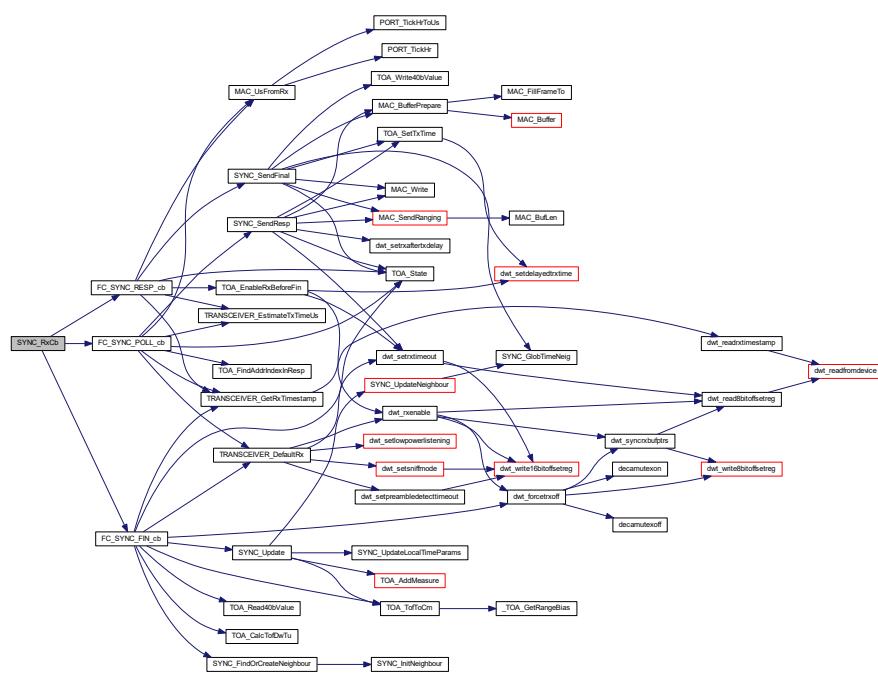
Parameters

in	<i>data</i>	pointer to data to parse
in	<i>info</i>	extra packet info

Returns

int 1 if packed was parser 0 otherwise

Here is the call graph for this function:



6.36.2.11 SYNC_RxToCb()

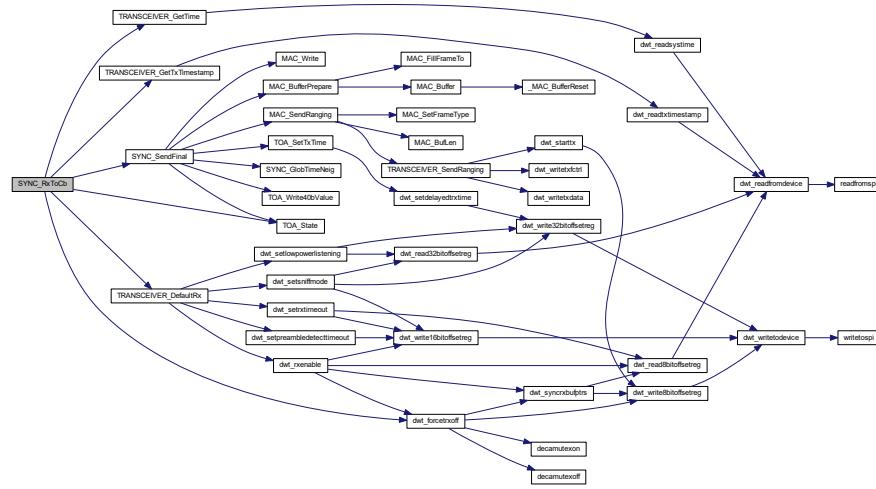
```
int SYNC_RxToCb ( )
```

sync rx timeout routine

Returns

int 1 if packed was parsed 0 otherwise

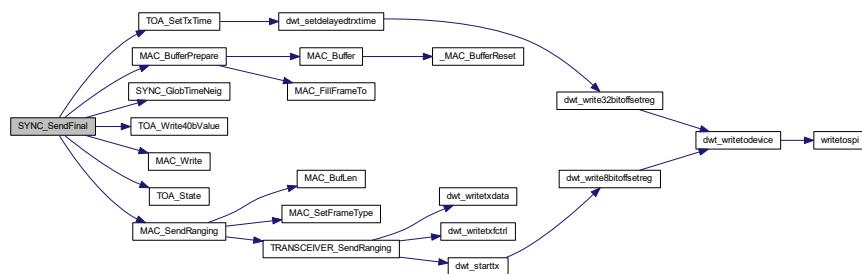
Here is the call graph for this function:



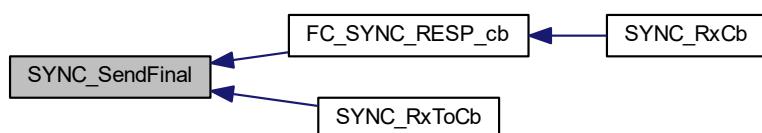
6.36.2.12 SYNC_SendFinal()

```
int SYNC_SendFinal ( )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.36.2.13 SYNC_SendPoll()

```
int SYNC_SendPoll (
    dev_addr_t dst,
    dev_addr_t anchors[],
    int anc_cnt )
```

Send Poll message to start SYNC process with a given anchor.

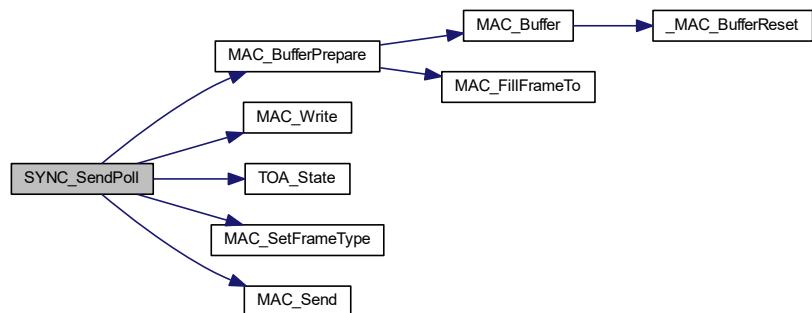
Parameters

in	<i>dst</i>	destination device address
in	<i>anchors</i>	list of anchor to measure ()
in	<i>anc_cnt</i>	counter of anchors to measure

Returns

int

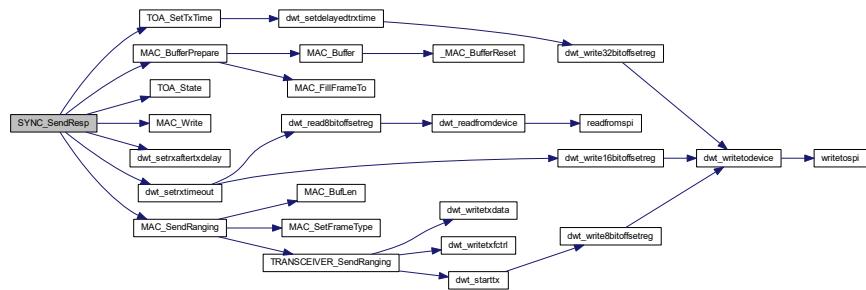
Here is the call graph for this function:



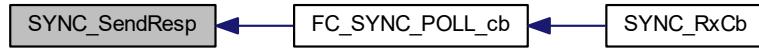
6.36.2.14 SYNC_SendResp()

```
int SYNC_SendResp (
    int64_t PollDwRxTs )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.36.2.15 SYNC_Smooth()

```
float SYNC_Smooth (
    float x,
    float range )
```

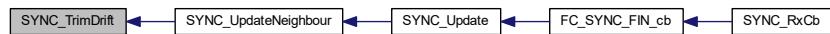
Here is the caller graph for this function:



6.36.2.16 SYNC_TrimDrift()

```
int64_t SYNC_TrimDrift (
    int64_t drift )
```

Here is the caller graph for this function:



6.36.2.17 SYNC_TxCb()

```
int SYNC_TxCb (
    int64_t TsDwTx )
```

sync tx callback

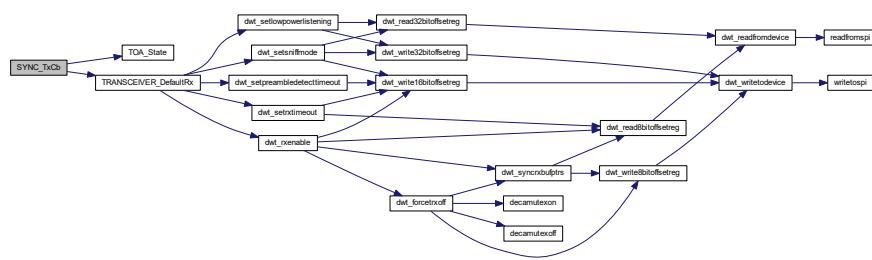
Parameters

in	TsDwTx	transmit timestamp in local dtu
----	--------	---------------------------------

Returns

int 1 if packed was parsed 0 otherwise

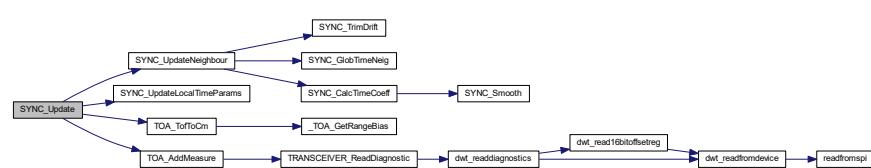
Here is the call graph for this function:



6.36.2.18 SYNC_Update()

```
void SYNC_Update (
    sync_neighbour_t * neig,
    int64_t ext_time,
    int64_t loc_time,
    int tof_dw )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.36.2.19 SYNC_UpdateLocalTimeParams()

```
void SYNC_UpdateLocalTimeParams (
    uint64_t transceiver_raw_time )
```

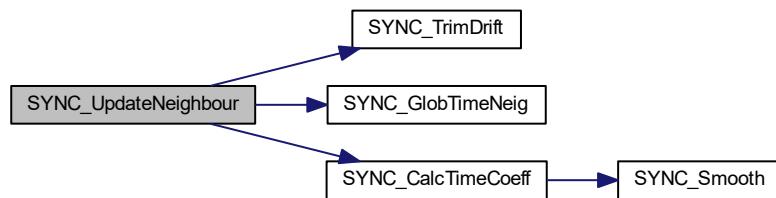
Here is the caller graph for this function:



6.36.2.20 SYNC_UpdateNeighbour()

```
void SYNC_UpdateNeighbour (
    sync_neighbour_t *neig,
    int64_t ext_time,
    int64_t loc_time,
    int tof_dw )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.36.3 Variable Documentation

6.36.3.1 bad_len_msg

```
const char bad_len_msg[] = "%s bad len %d!=%d"
```

6.36.3.2 mac

```
mac_instance_t mac
```

6.36.3.3 sync

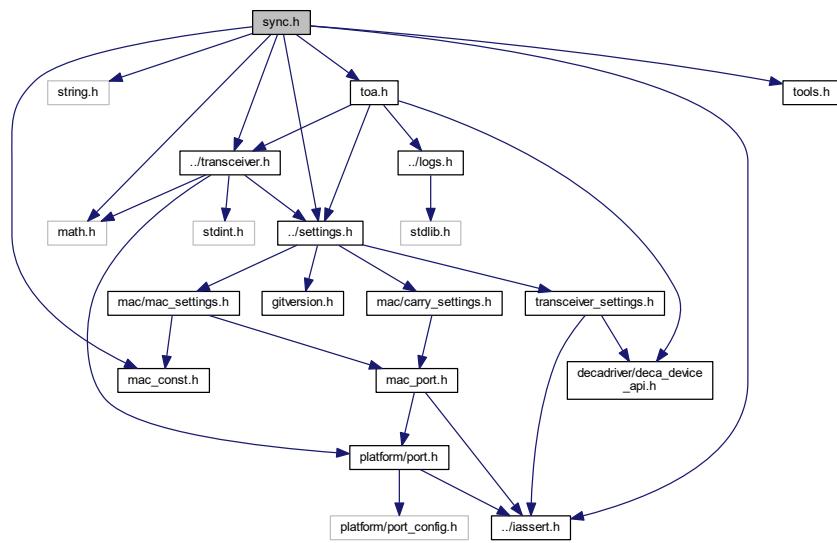
```
sync_instance_t sync
```

6.37 sync.h File Reference

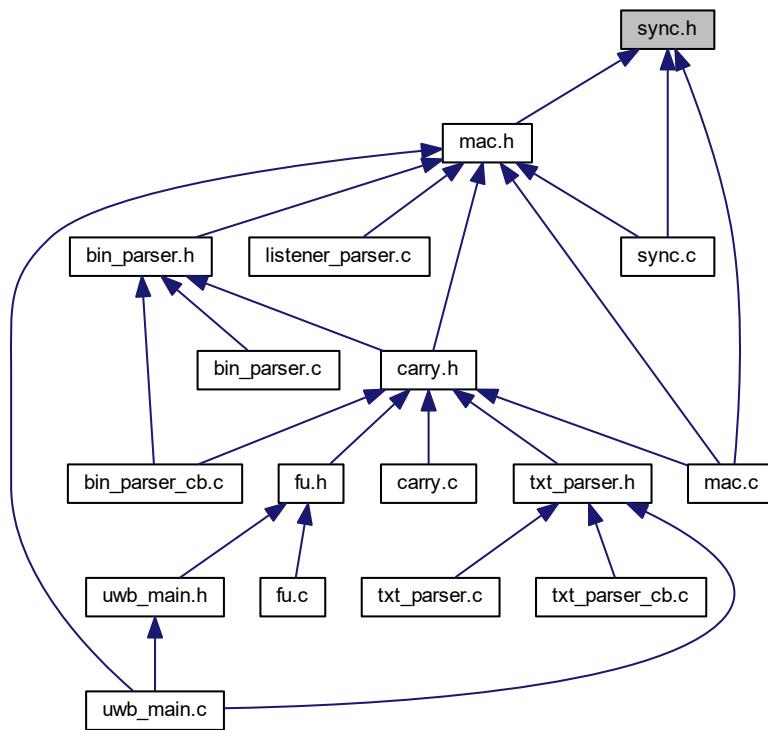
Time synchronization module.

```
#include <math.h>
#include <string.h>
#include "../iassert.h"
#include "../settings.h"
#include "../transceiver.h"
#include "mac_const.h"
#include "toa.h"
```

```
#include "tools.h"
Include dependency graph for sync.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `FC_SYNC_POLL_s`
- struct `FC_SYNC_RESP_s`
- struct `FC_SYNC_FIN_s`
- struct `sync_neighbour_t`
- struct `sync_instance_t`

Macros

- `#define SYNC_ASSERT(expr) IASSERT(expr)`
- `#define SYNC_TRACE_ENABLED 0`
set 1 to trace sync debug messages
- `#define SYNC_TRACE(...) ALL_UNUSED(__VA_ARGS__)`
- `#define SYNC_TIME_DUMP_ENABLED 0`
set 1 to trace sync time logging
- `#define SYNC_TIME_DUMP(...) ALL_UNUSED(__VA_ARGS__)`
- `#define SYNC_TRACE_TOA_ENABLED 0`
set 1 to trace sync time of arrival debug messages
- `#define SYNC_TRACE_TOA(...) ALL_UNUSED(__VA_ARGS__)`

Functions

- `void SYNC_Init ()`
Initialize Sync module.
- `int64_t SYNC_GlobTime (int64_t dw_ts)`
return global time
- `sync_neighbour_t * SYNC_FindOrCreateNeighbour (dev_addr_t addr, int tree_level)`
Search for neighbour with a given addr.
- `int SYNC_SendPoll (dev_addr_t dst, dev_addr_t anchors[], int anc_cnt)`
Send Poll message to start SYNC process with a given anchor.
- `int SYNC_RxCb (const void *data, const prot_packet_info_t *info)`
sync rx callback
- `int SYNC_RxToCb ()`
sync rx timeout routine
- `int SYNC_TxCb (int64_t TsDwTx)`
sync tx callback

6.37.1 Detailed Description

Time synchronization module.

Author

Karol Trzcinski

Date

2018-07-02

It is useful for time synchronization for collision avoidance and for tags localization. To achieve precise localization synchronization should be invoked frequently.

6.37.2 Macro Definition Documentation

6.37.2.1 SYNC_ASSERT

```
#define SYNC_ASSERT(  
    expr ) IASSERT(expr)
```

6.37.2.2 SYNC_TIME_DUMP

```
#define SYNC_TIME_DUMP(  
    ... ) ALL_UNUSED(__VA_ARGS__)
```

6.37.2.3 SYNC_TIME_DUMP_ENABLED

```
#define SYNC_TIME_DUMP_ENABLED 0
```

set 1 to trace sync time logging

6.37.2.4 SYNC_TRACE

```
#define SYNC_TRACE(  
    ... ) ALL_UNUSED(__VA_ARGS__)
```

6.37.2.5 SYNC_TRACE_ENABLED

```
#define SYNC_TRACE_ENABLED 0
```

set 1 to trace sync debug messages

6.37.2.6 SYNC_TRACE_TOA

```
#define SYNC_TRACE_TOA(  
    ... ) ALL_UNUSED(__VA_ARGS__)
```

6.37.2.7 SYNC_TRACE_TOA_ENABLED

```
#define SYNC_TRACE_TOA_ENABLED 0

set 1 to trace sync time of arrival debug messages
```

6.37.3 Function Documentation

6.37.3.1 SYNC_FindOrCreateNeighbour()

```
sync_neighbour_t* SYNC_FindOrCreateNeighbour (
    dev_addr_t addr,
    int tree_level )
```

Search for neighbour with a given addr.

When such device isn't in table then initialize new if there is enough memory for it.

Parameters

in	<i>addr</i>	neighbour address
in	<i>tree_level</i>	neighbour tree level

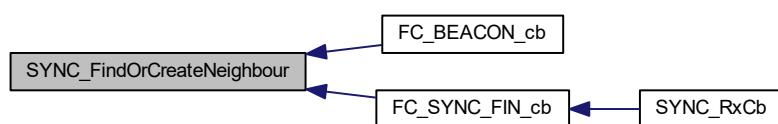
Returns

sync_neighbour_t* pointer to neighbour or zero

Here is the call graph for this function:



Here is the caller graph for this function:



6.37.3.2 SYNC_GlobTime()

```
int64_t SYNC_GlobTime (
    int64_t dw_ts )
```

return global time

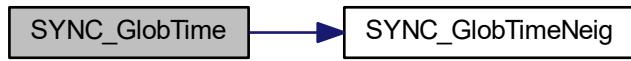
Parameters

in	<i>dw_ts</i>	local dwt time
----	--------------	----------------

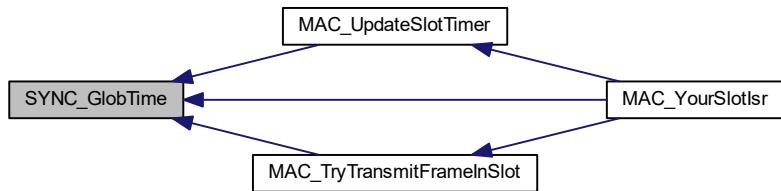
Returns

int64_t global dwt time

Here is the call graph for this function:



Here is the caller graph for this function:



6.37.3.3 SYNC_Init()

```
void SYNC_Init ( )
```

Initialize Sync module.

Here is the call graph for this function:



6.37.3.4 SYNC_RxCb()

```
int SYNC_RxCb (
    const void * data,
    const prot_packet_info_t * info )
```

sync rx callback

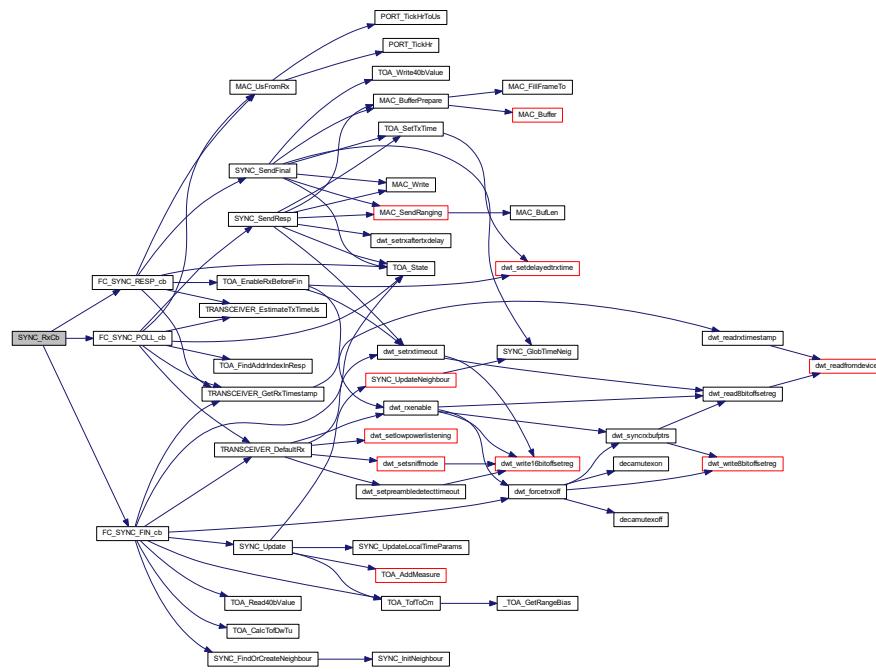
Parameters

in	<i>data</i>	pointer to data to parse
in	<i>info</i>	extra packet info

Returns

int 1 if packed was parser 0 otherwise

Here is the call graph for this function:



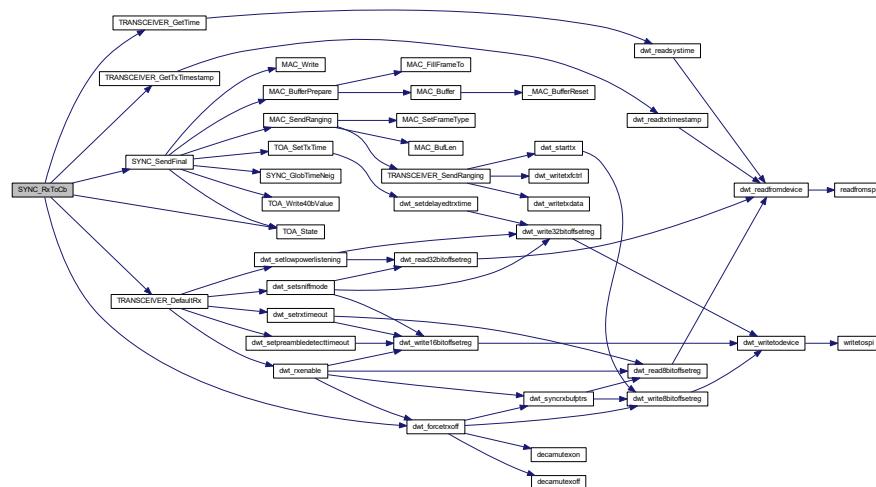
6.37.3.5 SYNC_RxToCb()

```
int SYNC_RxToCb ( )
sync rx timeout routine
```

Returns

int 1 if packed was parsed 0 otherwise

Here is the call graph for this function:



6.37.3.6 SYNC_SendPoll()

```
int SYNC_SendPoll (
    dev_addr_t dst,
    dev_addr_t anchors[],
    int anc_cnt )
```

Send Poll message to start SYNC process with a given anchor.

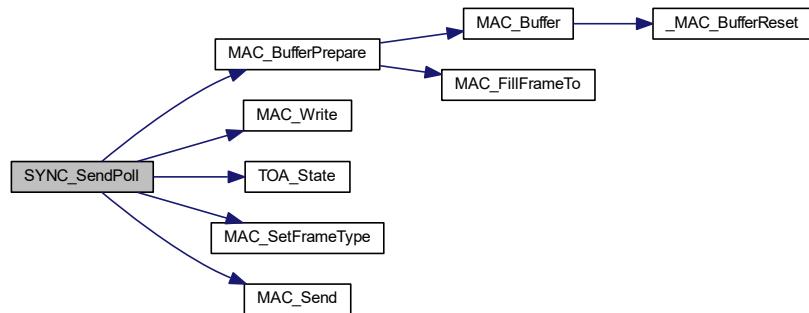
Parameters

in	<i>dst</i>	destination device address
in	<i>anchors</i>	list of anchor to measure ()
in	<i>anc_cnt</i>	counter of anchors to measure

Returns

int

Here is the call graph for this function:



6.37.3.7 SYNC_TxCb()

```
int SYNC_TxCb (
    int64_t TsDwTx )
```

sync tx callback

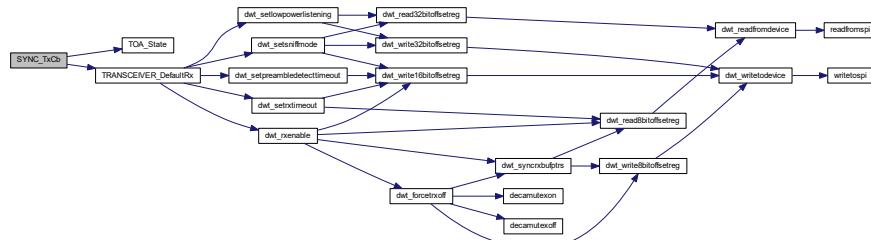
Parameters

in	<i>TsDwTx</i>	transmit timestamp in local dtu
----	---------------	---------------------------------

Returns

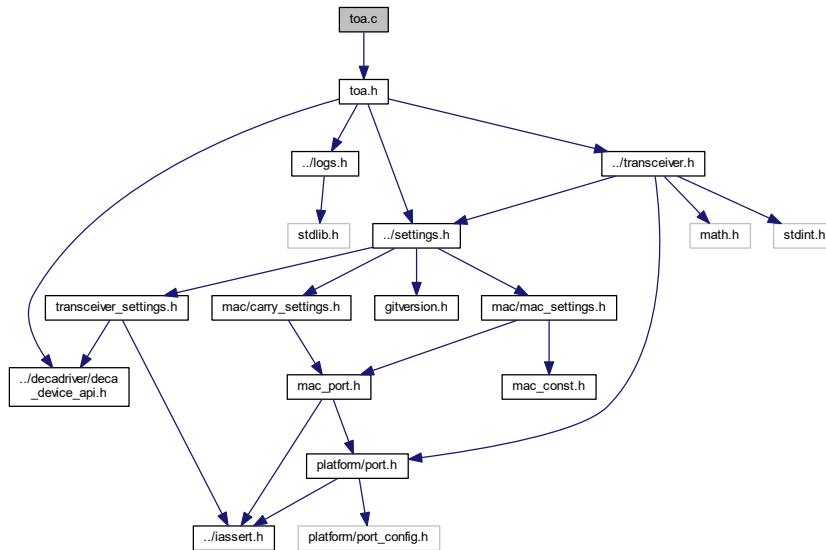
int 1 if packed was parsed 0 otherwise

Here is the call graph for this function:



6.38 toa.c File Reference

```
#include "toa.h"
Include dependency graph for toa.c:
```



Functions

- int `_TOA_GetRangeBias` (uint8 chan, int range, uint8 prf, int smartTxPower)
- void `TOA_State` (toa_core_t *toa, toa_state_t state)

change state of toa instance
- void `TOA_AddMeasure` (dev_addr_t addr, int distance)

add new measure to measures table
- int `TOA_FindAddrIndexInResp` (toa_core_t *toa, dev_addr_t addr)

- int **TOA_CalcTofDwTu** (const **toa_core_t** *toa, int resp_ind)
 - return index of a given address in a toa->addr_tab*
 - calculate TimeOfFlight in DecaWave TimeUnits or 0*
- float **TOA_CalcTofconst** (**toa_core_t** *toa, int resp_ind)
- int **TOA_GetRangeBias** (int cm)
- int **TOA_TofToCm** (float tof)
 - convert tof to cm including bias correction*
- int64_t **TOA_SetTxTime** (int64_t dw_time, uint32_t delay_us)
 - set delayed transmission parameters and return future TX timestamp*
- int **TOA_EnableRxBeforeFin** (const **toa_core_t** *toa, const **toa_settings_t** *tset, uint64_t DwPollRxTs)
 - enable receiver just before fin message*
- void **TOA_Write40bValue** (uint8_t *dst, int64_t val)
 - usefull when writing full timestamp*
- int64_t **TOA_Read40bValue** (const uint8_t *src)
 - usefull when reading full timestamp*

6.38.1 Function Documentation

6.38.1.1 **_TOA_GetRangeBias()**

```
int _TOA_GetRangeBias (
    uint8 chan,
    int range,
    uint8 prf,
    int smartTxPower )
```

Function: `dwt_getrangebias()`

Description: This function is used to return the range bias correction need for TWR with DW1000 units.

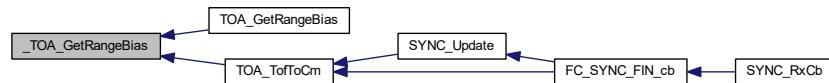
input parameters:

Parameters

<i>chan</i>	- specifies the operating channel (e.g. 1, 2, 3, 4, 5, 6 or 7)
<i>range</i>	- the calculated distance before correction
<i>prf</i>	- this is the PRF e.g. DWT_PRF_16M or DWT_PRF_64M

output parameters

returns correction needed in meters Here is the caller graph for this function:



6.38.1.2 TOA_AddMeasure()

```
void TOA_AddMeasure (
    dev_addr_t addr,
    int distance )
```

add new measure to measures table

Parameters

in	<i>addr</i>	of remote device
in	<i>distance</i>	in cm

Here is the call graph for this function:



Here is the caller graph for this function:



6.38.1.3 TOA_CalcTofconst()

```
float TOA_CalcTofconst (
    toa_core_t * toa,
    int resp_ind )
```

Here is the call graph for this function:



6.38.1.4 TOA_CalcTofDwTu()

```
int TOA_CalcTofDwTu (
    const toa_core_t * toa,
    int resp_ind )
```

calculate TimeOfFlight in DecaWave TimeUnits or 0

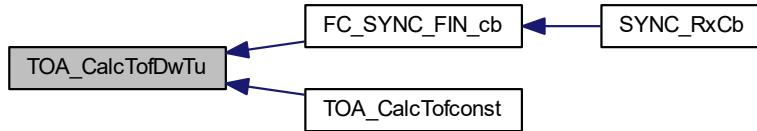
Parameters

in	<i>toa</i>	data
in	<i>resp_ind</i>	index of response

Returns

int tof in dtu

Here is the caller graph for this function:



6.38.1.5 TOA_EnableRxBeforeFin()

```
int TOA_EnableRxBeforeFin (
    const toa_core_t * toa,
    const toa_settings_t * tset,
    uint64_t DwPollRxTs )
```

enable receiver just before fin message

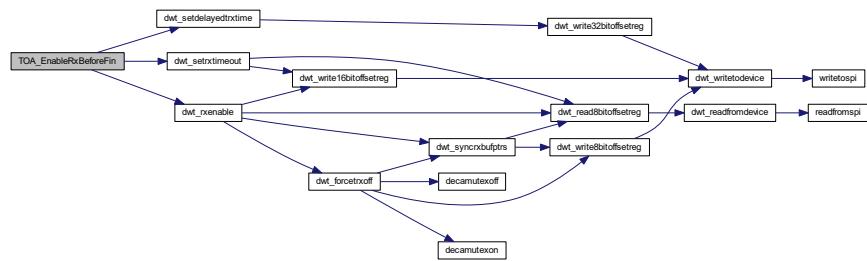
Parameters

in	<i>toa</i>	pointer to toa data
in	<i>tset</i>	pointer to toa settings
	<i>DwPollRxTs</i>	time in local dtu

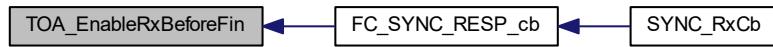
Returns

int 0 if success or error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.38.1.6 TOA_FindAddrIndexInResp()

```
int TOA_FindAddrIndexInResp (
    toa_core_t * toa,
    dev_addr_t addr )
```

return index of a given address in a toa->addr_tab

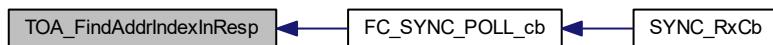
Parameters

in	<i>toa</i>	data
in	<i>addr</i>	address to find

Returns

int index of device of TOA_MAX_DEV_IN_POLL

Here is the caller graph for this function:



6.38.1.7 TOA_GetRangeBias()

```
int TOA_GetRangeBias (
    int cm )
```

Here is the call graph for this function:



6.38.1.8 TOA_Read40bValue()

```
int64_t TOA_Read40bValue (
    const uint8_t * src )
```

usefull when reading full timestamp

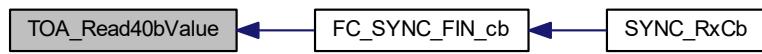
Parameters

in	src	pointer to address in LittleEndian
----	-----	------------------------------------

Returns

int64_t 40b address

Here is the caller graph for this function:



6.38.1.9 TOA_SetTxTime()

```
int64_t TOA_SetTxTime (
    int64_t dw_time,
    uint32_t delay_us )
```

set delayed transmission parameters and return future TX timestamp

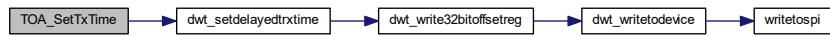
Parameters

in	<i>dw_time</i>	start local time in dru
in	<i>delay_us</i>	time delay in micro seconds

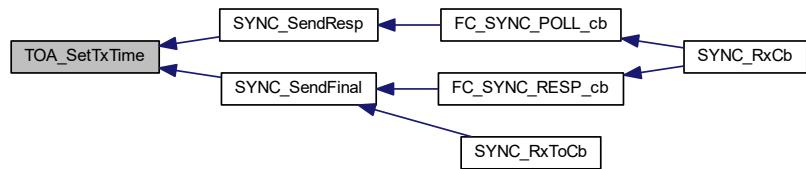
Returns

int64_t transmission time in dtu

Here is the call graph for this function:



Here is the caller graph for this function:



6.38.1.10 TOA_State()

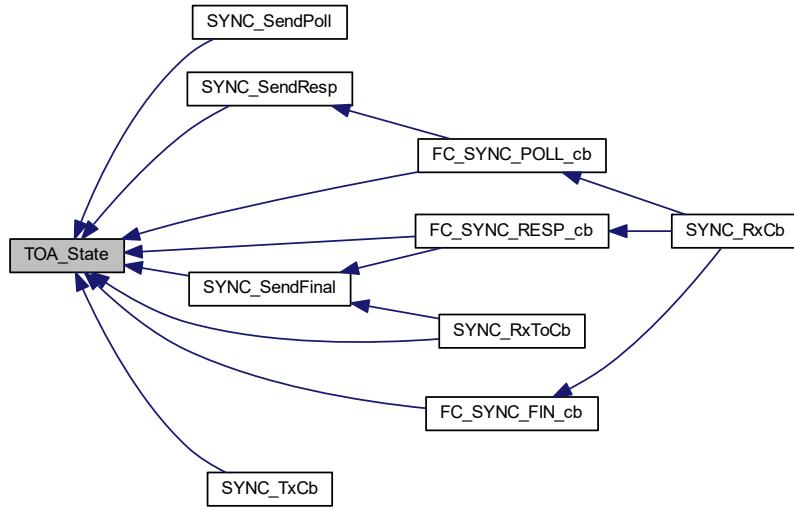
```
void TOA_State (
    toa_core_t * toa,
    toa_state_t state )
```

change state of toa instance

Parameters

<i>toa</i>	data
<i>state</i>	new state

Here is the caller graph for this function:



6.38.1.11 TOA_TofToCm()

```
int TOA_TofToCm (
    float tof )
```

convert tof to cm including bias correction

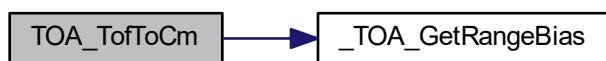
Parameters

in	<i>tof</i>	in seconds
----	------------	------------

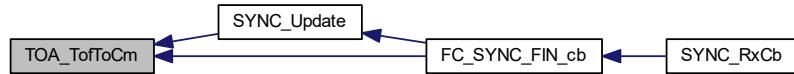
Returns

int distance in cm

Here is the call graph for this function:



Here is the caller graph for this function:



6.38.1.12 TOA_Write40bValue()

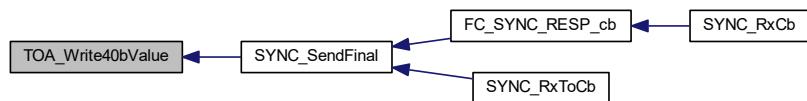
```
void TOA_Write40bValue (
    uint8_t * dst,
    int64_t val )
```

usefull when writing full timestamp

Parameters

out	<i>dst</i>	pointer to destination address
	<i>val</i>	value to write

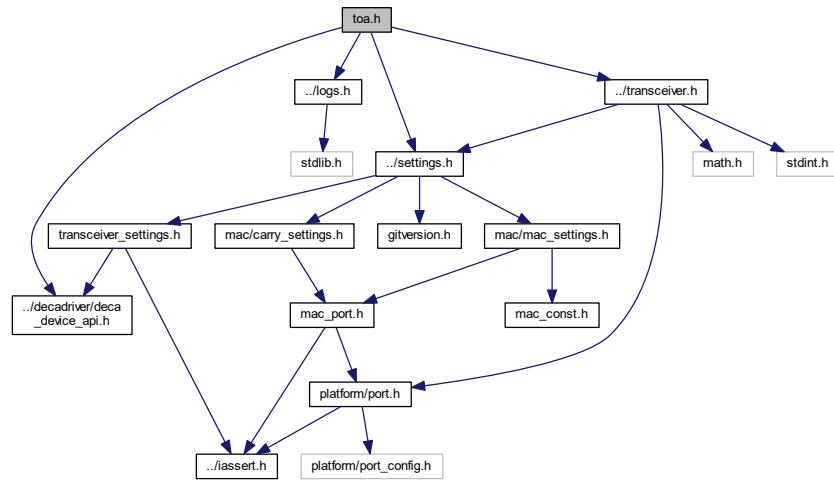
Here is the caller graph for this function:



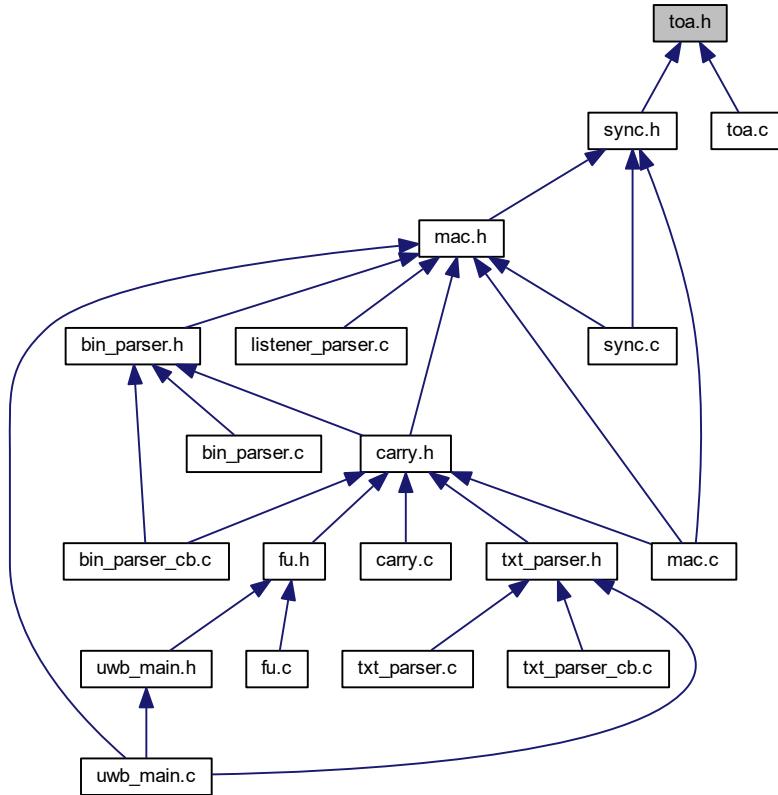
6.39 toa.h File Reference

```
#include "../decadriver/deca_device_api.h"
#include "../logs.h"
#include "../settings.h"
#include "../transceiver.h"
```

Include dependency graph for toa.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `toa_core_t`

Macros

- `#define SPEED_OF_LIGHT 299702547`
speed of light in air [m/s]

Enumerations

- enum `toa_state_t` {
 `TOA_IDLE, TOA_POLL_WAIT_TO_SEND, TOA_POLL_SENT, TOA_RESP_REC,`
`TOA_FIN_WAIT_TO_SEND, TOA_FIN_SENT, TOA_POLL_REC, TOA_RESP_WAIT_TO_SEND,`
`TOA_RESP_SENT, TOA_FIN_REC` }

Functions

- void `TOA_State` (`toa_core_t *toa, toa_state_t state`)
change state of toa instance
- void `TOA_AddMeasure` (`dev_addr_t addr, int distance`)
add new measure to measures table
- int `TOA_FindAddrIndexInResp` (`toa_core_t *toa, dev_addr_t addr`)
return index of a given address in a toa->addr_tab
- int `TOA_CalcTofDwTu` (`const toa_core_t *toa, int resp_ind`)
calculate TimeOfFlight in DecaWave TimeUnits or 0
- float `TOA_CalcTof` (`const toa_core_t *toa, int resp_ind`)
calculate TimeOfFlight in seconds
- int `TOA_TofToCm` (`float tof`)
convert tof to cm including bias correction
- int64_t `TOA_SetTxTime` (`int64_t dw_time, uint32_t delay_us`)
set delayed transmission parameters and return future TX timestamp
- int `TOA_EnableRxBeforeFin` (`const toa_core_t *toa, const toa_settings_t *tset, uint64_t DwPollRxTs`)
enable receiver just before fin message
- void `TOA_Write40bValue` (`uint8_t *dst, int64_t val`)
usefull when writing full timestamp
- int64_t `TOA_Read40bValue` (`const uint8_t *src`)
usefull when reading full timestamp

6.39.1 Macro Definition Documentation

6.39.1.1 SPEED_OF_LIGHT

```
#define SPEED_OF_LIGHT 299702547
```

speed of light in air [m/s]

6.39.2 Enumeration Type Documentation

6.39.2.1 toa_state_t

```
enum toa_state_t
```

Enumerator

TOA_IDLE
TOA_POLL_WAIT_TO_SEND
TOA_POLL_SENT
TOA_RESP_REC
TOA_FIN_WAIT_TO_SEND
TOA_FIN_SENT
TOA_POLL_REC
TOA_RESP_WAIT_TO_SEND
TOA_RESP_SENT
TOA_FIN_REC

6.39.3 Function Documentation

6.39.3.1 TOA_AddMeasure()

```
void TOA_AddMeasure (
    dev_addr_t addr,
    int distance )
```

add new measure to measures table

Parameters

in	<i>addr</i>	of remote device
in	<i>distance</i>	in cm

Here is the call graph for this function:



Here is the caller graph for this function:



6.39.3.2 TOA_CalcTof()

```
float TOA_CalcTof (
    const toa_core_t * toa,
    int resp_ind )
```

calculate TimeOfFlight in seconds

Parameters

in	<i>toa</i>	data
in	<i>resp_ind</i>	index of response

Returns

float tof in seconds

6.39.3.3 TOA_CalcTofDwTu()

```
int TOA_CalcTofDwTu (
    const toa_core_t * toa,
    int resp_ind )
```

calculate TimeOfFlight in DecaWave TimeUnits or 0

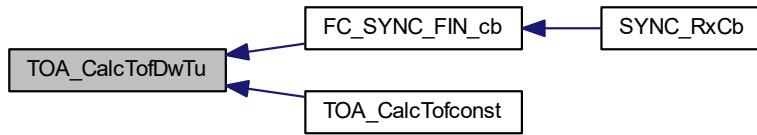
Parameters

in	<i>toa</i>	data
in	<i>resp_ind</i>	index of response

Returns

```
int tof in dtu
```

Here is the caller graph for this function:

**6.39.3.4 TOA_EnableRxBeforeFin()**

```
int TOA_EnableRxBeforeFin (
    const toa_core_t * toa,
    const toa_settings_t * tset,
    uint64_t DwPollRxTs )
```

enable receiver just before fin message

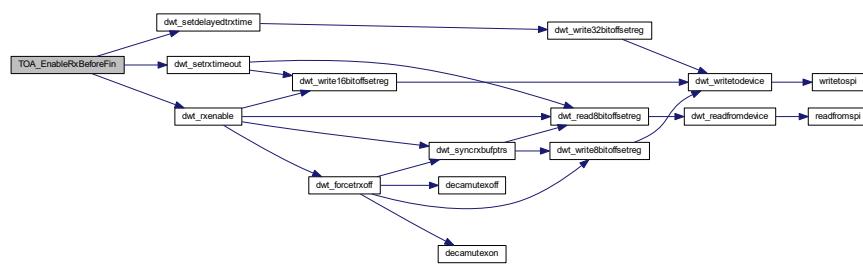
Parameters

in	<i>toa</i>	pointer to toa data
in	<i>tset</i>	pointer to toa settings
	<i>DwPollRxTs</i>	time in local dtu

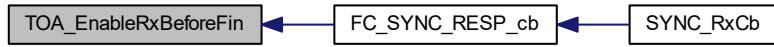
Returns

```
int 0 if sucess or error code
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.39.3.5 TOA_FindAddrIndexInResp()

```
int TOA_FindAddrIndexInResp (
    toa_core_t * toa,
    dev_addr_t addr )
```

return index of a given address in a toa->addr_tab

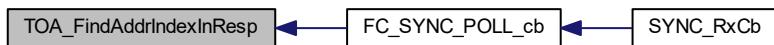
Parameters

in	<i>toa</i>	data
in	<i>addr</i>	address to find

Returns

int index of device of TOA_MAX_DEV_IN_POLL

Here is the caller graph for this function:



6.39.3.6 TOA_Read40bValue()

```
int64_t TOA_Read40bValue (
    const uint8_t * src )
```

usefull when reading full timestamp

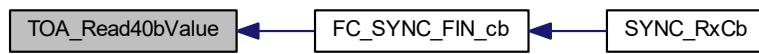
Parameters

in	src	pointer to address in LittleEndian
----	-----	------------------------------------

Returns

int64_t 40b address

Here is the caller graph for this function:

**6.39.3.7 TOA_SetTxTime()**

```
int64_t TOA_SetTxTime (
    int64_t dw_time,
    uint32_t delay_us )
```

set delayed transmission parameters and return future TX timestamp

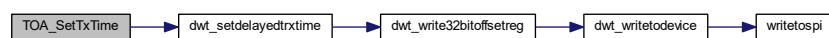
Parameters

in	<i>dw_time</i>	start local time in dru
in	<i>delay_us</i>	time delay in micro seconds

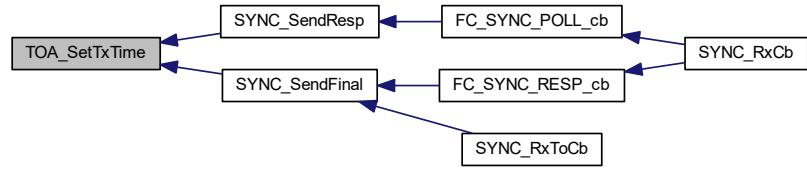
Returns

int64_t transmission time in dtu

Here is the call graph for this function:



Here is the caller graph for this function:



6.39.3.8 TOA_State()

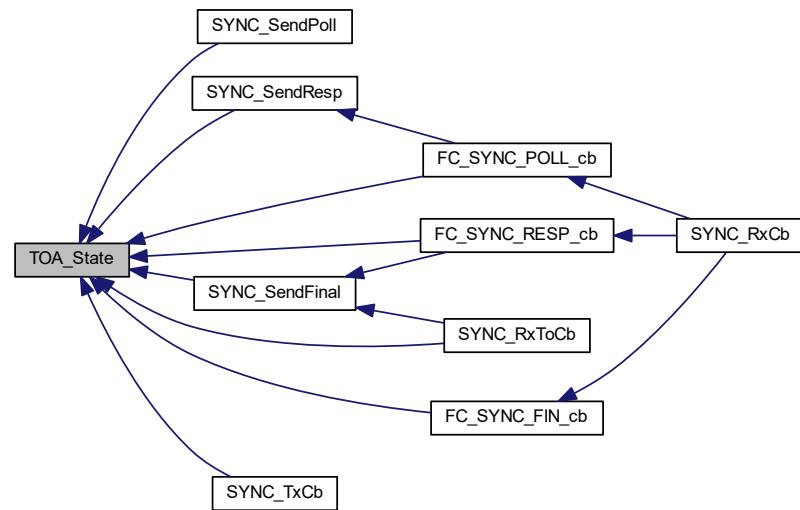
```
void TOA_State (
    toa_core_t * toa,
    toa_state_t state )
```

change state of toa instance

Parameters

<i>toa</i>	data
<i>state</i>	new state

Here is the caller graph for this function:



6.39.3.9 TOA_TofToCm()

```
int TOA_TofToCm (
    float tof )
```

convert tof to cm including bias correction

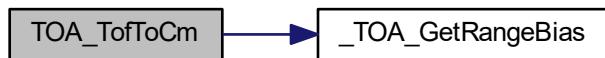
Parameters

in	<i>tof</i>	in seconds
----	------------	------------

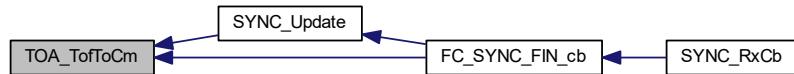
Returns

int distance in cm

Here is the call graph for this function:



Here is the caller graph for this function:



6.39.3.10 TOA_Write40bValue()

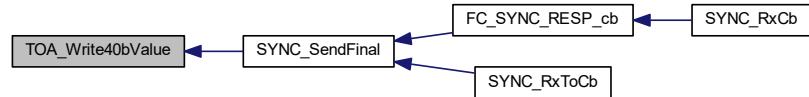
```
void TOA_Write40bValue (
    uint8_t * dst,
    int64_t val )
```

usefull when writing full timestamp

Parameters

out	<i>dst</i>	pointer to destination address
	<i>val</i>	value to write

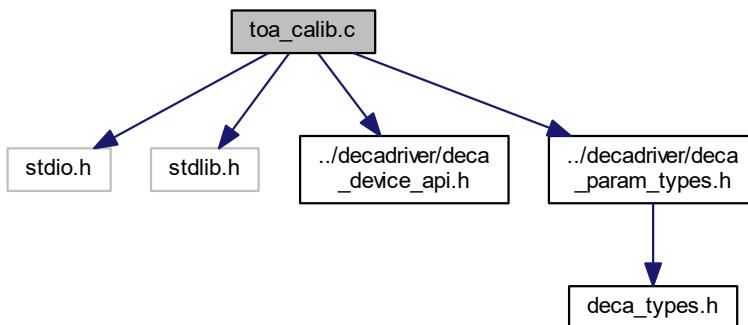
Here is the caller graph for this function:



6.40 toa_calib.c File Reference

```

#include <stdio.h>
#include <stdlib.h>
#include "../decadriver/deca_device_api.h"
#include "../decadriver/deca_param_types.h"
Include dependency graph for toa_calib.c:
  
```



Macros

- #define NUM_16M_OFFSET (37)
- #define NUM_16M_OFFSETWB (68)
- #define NUM_64M_OFFSET (26)
- #define NUM_64M_OFFSETWB (59)
- #define CM_OFFSET_16M_NB (-23)
- #define CM_OFFSET_16M_WB (-28)
- #define CM_OFFSET_64M_NB (-17)
- #define CM_OFFSET_64M_WB (-30)

Functions

- int _TOA_GetRangeBias (uint8 chan, int range, uint8 prf, int smartTxPower)

Variables

- const `uint8 chan_idxnb [NUM_CH_SUPPORTED]`
- const `uint8 chan_idxwb [NUM_CH_SUPPORTED]`
- const `uint8 range25cm16PRFn [4][NUM_16M_OFFSET]`
- const `uint8 range25cm16PRFwb [2][NUM_16M_OFFSETWB]`
- const `uint8 range25cm64PRFn [4][NUM_64M_OFFSET]`
- const `uint8 range25cm64PRFwb [2][NUM_64M_OFFSETWB]`

6.40.1 Macro Definition Documentation

6.40.1.1 CM_OFFSET_16M_NB

```
#define CM_OFFSET_16M_NB (-23)
```

6.40.1.2 CM_OFFSET_16M_WB

```
#define CM_OFFSET_16M_WB (-28)
```

6.40.1.3 CM_OFFSET_64M_NB

```
#define CM_OFFSET_64M_NB (-17)
```

6.40.1.4 CM_OFFSET_64M_WB

```
#define CM_OFFSET_64M_WB (-30)
```

6.40.1.5 NUM_16M_OFFSET

```
#define NUM_16M_OFFSET (37)
```

6.40.1.6 NUM_16M_OFFSETWB

```
#define NUM_16M_OFFSETWB (68)
```

6.40.1.7 NUM_64M_OFFSET

```
#define NUM_64M_OFFSET (26)
```

6.40.1.8 NUM_64M_OFFSETWB

```
#define NUM_64M_OFFSETWB (59)
```

6.40.2 Function Documentation

6.40.2.1 _TOA_GetRangeBias()

```
int _TOA_GetRangeBias (
    uint8 chan,
    int range,
    uint8 prf,
    int smartTxPower )
```

Function: dwt_getrangebias()

Description: This function is used to return the range bias correction need for TWR with DW1000 units.

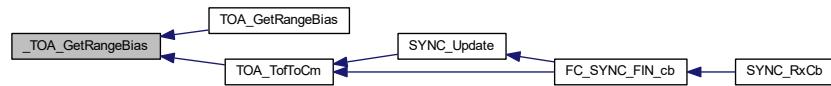
input parameters:

Parameters

<i>chan</i>	- specifies the operating channel (e.g. 1, 2, 3, 4, 5, 6 or 7)
<i>range</i>	- the calculated distance before correction
<i>prf</i>	- this is the PRF e.g. DWT_PRF_16M or DWT_PRF_64M

output parameters

returns correction needed in meters Here is the caller graph for this function:



6.40.3 Variable Documentation

6.40.3.1 chan_idxnb

```
const uint8 chan_idxnb[NUM_CH_SUPPORTED]
```

Initial value:

```
= {
  0, 0, 1, 2,
  0, 3, 0, 0}
```

6.40.3.2 chan_idxwb

```
const uint8 chan_idxwb[NUM_CH_SUPPORTED]
```

Initial value:

```
= {
  0, 0, 0, 0,
  0, 0, 0, 1}
```

6.40.3.3 range25cm16PRFnB

```
const uint8 range25cm16PRFnB[4][NUM_16M_OFFSET]
```

Initial value:

```
= {
  {1, 3, 4, 5, 7, 9, 11, 12, 13, 15, 18, 20, 23,
   25, 28, 30, 33, 36, 40, 43, 47, 50, 54, 58, 63, 66,
   71, 76, 82, 89, 98, 109, 127, 155, 222, 255},

  {1, 2, 4, 5, 6, 8, 9, 10, 12, 13, 15, 18, 20,
   22, 24, 27, 29, 32, 35, 38, 41, 44, 47, 51, 55, 58,
   62, 66, 71, 78, 85, 96, 111, 135, 194, 240, 255},

  {1, 2, 3, 4, 5, 7, 8, 9, 10, 12, 14, 16, 18,
   20, 22, 24, 26, 28, 31, 33, 36, 39, 42, 45, 49, 52,
   55, 59, 63, 69, 76, 85, 98, 120, 173, 213, 255},

  {1, 1, 2, 3, 4, 5, 6, 6, 7, 8, 9, 11, 12,
   14, 15, 16, 18, 20, 21, 23, 25, 27, 29, 31, 34, 36,
   38, 41, 44, 48, 53, 59, 68, 83, 120, 148, 255}}
```

6.40.3.4 range25cm16PRFwb

```
const uint8 range25cm16PRFwb[2] [NUM_16M_OFFSETWB]
```

Initial value:

```
=
{
    {7, 7, 8, 9, 9, 10, 11, 11,
     12, 13, 14, 15, 16, 17, 18, 19,
     20, 21, 22, 23, 24, 26, 27, 28,
     30, 31, 32, 34, 36, 38, 40, 42,
     44, 46, 48, 50, 52, 55, 57, 59,
     61, 63, 66, 68, 71, 74, 78, 81,
     85, 89, 94, 99, 104, 110, 116, 123,
     130, 139, 150, 164, 182, 207, 238, 255,
     255, 255, 255, 255},
    {
        4, 5, 5, 5, 6, 6, 7, 7,
        7, 8, 9, 9, 10, 10, 11, 11,
        12, 13, 13, 14, 15, 16, 17, 17,
        18, 19, 20, 21, 22, 23, 25, 26,
        27, 29, 30, 31, 32, 34, 35, 36,
        38, 39, 40, 42, 44, 46, 48, 50,
        52, 55, 58, 61, 64, 68, 72, 75,
        80, 85, 92, 101, 112, 127, 147, 168,
        182, 194, 205, 255}}
}
```

6.40.3.5 range25cm64PRFnB

```
const uint8 range25cm64PRFnB[4] [NUM_64M_OFFSET]
```

Initial value:

```
= {
    {1, 2, 2, 3, 4, 5, 7, 10, 13, 16, 19, 22, 24,
     27, 30, 32, 35, 38, 43, 48, 56, 78, 101, 120, 157, 255},
    {1, 2, 2, 3, 4, 4, 6, 9, 12, 14, 17, 19, 21,
     24, 26, 28, 31, 33, 37, 42, 49, 68, 89, 105, 138, 255},
    {1, 1, 2, 3, 3, 4, 5, 8, 10, 13, 15, 17, 19,
     21, 23, 25, 27, 30, 33, 37, 44, 60, 79, 93, 122, 255},
    {1, 1, 1, 2, 2, 3, 4, 6, 7, 9, 10, 12, 13, 15,
     16, 17, 19, 21, 23, 26, 30, 42, 55, 65, 85, 255}}
}
```

6.40.3.6 range25cm64PRFwb

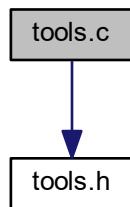
```
const uint8 range25cm64PRFwb[2] [NUM_64M_OFFSETWB]
```

Initial value:

```
=  
{  
    {7, 8, 8, 9, 9, 10, 11, 12,  
     13, 13, 14, 15, 16, 16, 17, 18,  
     19, 19, 20, 21, 22, 24, 25, 27,  
     28, 29, 30, 32, 33, 34, 35, 37,  
     39, 41, 43, 45, 48, 50, 53, 56,  
     60, 64, 68, 74, 81, 89, 98, 109,  
     122, 136, 146, 154, 162, 178, 220, 249,  
     255, 255, 255},  
  
    {4, 5, 5, 5, 6, 6, 7, 7,  
     8, 8, 9, 9, 10, 10, 10, 11,  
     11, 12, 13, 13, 14, 15, 16, 16,  
     17, 18, 19, 19, 20, 21, 22, 23,  
     24, 25, 26, 28, 29, 31, 33, 35,  
     37, 39, 42, 46, 50, 54, 60, 67,  
     75, 83, 90, 95, 100, 110, 135, 153,  
     172, 192, 255}}}
```

6.41 tools.c File Reference

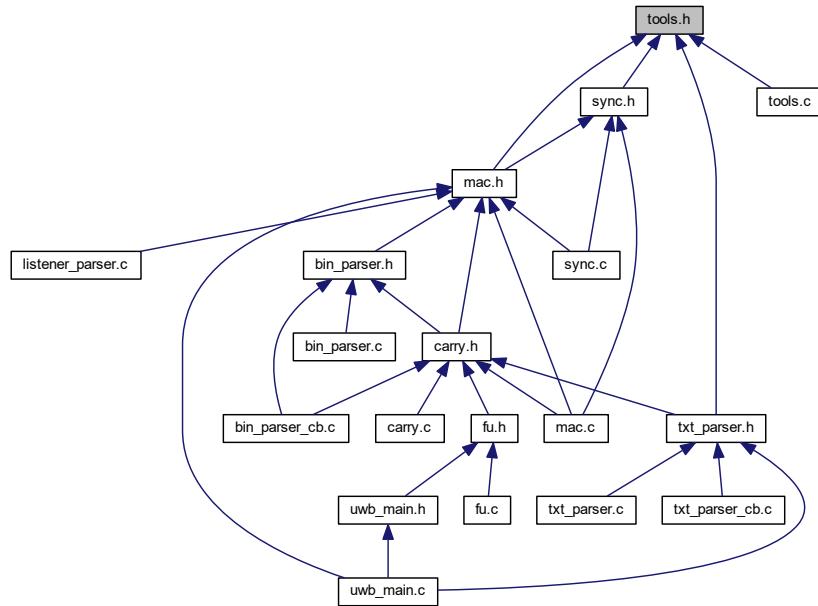
```
#include "tools.h"  
Include dependency graph for tools.c:
```



6.42 tools.h File Reference

file for small code snippets

This graph shows which files directly or indirectly include this file:



Macros

- `#define UNUSED_1(a) (void)(a)`
- `#define UNUSED_2(a, b) (void)(a),UNUSED_1(b)`
- `#define UNUSED_3(a, b, c) (void)(a),UNUSED_2(b,c)`
- `#define UNUSED_4(a, b, c, d) (void)(a),UNUSED_3(b,c,d)`
- `#define UNUSED_5(a, b, c, d, e) (void)(a),UNUSED_4(b,c,d,e)`
- `#define UNUSED_6(a, b, c, d, e, f) (void)(a),UNUSED_5(b,c,d,e,f)`
- `#define UNUSED_7(a, b, c, d, e, f, g) (void)(a),UNUSED_6(b,c,d,e,f,g)`
- `#define UNUSED_8(a, b, c, d, e, f, g, h) (void)(a),UNUSED_7(b,c,d,e,f,g,h)`
- `#define VA_NUM_ARGS_IMPL(_1, _2, _3, _4, _5, _6, _7, _8, _9, N, ...) N`
- `#define VA_NUM_ARGS(...) VA_NUM_ARGS_IMPL(__VA_ARGS__, 9, 8, 7, 6, 5, 4, 3, 2, 1)`
- `#define ALL_UNUSED_IMPL_(nargs) UNUSED_## nargs`
- `#define ALL_UNUSED_IMPL(nargs) ALL_UNUSED_IMPL_(nargs)`
- `#define ALL_UNUSED(...) ALL_UNUSED_IMPL(VA_NUM_ARGS(__VA_ARGS__)(__VA_ARGS__)`
useful to prevent compilator 'unused variable' warning
- `#define INCREMENT_CYCLE(VAR, START, STOP) (VAR=((VAR+1)==STOP?START:(VAR+1)))`
increment value and trim to <start,stop> range
- `#define DECREMENT_CYCLE(VAR, START, STOP) (VAR=(VAR==(START?STOP:VAR)-1))`
decrement value and trim to <start,stop> range
- `#define INCREMENT_MOD(VAR, MAX) INCREMENT_CYCLE(VAR,0,MAX)`
increment value and trim to <0,max> range
- `#define DECREMENT_MOD(VAR, MAX) DECREMENT_CYCLE(VAR,0,MAX)`
decrement value and trim to <0,max> range

6.42.1 Detailed Description

file for small code snippets

Author

Karol Trzcinski

Date

2018-07-02

6.42.2 Macro Definition Documentation

6.42.2.1 ALL_UNUSED

```
#define ALL_UNUSED( ... ) ALL_UNUSED_IMPL( VA_NUM_ARGS(__VA_ARGS__) ) __VA_ARGS__ )
```

useful to prevent compilator 'unused variable' warning

6.42.2.2 ALL_UNUSED_IMPL

```
#define ALL_UNUSED_IMPL( nargs ) ALL_UNUSED_IMPL_(nargs)
```

6.42.2.3 ALL_UNUSED_IMPL_

```
#define ALL_UNUSED_IMPL_( nargs ) UNUSED_## nargs
```

6.42.2.4 DECREMENT_CYCLE

```
#define DECREMENT_CYCLE( VAR, START, STOP ) (VAR=(VAR==(START?STOP:VAR)-1))
```

decrement value and trim to <start,stop> range

6.42.2.5 DECREMENT_MOD

```
#define DECREMENT_MOD (
    VAR,
    MAX ) DECREMENT_CYCLE (VAR, 0, MAX)
```

decrement value and trim to <0,max) range

6.42.2.6 INCREMENT_CYCLE

```
#define INCREMENT_CYCLE (
    VAR,
    START,
    STOP ) (VAR= ( (VAR+1) ==STOP?START:(VAR+1) ))
```

increment value and trim to <start,stop) range

6.42.2.7 INCREMENT_MOD

```
#define INCREMENT_MOD (
    VAR,
    MAX ) INCREMENT_CYCLE (VAR, 0, MAX)
```

increment value and trim to <0,max) range

6.42.2.8 UNUSED_1

```
#define UNUSED_1 (
    a ) (void) (a)
```

6.42.2.9 UNUSED_2

```
#define UNUSED_2 (
    a,
    b ) (void) (a), UNUSED_1 (b)
```

6.42.2.10 UNUSED_3

```
#define UNUSED_3 (
    a,
    b,
    c ) (void) (a), UNUSED_2 (b, c)
```

6.42.2.11 UNUSED_4

```
#define UNUSED_4 (
    a,
    b,
    c,
    d ) (void) (a), UNUSED_3 (b, c, d)
```

6.42.2.12 UNUSED_5

```
#define UNUSED_5 (
    a,
    b,
    c,
    d,
    e ) (void) (a), UNUSED_4 (b, c, d, e)
```

6.42.2.13 UNUSED_6

```
#define UNUSED_6 (
    a,
    b,
    c,
    d,
    e,
    f ) (void) (a), UNUSED_5 (b, c, d, e, f)
```

6.42.2.14 UNUSED_7

```
#define UNUSED_7 (
    a,
    b,
    c,
    d,
    e,
    f,
    g ) (void) (a), UNUSED_6 (b, c, d, e, f, g)
```

6.42.2.15 UNUSED_8

```
#define UNUSED_8(  
    a,  
    b,  
    c,  
    d,  
    e,  
    f,  
    g,  
    h ) (void) (a),UNUSED_7(b,c,d,e,f,g,h)
```

6.42.2.16 VA_NUM_ARGS

```
#define VA_NUM_ARGS(  
... ) VA_NUM_ARGS_IMPL(__VA_ARGS__, 9, 8, 7, 6, 5, 4, 3, 2, 1)
```

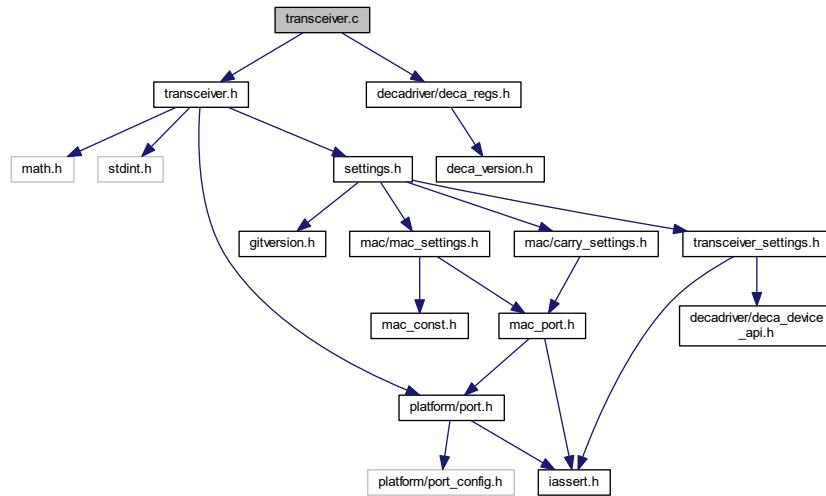
6.42.2.17 VA_NUM_ARGS_IMPL

```
#define VA_NUM_ARGS_IMPL(  
    _1,  
    _2,  
    _3,  
    _4,  
    _5,  
    _6,  
    _7,  
    _8,  
    _9,  
    N,  
    ... ) N
```

6.43 transceiver.c File Reference

```
#include "transceiver.h"  
#include "decadrive/deca_regs.h"
```

Include dependency graph for transceiver.c:



Functions

- void **TRANSCEIVER_Init ()**
Initialize transceiver based on current settings.
- void **TRANSCEIVER_SetCb (dwt_cb_t tx_cb, dwt_cb_t rx_cb, dwt_cb_t rxto_cb, dwt_cb_t rxerr_cb)**
connect event callbacks
- void **TRANSCEIVER_SetAddr (pan_dev_addr_t pan_addr, dev_addr_t dev_addr)**
set device address
- void **TRANSCEIVER_DefaultRx ()**
turn on receiver
- void **TRANSCEIVER_EnterDeepSleep ()**
move transmitter to low power mode
- void **TRANSCEIVER_WakeUp (uint8_t *buf, int len)**
- int64_t **TRANSCEIVER_GetRxTimestamp (void)**
return last packed receive timestamp
- int64_t **TRANSCEIVER_GetTxTimestamp (void)**
return last packed transmit timestamp
- int64_t **TRANSCEIVER_GetTime ()**
return time in transceiver
- void **TRANSCEIVER_ReadDiagnostic (int16_t *cRSSI, int16_t *cFPP, int16_t *cSNR)**
read values from last rx from transceiver
- int **TRANSCEIVER_Send (const void *buf, unsigned int len)**
immediately send data via transceiver
- int **TRANSCEIVER_SendRanging (const void *buf, unsigned int len, uint8_t flags)**
send ranging data via transceiver
- void **TRANSCEIVER_Read (void *buf, unsigned int len)**
read data from device
- int **TRANSCEIVER_EstimateTxTimeUs (unsigned int len)**
estimate packet transmission time

Variables

- int `transceiver_br` = 0
- int `transceiver_plen` = 0
- int `transceiver_pac` = 0
- int `transceiver_sfd` = 0

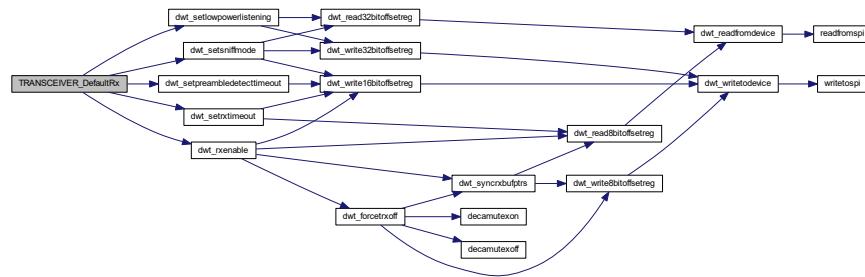
6.43.1 Function Documentation

6.43.1.1 TRANSCEIVER_DefaultRx()

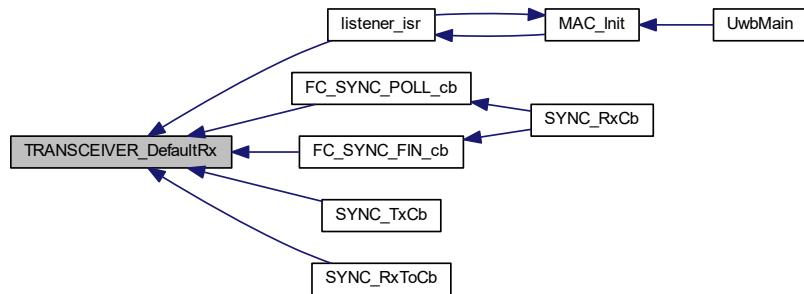
```
void TRANSCEIVER_DefaultRx ( )
```

turn on receiver

Here is the call graph for this function:



Here is the caller graph for this function:

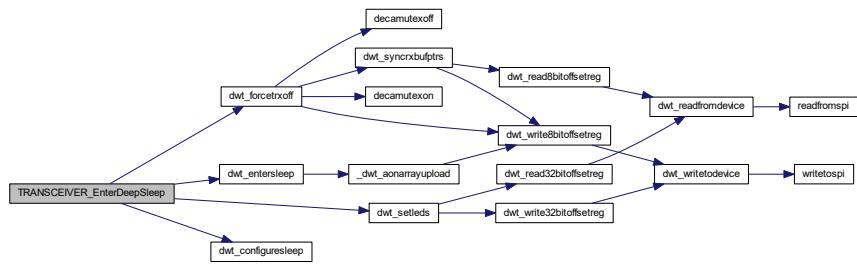


6.43.1.2 TRANSCEIVER_EnterDeepSleep()

```
void TRANSCEIVER_EnterDeepSleep ( )
```

move transmitter to low power mode

Here is the call graph for this function:



Here is the caller graph for this function:



6.43.1.3 TRANSCEIVER_EstimateTxTimeUs()

```
int TRANSCEIVER_EstimateTxTimeUs (
    unsigned int len )
```

estimate packet transmission time

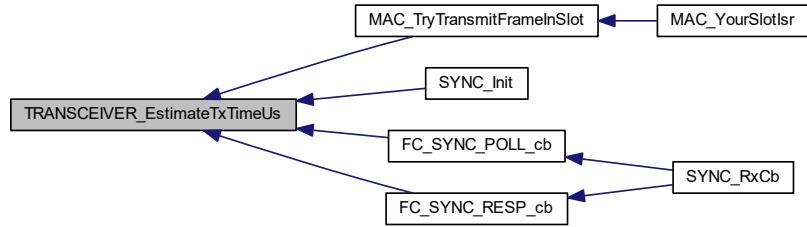
Parameters

in	<i>len</i>	packed length (without preamble, SFD and CRC len)
----	------------	---

Returns

```
int estimated time in micro seconds
```

Here is the caller graph for this function:

**6.43.1.4 TRANSCEIVER_GetRxTimestamp()**

```
int64_t TRANSCEIVER_GetRxTimestamp (
    void )
```

return last packed receive timestamp

timestamp is saved at the begin of SFD detection

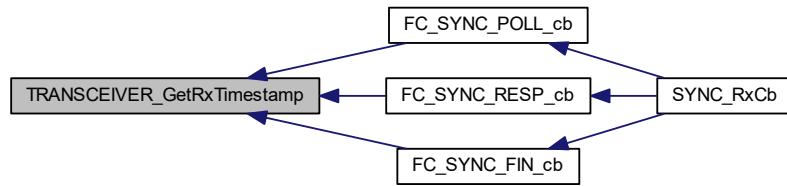
Returns

```
int64_t event timestamp in dw time units
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.43.1.5 TRANSCEIVER_GetTime()

```
int64_t TRANSCEIVER_GetTime( )
```

return time in transceiver

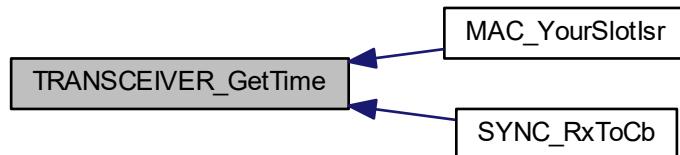
Returns

int64_t time in transceiver time units

Here is the call graph for this function:



Here is the caller graph for this function:



6.43.1.6 TRANSCEIVER_GetTxTimestamp()

```
int64_t TRANSCEIVER_GetTxTimestamp( void )
```

return last packed transmit timestamp

timestamp is saved at the begin of SFD transmission

Returns

```
int64_t event timestamp in dw time units
```

Here is the call graph for this function:



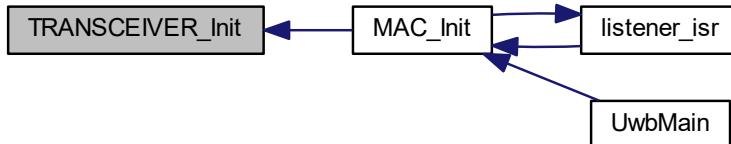
Here is the caller graph for this function:

**6.43.1.7 TRANSCEIVER_Init()**

```
void TRANSCEIVER_Init( )
```

Initialize transceiver based on current settings.

Here is the caller graph for this function:

**6.43.1.8 TRANSCEIVER_Read()**

```
void TRANSCEIVER_Read( void * buf,
                      unsigned int len )
```

read data from device

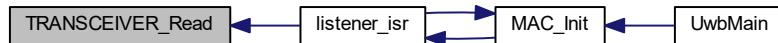
Parameters

<i>out</i>	<i>buf</i>	pointer to data buffer
<i>in</i>	<i>len</i>	data length in bytes

Here is the call graph for this function:



Here is the caller graph for this function:

**6.43.1.9 TRANSCEIVER_ReadDiagnostic()**

```

void TRANSCEIVER_ReadDiagnostic (
    int16_t * cRSSI,
    int16_t * cFPP,
    int16_t * cSNR )
  
```

read values from last rx from transceiver

Parameters

<i>out</i>	<i>cRSSI</i>	Received Signal Strength Indicator in centy dBm
<i>out</i>	<i>cFPP</i>	First Power Path in centy dBm
<i>out</i>	<i>cSNR</i>	Signal to Noise Ratio in centy dB

Here is the call graph for this function:



Here is the caller graph for this function:



6.43.1.10 TRANSCEIVER_Send()

```
int TRANSCEIVER_Send (
    const void * buf,
    unsigned int len )
```

immediately send data via transceiver

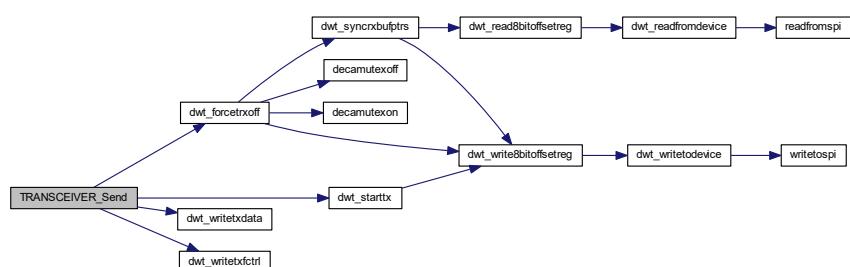
Parameters

in	<i>buf</i>	pointer to data buffer
in	<i>len</i>	data length in bytes

Returns

int 0 if success, error code otherwise

Here is the call graph for this function:



6.43.1.11 TRANSCEIVER_SendRanging()

```
int TRANSCEIVER_SendRanging (
    const void * buf,
    unsigned int len,
    uint8_t flags )
```

send ranging data via transceiver

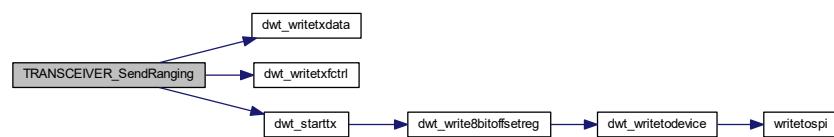
Parameters

in	<i>buf</i>	pointer to data buffer
in	<i>len</i>	data length in bytes
in	<i>flags</i>	flags for transceiver: <ul style="list-style-type: none">• DWT_START_TX_IMMEDIATE• DWT_START_TX_DELAYED• DWT_RESPONSE_EXPECTED

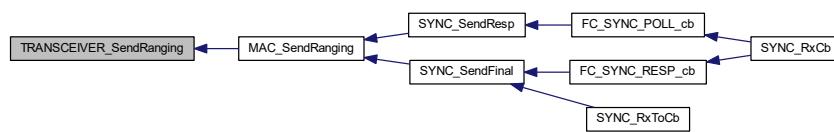
Returns

int 0 if success, error code otherwise

Here is the call graph for this function:



Here is the caller graph for this function:

**6.43.1.12 TRANSCEIVER_SetAddr()**

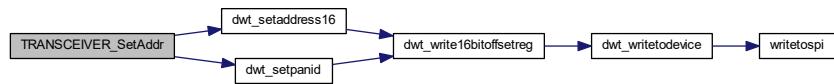
```
void TRANSCEIVER_SetAddr (
    pan_dev_addr_t pan_addr,
    dev_addr_t dev_addr )
```

set device address

Parameters

in	<i>pan_addr</i>	personal access network identifier
in	<i>dev_addr</i>	device address

Here is the call graph for this function:



6.43.1.13 TRANSCEIVER_SetCb()

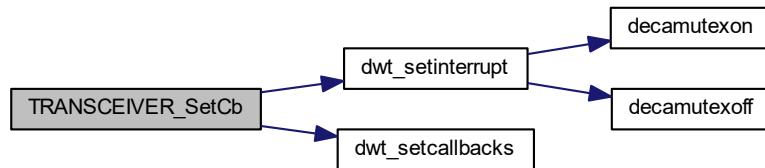
```
void TRANSCEIVER_SetCb (
    dwt_cb_t tx_cb,
    dwt_cb_t rx_cb,
    dwt_cb_t rxto_cb,
    dwt_cb_t rxerr_cb )
```

connect event callbacks

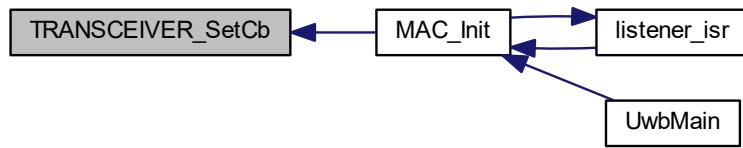
Parameters

in	<i>tx_cb</i>	transmitted message callback
in	<i>rx_cb</i>	received message callback
in	<i>rxto_cb</i>	timeout during receiving package callback
in	<i>rxerr_cb</i>	error during receiving package callback

Here is the call graph for this function:



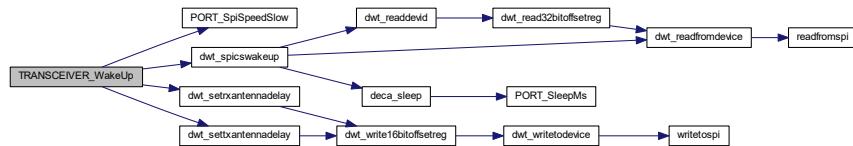
Here is the caller graph for this function:



6.43.1.14 TRANSCEIVER_WakeUp()

```
void TRANSCEIVER_WakeUp (
    uint8_t * buf,
    int len )
```

Here is the call graph for this function:



6.43.2 Variable Documentation

6.43.2.1 transceiver_br

```
int transceiver_br = 0
```

6.43.2.2 transceiver_pac

```
int transceiver_pac = 0
```

6.43.2.3 transceiver_plen

```
int transceiver_plen = 0
```

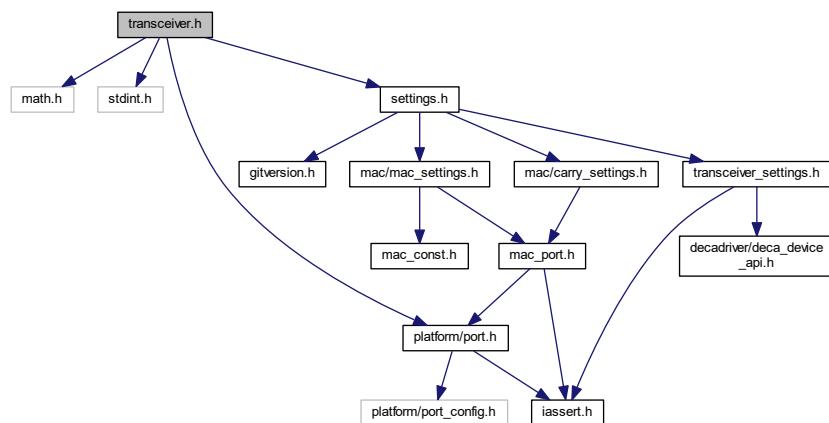
6.43.2.4 transceiver_sfd

```
int transceiver_sfd = 0
```

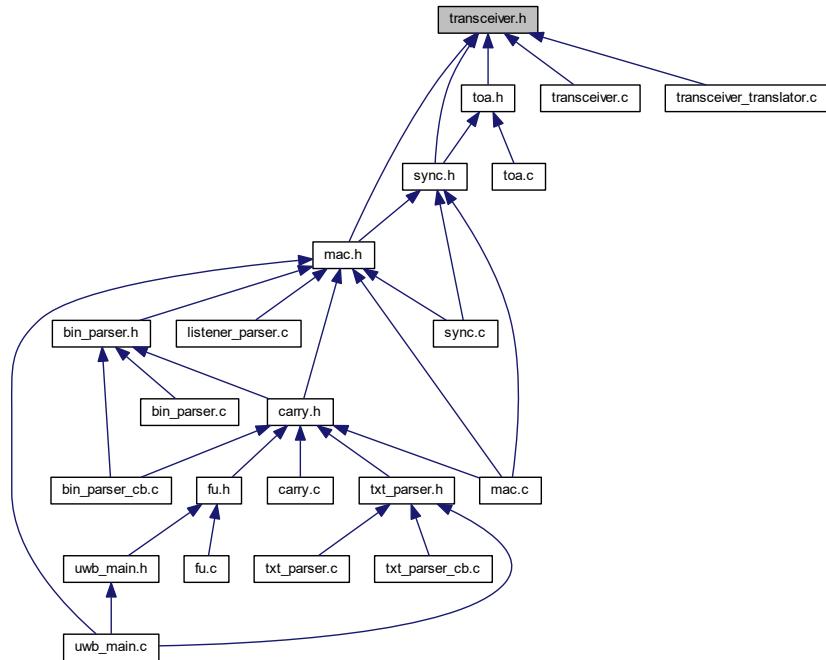
6.44 transceiver.h File Reference

transceiver

```
#include <math.h>
#include <stdint.h>
#include "platform/port.h"
#include "settings.h"
Include dependency graph for transceiver.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- #define **UUS_TO_DWT_TIME** (65536ul)
- #define **MASK_40BIT** (0xffffffffffffL)

Functions

- void **TRANSCEIVER_Init** ()

Initialize transceiver based on current settings.
- void **TRANSCEIVER_SetAddr** (**pan_dev_addr_t** pan_addr, **dev_addr_t** dev_addr)

set device address
- void **TRANSCEIVER_SetCb** (**dwt_cb_t** tx_cb, **dwt_cb_t** rx_cb, **dwt_cb_t** rxto_cb, **dwt_cb_t** rxerr_cb)

connect event callbacks
- void **TRANSCEIVER_EnterDeepSleep** ()

move transmitter to low power mode
- int **TRANSCEIVER_Send** (const void *buf, unsigned int len)

immediately send data via transceiver
- int **TRANSCEIVER_SendRanging** (const void *buf, unsigned int len, uint8_t flags)

send ranging data via transceiver
- void **TRANSCEIVER_Read** (void *buf, unsigned int len)

read data from device
- void **TRANSCEIVER_DefaultRx** ()

turn on receiver
- int64_t **TRANSCEIVER_GetTime** ()

return time in transceiver

- int64_t **TRANSCEIVER_GetRxTimestamp** (void)
return last packed receive timestamp
- int64_t **TRANSCEIVER_GetTxTimestamp** (void)
return last packed transmit timestamp
- void **TRANSCEIVER_ReadDiagnostic** (int16_t *cRSSI, int16_t *cFPP, int16_t *cSNR)
read values from last rx from transceiver
- int **TRANSCEIVER_EstimateTxTimeUs** (unsigned int len)
estimate packet transmission time

6.44.1 Detailed Description

transceiver

Author

Karol Trzcinski

Date

2018-06-28

6.44.2 Macro Definition Documentation

6.44.2.1 MASK_40BIT

```
#define MASK_40BIT (0xffffffffffffL)
```

6.44.2.2 UUS_TO_DWT_TIME

```
#define UUS_TO_DWT_TIME (65536ul)
```

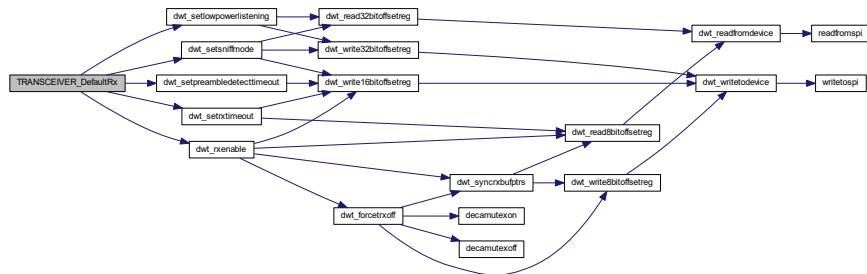
6.44.3 Function Documentation

6.44.3.1 TRANSCEIVER_DefaultRx()

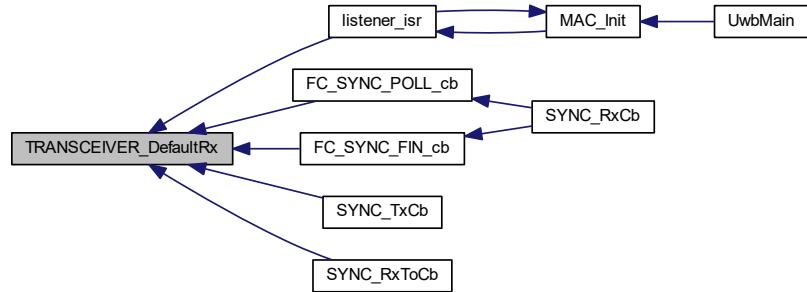
```
void TRANSCEIVER_DefaultRx ( )
```

turn on receiver

Here is the call graph for this function:



Here is the caller graph for this function:

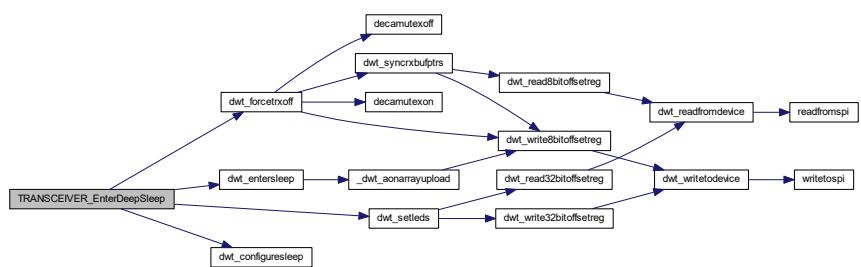


6.44.3.2 TRANSCEIVER_EnterDeepSleep()

```
void TRANSCEIVER_EnterDeepSleep ( )
```

move transmitter to low power mode

Here is the call graph for this function:



Here is the caller graph for this function:



6.44.3.3 TRANSCEIVER_EstimateTxTimeUs()

```
int TRANSCEIVER_EstimateTxTimeUs (
    unsigned int len )
```

estimate packet transmission time

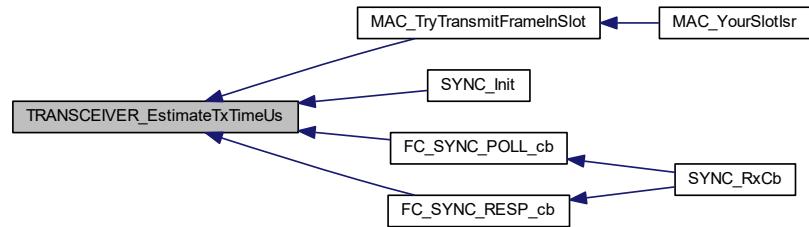
Parameters

in	len	packed length (without preamble, SFD and CRC len)
----	-----	---

Returns

int estimated time in micro seconds

Here is the caller graph for this function:



6.44.3.4 TRANSCEIVER_GetRxTimestamp()

```
int64_t TRANSCEIVER_GetRxTimestamp (
    void )
```

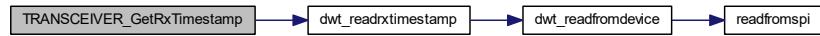
return last packed receive timestamp

timestamp is saved at the begin of SFD detection

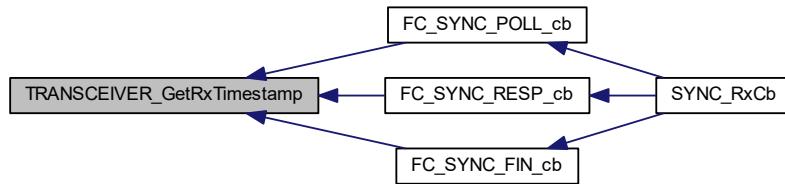
Returns

```
int64_t event timestamp in dw time units
```

Here is the call graph for this function:



Here is the caller graph for this function:

**6.44.3.5 TRANSCEIVER_GetTime()**

```
int64_t TRANSCEIVER_GetTime( )
```

return time in transceiver

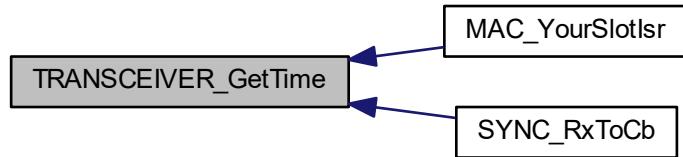
Returns

```
int64_t time in transceiver time units
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.44.3.6 `TRANSCEIVER_GetTxTimestamp()`

```
int64_t TRANSCEIVER_GetTxTimestamp ( void )
```

return last packed transmit timestamp

timestamp is saved at the begin of SFD transmission

Returns

`int64_t` event timestamp in dw time units

Here is the call graph for this function:



Here is the caller graph for this function:

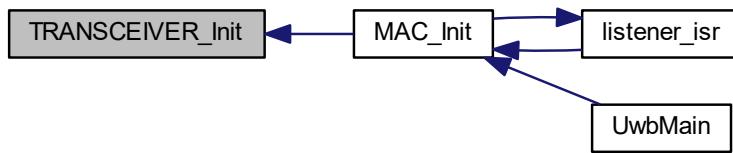


6.44.3.7 TRANSCEIVER_Init()

```
void TRANSCEIVER_Init( )
```

Initialize transceiver based on current settings.

Here is the caller graph for this function:



6.44.3.8 TRANSCEIVER_Read()

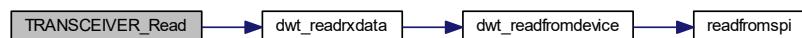
```
void TRANSCEIVER_Read(
    void * buf,
    unsigned int len )
```

read data from device

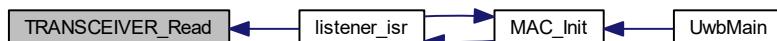
Parameters

<code>out</code>	<code>buf</code>	pointer to data buffer
<code>in</code>	<code>len</code>	data length in bytes

Here is the call graph for this function:



Here is the caller graph for this function:



6.44.3.9 TRANSCEIVER_ReadDiagnostic()

```
void TRANSCEIVER_ReadDiagnostic (
    int16_t * cRSSI,
    int16_t * cFPP,
    int16_t * cSNR )
```

read values from last rx from transceiver

Parameters

out	<i>cRSSI</i>	Received Signal Strength Indicator in centy dBm
out	<i>cFPP</i>	First Power Path in centy dBm
out	<i>cSNR</i>	Signal to Noise Ratio in centy dB

Here is the call graph for this function:



Here is the caller graph for this function:



6.44.3.10 TRANSCEIVER_Send()

```
int TRANSCEIVER_Send (
    const void * buf,
    unsigned int len )
```

immediately send data via transceiver

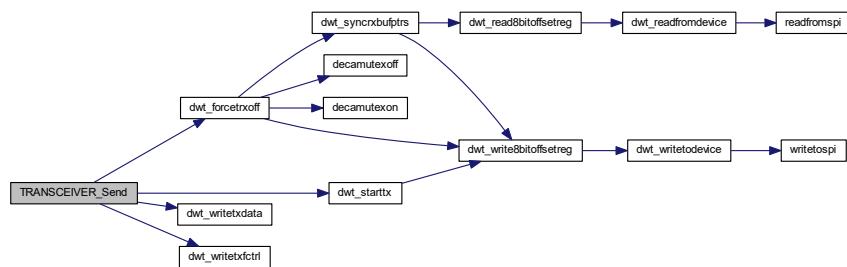
Parameters

in	<i>buf</i>	pointer to data buffer
in	<i>len</i>	data length in bytes

Returns

int 0 if success, error code otherwise

Here is the call graph for this function:

**6.44.3.11 TRANSCEIVER_SendRanging()**

```
int TRANSCEIVER_SendRanging (
    const void * buf,
    unsigned int len,
    uint8_t flags )
```

send ranging data via transceiver

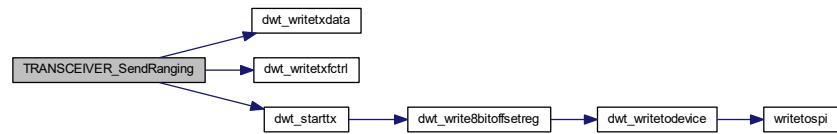
Parameters

in	<i>buf</i>	pointer to data buffer
in	<i>len</i>	data length in bytes
in	<i>flags</i>	flags for transceiver: <ul style="list-style-type: none">• DWT_START_TX_IMMEDIATE• DWT_START_TX_DELAYED• DWT_RESPONSE_EXPECTED

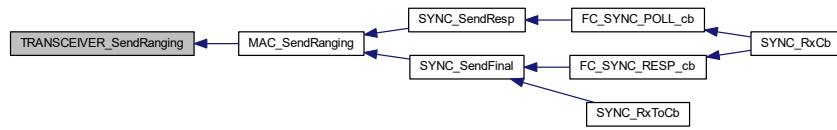
Returns

int 0 if success, error code otherwise

Here is the call graph for this function:



Here is the caller graph for this function:

**6.44.3.12 TRANSCEIVER_SetAddr()**

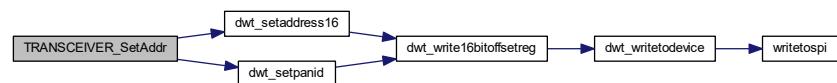
```
void TRANSCEIVER_SetAddr (
    pan_dev_addr_t pan_addr,
    dev_addr_t dev_addr )
```

set device address

Parameters

in	<i>pan_addr</i>	personal access network identifier
in	<i>dev_addr</i>	device address

Here is the call graph for this function:



6.44.3.13 TRANSCEIVER_SetCb()

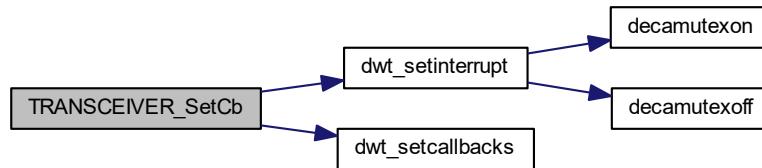
```
void TRANSCEIVER_SetCb (
    dwt_cb_t tx_cb,
    dwt_cb_t rx_cb,
    dwt_cb_t rxto_cb,
    dwt_cb_t rxerr_cb )
```

connect event callbacks

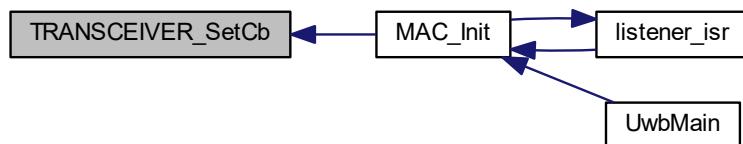
Parameters

in	<i>tx_cb</i>	transmitted message callback
in	<i>rx_cb</i>	received message callback
in	<i>rxto_cb</i>	timeout during receiving package callback
in	<i>rxerr_cb</i>	error during receiving package callback

Here is the call graph for this function:



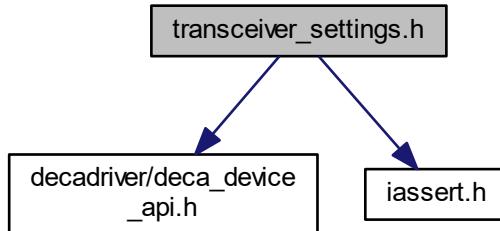
Here is the caller graph for this function:



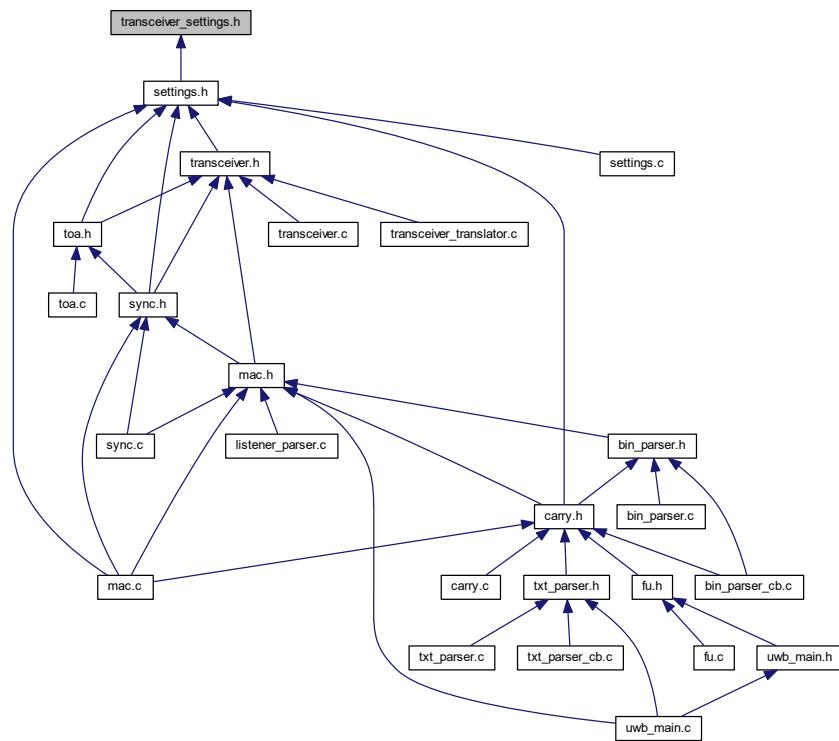
6.45 transceiver_settings.h File Reference

transceiver settings struct typedef with default values

```
#include "decadriver/deca_device_api.h"
#include "iassert.h"
Include dependency graph for transceiver_settings.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `transceiver_settings_t`
transceiver settings

Macros

- #define TRANSCEIVER_ASSERT(expr) IASSERT(expr)
- #define TRANSCEIVER_SETTINGS_DEF
default values for transceiver settings

6.45.1 Detailed Description

transceiver settings struct typedef with default values

Author

Karol Trzcinski

Date

2018-07-02

tutorial:

chan rf communication channel

- 2 - Fc 3993.6MHz BW 499.2MHz
- 5 - FC 6489.6MHz BW 499.2MHz

rxPAC Preamble Accumulation Count. This reports the number of symbols of preamble accumulated.

- DWT_PAC8 for preamble len 64..128
- DWT_PAC16 for preamble len 256..512
- DWT_PAC32 for preamble len 1024..4096

nsSFD non standard Start of Frame Delimiter

- 0 for preable len < X
- 1 for pramble len >= X

sfdTo Start of Frame Delimiter Timeout

- 0 for automatically calculate
- 1 + plen + sfd_len - pac

PGdly Transmitter Calibration – Pulse Generator Delay. This effectively sets the width of transmitted pulses effectively setting the output bandwidth. The value used here depends on the radio TX channel selected.

- 0 for autmatically calculate default value from datasheet

6.45.2 Macro Definition Documentation

6.45.2.1 TRANSCEIVER_ASSERT

```
#define TRANSCEIVER_ASSERT(
    expr ) IASSERT(expr)
```

6.45.2.2 TRANSCEIVER_SETTINGS_DEF

```
#define TRANSCEIVER_SETTINGS_DEF
```

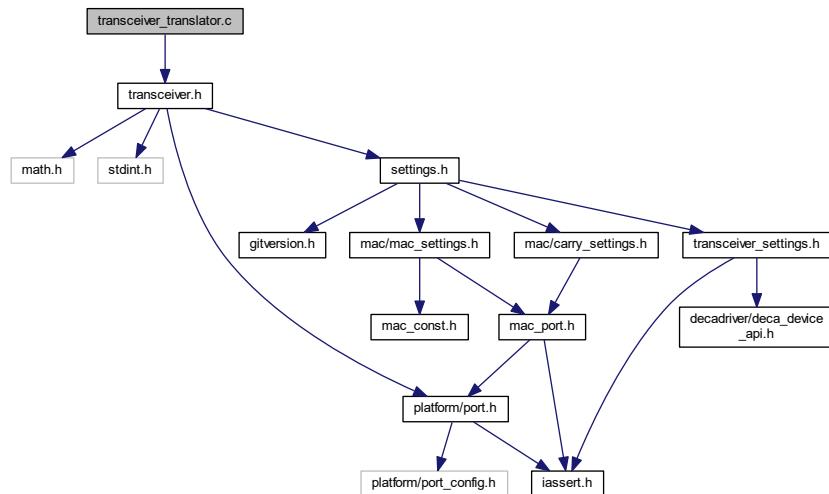
Value:

```
{
    .ant_dly_rx = 16436,
    .ant_dly_tx = 16436,
    .low_power_mode = 0,
    .dwt_config.chan = 5,
    .dwt_config.prf = DWT_PRF_64M,
    .dwt_config.txPreambLength = DWT_PLEN_128,
    .dwt_config.rxPAC = DWT_PAC8,
    .dwt_config.txCode = 10,
    .dwt_config.rxCode = 10,
    .dwt_config.nsSFD = 1,
    .dwt_config.dataRate = DWT_BR_6M8,
    .dwt_config.phrMode = DWT_PHRMODE_STD,
    .dwt_config.sfdTO = 0,
    .dwt_txconfig.PGdly = 0,
}
```

default values for transceiver settings

6.46 transceiver_translator.c File Reference

```
#include "transceiver.h"
Include dependency graph for transceiver_translator.c:
```



Functions

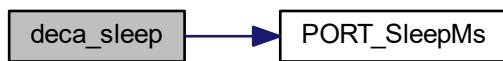
- void `deca_sleep` (unsigned int time_ms)

6.46.1 Function Documentation

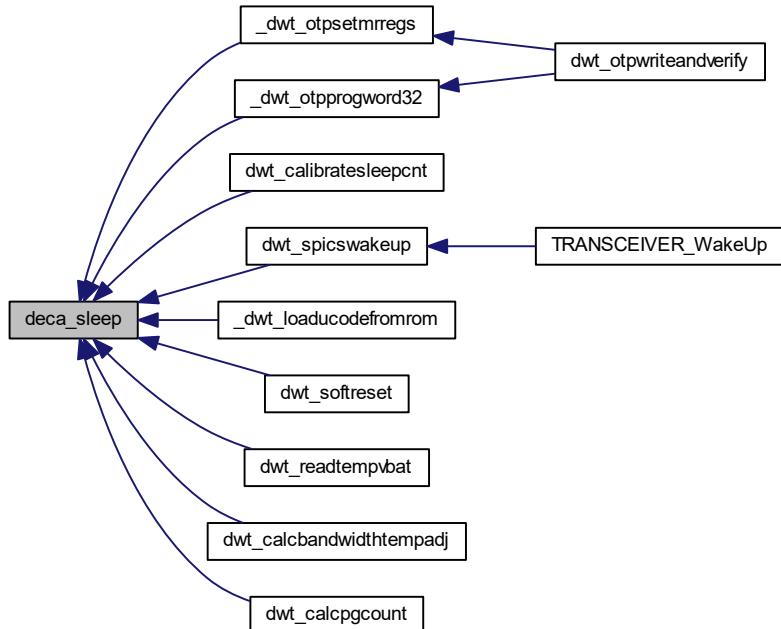
6.46.1.1 `deca_sleep()`

```
void deca_sleep (
    unsigned int time_ms )
```

Here is the call graph for this function:

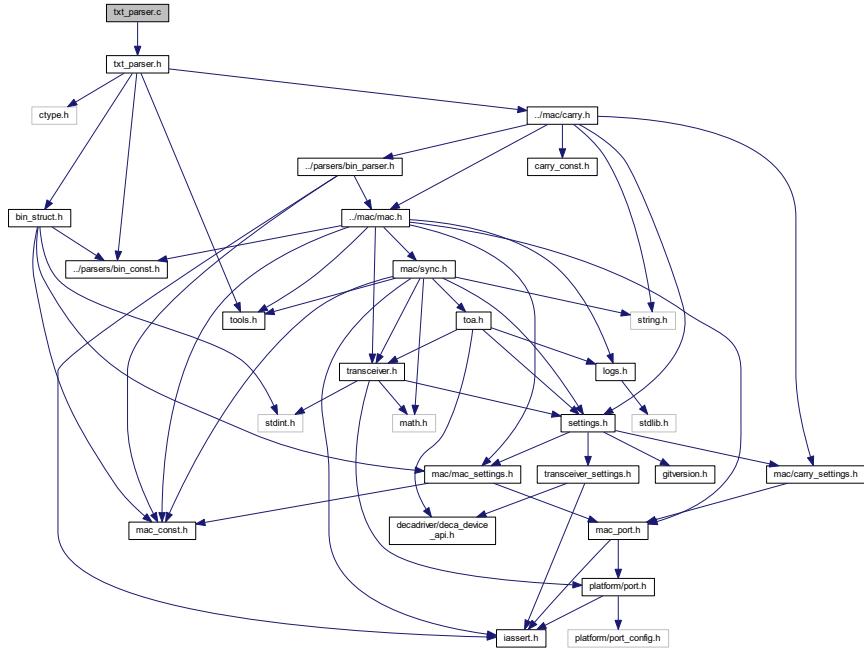


Here is the caller graph for this function:



6.47 txt_parser.c File Reference

```
#include "txt_parser.h"
Include dependency graph for txt_parser.c:
```



Functions

- `cchar * TXT_PointParamNumber (const txt_buf_t *buf, cchar *cmd, int num)`
 convert text number to int from circular buffer
- `int TXT_Atol (const txt_buf_t *buf, cchar *ptr, int base)`
 get parametr number value, form "cmd:value" from circular buffer
- `int TXT_GetParam (const txt_buf_t *buf, cchar *cmd, int base)`
 get parametr number value, form "cmd:value" from circular buffer
- `bool TXT_StartsWith (const txt_buf_t *buf, cchar *cmd)`
 check if command starts with in circular buffer
- `void TXT_Parse (const txt_buf_t *buf)`
- `void TXT_Input (const char *str, int len)`
 take input to text parser circular buffer
- `void TXT_Control ()`
 look for any new text message to parse

Variables

- `char _txt_buf_raw [256]`

6.47.1 Function Documentation

6.47.1.1 TXT_Atol()

```
int TXT_Atoi (
    const txt_buf_t * buf,
    cchar * ptr,
    int base )
```

convert text number to int from circular buffer

Parameters

in	<i>buf</i>	buffer data, especially buf->end and buf->start
in	<i>ptr</i>	pointer to number in text
in	<i>base</i>	number base, eg. 10 or 16

Returns

int number value or 0

Here is the caller graph for this function:



6.47.1.2 TXT_Control()

```
void TXT_Control ( )
```

look for any new text message to parse

This function should be called frequently from main program loop to start parsing new text message. Here is the caller graph for this function:



6.47.1.3 TXT_GetParam()

```
int TXT_GetParam (
    const txt_buf_t * buf,
    cchar * cmd,
    int base )
```

get parametr number value, form "cmd:value" from circular buffer

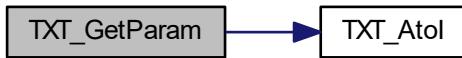
Parameters

in	<i>buf</i>	buffer to search in
in	<i>cmd</i>	parameter name
in	<i>base</i>	number base, eg. 10 or 16

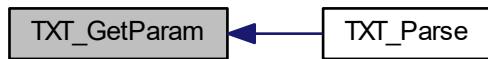
Returns

int parameter value or -1

Here is the call graph for this function:



Here is the caller graph for this function:



6.47.1.4 TXT_Input()

```
void TXT_Input (
    const char * str,
    int len )
```

take input to text parser circular buffer

Ignore \r and split by \n

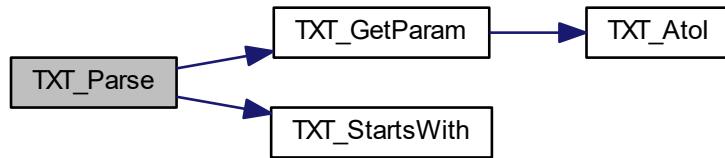
Parameters

in	<i>str</i>	pointer to new input string
in	<i>len</i>	length of new input string

6.47.1.5 *TXT_Parse()*

```
void TXT_Parse (
    const txt_buf_t * buf )
```

Here is the call graph for this function:

**6.47.1.6 *TXT_PointParamNumber()***

```
cchar* TXT_PointParamNumber (
    const txt_buf_t * buf,
    cchar * cmd,
    int num )
```

Parameters

in	<i>buf</i>	buffer data, especially buf->end and buf->start
in	<i>cmd</i>	pointer to place where start searching
in	<i>num</i>	number of parameter to point

Returns

`cchar*` pointer to parameter number `num` or 0

6.47.1.7 TXT_StartsWith()

```
bool TXT_StartsWith (
    const txt_buf_t * buf,
    cchar * cmd )
```

check if command starts with in circular buffer

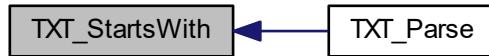
Parameters

in	<i>buf</i>	buffer to search in
in	<i>cmd</i>	command start message

Returns

true buf->cmd starts with cmd
 false buf->cmd doesn't starts with cmd

Here is the caller graph for this function:



6.47.2 Variable Documentation

6.47.2.1 _txt_buf_raw

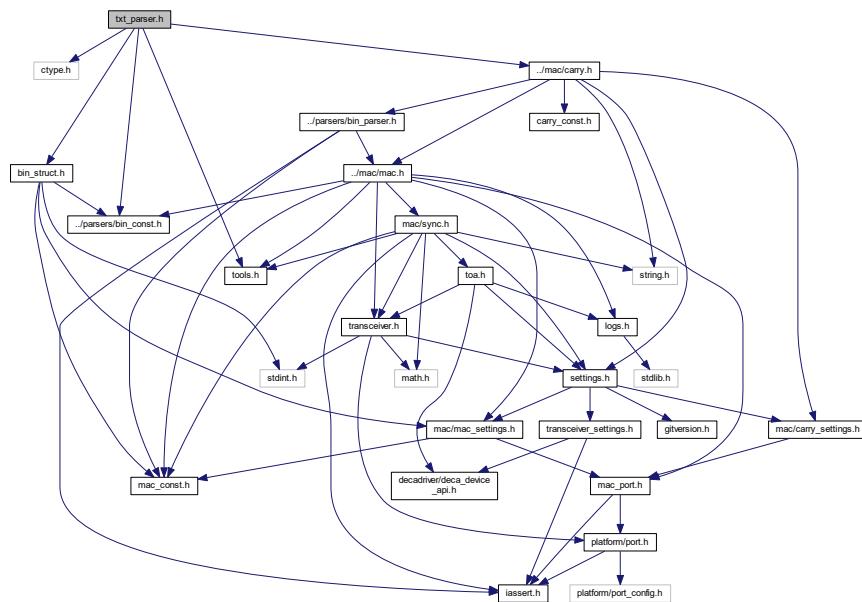
```
char _txt_buf_raw[256]
```

6.48 txt_parser.h File Reference

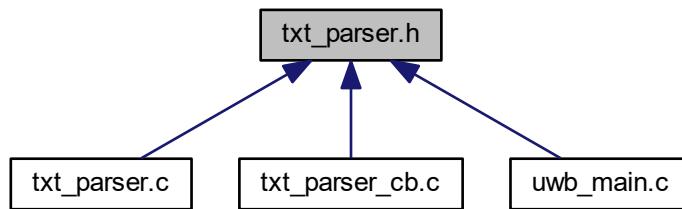
text parser engine

```
#include <ctype.h>
#include "tools.h"
#include "../mac/carry.h"
#include "bin_const.h"
```

```
#include "bin_struct.h"
Include dependency graph for txt_parser.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `txt_buf_t`
text parser engine circular buffer structure
- struct `txt_cb_t`
text parser engine callback struct

Typedefs

- `typedef const char cchar`
- `typedef void(* txt_parser_cb) (const txt_buf_t *buf, const prot_packet_info_t *info)`
text parser engine callback definition

Functions

- `cchar * TXT_PointParamNumber (const txt_buf_t *buf, cchar *cmd, int num)`
convert text number to int from circular buffer
- `int TXT_Atol (const txt_buf_t *buf, cchar *ptr, int base)`
get parametr number value, form "cmd:value" from circular buffer
- `int TXT_GetParam (const txt_buf_t *buf, cchar *cmd, int base)`
get parametr number value, form "cmd:value" from circular buffer
- `bool TXT_StartsWith (const txt_buf_t *buf, cchar *cmd)`
check if command starts with in circular buffer
- `void TXT_Input (const char *str, int len)`
take input to text parser circular buffer
- `void TXT_Control ()`
look for any new text message to parse

6.48.1 Detailed Description

text parser engine

Each message should be transformed from text to correct struct and function code. Struct with function code should be transformed to binary message and parsed by binary parser. Such mechanism helps to keep code consistent and help to avoid code doubling in text and binary parser.

Author

Karol Trzcinski

Date

2018-06-28

6.48.2 Typedef Documentation

6.48.2.1 cchar

```
typedef const char cchar
```

6.48.2.2 txt_parser_cb

```
typedef void(* txt_parser_cb) (const txt_buf_t *buf, const prot_packet_info_t *info)
```

text parser engine callback definition

6.48.3 Function Documentation

6.48.3.1 TXT_Atol()

```
int TXT_Atol (
    const txt_buf_t * buf,
    cchar * ptr,
    int base )
```

convert text number to int from circular buffer

Parameters

in	<i>buf</i>	buffer data, especially buf->end and buf->start
in	<i>ptr</i>	pointer to number in text
in	<i>base</i>	number base, eg. 10 or 16

Returns

int number value or 0

Here is the caller graph for this function:



6.48.3.2 TXT_Control()

void TXT_Control ()

look for any new text message to parse

This function should be called frequently from main program loop to start parsing new text message. Here is the caller graph for this function:



6.48.3.3 TXT_GetParam()

```

int TXT_GetParam (
    const txt_buf_t * buf,
    cchar * cmd,
    int base )
  
```

get parametr number value, form "cmd:value" from circular buffer

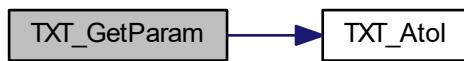
Parameters

in	<i>buf</i>	buffer to search in
in	<i>cmd</i>	parameter name
in	<i>base</i>	number base, eg. 10 or 16

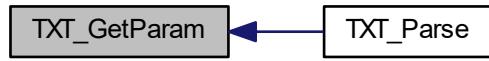
Returns

int parameter value or -1

Here is the call graph for this function:



Here is the caller graph for this function:

**6.48.3.4 TXT_Input()**

```
void TXT_Input (
    const char * str,
    int len )
```

take input to text parser circular buffer

Ignore \r and split by \n

Parameters

in	<i>str</i>	pointer to new input string
in	<i>len</i>	length of new input string

6.48.3.5 `TXT_PointParamNumber()`

```
cchar* TXT_PointParamNumber (
    const txt_buf_t * buf,
    cchar * cmd,
    int num )
```

Parameters

in	<i>buf</i>	buffer data, especially buf->end and buf->start
in	<i>cmd</i>	pointer to place where start searching
in	<i>num</i>	number of parameter to point

Returns

`cchar*` pointer to parameter number `num` or 0

6.48.3.6 `TXT_StartsWith()`

```
bool TXT_StartsWith (
    const txt_buf_t * buf,
    cchar * cmd )
```

check if command starts with in circular buffer

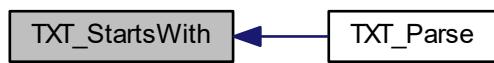
Parameters

in	<i>buf</i>	buffer to search in
in	<i>cmd</i>	command start message

Returns

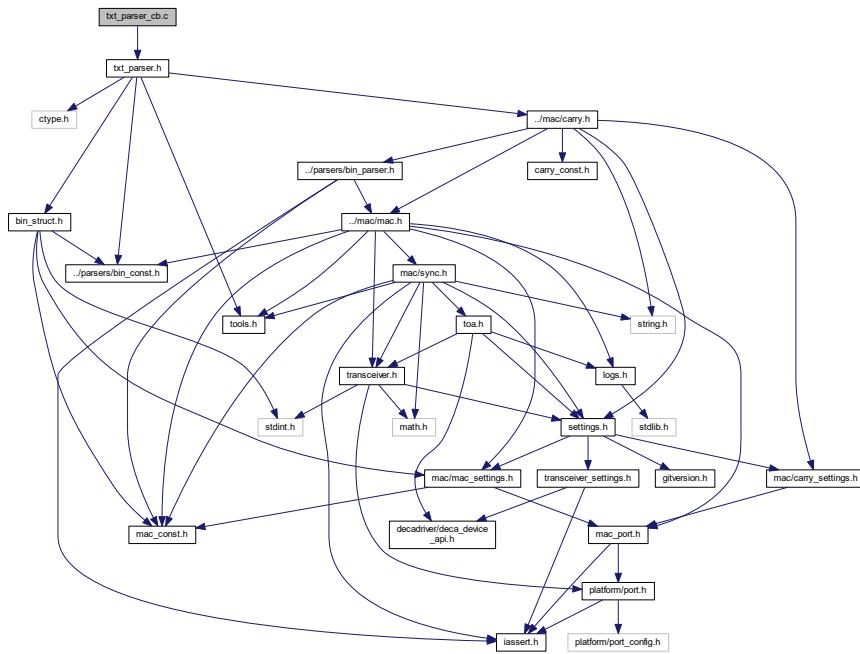
true `buf->cmd` starts with `cmd`
false `buf->cmd` doesn't starts with `cmd`

Here is the caller graph for this function:



6.49 txt_parser_cb.c File Reference

```
#include "txt_parser.h"
Include dependency graph for txt_parser_cb.c:
```



Functions

- void `_TXT_Finalize (mac_buf_t *buf, const prot_packet_info_t *info)`
- void `_TXT_Ask (const prot_packet_info_t *info, FC_t FC)`
- void `TXT_StatCb (const txt_buf_t *buf, const prot_packet_info_t *info)`
- void `TXT_VersionCb (const txt_buf_t *buf, const prot_packet_info_t *info)`
- void `TXT_HangCb (const txt_buf_t *buf, const prot_packet_info_t *info)`
- void `TXT_RFSet (const txt_buf_t *buf, const prot_packet_info_t *info)`
- void `TXT_TestCb (const txt_buf_t *buf, const prot_packet_info_t *info)`

Variables

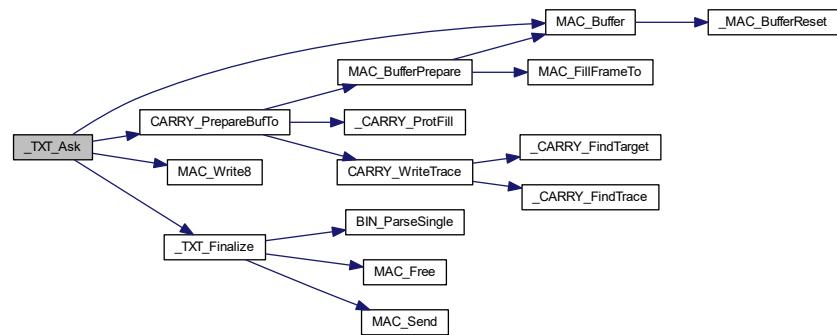
- const `txt_cb_t txt_cb_tab []`
- const int `txt_cb_len = sizeof(txt_cb_tab) / sizeof(*txt_cb_tab)`

6.49.1 Function Documentation

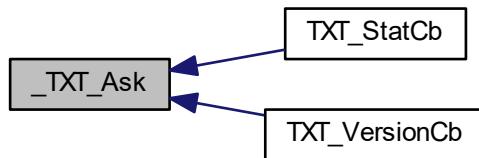
6.49.1.1 _TXT_Ask()

```
void _TXT_Ask (
    const prot_packet_info_t * info,
    FC_t FC )
```

Here is the call graph for this function:



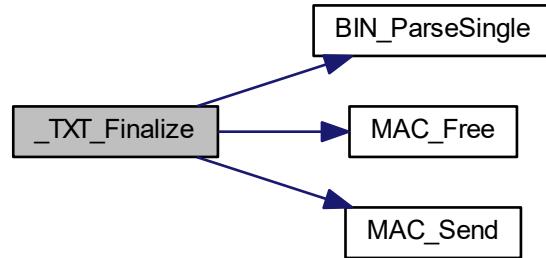
Here is the caller graph for this function:



6.49.1.2 _TXT_Finalize()

```
void _TXT_Finalize (
    mac_buf_t * buf,
    const prot_packet_info_t * info )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.49.1.3 `TXT_HangCb()`

```
void TXT_HangCb (
    const txt_buf_t * buf,
    const prot_packet_info_t * info )
```

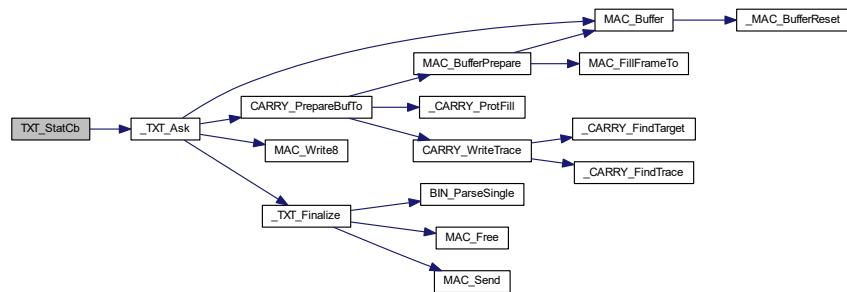
6.49.1.4 `TXT_RFSet()`

```
void TXT_RFSet (
    const txt_buf_t * buf,
    const prot_packet_info_t * info )
```

6.49.1.5 TXT_StatCb()

```
void TXT_StatCb (
    const txt_buf_t * buf,
    const prot_packet_info_t * info )
```

Here is the call graph for this function:



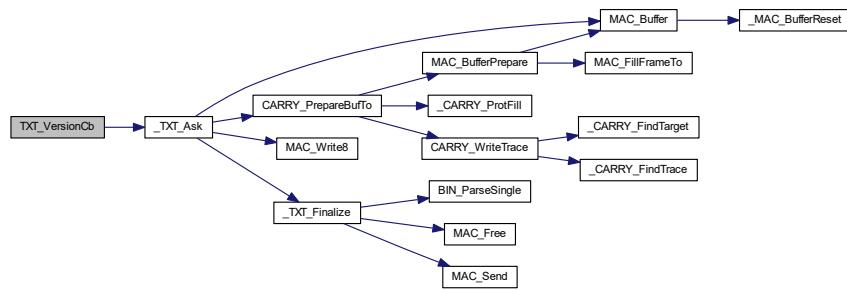
6.49.1.6 TXT_TestCb()

```
void TXT_TestCb (
    const txt_buf_t * buf,
    const prot_packet_info_t * info )
```

6.49.1.7 TXT_VersionCb()

```
void TXT_VersionCb (
    const txt_buf_t * buf,
    const prot_packet_info_t * info )
```

Here is the call graph for this function:



6.49.2 Variable Documentation

6.49.2.1 txt_cb_len

```
const int txt_cb_len = sizeof(txt_cb_tab) / sizeof(*txt_cb_tab)
```

6.49.2.2 txt_cb_tab

```
const txt_cb_t txt_cb_tab[ ]
```

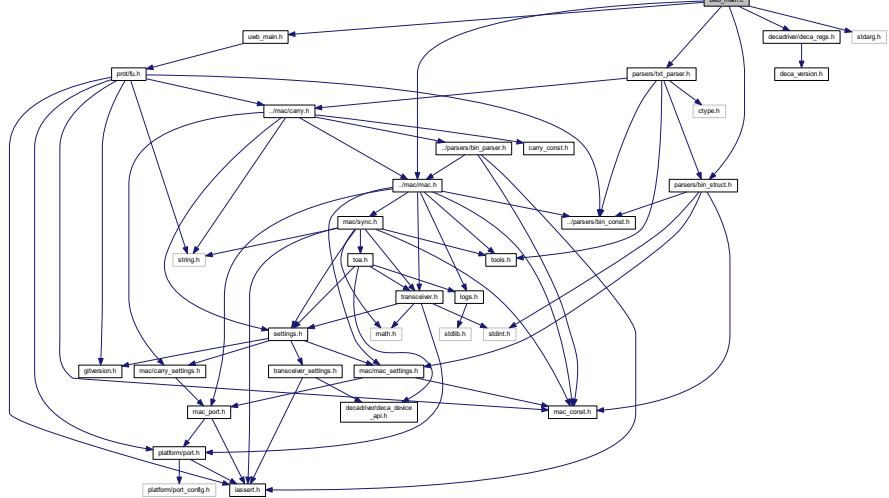
Initial value:

```
= {{"stat",  TXT_StatCb},  
     {"version",  TXT_VersionCb},  
     {"_hang",  TXT_HangCb},  
     {"rfset",  TXT_RFSet},  
     {"test",  TXT_TestCb}}
```

6.50 uwb_main.c File Reference

```
#include "uwb_main.h"
#include "mac/mac.h"
#include "parsers/bin_struct.h"
#include "parsers/txt_parser.h"
#include "decadriver/deca_regs.h"
#include "stdarg.h"
```

Include dependency graph for aws_main.h



Functions

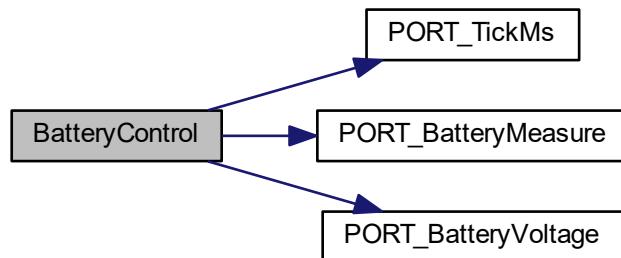
- void `SendTurnOnMessage ()`
- void `SendTurnOffMessage (uint8_t reason)`
- void `SendBeaconMessage ()`
- void `Desynchronize ()`
- void `CheckSleepMode ()`
- void `TurnOff ()`
- void `BatteryControl ()`
- void `diagnostic ()`
- void `BeaconSender ()`
- void `RangingControl ()`
- void `UwbMain ()`
initialize UWB core and go to infinity loop
- void `str_append (char *buf, size_t size, char *frm,...)`

6.50.1 Function Documentation

6.50.1.1 BatteryControl()

```
void BatteryControl ( )
```

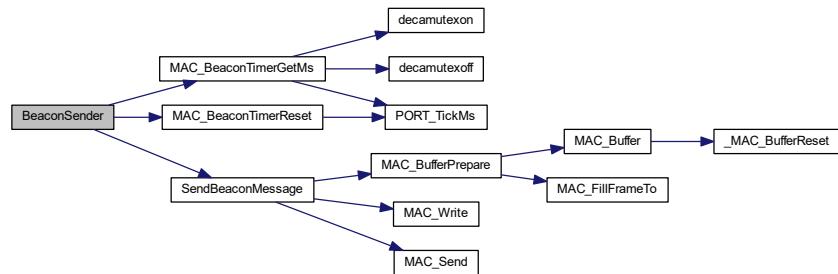
Here is the call graph for this function:



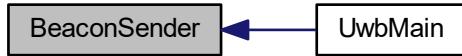
6.50.1.2 BeaconSender()

```
void BeaconSender ( )
```

Here is the call graph for this function:



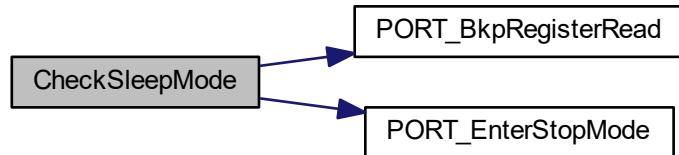
Here is the caller graph for this function:



6.50.1.3 CheckSleepMode()

```
void CheckSleepMode ( )
```

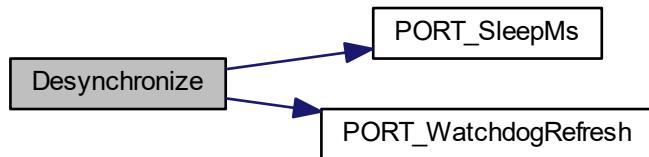
Here is the call graph for this function:



6.50.1.4 Desynchronize()

```
void Desynchronize ( )
```

Here is the call graph for this function:



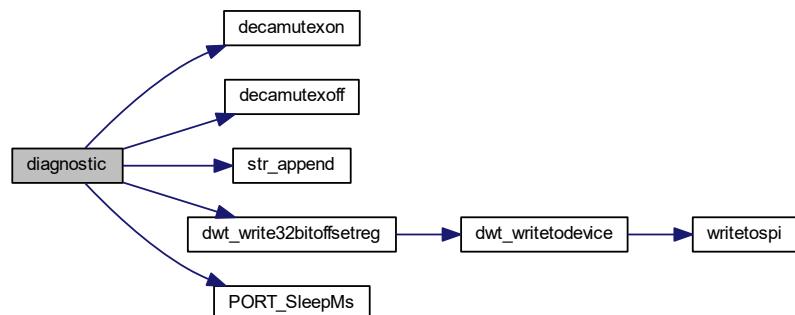
Here is the caller graph for this function:



6.50.1.5 diagnostic()

```
void diagnostic ( )
```

Here is the call graph for this function:



6.50.1.6 RangingControl()

```
void RangingControl ( )
```

Here is the call graph for this function:



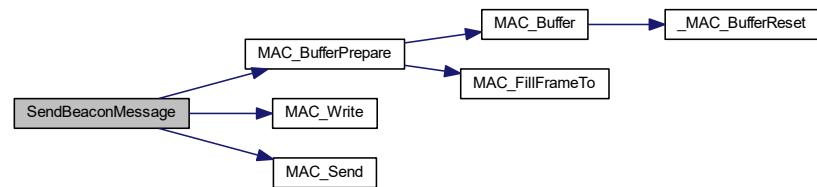
Here is the caller graph for this function:



6.50.1.7 SendBeaconMessage()

```
void SendBeaconMessage ( )
```

Here is the call graph for this function:



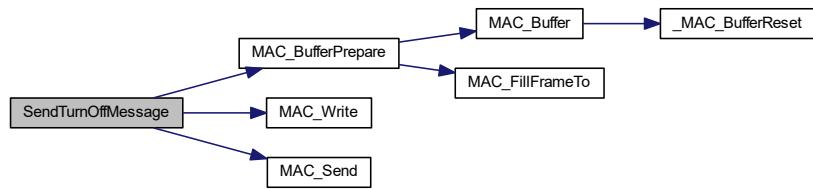
Here is the caller graph for this function:



6.50.1.8 SendTurnOffMessage()

```
void SendTurnOffMessage (
    uint8_t reason )
```

Here is the call graph for this function:



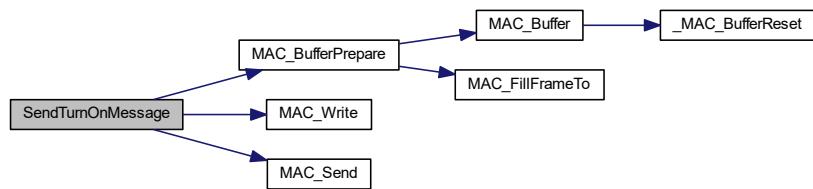
Here is the caller graph for this function:



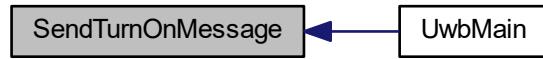
6.50.1.9 SendTurnOnMessage()

```
void SendTurnOnMessage ( )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.50.1.10 str_append()

```

void str_append (
    char * buf,
    size_t size,
    char * frm,
    ...
)
  
```

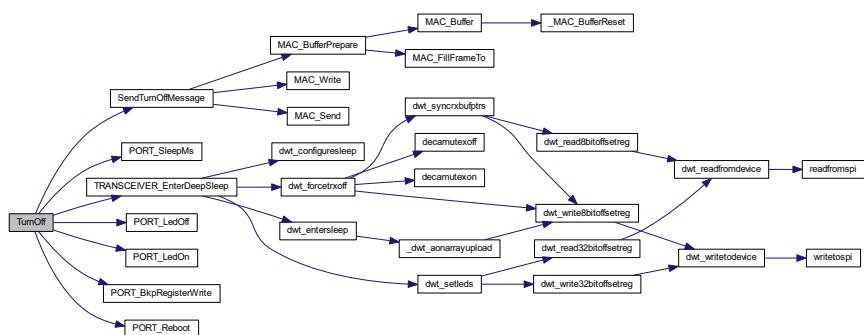
Here is the caller graph for this function:



6.50.1.11 TurnOff()

```
void TurnOff ( )
```

Here is the call graph for this function:



6.50.1.12 UwbMain()

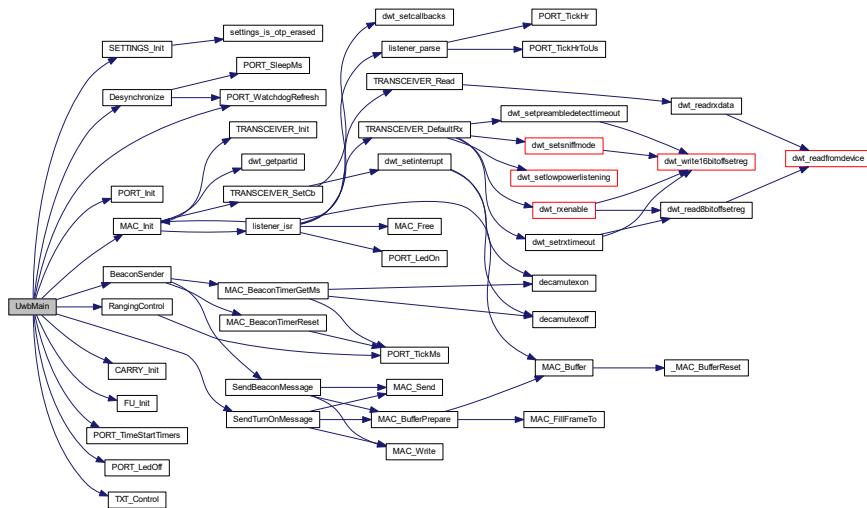
```
void UwbMain ( )
```

initialize UWB core and go to infinity loop

Returns

void

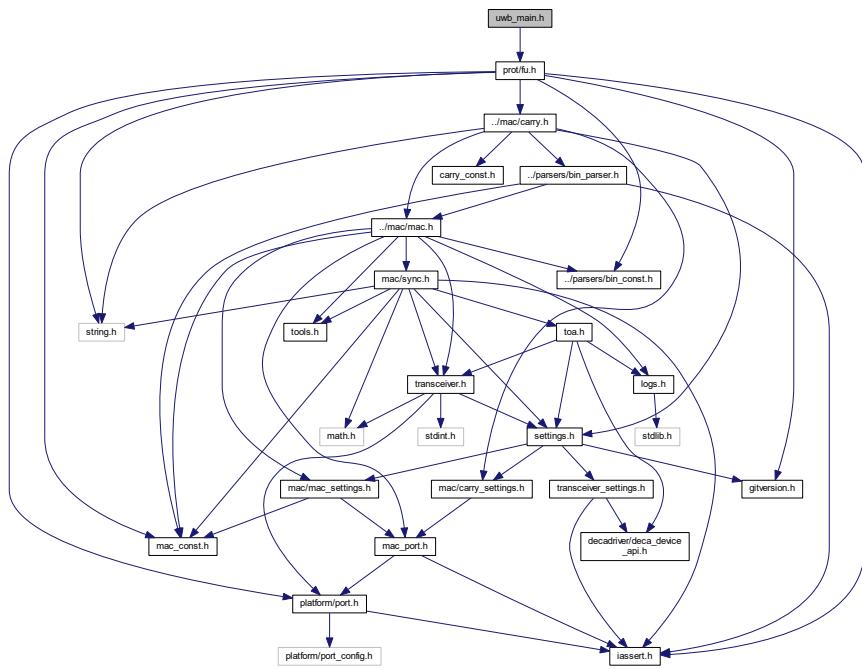
Here is the call graph for this function:



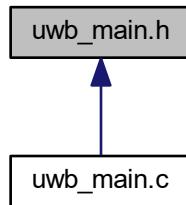
6.51 uwb_main.h File Reference

main core function

```
#include <prot/fu.h>
Include dependency graph for uwb_main.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- void `UwbMain ()`
initialize UWB core and go to infinity loop

6.51.1 Detailed Description

main core function

Author

Karol Trzcinski

Date

2018-06-28

6.51.2 Function Documentation

6.51.2.1 UwbMain()

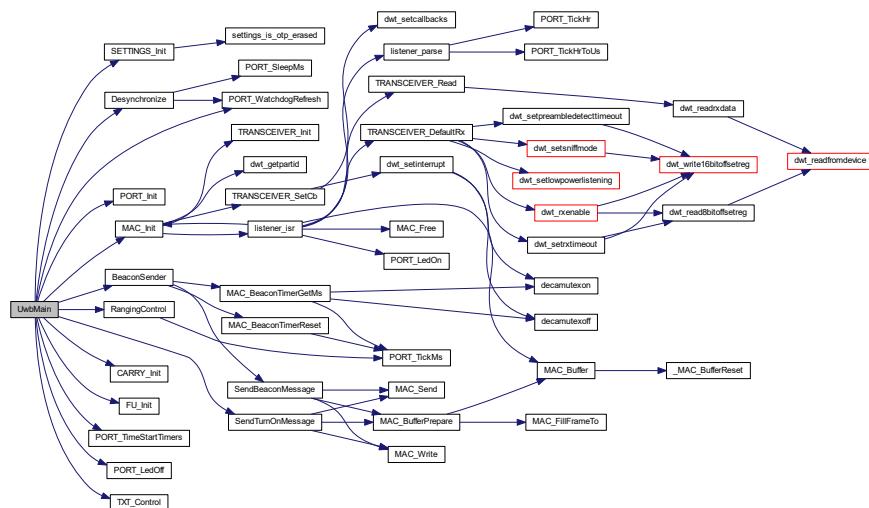
```
void UwbMain ( )
```

initialize UWB core and go to infinity loop

Returns

```
void
```

Here is the call graph for this function:



Index

_CARRY_FindTarget
 carry.c, 87
_CARRY_FindTrace
 carry.c, 88
_CARRY_ProtFill
 carry.c, 88
_DECA_INT16_
 deca_device_api.h, 162
 deca_types.h, 375
_DECA_INT32_
 deca_device_api.h, 162
 deca_types.h, 375
_DECA_INT8_
 deca_device_api.h, 162
 deca_types.h, 375
_DECA_UINT16_
 deca_device_api.h, 162
 deca_types.h, 376
_DECA_UINT32_
 deca_device_api.h, 162
 deca_types.h, 376
_DECA_UINT8_
 deca_device_api.h, 162
 deca_types.h, 376
_DEF_SLOT_CNT
 mac_settings.h, 446
_DEF_SLOT_SUM_TIME
 mac_settings.h, 446
_DEF_SLOT_TIME
 mac_settings.h, 446
_MAC_BufGetOldestToTx
 mac.c, 406
_MAC_BufferReset
 mac.c, 406
_TOA_GetRangeBias
 toa.c, 502
 toa_calib.c, 521
_TXT_Ask
 txt_parser_cb.c, 567
_TXT_Finalize
 txt_parser_cb.c, 568
_F_HASH_
 gitversion.h, 396
_F_MAJOR_
 gitversion.h, 396
_F_MINOR_
 gitversion.h, 396
_H_VERSION_
 settings.h, 475

_dwt_aonarrayupload
 deca_device.c, 109
_dwt_aonconfigupload
 deca_device.c, 109
_dwt_computetxpowersetting
 deca_device.c, 110
_dwt_configlde
 deca_device.c, 110
_dwt_disablesequencing
 deca_device.c, 111
_dwt_enableclocks
 deca_device.c, 111
_dwt_loaducodefromrom
 deca_device.c, 112
_dwt_otpprogword32
 deca_device.c, 113
_dwt_otpread
 deca_device.c, 113
_dwt_otpsetmrregs
 deca_device.c, 114
_settings_otp
 settings.c, 473
_txt_buf_raw
 txt_parser.c, 561

ACC_MEM_ID
 deca_regs.h, 251
ACC_MEM_LEN
 deca_regs.h, 251
ACK_RESP_T_ACK_TIM_MASK
 deca_regs.h, 251
ACK_RESP_T_ACK_TIM_OFFSET
 deca_regs.h, 251
ACK_RESP_T_ID
 deca_regs.h, 252
ACK_RESP_T_LEN
 deca_regs.h, 252
ACK_RESP_T_MASK
 deca_regs.h, 252
ACK_RESP_T_W4R_TIM_MASK
 deca_regs.h, 252
ACK_RESP_T_W4R_TIM_OFFSET
 deca_regs.h, 252
ACK_TIM_MASK
 deca_regs.h, 252
ADDR_ANCHOR_FLAG
 mac_const.h, 439
ADDR_BROADCAST
 mac_const.h, 439
AGC_CFG_STS_ID

deca_regs.h, 252
AGC_CTRL1_DIS_AM
 deca_regs.h, 253
AGC_CTRL1_LEN
 deca_regs.h, 253
AGC_CTRL1_MASK
 deca_regs.h, 253
AGC_CTRL1_OFFSET
 deca_regs.h, 253
AGC_CTRL_ID
 deca_regs.h, 253
AGC_CTRL_LEN
 deca_regs.h, 253
AGC_STAT1_EDG1_MASK
 deca_regs.h, 253
AGC_STAT1_EDG2_MASK
 deca_regs.h, 254
AGC_STAT1_LEN
 deca_regs.h, 254
AGC_STAT1_MASK
 deca_regs.h, 254
AGC_STAT1_OFFSET
 deca_regs.h, 254
AGC_TUNE1_16M
 deca_regs.h, 254
AGC_TUNE1_64M
 deca_regs.h, 254
AGC_TUNE1_LEN
 deca_regs.h, 254
AGC_TUNE1_MASK
 deca_regs.h, 255
AGC_TUNE1_OFFSET
 deca_regs.h, 255
AGC_TUNE2_LEN
 deca_regs.h, 255
AGC_TUNE2_MASK
 deca_regs.h, 255
AGC_TUNE2_OFFSET
 deca_regs.h, 255
AGC_TUNE2_VAL
 deca_regs.h, 255
AGC_TUNE3_LEN
 deca_regs.h, 255
AGC_TUNE3_MASK
 deca_regs.h, 255
AGC_TUNE3_OFFSET
 deca_regs.h, 256
AGC_TUNE3_VAL
 deca_regs.h, 256
ALL_UNUSED_IMPL_
 tools.h, 526
ALL_UNUSED_IMPL
 tools.h, 526
ALL_UNUSED
 tools.h, 526
AON_ADDR_LEN
 deca_regs.h, 256
AON_ADDR_LPOSC_CAL_0
deca_regs.h, 256
AON_ADDR_LPOSC_CAL_1
 deca_regs.h, 256
AON_ADDR_OFFSET
 deca_regs.h, 256
AON_CFG0_LEN
 deca_regs.h, 256
AON_CFG0_LPCLKDIVA_MASK
 deca_regs.h, 257
AON_CFG0_LPCLKDIVA_SHIFT
 deca_regs.h, 257
AON_CFG0_LPDIV_EN
 deca_regs.h, 257
AON_CFG0_OFFSET
 deca_regs.h, 257
AON_CFG0_SLEEP_EN
 deca_regs.h, 257
AON_CFG0_SLEEP_SHIFT
 deca_regs.h, 257
AON_CFG0_SLEEP_TIM_OFFSET
 deca_regs.h, 258
AON_CFG0_SLEEP_TIM
 deca_regs.h, 257
AON_CFG0_WAKE_CNT
 deca_regs.h, 258
AON_CFG0_WAKE_PIN
 deca_regs.h, 258
AON_CFG0_WAKE_SPI
 deca_regs.h, 258
AON_CFG1_LEN
 deca_regs.h, 258
AON_CFG1_LPOSC_CAL
 deca_regs.h, 258
AON_CFG1_MASK
 deca_regs.h, 258
AON_CFG1_OFFSET
 deca_regs.h, 259
AON_CFG1_SLEEP_CEN
 deca_regs.h, 259
AON_CFG1_SMXX
 deca_regs.h, 259
AON_CTRL_DCA_ENAB
 deca_regs.h, 259
AON_CTRL_DCA_READ
 deca_regs.h, 259
AON_CTRL_LEN
 deca_regs.h, 259
AON_CTRL_MASK
 deca_regs.h, 259
AON_CTRL_OFFSET
 deca_regs.h, 260
AON_CTRL_RESTORE
 deca_regs.h, 260
AON_CTRL_SAVE
 deca_regs.h, 260
AON_CTRL_UPL_CFG
 deca_regs.h, 260
AON_ID

deca_regs.h, 260
AON_LEN
 deca_regs.h, 260
AON_RDAT_LEN
 deca_regs.h, 260
AON_RDAT_OFFSET
 deca_regs.h, 261
AON_WCFG_LEN
 deca_regs.h, 261
AON_WCFG_MASK
 deca_regs.h, 261
AON_WCFG_OFFSET
 deca_regs.h, 261
AON_WCFG_ONW_L64P
 deca_regs.h, 261
AON_WCFG_ONW_LDC
 deca_regs.h, 261
AON_WCFG_ONW_LEUI
 deca_regs.h, 261
AON_WCFG_ONW_LLDE
 deca_regs.h, 262
AON_WCFG_ONW_LLDO
 deca_regs.h, 262
AON_WCFG_ONW_RADC
 deca_regs.h, 262
AON_WCFG_ONW_RX
 deca_regs.h, 262
AON_WCFG_PRES_SLEEP
 deca_regs.h, 262
ARFE
 dwt_deviceentcnts_t, 18
addr
 carry_target_t, 12
 mac_settings_t, 48
 sync_neighbour_t, 58
addr_tab
 toa_core_t, 61
agc_cfg_struct, 9
 lo32, 9
 target, 9
agc_config
 deca_param_types.h, 224
 deca_params_init.c, 227
anc_in_poll_cnt
 toa_core_t, 61
ant_dly_rx
 transceiver_settings_t, 64
ant_dly_tx
 transceiver_settings_t, 64
B20_SIGN_EXTEND_MASK
 deca_device.c, 106
B20_SIGN_EXTEND_TEST
 deca_device.c, 106
BASE64_Decode
 base64.c, 69
 base64.h, 72
BASE64_Encode
 base64.c, 70
base64.h, 72
 BIN_ASSERT
 bin_parser.h, 79
 BIN_Parse
 bin_parser.c, 76
 bin_parser.h, 80
 BIN_ParseSingle
 bin_parser.c, 77
 bin_parser.h, 80
 BIN_SEND_RESP
 bin_parser_cb.c, 82
 BOOSTNORM_MASK
 deca_regs.h, 262
 BOOSTP125_MASK
 deca_regs.h, 262
 BOOSTP250_MASK
 deca_regs.h, 263
 BOOSTP500_MASK
 deca_regs.h, 263
bad_len_msg
 sync.c, 492
base64.c, 69
 BASE64_Decode, 69
 BASE64_Encode, 70
 BASE64_TextSize, 71
base64.h, 71
 BASE64_Decode, 72
 BASE64_Encode, 72
 BASE64_TextSize, 73
battery_mV
 FC_STAT_s, 28
BatteryControl
 uwb_main.c, 572
beacon_timer_timestamp
 mac_instance_t, 45
BeaconSender
 uwb_main.c, 572
bin_const.h, 73
 FC_t, 74
bin_parser.c, 75
 BIN_Parse, 76
 BIN_ParseSingle, 77
bin_parser.h, 77
 BIN_ASSERT, 79
 BIN_Parse, 80
 BIN_ParseSingle, 80
 prot_parser_cb, 79
bin_parser_cb.c, 81
 BIN_SEND_RESP, 82
 FC_BEACON_cb, 83
 FC_STAT_ASK_cb, 83
 FC_TURN_ON_cb, 84
 FC_VERSION_ASK_cb, 84
 prot_cb_len, 84
 prot_cb_tab, 85

bin_struct.h, 85
 boot_reserved
 settings_version_t, 55
 buf
 mac_buf_t, 43
 mac_instance_t, 46
 buf_get_ind
 mac_instance_t, 46
 CARRY_ASSERT
 carry_settings.h, 102
 CARRY_FLAG_ACK_REQ
 carry_const.h, 98
 CARRY_FLAG_REROUTE
 carry_const.h, 98
 CARRY_FLAG_TARGET_DEV
 carry_const.h, 98
 CARRY_FLAG_TARGET_MASK
 carry_const.h, 99
 CARRY_FLAG_TARGET_SERVER
 carry_const.h, 99
 CARRY_FLAG_TARGET_SINK
 carry_const.h, 99
 CARRY_GetBufTo
 carry.h, 94
 CARRY_HEAD_MIN_LEN
 carry.h, 94
 CARRY_HOPS_NUM_MASK
 carry_const.h, 99
 CARRY_Init
 carry.c, 88
 carry.h, 94
 CARRY_MAX_HOPS
 carry_const.h, 99
 CARRY_MAX_TARGETS
 carry_settings.h, 102
 CARRY_MAX_TRACE
 carry_settings.h, 102
 CARRY_ParseMessage
 carry.c, 89
 carry.h, 95
 CARRY_PrepBufTo
 carry.c, 90
 carry.h, 96
 CARRY_SETTINGS_DEF
 carry_settings.h, 102
 CARRY_WriteTrace
 carry.c, 90
 carry.h, 96
 CHAN_CTRL_DWSFD_SHIFT
 deca_regs.h, 263
 CHAN_CTRL_DWSFD
 deca_regs.h, 263
 CHAN_CTRL_ID
 deca_regs.h, 263
 CHAN_CTRL_LEN
 deca_regs.h, 263
 CHAN_CTRL_MASK
 deca_regs.h, 263
 CHAN_CTRL_RNSSFD_SHIFT
 deca_regs.h, 264
 CHAN_CTRL_RNSSFD
 deca_regs.h, 264
 CHAN_CTRL_RX_CHAN_MASK
 deca_regs.h, 264
 CHAN_CTRL_RX_CHAN_SHIFT
 deca_regs.h, 264
 CHAN_CTRL_RX_PCOD_MASK
 deca_regs.h, 264
 CHAN_CTRL_RX_PCOD_SHIFT
 deca_regs.h, 264
 CHAN_CTRL_RXFPRF_16
 deca_regs.h, 264
 CHAN_CTRL_RXFPRF_4
 deca_regs.h, 265
 CHAN_CTRL_RXFPRF_64
 deca_regs.h, 265
 CHAN_CTRL_RXFPRF_MASK
 deca_regs.h, 265
 CHAN_CTRL_RXFPRF_SHIFT
 deca_regs.h, 265
 CHAN_CTRL_TNSSFD_SHIFT
 deca_regs.h, 265
 CHAN_CTRL_TNSSFD
 deca_regs.h, 265
 CHAN_CTRL_TX_CHAN_MASK
 deca_regs.h, 265
 CHAN_CTRL_TX_CHAN_SHIFT
 deca_regs.h, 266
 CHAN_CTRL_TX_PCOD_MASK
 deca_regs.h, 266
 CHAN_CTRL_TX_PCOD_SHIFT
 deca_regs.h, 266
 CIR_MXG_MASK
 deca_regs.h, 266
 CIR_MXG_SHIFT
 deca_regs.h, 266
 CM_OFFSET_16M_NB
 toa_calib.c, 520
 CM_OFFSET_16M_WB
 toa_calib.c, 520
 CM_OFFSET_64M_NB
 toa_calib.c, 520
 CM_OFFSET_64M_WB
 toa_calib.c, 520
 CRCB
 dwt_deviceentcnts_t, 18
 CRCG
 dwt_deviceentcnts_t, 19
 carry
 carry.c, 91
 prot_packet_info_t, 51
 settings_t, 54
 carry.c, 87
 _CARRY_FindTarget, 87
 _CARRY_FindTrace, 88
 _CARRY_ProtFill, 88

CARRY_Init, 88
CARRY_ParseMessage, 89
CARRY_PrepareBufTo, 90
CARRY_WriteTrace, 90
carry, 91
carry.h, 92
 CARRY_GetBufTo, 94
 CARRY_HEAD_MIN_LEN, 94
 CARRY_Init, 94
 CARRY_ParseMessage, 95
 CARRY_PrepareBufTo, 96
 CARRY_WriteTrace, 96
carry_const.h, 98
 CARRY_FLAG_ACK_REQ, 98
 CARRY_FLAG_REROUTE, 98
 CARRY_FLAG_TARGET_DEV, 98
 CARRY_FLAG_TARGET_MASK, 99
 CARRY_FLAG_TARGET_SERVER, 99
 CARRY_FLAG_TARGET_SINK, 99
 CARRY_HOPS_NUM_MASK, 99
 CARRY_MAX_HOPS, 99
carry_instance_t, 10
 isConnectedToServer, 10
 target, 10
 toSinkId, 11
carry_settings.h, 100
 CARRY_ASSERT, 102
 CARRY_MAX_TARGETS, 102
 CARRY_MAX_TRACE, 102
 CARRY_SETTINGS_DEF, 102
carry_settings_t, 11
 trace_max_fail_cnt, 11
 trace_max_inactive_time, 11
carry_target_t, 12
 addr, 12
 last_update_time, 13
 trace, 13
carry_trace_t, 13
 fail_cnt, 14
 last_update_time, 14
 pass_cnt, 14
 path, 14
 path_len, 14
cb
 prot_cb_t, 50
 txt_cb_t, 67
cbData
 dwt_local_data_t, 21
cbRxErr
 dwt_local_data_t, 21
cbRxOk
 dwt_local_data_t, 21
cbRxTo
 dwt_local_data_t, 21
cbTxDone
 dwt_local_data_t, 22
cchar
 txt_parser.h, 563
chan
 dwt_config_t, 16
chan_idx
 deca_param_types.h, 224
 deca_params_init.c, 227
chan_idxnb
 toa_calib.c, 522
chan_idxwb
 toa_calib.c, 522
CheckSleepMode
 uwb_main.c, 573
cmd
 txt_buf_t, 66
 txt_cb_t, 67
cnt
 txt_buf_t, 66
control
 mac_buf_t, 43
DA_ATTN_STEP
 deca_param_types.h, 222
DBG
 port.h, 450
DECREMENT_CYCLE
 tools.h, 526
DECREMENT_MOD
 tools.h, 526
DEF_SETTINGS
 settings.h, 475
DEV_ID_ID
 deca_regs.h, 266
DEV_ID_LEN
 deca_regs.h, 266
DEV_ID_MODEL_MASK
 deca_regs.h, 267
DEV_ID_REV_MASK
 deca_regs.h, 267
DEV_ID_RIDTAG_MASK
 deca_regs.h, 267
DEV_ID_VER_MASK
 deca_regs.h, 267
DIAG_TMC_LEN
 deca_regs.h, 267
DIAG_TMC_MASK
 deca_regs.h, 267
DIAG_TMC_TX_PSTM
 deca_regs.h, 268
DIG_DIAG_ID
 deca_regs.h, 268
DIG_DIAG_LEN
 deca_regs.h, 268
dPtr
 mac_buf_t, 43
DRX_CARRIER_INT_LEN
 deca_regs.h, 268
DRX_CARRIER_INT_MASK
 deca_regs.h, 268

DRX_CARRIER_INT_OFFSET
 deca_regs.h, 268

DRX_CONF_ID
 deca_regs.h, 268

DRX_CONF_LEN
 deca_regs.h, 269

DRX_PRETOC_LEN
 deca_regs.h, 269

DRX_PRETOC_MASK
 deca_regs.h, 269

DRX_PRETOC_OFFSET
 deca_regs.h, 269

DRX_SFDTOC_LEN
 deca_regs.h, 269

DRX_SFDTOC_MASK
 deca_regs.h, 269

DRX_SFDTOC_OFFSET
 deca_regs.h, 269

DRX_TUNE0b_110K_NSTD
 deca_regs.h, 270

DRX_TUNE0b_110K_STD
 deca_regs.h, 270

DRX_TUNE0b_6M8_NSTD
 deca_regs.h, 270

DRX_TUNE0b_6M8_STD
 deca_regs.h, 270

DRX_TUNE0b_850K_NSTD
 deca_regs.h, 270

DRX_TUNE0b_850K_STD
 deca_regs.h, 270

DRX_TUNE0b_LEN
 deca_regs.h, 270

DRX_TUNE0b_MASK
 deca_regs.h, 270

DRX_TUNE0b_OFFSET
 deca_regs.h, 271

DRX_TUNE1a_LEN
 deca_regs.h, 271

DRX_TUNE1a_MASK
 deca_regs.h, 271

DRX_TUNE1a_OFFSET
 deca_regs.h, 271

DRX_TUNE1a_PRF16
 deca_regs.h, 271

DRX_TUNE1a_PRF64
 deca_regs.h, 271

DRX_TUNE1b_110K
 deca_regs.h, 271

DRX_TUNE1b_6M8_PRE64
 deca_regs.h, 271

DRX_TUNE1b_850K_6M8
 deca_regs.h, 272

DRX_TUNE1b_LEN
 deca_regs.h, 272

DRX_TUNE1b_MASK
 deca_regs.h, 272

DRX_TUNE1b_OFFSET
 deca_regs.h, 272

DRX_TUNE2_LEN
 deca_regs.h, 272

DRX_TUNE2_MASK
 deca_regs.h, 272

DRX_TUNE2_OFFSET
 deca_regs.h, 272

DRX_TUNE2_PRF16_PAC16
 deca_regs.h, 272

DRX_TUNE2_PRF16_PAC32
 deca_regs.h, 273

DRX_TUNE2_PRF16_PAC64
 deca_regs.h, 273

DRX_TUNE2_PRF16_PAC8
 deca_regs.h, 273

DRX_TUNE2_PRF64_PAC16
 deca_regs.h, 273

DRX_TUNE2_PRF64_PAC32
 deca_regs.h, 273

DRX_TUNE2_PRF64_PAC64
 deca_regs.h, 273

DRX_TUNE2_PRF64_PAC8
 deca_regs.h, 273

DRX_TUNE4H_LEN
 deca_regs.h, 273

DRX_TUNE4H_MASK
 deca_regs.h, 274

DRX_TUNE4H_OFFSET
 deca_regs.h, 274

DRX_TUNE4H_PRE128PLUS
 deca_regs.h, 274

DRX_TUNE4H_PRE64
 deca_regs.h, 274

DW1000_DEVICE_DRIVER_VER_STRING
 deca_version.h, 378

DW1000_DRIVER_VERSION
 deca_version.h, 378

DW_NS_SFD_LEN_110K
 deca_regs.h, 274

DW_NS_SFD_LEN_6M8
 deca_regs.h, 274

DW_NS_SFD_LEN_850K
 deca_regs.h, 274

DWT_BR_110K
 deca_device_api.h, 163

DWT_BR_6M8
 deca_device_api.h, 163

DWT_BR_850K
 deca_device_api.h, 163

DWT_CB_DATA_RX_FLAG_RNG
 deca_device_api.h, 163

DWT_CONFIG
 deca_device_api.h, 163

DWT_DEVICE_ID
 deca_device_api.h, 163

DWT_ERROR
 deca_device_api.h, 163

DWT_FF_ACK_EN
 deca_device_api.h, 164

DWT_FF_BEACON_EN
 deca_device_api.h, 164

DWT_FF_COORD_EN
 deca_device_api.h, 164

DWT_FF_DATA_EN
 deca_device_api.h, 164

DWT_FF_MAC_EN
 deca_device_api.h, 164

DWT_FF_NOTYPE_EN
 deca_device_api.h, 164

DWT_FF_RSVD_EN
 deca_device_api.h, 164

DWT_IDLE_ON_DLY_ERR
 deca_device_api.h, 164

DWT_INT_ARFE
 deca_device_api.h, 165

DWT_INT_LDED
 deca_device_api.h, 165

DWT_INT_RFCE
 deca_device_api.h, 165

DWT_INT_RFCG
 deca_device_api.h, 165

DWT_INT_RFSL
 deca_device_api.h, 165

DWT_INT_RFTO
 deca_device_api.h, 165

DWT_INT_RPHE
 deca_device_api.h, 165

DWT_INT_RXOVR
 deca_device_api.h, 165

DWT_INT_RXPTO
 deca_device_api.h, 166

DWT_INT_SFDT
 deca_device_api.h, 166

DWT_INT_TFRS
 deca_device_api.h, 166

DWT_LEDS_DISABLE
 deca_device_api.h, 166

DWT_LEDS_ENABLE
 deca_device_api.h, 166

DWT_LEDS_INIT_BLINK
 deca_device_api.h, 166

DWT_LOADEUI
 deca_device_api.h, 166

DWT_LOADNONE
 deca_device_api.h, 166

DWT_LOADOPSET
 deca_device_api.h, 167

DWT_LOADUCODE
 deca_device_api.h, 167

DWT_NO_SYNC_PTRS
 deca_device_api.h, 167

DWT_NUM_DW_DEV
 deca_device_api.h, 167

DWT_OPSET_64LEN
 deca_device_api.h, 167

DWT_OPSET_DEFLT
 deca_device_api.h, 167

DWT_OPSET_TIGHT
 deca_device_api.h, 167

DWT_PAC16
 deca_device_api.h, 167

DWT_PAC32
 deca_device_api.h, 168

DWT_PAC64
 deca_device_api.h, 168

DWT_PAC8
 deca_device_api.h, 168

DWT_PHRMODE_EXT
 deca_device_api.h, 168

DWT_PHRMODE_STD
 deca_device_api.h, 168

DWT_PLEN_1024
 deca_device_api.h, 168

DWT_PLEN_128
 deca_device_api.h, 168

DWT_PLEN_1536
 deca_device_api.h, 169

DWT_PLEN_2048
 deca_device_api.h, 169

DWT_PLEN_256
 deca_device_api.h, 169

DWT_PLEN_4096
 deca_device_api.h, 169

DWT_PLEN_512
 deca_device_api.h, 169

DWT_PLEN_64
 deca_device_api.h, 169

DWT_PRESRV_SLEEP
 deca_device_api.h, 169

DWT_PRF_16M
 deca_device_api.h, 170

DWT_PRF_64M
 deca_device_api.h, 170

DWT_RESPONSE_EXPECTED
 deca_device_api.h, 170

DWT_RX_EN
 deca_device_api.h, 170

DWT_SFDTOC_DEF
 deca_device_api.h, 170

DWT_SLP_EN
 deca_device_api.h, 170

DWT_START_RX_DELAYED
 deca_device_api.h, 171

DWT_START_RX_IMMEDIATE
 deca_device_api.h, 171

DWT_START_TX_DELAYED
 deca_device_api.h, 171

DWT_START_TX_IMMEDIATE
 deca_device_api.h, 171

DWT_SUCCESS
 deca_device_api.h, 171

DWT_TANDV
 deca_device_api.h, 171

DWT_TIME_UNITS
 deca_device_api.h, 171

DWT_WAKE_CS
 deca_device_api.h, 172

DWT_WAKE_SLP_CNT
 deca_device_api.h, 172

DWT_WAKE_WK
 deca_device_api.h, 172

DWT_XTAL_EN
 deca_device_api.h, 172

DX_TIME_ID
 deca_regs.h, 274

DX_TIME_LEN
 deca_regs.h, 275

data
 FU_prot, 39
 mac_buf_t, 43

dataRate
 dwt_config_t, 16

datalength
 dwt_cb_data_t, 15

dblbuffer
 dwt_local_data_t, 22

deca_device.c, 103
 _dwt_aonarrayupload, 109
 _dwt_aonconfigupload, 109
 _dwt_computetxpowersetting, 110
 _dwt_configidle, 110
 _dwt_disablesequencing, 111
 _dwt_enableclocks, 111
 _dwt_loaducodefromrom, 112
 _dwt_otpprogword32, 113
 _dwt_otpread, 113
 _dwt_otpsetmrregs, 114
 B20_SIGN_EXTEND_MASK, 106
 B20_SIGN_EXTEND_TEST, 106
 dwt_calcbandwidthtempadj, 115
 dwt_calcpgcount, 115
 dwt_calcpowertempadj, 116
 dwt_calibratesleepcnt, 116
 dwt_checkirq, 116
 dwt_configcontinuousframemode, 117
 dwt_configcwmode, 117
 dwt_configeventcounters, 117
 dwt_configure, 118
 dwt_configuresleep, 118
 dwt_configuresleepcnt, 118
 dwt_configuretxrf, 119
 dwt_enableautoack, 119
 dwt_enableframefilter, 119
 dwt_entersleep, 120
 dwt_entersleepaftertx, 120
 dwt_forcetrxoff, 121
 dwt_geteui, 121
 dwt_getinitxtaltrim, 122
 dwt_getlotid, 122
 dwt_getpartid, 122
 dwt_initialise, 123
 dwt_isr, 123
 dwt_loadopsettabfromotp, 123
 dwt_lowpowerlistenisr, 124
 dwt_otpread, 124
 dwt_otprevision, 124
 dwt_otpwriteandverify, 125
 dwt_read16bitoffsetreg, 125
 dwt_read32bitoffsetreg, 126
 dwt_read8bitoffsetreg, 126
 dwt_readaccdata, 127
 dwt_readcarrierintegrator, 127
 dwt_readdevid, 128
 dwt_readdiagnostics, 128
 dwt_readeventcounters, 129
 dwt_readfromdevice, 129
 dwt_readrxdata, 131
 dwt_readrxtimestamp, 132
 dwt_readrxtimestampphi32, 132
 dwt_readrxtimestamplo32, 133
 dwt_readsystime, 133
 dwt_readsystimestampphi32, 134
 dwt_readtempvbat, 134
 dwt_readtxtimestamp, 134
 dwt_readtxtimestampphi32, 135
 dwt_readtxtimestamplo32, 135
 dwt_readwakeuptemp, 136
 dwt_readwakeupvbat, 136
 dwt_rxenable, 136
 dwt_rxreset, 137
 dwt_setaddress16, 138
 dwt_setcallbacks, 138
 dwt_setdblrxbuffmode, 139
 dwt_setdelayedtrxtime, 139
 dwt_seteui, 139
 dwt_setfinegraintxseq, 140
 dwt_setgpiodirection, 140
 dwt_setgpiovalue, 141
 dwt_setinterrupt, 141
 dwt_setleds, 142
 dwt_setlnapemode, 142
 dwt_setlocaldataptr, 143
 dwt_setlowpowerlistening, 143
 dwt_setpanid, 143
 dwt_setpreambledetecttimeout, 144
 dwt_setrxaftertxdelay, 144
 dwt_setrxantennadelay, 145
 dwt_setrxtimeout, 145
 dwt_setsmarttxpower, 146
 dwt_setsniffmode, 146
 dwt_setsnoozetime, 147
 dwt_settxantennadelay, 147
 dwt_setxtaltrim, 148
 dwt_softreset, 148
 dwt_spicswakeups, 149
 dwt_starttx, 149
 dwt_syncrxbufptrs, 150
 dwt_write16bitoffsetreg, 151
 dwt_write32bitoffsetreg, 152
 dwt_write8bitoffsetreg, 153
 dwt_writetodevice, 153

dwt_writetxdata, 155
dwt_writetxfctrl, 156
ENABLE_ALL_SEQ, 107
FCTRL_ACK_REQ_MASK, 107
FCTRL_LEN_MAX, 107
FORCE_LDE, 107
FORCE OTP OFF, 107
FORCE OTP ON, 107
FORCE_SYS_PLL, 107
FORCE_SYS_XTI, 107
FORCE_TX_PLL, 108
LDOTUNE_ADDRESS, 108
LOTID_ADDRESS, 108
PARTID_ADDRESS, 108
READ_ACC_OFF, 108
READ_ACC_ON, 108
VBAT_ADDRESS, 108
VTEMP_ADDRESS, 108
XTRIM_ADDRESS, 109

deca_device_api.h, 156
 __DECA_INT16_, 162
 __DECA_INT32_, 162
 __DECA_INT8_, 162
 __DECA_UINT16_, 162
 __DECA_UINT32_, 162
 __DECA_UINT8_, 162
 DWT_BR_110K, 163
 DWT_BR_6M8, 163
 DWT_BR_850K, 163
 DWT_CB_DATA_RX_FLAG RNG, 163
 DWT_CONFIG, 163
 DWT_DEVICE_ID, 163
 DWT_ERROR, 163
 DWT_FF_ACK_EN, 164
 DWT_FF_BEACON_EN, 164
 DWT_FF_COORD_EN, 164
 DWT_FF_DATA_EN, 164
 DWT_FF_MAC_EN, 164
 DWT_FF_NOTYPE_EN, 164
 DWT_FF_RSVD_EN, 164
 DWT_IDLE_ON_DLY_ERR, 164
 DWT_INT_ARFE, 165
 DWT_INT_LDED, 165
 DWT_INT_RFCE, 165
 DWT_INT_RFCG, 165
 DWT_INT_RFSL, 165
 DWT_INT_RFTO, 165
 DWT_INT_RPHE, 165
 DWT_INT_RXOVR, 165
 DWT_INT_RXPTO, 166
 DWT_INT_SFDT, 166
 DWT_INT_TFRS, 166
 DWT_LEDS_DISABLE, 166
 DWT_LEDS_ENABLE, 166
 DWT_LEDS_INIT_BLINK, 166
 DWT_LOADEUI, 166
 DWT_LOADNONE, 166
 DWT_LOADOPSET, 167

 DWT_LOADUCODE, 167
 DWT_NO_SYNC_PTRS, 167
 DWT_NUM_DW_DEV, 167
 DWT_OPSET_64LEN, 167
 DWT_OPSET_DEFLT, 167
 DWT_OPSET_TIGHT, 167
 DWT_PAC16, 167
 DWT_PAC32, 168
 DWT_PAC64, 168
 DWT_PAC8, 168
 DWT_PHRMODE_EXT, 168
 DWT_PHRMODE_STD, 168
 DWT_PLEN_1024, 168
 DWT_PLEN_128, 168
 DWT_PLEN_1536, 169
 DWT_PLEN_2048, 169
 DWT_PLEN_256, 169
 DWT_PLEN_4096, 169
 DWT_PLEN_512, 169
 DWT_PLEN_64, 169
 DWT_PRESRV_SLEEP, 169
 DWT_PRF_16M, 170
 DWT_PRF_64M, 170
 DWT_RESPONSE_EXPECTED, 170
 DWT_RX_EN, 170
 DWT_SFDTOC_DEF, 170
 DWT_SLP_EN, 170
 DWT_START_RX_DELAYED, 171
 DWT_START_RX_IMMEDIATE, 171
 DWT_START_TX_DELAYED, 171
 DWT_START_TX_IMMEDIATE, 171
 DWT_SUCCESS, 171
 DWT_TANDV, 171
 DWT_TIME_UNITS, 171
 DWT_WAKE_CS, 172
 DWT_WAKE_SLP_CNT, 172
 DWT_WAKE_WK, 172
 DWT_XTAL_EN, 172
 deca_sleep, 174
 decalrqStatus_t, 173
 decamutexoff, 175
 decamutexon, 176
 dwt_calcbandwidthtempadj, 176
 dwt_calcpgcount, 177
 dwt_calcpowertempadj, 177
 dwt_calibratesleepcnt, 178
 dwt_cb_t, 173
 dwt_checkirq, 178
 dwt_configcontinuousframemode, 179
 dwt_configcwmode, 179
 dwt_configeventcounters, 179
 dwt_configure, 180
 dwt_configuresleep, 180
 dwt_configuresleepcnt, 180
 dwt_configuretxrf, 181
 dwt_enableautoack, 181
 dwt_enableframefilter, 181
 dwt_entersleep, 182

dwt_entersleepaftertx, 182
 dwt_forcetrxoff, 183
 dwt_geteui, 183
 dwt_getinitxtaltrim, 184
 dwt_getlotid, 184
 dwt_getpartid, 184
 dwt_initialise, 185
 dwt_isr, 185
 dwt_loadopsettabfromotp, 185
 dwt_lowpowerlistenisr, 186
 dwt_otpread, 186
 dwt_otprevision, 186
 dwt_otpwriteandverify, 187
 dwt_read16bitoffsetreg, 187
 dwt_read32bitoffsetreg, 188
 dwt_read32bitreg, 170
 dwt_read8bitoffsetreg, 188
 dwt_readaccdata, 189
 dwt_readcarrierintegrator, 189
 dwt_readdovid, 190
 dwt_readdiagnostics, 190
 dwt_readeventcounters, 191
 dwt_readfromdevice, 191
 dwt_readrxdata, 193
 dwt_readrxtimestamp, 194
 dwt_readrxtimestampphi32, 194
 dwt_readrxtimestamplo32, 195
 dwt_readsystime, 195
 dwt_readsystimestampphi32, 196
 dwt_readtempvbat, 196
 dwt_readtxtimestamp, 196
 dwt_readtxtimestampphi32, 197
 dwt_readtxtimestamplo32, 197
 dwt_readwakeuptemp, 198
 dwt_readwakeupvbat, 198
 dwt_rxenable, 198
 dwt_rxreset, 199
 dwt_setaddress16, 200
 dwt_setcallbacks, 200
 dwt_setdblrxbuffmode, 201
 dwt_setdelayedtrxtime, 201
 dwt_seteui, 201
 dwt_setfinegraintxseq, 202
 dwt_setgpiodirection, 202
 dwt_setgpiovalue, 203
 dwt_setinterrupt, 203
 dwt_setleds, 204
 dwt_setlnapamode, 204
 dwt_setlocaldataptr, 205
 dwt_setlowpowerlistening, 205
 dwt_setpanid, 205
 dwt_setpreambledetecttimeout, 206
 dwt_setrxaftertxdelay, 206
 dwt_setrxantennadelay, 207
 dwt_setrxtimeout, 207
 dwt_setsmartxpower, 208
 dwt_setsniffmode, 208
 dwt_setsnoozetime, 209
 dwt_settxantennadelay, 209
 dwt_setxtaltrim, 210
 dwt_softreset, 210
 dwt_spicswakeups, 211
 dwt_starttx, 211
 dwt_syncrxbufptrs, 212
 dwt_write16bitoffsetreg, 213
 dwt_write32bitoffsetreg, 214
 dwt_write32bitreg, 172
 dwt_write8bitoffsetreg, 215
 dwt_writetodevice, 215
 dwt_writetxdata, 217
 dwt_writetxctrl, 218
 FREQ_OFFSET_MULTIPLIER_110KB, 172
 FREQ_OFFSET_MULTIPLIER, 172
 HERTZ_TO_PPM_MULTIPLIER_CHAN_1, 173
 HERTZ_TO_PPM_MULTIPLIER_CHAN_2, 173
 HERTZ_TO_PPM_MULTIPLIER_CHAN_3, 173
 HERTZ_TO_PPM_MULTIPLIER_CHAN_5, 173
 int16, 173
 int32, 174
 int8, 174
 readfromspi, 218
 uint16, 174
 uint32, 174
 uint8, 174
 writetospi, 219
 deca_param_types.h, 220
 agc_config, 224
 chan_idx, 224
 DA_ATTN_STEP, 222
 digital_bb_config, 224
 dtune1, 225
 dwnsSFDlen, 225
 fs_pll_cfg, 225
 fs_pll_tune, 225
 LDE_PARAM1, 222
 LDE_PARAM3_16, 222
 LDE_PARAM3_64, 222
 lde_replicaCoeff, 225
 MIXER_GAIN_STEP, 223
 N_STD_FACTOR, 223
 NUM_BR, 223
 NUM_BW, 223
 NUM_CH_SUPPORTED, 223
 NUM_CH, 223
 NUM_PACS, 223
 NUM_PRF, 223
 NUM_SFD, 224
 PCODES, 224
 PEAK_MULTIPLIER, 224
 rx_config, 225
 sftsh, 225
 tx_config, 225
 txpwr_compensation, 226
 XMLPARAMS_VERSION, 224
 deca_params_init.c, 226
 agc_config, 227

chan_idx, 227
digital_bb_config, 227
dtune1, 227
dwnsSFDlen, 228
fs_pll_cfg, 228
fs_pll_tune, 228
lde_replicaCoeff, 228
rx_config, 229
sftsh, 229
tx_config, 229
txpwr_compensation, 230
deca_regs.h, 230
ACC_MEM_ID, 251
ACC_MEM_LEN, 251
ACK_RESP_T_ACK_TIM_MASK, 251
ACK_RESP_T_ACK_TIM_OFFSET, 251
ACK_RESP_T_ID, 252
ACK_RESP_T_LEN, 252
ACK_RESP_T_MASK, 252
ACK_RESP_T_W4R_TIM_MASK, 252
ACK_RESP_T_W4R_TIM_OFFSET, 252
ACK_TIM_MASK, 252
AGC_CFG_STS_ID, 252
AGC_CTRL1_DIS_AM, 253
AGC_CTRL1_LEN, 253
AGC_CTRL1_MASK, 253
AGC_CTRL1_OFFSET, 253
AGC_CTRL_ID, 253
AGC_CTRL_LEN, 253
AGC_STAT1_EDG1_MASK, 253
AGC_STAT1_EDG2_MASK, 254
AGC_STAT1_LEN, 254
AGC_STAT1_MASK, 254
AGC_STAT1_OFFSET, 254
AGC_TUNE1_16M, 254
AGC_TUNE1_64M, 254
AGC_TUNE1_LEN, 254
AGC_TUNE1_MASK, 255
AGC_TUNE1_OFFSET, 255
AGC_TUNE2_LEN, 255
AGC_TUNE2_MASK, 255
AGC_TUNE2_OFFSET, 255
AGC_TUNE2_VAL, 255
AGC_TUNE3_LEN, 255
AGC_TUNE3_MASK, 255
AGC_TUNE3_OFFSET, 256
AGC_TUNE3_VAL, 256
AON_ADDR_LEN, 256
AON_ADDR_LPOSC_CAL_0, 256
AON_ADDR_LPOSC_CAL_1, 256
AON_ADDR_OFFSET, 256
AON_CFG0_LEN, 256
AON_CFG0_LPCLKDIVA_MASK, 257
AON_CFG0_LPCLKDIVA_SHIFT, 257
AON_CFG0_LPDIV_EN, 257
AON_CFG0_OFFSET, 257
AON_CFG0_SLEEP_EN, 257
AON_CFG0_SLEEP_SHIFT, 257
AON_CFG0_SLEEP_TIM_OFFSET, 258
AON_CFG0_SLEEP_TIM, 257
AON_CFG0_WAKE_CNT, 258
AON_CFG0_WAKE_PIN, 258
AON_CFG0_WAKE_SPI, 258
AON_CFG1_LEN, 258
AON_CFG1_LPOSC_CAL, 258
AON_CFG1_MASK, 258
AON_CFG1_OFFSET, 259
AON_CFG1_SLEEP_CEN, 259
AON_CFG1_SMXX, 259
AON_CTRL_DCA_ENAB, 259
AON_CTRL_DCA_READ, 259
AON_CTRL_LEN, 259
AON_CTRL_MASK, 259
AON_CTRL_OFFSET, 260
AON_CTRL_RESTORE, 260
AON_CTRL_SAVE, 260
AON_CTRL_UPL_CFG, 260
AON_ID, 260
AON_LEN, 260
AON_RDAT_LEN, 260
AON_RDAT_OFFSET, 261
AON_WCFG_LEN, 261
AON_WCFG_MASK, 261
AON_WCFG_OFFSET, 261
AON_WCFG_ONW_L64P, 261
AON_WCFG_ONW_LDC, 261
AON_WCFG_ONW_LEUI, 261
AON_WCFG_ONW_LLDE, 262
AON_WCFG_ONW_LLDO, 262
AON_WCFG_ONW_RADC, 262
AON_WCFG_ONW_RX, 262
AON_WCFG_PRES_SLEEP, 262
BOOSTNORM_MASK, 262
BOOSTP125_MASK, 262
BOOSTP250_MASK, 263
BOOSTP500_MASK, 263
CHAN_CTRL_DWSFD_SHIFT, 263
CHAN_CTRL_DWSFD, 263
CHAN_CTRL_ID, 263
CHAN_CTRL_LEN, 263
CHAN_CTRL_MASK, 263
CHAN_CTRL_RNSSFD_SHIFT, 264
CHAN_CTRL_RNSSFD, 264
CHAN_CTRL_RX_CHAN_MASK, 264
CHAN_CTRL_RX_CHAN_SHIFT, 264
CHAN_CTRL_RX_PCOD_MASK, 264
CHAN_CTRL_RX_PCOD_SHIFT, 264
CHAN_CTRL_RXFPRF_16, 264
CHAN_CTRL_RXFPRF_4, 265
CHAN_CTRL_RXFPRF_64, 265
CHAN_CTRL_RXFPRF_MASK, 265
CHAN_CTRL_RXFPRF_SHIFT, 265
CHAN_CTRL_TNSSFD_SHIFT, 265
CHAN_CTRL_TNSSFD, 265
CHAN_CTRL_TX_CHAN_MASK, 265
CHAN_CTRL_TX_CHAN_SHIFT, 266

CHAN_CTRL_TX_PCOD_MASK, 266
 CHAN_CTRL_TX_PCOD_SHIFT, 266
 CIR_MXG_MASK, 266
 CIR_MXG_SHIFT, 266
 DEV_ID_ID, 266
 DEV_ID_LEN, 266
 DEV_ID_MODEL_MASK, 267
 DEV_ID_REV_MASK, 267
 DEV_ID_RIDTAG_MASK, 267
 DEV_ID_VER_MASK, 267
 DIAG_TMC_LEN, 267
 DIAG_TMC_MASK, 267
 DIAG_TMC_OFFSET, 267
 DIAG_TMC_TX_PSTM, 268
 DIG_DIAG_ID, 268
 DIG_DIAG_LEN, 268
 DRX_CARRIER_INT_LEN, 268
 DRX_CARRIER_INT_MASK, 268
 DRX_CARRIER_INT_OFFSET, 268
 DRX_CONF_ID, 268
 DRX_CONF_LEN, 269
 DRX_PRETOC_LEN, 269
 DRX_PRETOC_MASK, 269
 DRX_PRETOC_OFFSET, 269
 DRX_SFDTOC_LEN, 269
 DRX_SFDTOC_MASK, 269
 DRX_SFDTOC_OFFSET, 269
 DRX_TUNE0b_110K_NSTD, 270
 DRX_TUNE0b_110K_STD, 270
 DRX_TUNE0b_6M8_NSTD, 270
 DRX_TUNE0b_6M8_STD, 270
 DRX_TUNE0b_850K_NSTD, 270
 DRX_TUNE0b_850K_STD, 270
 DRX_TUNE0b_LEN, 270
 DRX_TUNE0b_MASK, 270
 DRX_TUNE0b_OFFSET, 271
 DRX_TUNE1a_LEN, 271
 DRX_TUNE1a_MASK, 271
 DRX_TUNE1a_OFFSET, 271
 DRX_TUNE1a_PRF16, 271
 DRX_TUNE1a_PRF64, 271
 DRX_TUNE1b_110K, 271
 DRX_TUNE1b_6M8_PRE64, 271
 DRX_TUNE1b_850K_6M8, 272
 DRX_TUNE1b_LEN, 272
 DRX_TUNE1b_MASK, 272
 DRX_TUNE1b_OFFSET, 272
 DRX_TUNE2_LEN, 272
 DRX_TUNE2_MASK, 272
 DRX_TUNE2_OFFSET, 272
 DRX_TUNE2_PRF16_PAC16, 272
 DRX_TUNE2_PRF16_PAC32, 273
 DRX_TUNE2_PRF16_PAC64, 273
 DRX_TUNE2_PRF16_PAC8, 273
 DRX_TUNE2_PRF64_PAC16, 273
 DRX_TUNE2_PRF64_PAC32, 273
 DRX_TUNE2_PRF64_PAC64, 273
 DRX_TUNE2_PRF64_PAC8, 273

DRX_TUNE4H_LEN, 273
 DRX_TUNE4H_MASK, 274
 DRX_TUNE4H_OFFSET, 274
 DRX_TUNE4H_PRE128PLUS, 274
 DRX_TUNE4H_PRE64, 274
 DW_NS_SFD_LEN_110K, 274
 DW_NS_SFD_LEN_6M8, 274
 DW_NS_SFD_LEN_850K, 274
 DX_TIME_ID, 274
 DX_TIME_LEN, 275
 EC_CTRL_LEN, 275
 EC_CTRL_MASK, 275
 EC_CTRL_OFFSET, 275
 EC_CTRL_OSRSM, 275
 EC_CTRL_OSTRM, 275
 EC_CTRL_OSTSM, 275
 EC_CTRL_PLLCK, 276
 EC_CTRL_WAIT_MASK, 276
 EC_GOLP_LEN, 276
 EC_GOLP_MASK, 276
 EC_GOLP_OFFSET_EXT_MASK, 276
 EC_GOLP, 276
 EC_RXTC_LEN, 276
 EC_RXTC_MASK, 277
 EC_RXTC_OFFSET, 277
 EUI_64_ID, 277
 EUI_64_LEN, 277
 EUI_64_OFFSET, 277
 EVC_CLR, 277
 EVC_CTRL_LEN, 277
 EVC_CTRL_MASK, 278
 EVC_CTRL_OFFSET, 278
 EVC_EN, 278
 EVC_FCE_LEN, 278
 EVC_FCE_MASK, 278
 EVC_FCE_OFFSET, 278
 EVC_FCG_LEN, 278
 EVC_FCG_MASK, 279
 EVC_FCG_OFFSET, 279
 EVC_FFR_LEN, 279
 EVC_FFR_MASK, 279
 EVC_FFR_OFFSET, 279
 EVC_FWTO_LEN, 279
 EVC_FWTO_MASK, 279
 EVC_FWTO_OFFSET, 280
 EVC_HPW_LEN, 280
 EVC_HPW_MASK, 280
 EVC_HPW_OFFSET, 280
 EVC_OVR_LEN, 280
 EVC_OVR_MASK, 280, 281
 EVC_OVR_OFFSET, 281
 EVC_PHE_LEN, 281
 EVC_PHE_MASK, 281
 EVC_PHE_OFFSET, 281
 EVC_PTO_LEN, 281
 EVC_PTO_MASK, 281
 EVC_PTO_OFFSET, 282
 EVC_RES1_OFFSET, 282

EVC_RSE_LEN, 282
EVC_RSE_MASK, 282
EVC_RSE_OFFSET, 282
EVC_STO_OFFSET, 282
EVC_TPW_LEN, 282
EVC_TPW_MASK, 283
EVC_TPW_OFFSET, 283
EVC_TXFS_LEN, 283
EVC_TXFS_MASK, 283
EVC_TXFS_OFFSET, 283
EXT_SYNC_ID, 283
EXT_SYNC_LEN, 283
FP_AMPL2_MASK, 284
FP_AMPL2_SHIFT, 284
FP_AMPL3_MASK, 284
FP_AMPL3_SHIFT, 284
FS_CTRL_ID, 284
FS_CTRL_LEN, 284
FS_PLLCFG_CH1, 284
FS_PLLCFG_CH2, 285
FS_PLLCFG_CH3, 285
FS_PLLCFG_CH4, 285
FS_PLLCFG_CH5, 285
FS_PLLCFG_CH7, 285
FS_PLLCFG_LEN, 285
FS_PLLCFG_OFFSET, 285
FS_PLLTUNE_CH1, 285
FS_PLLTUNE_CH2, 286
FS_PLLTUNE_CH3, 286
FS_PLLTUNE_CH4, 286
FS_PLLTUNE_CH5, 286
FS_PLLTUNE_CH7, 286
FS_PLLTUNE_LEN, 286
FS_PLLTUNE_OFFSET, 286
FS_RES1_LEN, 286
FS_RES1_OFFSET, 287
FS_RES2_LEN, 287
FS_RES2_OFFSET, 287
FS_RES3_LEN, 287
FS_RES3_OFFSET, 287
FS_XTALT_LEN, 287
FS_XTALT_MASK, 287
FS_XTALT_MIDRANGE, 288
FS_XTALT_OFFSET, 288
GDM0, 288
GDM1, 288
GDM2, 288
GDM3, 288
GDM4, 288
GDM5, 289
GDM6, 289
GDM7, 289
GDM8, 289
GDP0, 289
GDP1, 289
GDP2, 289
GDP3, 290
GDP4, 290
GDP5, 290
GDP6, 290
GDP7, 290
GDP8, 290
GIBES0, 290
GIBES1, 290
GIBES2, 291
GIBES3, 291
GIBES4, 291
GIBES5, 291
GIBES6, 291
GIBES7, 291
GIBES8, 291
GICLR0, 291
GICLR1, 292
GICLR2, 292
GICLR3, 292
GICLR4, 292
GICLR5, 292
GICLR6, 292
GICLR7, 292
GICLR8, 293
GIDBE0, 293
GIDBE1, 293
GIDBE2, 293
GIDBE3, 293
GIDBE4, 293
GIDBE5, 293
GIDBE6, 294
GIDBE7, 294
GIDBE8, 294
GIMOD0, 294
GIMOD1, 294
GIMOD2, 294
GIMOD3, 294
GIMOD4, 295
GIMOD5, 295
GIMOD6, 295
GIMOD7, 295
GIMOD8, 295
GIRQE0, 295
GIRQE1, 295
GIRQE2, 295
GIRQE3, 296
GIRQE4, 296
GIRQE5, 296
GIRQE6, 296
GIRQE7, 296
GIRQE8, 296
GIRQx0, 296
GIRQx1, 296
GIRQx2, 297
GIRQx3, 297
GIRQx4, 297
GIRQx5, 297
GIRQx6, 297
GIRQx7, 297
GIRQx8, 297

GISEN0, 297
GISEN1, 298
GISEN2, 298
GISEN3, 298
GISEN4, 298
GISEN5, 298
GISEN6, 298
GISEN7, 298
GISEN8, 299
GPIO_CTRL_ID, 299
GPIO_CTRL_LEN, 299
GPIO_DIR_LEN, 299
GPIO_DIR_MASK, 299
GPIO_DIR_OFFSET, 299
GPIO_DOUT_LEN, 299
GPIO_DOUT_MASK, 300
GPIO_DOUT_OFFSET, 300
GPIO_IBES_LEN, 300
GPIO_IBES_MASK, 300
GPIO_IBES_OFFSET, 300
GPIO_ICLR_LEN, 300
GPIO_ICLR_MASK, 300
GPIO_ICLR_OFFSET, 301
GPIO_IDBE_LEN, 301
GPIO_IDBE_MASK, 301
GPIO_IDBE_OFFSET, 301
GPIO_IMODE_LEN, 301
GPIO_IMODE_MASK, 301
GPIO_IMODE_OFFSET, 301
GPIO_IRQE_LEN, 302
GPIO_IRQE_MASK, 302
GPIO_IRQE_OFFSET, 302
GPIO_ISEN_LEN, 302
GPIO_ISEN_MASK, 302
GPIO_ISEN_OFFSET, 302
GPIO_MODE_LEN, 302
GPIO_MODE_MASK, 303
GPIO_MODE_OFFSET, 303
GPIO_MSGP0_MASK, 303
GPIO_MSGP1_MASK, 303
GPIO_MSGP2_MASK, 303
GPIO_MSGP3_MASK, 303
GPIO_MSGP4_MASK, 303
GPIO_MSGP5_MASK, 303
GPIO_MSGP6_MASK, 304
GPIO_MSGP7_MASK, 304
GPIO_MSGP8_MASK, 304
GPIO_PIN2_RXLED, 304
GPIO_PIN3_TXLED, 304
GPIO_PIN4_EXTPA, 304
GPIO_PIN5_EXTTXE, 304
GPIO_PIN6_EXTRXE, 304
GPIO_RAW_LEN, 305
GPIO_RAW_MASK, 305
GPIO_RAW_OFFSET, 305
GRAWP0, 305
GRAWP1, 305
GRAWP2, 305
GRAWP3, 305
GRAWP4, 306
GRAWP5, 306
GRAWP6, 306
GRAWP7, 306
GRAWP8, 306
GxM0, 306
GxM1, 306
GxM2, 306
GxM3, 307
GxM4, 307
GxM5, 307
GxM6, 307
GxM7, 307
GxM8, 307
GxP0, 307
GxP1, 307
GxP2, 308
GxP3, 308
GxP4, 308
GxP5, 308
GxP6, 308
GxP7, 308
GxP8, 308
LDE_CFG1_LEN, 308
LDE_CFG1_NSTDEV_MASK, 309
LDE_CFG1_OFFSET, 309
LDE_CFG1_PMULT_MASK, 309
LDE_CFG2_LEN, 309
LDE_CFG2_OFFSET, 309
LDE_IF_ID, 309
LDE_IF_LEN, 309
LDE_PPAMPL_LEN, 310
LDE_PPAMPL_OFFSET, 310
LDE_PPIDX_LEN, 310
LDE_PPIDX_OFFSET, 310
LDE_REPC_LEN, 310
LDE_REPC_OFFSET, 310
LDE_REPC_PCODE_1, 310
LDE_REPC_PCODE_10, 311
LDE_REPC_PCODE_11, 311
LDE_REPC_PCODE_12, 311
LDE_REPC_PCODE_13, 311
LDE_REPC_PCODE_14, 311
LDE_REPC_PCODE_15, 311
LDE_REPC_PCODE_16, 311
LDE_REPC_PCODE_17, 311
LDE_REPC_PCODE_18, 312
LDE_REPC_PCODE_19, 312
LDE_REPC_PCODE_2, 312
LDE_REPC_PCODE_20, 312
LDE_REPC_PCODE_21, 312
LDE_REPC_PCODE_22, 312
LDE_REPC_PCODE_23, 312
LDE_REPC_PCODE_24, 312
LDE_REPC_PCODE_3, 313
LDE_REPC_PCODE_4, 313
LDE_REPC_PCODE_5, 313

LDE_REPCPCODE_6, 313
LDE_REPCPCODE_7, 313
LDE_REPCPCODE_8, 313
LDE_REPCPCODE_9, 313
LDE_RXANTD_LEN, 313
LDE_RXANTD_OFFSET, 314
LDE_THRESH_LEN, 314
LDE_THRESH_OFFSET, 314
OTP_ADDR_LEN, 314
OTP_ADDR_MASK, 314
OTP_ADDR, 314
OTP_CTRL_LDELOAD, 315
OTP_CTRL_LEN, 315
OTP_CTRL_MASK, 315
OTP_CTRL_OTPPROG, 315
OTP_CTRL_OTPRDEN, 315
OTP_CTRL_OTPREAD, 315
OTP_CTRL, 314
OTP_IF_ID, 315
OTP_IF_LEN, 316
OTP_RDAT_LEN, 316
OTP_RDAT, 316
OTP_SF_LDO_KICK, 316
OTP_SF_LEN, 316
OTP_SF_MASK, 316
OTP_SF_OPS_KICK, 317
OTP_SF_OPS_SEL_L64, 317
OTP_SF_OPS_SEL_MASK, 317
OTP_SF_OPS_SEL_SHFT, 317
OTP_SF_OPS_SEL_TIGHT, 317
OTP_SRDAT_LEN, 317
OTP_SRDAT, 317
OTP_STAT_LEN, 318
OTP_STAT_MASK, 318
OTP_STAT_OTPPRGD, 318
OTP_STAT_OTPVPOK, 318
OTP_STAT, 318
OTP_SF, 316
OTP_WDAT_LEN, 318
OTP_WDAT, 318
PANADR_ID, 319
PANADR_LEN, 319
PANADR_PAN_ID_MASK, 319
PANADR_PAN_ID_OFFSET, 319
PANADR_SHORT_ADDR_MASK, 319
PANADR_SHORT_ADDR_OFFSET, 319
PMSC_CTRL0_FACE, 319
PMSC_CTRL0_GPDCE, 320
PMSC_CTRL0_KHZCLEN, 320
PMSC_CTRL0_LEN, 320
PMSC_CTRL0_MASK, 320
PMSC_CTRL0_OFFSET, 320
PMSC_CTRL0_PLL2_SEQ_EN, 320
PMSC_CTRL0_RESET_ALL, 320
PMSC_CTRL0_RESET_CLEAR, 320
PMSC_CTRL0_RESET_RX, 321
PMSC_CTRL0_RXCLKS_125M, 321
PMSC_CTRL0_RXCLKS_19M, 321
PMSC_CTRL0_RXCLKS_AUTO, 321
PMSC_CTRL0_RXCLKS_OFF, 321
PMSC_CTRL0_SOFTRESET_OFFSET, 321
PMSC_CTRL0_SYSCLKS_125M, 321
PMSC_CTRL0_SYSCLKS_19M, 322
PMSC_CTRL0_SYSCLKS_AUTO, 322
PMSC_CTRL0_TXCLKS_125M, 322
PMSC_CTRL0_TXCLKS_19M, 322
PMSC_CTRL0_TXCLKS_AUTO, 322
PMSC_CTRL0_TXCLKS_OFF, 322
PMSC_CTRL1_ARX2INIT, 322
PMSC_CTRL1_ARXSLP, 323
PMSC_CTRL1_ATXSLP, 323
PMSC_CTRL1_KHZCLKDIV_MASK, 323
PMSC_CTRL1_LDERUNE, 323
PMSC_CTRL1_LEN, 323
PMSC_CTRL1_MASK, 323
PMSC_CTRL1_OFFSET, 323
PMSC_CTRL1_PKTSEQ_DISABLE, 324
PMSC_CTRL1_PKTSEQ_ENABLE, 324
PMSC_CTRL1_PLLSYN, 324
PMSC_CTRL1_SNOZE, 324
PMSC_CTRL1_SNOZR, 324
PMSC_ID, 324
PMSC_LEDC_BLINK_NOW_ALL, 324
PMSC_LEDC_BLINK_TIM_MASK, 325
PMSC_LEDC_BLINK_TIME_DEF, 325
PMSC_LEDC_BLNKEN, 325
PMSC_LEDC_LEN, 325
PMSC_LEDC_MASK, 325
PMSC_LED_OFFSET, 325
PMSC_LEN, 325
PMSC_RES1_OFFSET, 326
PMSC_RES2_OFFSET, 326
PMSC_RES3_OFFSET, 326
PMSC_SNOZT_LEN, 326
PMSC_SNOZT_OFFSET, 326
PMSC_TXFINESEQ_DISABLE, 326
PMSC_TXFINESEQ_ENABLE, 326
PMSC_TXFINESEQ_OFFSET, 327
REG_05_ID_RESERVED, 327
REG_07_ID_RESERVED, 327
REG_0B_ID_RESERVED, 327
REG_16_ID_RESERVED, 327
REG_1B_ID_RESERVED, 327
REG_1C_ID_RESERVED, 328
REG_20_ID_RESERVED, 328
REG_22_ID_RESERVED, 328
REG_29_ID_RESERVED, 328
REG_30_ID_RESERVED, 328
REG_31_ID_RESERVED, 328
REG_32_ID_RESERVED, 329
REG_33_ID_RESERVED, 329
REG_34_ID_RESERVED, 329
REG_35_ID_RESERVED, 329
REG_37_ID_RESERVED, 329
REG_38_ID_RESERVED, 329
REG_39_ID_RESERVED, 329

REG_3A_ID_RESERVED, 330
 REG_3B_ID_RESERVED, 330
 REG_3C_ID_RESERVED, 330
 REG_3D_ID_RESERVED, 330
 REG_3E_ID_RESERVED, 330
 REG_3F_ID_RESERVED, 330
 RF_CONF_ID, 330
 RF_CONF_LEN, 331
 RF_CONF_PGMIXBIASEN_MASK, 331
 RF_CONF_PLLEN_MASK, 331
 RF_CONF_RXEN_MASK, 331
 RF_CONF_TXALLEN_MASK, 331
 RF_CONF_TXBLOCKSEN_MASK, 331
 RF_CONF_TXEN_MASK, 331
 RF_CONF_TXPLLPOWEN_MASK, 332
 RF_CONF_TXPOW_MASK, 332
 RF_RXCTRLH_LEN, 332
 RF_RXCTRLH_NBW, 332
 RF_RXCTRLH_OFFSET, 332
 RF_RXCTRLH_WBW, 332
 RF_STATUS_OFFSET, 332
 RF_TXCTRL_CH1, 332
 RF_TXCTRL_CH2, 333
 RF_TXCTRL_CH3, 333
 RF_TXCTRL_CH4, 333
 RF_TXCTRL_CH5, 333
 RF_TXCTRL_CH7, 333
 RF_TXCTRL_LEN, 333
 RF_TXCTRL_OFFSET, 333
 RF_TXCTRL_TXMTUNE_MASK, 334
 RF_TXCTRL_TXTXMQ_MASK, 334
 RX_BUFFER_ID, 334
 RX_BUFFER_LEN, 334
 RX_EQUAL_CIR_MXG_MASK, 334
 RX_EQUAL_CIR_MXG_SHIFT, 334
 RX_EQUAL_FP_AMPL2_MASK, 334
 RX_EQUAL_FP_AMPL2_SHIFT, 335
 RX_EQUAL_FP_AMPL3_MASK, 335
 RX_EQUAL_FP_AMPL3_SHIFT, 335
 RX_EQUAL_STD_NOISE_MASK, 335
 RX_EQUAL_STD_NOISE_SHIFT, 335
 RX_FINFO_ID, 335
 RX_FINFO_LEN, 335
 RX_FINFO_MASK_32, 336
 RX_FINFO_OFFSET, 336
 RX_FINFO_RNG_SHIFT, 336
 RX_FINFO_RNG, 336
 RX_FINFO_RXBR_110k, 336
 RX_FINFO_RXBR_6M, 336
 RX_FINFO_RXBR_850k, 336
 RX_FINFO_RXBR_MASK, 337
 RX_FINFO_RXBR_SHIFT, 337
 RX_FINFO_RXFL_MASK_1023, 337
 RX_FINFO_RXFLE_MASK, 337
 RX_FINFO_RXFLEN_MASK, 337
 RX_FINFO_RXNSPL_MASK, 337
 RX_FINFO_RXPACC_MASK, 337
 RX_FINFO_RXPACC_SHIFT, 338
 RX_FINFO_RXPEL_1024, 338
 RX_FINFO_RXPEL_128, 338
 RX_FINFO_RXPEL_1536, 338
 RX_FINFO_RXPEL_2048, 338
 RX_FINFO_RXPEL_256, 338
 RX_FINFO_RXPEL_4096, 338
 RX_FINFO_RXPEL_512, 338
 RX_FINFO_RXPEL_64, 339
 RX_FINFO_RXPEL_MASK, 339
 RX_FINFO_RXPRF_16M, 339
 RX_FINFO_RXPRF_64M, 339
 RX_FINFO_RXPRF_MASK, 339
 RX_FINFO_RXPRF_SHIFT, 339
 RX_FINFO_RXPSR_MASK, 339
 RX_FQUAL_ID, 339
 RX_FQUAL_LEN, 340
 RX_FWTO_ID, 340
 RX_FWTO_LEN, 340
 RX_FWTO_MASK, 340
 RX_FWTO_OFFSET, 340
 RX_SNIFF_ID, 340
 RX_SNIFF_LEN, 340
 RX_SNIFF_MASK, 341
 RX_SNIFF_OFFSET, 341
 RX_SNIFF_SNIFF_OFFT_MASK, 341
 RX_SNIFF_SNIFF_ONT_MASK, 341
 RX_STAMP_LEN, 341
 RX_TIME_FP_AMPL1_OFFSET, 341
 RX_TIME_FP_INDEX_OFFSET, 341
 RX_TIME_FP_RAWST_OFFSET, 342
 RX_TIME_ID, 342
 RX_TIME_LLEN, 342
 RX_TIME_RX_STAMP_LEN, 342
 RX_TIME_RX_STAMP_OFFSET, 342
 RX_TTCKI_ID, 342
 RX_TTCKI_LEN, 342
 RX_TTCKO_ID, 343
 RX_TTCKO_LEN, 343
 RX_TTCKO_MASK_32, 343
 RX_TTCKO_RCPhase_MASK, 343
 RX_TTCKO_RSMPDEL_MASK, 343
 RX_TTCKO_RXTOFS_MASK, 343
 SNIFF_OFFT_MASK, 343
 SNIFF_ONT_MASK, 344
 STD_NOISE_MASK, 344
 STD_NOISE_SHIFT, 344
 SYS_CFG_ACKPEND, 344
 SYS_CFG_AUTOACK, 344
 SYS_CFG_DIS_DRXB, 344
 SYS_CFG_DIS_FCE, 344
 SYS_CFG_DIS_PHE, 344
 SYS_CFG_DIS_RSDE, 345
 SYS_CFG_DIS_STXP, 345
 SYS_CFG_FCS_INIT2F, 345
 SYS_CFG_FF_ALL_EN, 345
 SYS_CFG_FFA4, 345
 SYS_CFG_FFA5, 345
 SYS_CFG_FFAA, 345

SYS_CFG_FFAB, 346
SYS_CFG_FFAD, 346
SYS_CFG_FFAM, 346
SYS_CFG_FFAR, 346
SYS_CFG_FFBC, 346
SYS_CFG_FFE, 346
SYS_CFG_HIRQ_POL, 346
SYS_CFG_ID, 347
SYS_CFG_LEN, 347
SYS_CFG_MASK, 347
SYS_CFG_PHR_MODE_00, 347
SYS_CFG_PHR_MODE_11, 347
SYS_CFG_PHR_MODE_SHFT, 347
SYS_CFG_RXAUTR, 347
SYS_CFG_RXM110K, 348
SYS_CFG_RXWTOE, 348
SYS_CFG_SPI_EDGE, 348
SYS_CTRL_CANSFCS, 348
SYS_CTRL_HRBTOFFSET, 348
SYS_CTRL_HRBTT, 348
SYS_CTRL_HSRBTTOGGLE, 348
SYS_CTRL_ID, 349
SYS_CTRL_LEN, 349
SYS_CTRL_MASK_32, 349
SYS_CTRL_OFFSET, 349
SYS_CTRL_RXDLYE, 349
SYS_CTRL_RXENAB, 349
SYS_CTRL_SFCST, 349
SYS_CTRL_TRXOFF, 350
SYS_CTRL_TXDLYS, 350
SYS_CTRL_TXSTRT, 350
SYS_CTRL_WAIT4RESP, 350
SYS_MASK_ID, 350
SYS_MASK_LEN, 350
SYS_MASK_MAAT, 350
SYS_MASK_MAFFREJ, 351
SYS_MASK_MASK_32, 351
SYS_MASK_MCPLLLL, 351
SYS_MASK_MCPLOCK, 351
SYS_MASK_MESYNCR, 351
SYS_MASK_MGPIORQ, 351
SYS_MASK_MHPDWARN, 351
SYS_MASK_MLDEDONE, 352
SYS_MASK_MLDEERR, 352
SYS_MASK_MRFPLLLL, 352
SYS_MASK_MRXdFR, 352
SYS_MASK_MRXFCE, 352
SYS_MASK_MRXFCG, 352
SYS_MASK_MRXOVRR, 352
SYS_MASK_MRXPHD, 352
SYS_MASK_MRXPHE, 353
SYS_MASK_MRXP RD, 353
SYS_MASK_MRXP TO, 353
SYS_MASK_MRXRFSL, 353
SYS_MASK_MRXRFTO, 353
SYS_MASK_MRXSFDTO, 353
SYS_MASK_MRXSFDD, 353
SYS_MASK_MSLP2INIT, 353
SYS_MASK_MTXBERR, 354
SYS_MASK_MTXFRB, 354
SYS_MASK_MTXFRS, 354
SYS_MASK_MTYPHS, 354
SYS_MASK_MTYPRS, 354
SYS_STATE_ID, 354
SYS_STATE_LEN, 354
SYS_STATUS_AAT, 355
SYS_STATUS_AFFREJ, 355
SYS_STATUS_ALL_DBLBUFF, 355
SYS_STATUS_ALL_RX_ERR, 355
SYS_STATUS_ALL_RX_GOOD, 355
SYS_STATUS_ALL_RX_TO, 355
SYS_STATUS_ALL_TX, 356
SYS_STATUS_CLKPLL_LL, 356
SYS_STATUS_CPLOCK, 356
SYS_STATUS_ESYNCR, 356
SYS_STATUS_GPIOIRQ, 356
SYS_STATUS_HPDWARN, 356
SYS_STATUS_HSRBP, 356
SYS_STATUS_ICRBP, 357
SYS_STATUS IRQS, 357
SYS_STATUS_ID, 357
SYS_STATUS_LDEDONE, 357
SYS_STATUS_LDEERR, 357
SYS_STATUS_LEN, 357
SYS_STATUS_MASK_32, 357
SYS_STATUS_OFFSET, 358
SYS_STATUS_RFPLL_LL, 358
SYS_STATUS_RXDFR, 358
SYS_STATUS_RXFCE, 358
SYS_STATUS_RXFCG, 358
SYS_STATUS_RXOVRR, 358
SYS_STATUS_RXPHD, 359
SYS_STATUS_RXPHE, 359
SYS_STATUS_RXPREJ, 359
SYS_STATUS_RXPRD, 359
SYS_STATUS_RXPTO, 359
SYS_STATUS_RXRFSL, 359
SYS_STATUS_RXRFTO, 359
SYS_STATUS_RXRSCS, 359
SYS_STATUS_RXSF DTO, 360
SYS_STATUS_RXSFDD, 360
SYS_STATUS_SLP2INIT, 360
SYS_STATUS_TXBERR, 360
SYS_STATUS_TXERR, 360
SYS_STATUS_TXFRB, 360
SYS_STATUS_TXFRS, 360
SYS_STATUS_TXP HS, 361
SYS_STATUS_TXPRS, 361
SYS_STATUS_TXPUTE, 361
SYS_STATUS_reserved, 358
SYS_TIME_ID, 361
SYS_TIME_LEN, 361
SYS_TIME_OFFSET, 361
TC_PGCAL_STATUS_DELAY_MASK, 361
TC_PGCAL_STATUS_LEN, 362
TC_PGCAL_STATUS_OFFSET, 362

TC_PGCCTRL_AUTOCAL, 362
 TC_PGCCTRL_CALSTART, 362
 TC_PGCCTRL_DIR_CONV, 362
 TC_PGCCTRL_LEN, 362
 TC_PGCCTRL_OFFSET, 362
 TC_PGCCTRL_ON_TX, 362
 TC_PGCCTRL_TMEAS_MASK, 363
 TC_PGDELAY_CH1, 363
 TC_PGDELAY_CH2, 363
 TC_PGDELAY_CH3, 363
 TC_PGDELAY_CH4, 363
 TC_PGDELAY_CH5, 363
 TC_PGDELAY_CH7, 363
 TC_PGDELAY_LEN, 364
 TC_PGDELAY_OFFSET, 364
 TC_PGTTEST_CW, 364
 TC_PGTTEST_LEN, 364
 TC_PGTTEST_NORMAL, 364
 TC_PGTTEST_OFFSET, 364
 TC_SARL_SAR_LTEMP_OFFSET, 364
 TC_SARL_SAR_LVBAT_OFFSET, 365
 TC_SARL_SAR_C, 364
 TC_SARW_SAR_WTEMP_OFFSET, 365
 TC_SARW_SAR_WVBAT_OFFSET, 365
 TX_ANTD_ID, 365
 TX_ANTD_LEN, 365
 TX_ANTD_OFFSET, 365
 TX_BUFFER_ID, 365
 TX_BUFFER_LEN, 366
 TX_CAL_ID, 366
 TX_CAL_LEN, 366
 TX_FCTRL_FLE_MASK, 366
 TX_FCTRL_IFSDELAY_MASK, 366
 TX_FCTRL_ID, 366
 TX_FCTRL_LEN, 366
 TX_FCTRL_PE_MASK, 367
 TX_FCTRL_PE_SHFT, 367
 TX_FCTRL_SAFE_MASK_32, 367
 TX_FCTRL_TFLE_MASK, 367
 TX_FCTRL_TFLEN_MASK, 367
 TX_FCTRL_TR_SHFT, 367
 TX_FCTRL_TXBOFFS_MASK, 368
 TX_FCTRL_TXBOFFS_SHFT, 368
 TX_FCTRL_TXBR_110k, 368
 TX_FCTRL_TXBR_6M, 368
 TX_FCTRL_TXBR_850k, 368
 TX_FCTRL_TXBR_MASK, 368
 TX_FCTRL_TXBR_SHFT, 368
 TX_FCTRL_TXPRF_16M, 369
 TX_FCTRL_TXPRF_4M, 369
 TX_FCTRL_TXPRF_64M, 369
 TX_FCTRL_TXPRF_MASK, 369
 TX_FCTRL_TXPRF_SHFT, 369
 TX_FCTRL_TXPSR_MASK, 369
 TX_FCTRL_TXPSR_PE_1024, 369
 TX_FCTRL_TXPSR_PE_128, 370
 TX_FCTRL_TXPSR_PE_1536, 370
 TX_FCTRL_TXPSR_PE_16, 370
 TX_FCTRL_TXPSR_PE_2048, 370
 TX_FCTRL_TXPSR_PE_256, 370
 TX_FCTRL_TXPSR_PE_4096, 370
 TX_FCTRL_TXPSR_PE_512, 370
 TX_FCTRL_TXPSR_PE_64, 370
 TX_FCTRL_TXPSR_PE_MASK, 371
 TX_FCTRL_TXPSR_SHFT, 371
 TX_FCTRL_TR, 367
 TX_POWER_BOOSTNORM_MASK, 371
 TX_POWER_BOOSTNORM_SHIFT, 371
 TX_POWER_BOOSTP125_MASK, 371
 TX_POWER_BOOSTP125_SHIFT, 371
 TX_POWER_BOOSTP250_MASK, 371
 TX_POWER_BOOSTP250_SHIFT, 372
 TX_POWER_BOOSTP500_MASK, 372
 TX_POWER_BOOSTP500_SHIFT, 372
 TX_POWER_ID, 372
 TX_POWER_LEN, 372
 TX_POWER_MAN_DEFAULT, 372
 TX_POWER_TXPOWPHR_MASK, 372
 TX_POWER_TXPOWSD_MASK, 373
 TX_STAMP_LEN, 373
 TX_TIME_ID, 373
 TX_TIME_LLEN, 373
 TX_TIME_TX_RAWST_OFFSET, 373
 TX_TIME_TX_STAMP_LEN, 373
 TX_TIME_TX_STAMP_OFFSET, 373
 USR_SFD_ID, 374
 USR_SFD_LEN, 374
 W4R_TIM_MASK, 374
deca_sleep
 deca_device_api.h, 174
 transceiver_translator.c, 556
deca_types.h, 374
 _DECA_INT16_, 375
 _DECA_INT32_, 375
 _DECA_INT8_, 375
 _DECA_UINT16_, 376
 _DECA_UINT32_, 376
 _DECA_UINT8_, 376
 int16, 376
 int32, 376
 int8, 376
 NULL, 376
 uint16, 377
 uint32, 377
 uint8, 377
deca_version.h, 377
 DW1000_DEVICE_DRIVER_VER_STRING, 378
 DW1000_DRIVER_VERSION, 378
decalrqStatus_t
 deca_device_api.h, 173
decamutexoff
 deca_device_api.h, 175
 port.h, 450
decamutexon
 deca_device_api.h, 176
 port.h, 451

Desynchronize
 wub_main.c, 573

dev_addr_t
 mac_const.h, 441

diagnostic
 wub_main.c, 574

digital_bb_config
 deca_param_types.h, 224
 deca_params_init.c, 227

direct_src
 prot_packet_info_t, 52

drift
 sync_neighbour_t, 58

dst
 mac_buf_t, 43

dtune1
 deca_param_types.h, 225
 deca_params_init.c, 227

dwnsSFDlen
 deca_param_types.h, 225
 deca_params_init.c, 228

dwt_calcbandwidthtempadj
 deca_device.c, 115
 deca_device_api.h, 176

dwt_calcpgcount
 deca_device.c, 115
 deca_device_api.h, 177

dwt_calcpowertempadj
 deca_device.c, 116
 deca_device_api.h, 177

dwt_calibratesleepcnt
 deca_device.c, 116
 deca_device_api.h, 178

dwt_cb_data_t, 15
 datalength, 15
 fctrl, 15
 rx_flags, 15
 status, 15

dwt_cb_t
 deca_device_api.h, 173

dwt_checkirq
 deca_device.c, 116
 deca_device_api.h, 178

dwt_config
 transceiver_settings_t, 65

dwt_config_t, 15
 chan, 16
 dataRate, 16
 nsSFD, 16
 phrMode, 17
 prf, 17
 rxCode, 17
 rxPAC, 17
 sfdTO, 17
 txCode, 17
 txPreambLength, 18

dwt_configcontinuousframemode
 deca_device.c, 117

deca_device_api.h, 179

dwt_configcwmode
 deca_device.c, 117
 deca_device_api.h, 179

dwt_configeventcounters
 deca_device.c, 117
 deca_device_api.h, 179

dwt_configure
 deca_device.c, 118
 deca_device_api.h, 180

dwt_configuresleep
 deca_device.c, 118
 deca_device_api.h, 180

dwt_configuresleepcnt
 deca_device.c, 118
 deca_device_api.h, 180

dwt_configuretxrf
 deca_device.c, 119
 deca_device_api.h, 181

dwt_deviceentcnts_t, 18
 ARFE, 18
 CRCB, 18
 CRCG, 19
 HPW, 19
 OVER, 19
 PHE, 19
 PTO, 19
 RSL, 19
 RTO, 19
 SFDTO, 19
 TXF, 20
 TXW, 20

dwt_enableautoack
 deca_device.c, 119
 deca_device_api.h, 181

dwt_enableframefilter
 deca_device.c, 119
 deca_device_api.h, 181

dwt_entersleep
 deca_device.c, 120
 deca_device_api.h, 182

dwt_entersleepaftertx
 deca_device.c, 120
 deca_device_api.h, 182

dwt_forcetxoff
 deca_device.c, 121
 deca_device_api.h, 183

dwt_geteui
 deca_device.c, 121
 deca_device_api.h, 183

dwt_getinitxtaltrim
 deca_device.c, 122
 deca_device_api.h, 184

dwt_getlotid
 deca_device.c, 122
 deca_device_api.h, 184

dwt_getpartid
 deca_device.c, 122

dwt_readdiagnostics
 deca_device.c, 128
 deca_device_api.h, 190
dwt_readeventcounters
 deca_device.c, 129
 deca_device_api.h, 191
dwt_readfromdevice
 deca_device.c, 129
 deca_device_api.h, 191
dwt_readrxdata
 deca_device.c, 131
 deca_device_api.h, 193
dwt_readrxtimestamp
 deca_device.c, 132
 deca_device_api.h, 194
dwt_readrxtimestamphi32
 deca_device.c, 132
 deca_device_api.h, 194
dwt_readrxtimestamplo32
 deca_device.c, 133
 deca_device_api.h, 195
dwt_readsystime
 deca_device.c, 133
 deca_device_api.h, 195
dwt_readsystimestamphi32
 deca_device.c, 134
 deca_device_api.h, 196
dwt_readtempvbat
 deca_device.c, 134
 deca_device_api.h, 196
dwt_readtxtimestamp
 deca_device.c, 134
 deca_device_api.h, 196
dwt_readtxtimestamphi32
 deca_device.c, 135
 deca_device_api.h, 197
dwt_readtxtimestamplo32
 deca_device.c, 135
 deca_device_api.h, 197
dwt_readwakeuptemp
 deca_device.c, 136
 deca_device_api.h, 198
dwt_readwakeupsbat
 deca_device.c, 136
 deca_device_api.h, 198
dwt_rxdiag_t, 23
 firstPath, 23
 firstPathAmp1, 24
 firstPathAmp2, 24
 firstPathAmp3, 24
 maxGrowthCIR, 24
 maxNoise, 24
 rxPreamCount, 24
 stdNoise, 24
dwt_rxenable
 deca_device.c, 136
 deca_device_api.h, 198
dwt_rxreset

deca_device.c, 137
deca_device_api.h, 199
dwt_setaddress16
 deca_device.c, 138
 deca_device_api.h, 200
dwt_setcallbacks
 deca_device.c, 138
 deca_device_api.h, 200
dwt_setdblrxbuffmode
 deca_device.c, 139
 deca_device_api.h, 201
dwt_setdelayedtrxtime
 deca_device.c, 139
 deca_device_api.h, 201
dwt_seteui
 deca_device.c, 139
 deca_device_api.h, 201
dwt_setfinegraintxseq
 deca_device.c, 140
 deca_device_api.h, 202
dwt_setgpiodirection
 deca_device.c, 140
 deca_device_api.h, 202
dwt_setgpiovalue
 deca_device.c, 141
 deca_device_api.h, 203
dwt_setinterrupt
 deca_device.c, 141
 deca_device_api.h, 203
dwt_setleds
 deca_device.c, 142
 deca_device_api.h, 204
dwt_setlnapemode
 deca_device.c, 142
 deca_device_api.h, 204
dwt_setlocaldataptr
 deca_device.c, 143
 deca_device_api.h, 205
dwt_setlowpowerlistening
 deca_device.c, 143
 deca_device_api.h, 205
dwt_setpanid
 deca_device.c, 143
 deca_device_api.h, 205
dwt_setpreambledetecttimeout
 deca_device.c, 144
 deca_device_api.h, 206
dwt_setrxaftertxdelay
 deca_device.c, 144
 deca_device_api.h, 206
dwt_setrxantennadelay
 deca_device.c, 145
 deca_device_api.h, 207
dwt_setrxtimeout
 deca_device.c, 145
 deca_device_api.h, 207
dwt_setsmarttxpower
 deca_device.c, 146
 deca_device_api.h, 208
dwt_setsniffmode
 deca_device.c, 146
 deca_device_api.h, 208
dwt_setsnoozetime
 deca_device.c, 147
 deca_device_api.h, 209
dwt_settxantennadelay
 deca_device.c, 147
 deca_device_api.h, 209
dwt_setxtaltrim
 deca_device.c, 148
 deca_device_api.h, 210
dwt_softreset
 deca_device.c, 148
 deca_device_api.h, 210
dwt_spicswakeup
 deca_device.c, 149
 deca_device_api.h, 211
dwt_starttx
 deca_device.c, 149
 deca_device_api.h, 211
dwt_syncrxbufptrs
 deca_device.c, 150
 deca_device_api.h, 212
dwt_txconfig
 transceiver_settings_t, 65
dwt_txconfig_t, 25
 PGdly, 25
 power, 25
dwt_write16bitoffsetreg
 deca_device.c, 151
 deca_device_api.h, 213
dwt_write32bitoffsetreg
 deca_device.c, 152
 deca_device_api.h, 214
dwt_write32bitreg
 deca_device_api.h, 172
dwt_write8bitoffsetreg
 deca_device.c, 153
 deca_device_api.h, 215
dwt_writetodevice
 deca_device.c, 153
 deca_device_api.h, 215
dwt_writetxdata
 deca_device.c, 155
 deca_device_api.h, 217
dwt_writetxfctrl
 deca_device.c, 156
 deca_device_api.h, 218

EC_CTRL_LEN
 deca_regs.h, 275
EC_CTRL_MASK
 deca_regs.h, 275
EC_CTRL_OFFSET
 deca_regs.h, 275
EC_CTRL_OSRSRM
 deca_regs.h, 275

EC_CTRL_OSTRM
 deca_regs.h, 275
EC_CTRL_OSTSM
 deca_regs.h, 275
EC_CTRL_PLLCK
 deca_regs.h, 276
EC_CTRL_WAIT_MASK
 deca_regs.h, 276
EC_GOLP_LEN
 deca_regs.h, 276
EC_GOLP_MASK
 deca_regs.h, 276
EC_GOLP_OFFSET_EXT_MASK
 deca_regs.h, 276
EC_GOLP
 deca_regs.h, 276
EC_RXTC_LEN
 deca_regs.h, 276
EC_RXTC_MASK
 deca_regs.h, 277
EC_RXTC_OFFSET
 deca_regs.h, 277
ENABLE_ALL_SEQ
 deca_device.c, 107
EUI_64_ID
 deca_regs.h, 277
EUI_64_LEN
 deca_regs.h, 277
EUI_64_OFFSET
 deca_regs.h, 277
EVC_CLR
 deca_regs.h, 277
EVC_CTRL_LEN
 deca_regs.h, 277
EVC_CTRL_MASK
 deca_regs.h, 278
EVC_CTRL_OFFSET
 deca_regs.h, 278
EVC_EN
 deca_regs.h, 278
EVC_FCE_LEN
 deca_regs.h, 278
EVC_FCE_MASK
 deca_regs.h, 278
EVC_FCE_OFFSET
 deca_regs.h, 278
EVC_FCG_LEN
 deca_regs.h, 278
EVC_FCG_MASK
 deca_regs.h, 279
EVC_FCG_OFFSET
 deca_regs.h, 279
EVC_FFR_LEN
 deca_regs.h, 279
EVC_FFR_MASK
 deca_regs.h, 279
EVC_FFR_OFFSET
 deca_regs.h, 279

EVC_FWTO_LEN
 deca_regs.h, 279
EVC_FWTO_MASK
 deca_regs.h, 279
EVC_FWTO_OFFSET
 deca_regs.h, 280
EVC_HPW_LEN
 deca_regs.h, 280
EVC_HPW_MASK
 deca_regs.h, 280
EVC_HPW_OFFSET
 deca_regs.h, 280
EVC_OVR_LEN
 deca_regs.h, 280
EVC_OVR_MASK
 deca_regs.h, 280, 281
EVC_OVR_OFFSET
 deca_regs.h, 281
EVC_PHE_LEN
 deca_regs.h, 281
EVC_PHE_MASK
 deca_regs.h, 281
EVC_PHE_OFFSET
 deca_regs.h, 281
EVC_PTO_LEN
 deca_regs.h, 281
EVC_PTO_MASK
 deca_regs.h, 281
EVC_PTO_OFFSET
 deca_regs.h, 282
EVC_RES1_OFFSET
 deca_regs.h, 282
EVC_RSE_LEN
 deca_regs.h, 282
EVC_RSE_MASK
 deca_regs.h, 282
EVC_RSE_OFFSET
 deca_regs.h, 282
EVC_STO_OFFSET
 deca_regs.h, 282
EVC_TPW_LEN
 deca_regs.h, 282
EVC_TPW_MASK
 deca_regs.h, 283
EVC_TPW_OFFSET
 deca_regs.h, 283
EVC_TXFS_LEN
 deca_regs.h, 283
EVC_TXFS_MASK
 deca_regs.h, 283
EVC_TXFS_OFFSET
 deca_regs.h, 283
EXT_SYNC_ID
 deca_regs.h, 283
EXT_SYNC_LEN
 deca_regs.h, 283
end
 txt_buf_t, 66

err_cnt
 FC_STAT_s, 28
extra
 FU_prot, 39

f_hash
 settings_version_t, 55
f_major
 settings_version_t, 55
f_minor
 settings_version_t, 56
FC_BEACON_cb
 bin_parser_cb.c, 83
FC_BEACON_s, 25
 FC, 26
 len, 26
 serial, 26
FC_CARRY_s, 26
 flag_hops, 27
 hops, 27
 src_addr, 27
FC_STAT_ASK_cb
 bin_parser_cb.c, 83
FC_STAT_s, 28
 battery_mV, 28
 err_cnt, 28
 FC, 28
 len, 29
 rx_cnt, 29
 to_cnt, 29
 tx_cnt, 29
FC_SYNC_FIN_cb
 sync.c, 480
FC_SYNC_FIN_s, 29
 FC, 30
 len, 30
 slot_num, 30
 tree_level, 30
 TsFinTxBuf, 30
 TsOffset, 31
 TsPollTx, 31
 TsRespRx, 31
FC_SYNC_POLL_cb
 sync.c, 480
FC_SYNC_POLL_s, 31
 FC, 32
 len, 32
 num_poll_anchor, 32
 poll_addr, 32
FC_SYNC_RESP_cb
 sync.c, 481
FC_SYNC_RESP_s, 32
 FC, 33
 len, 33
 TsPollRx, 33
 TsRespTx, 33
FC_TURN_OFF_s, 33
 FC, 34
 len, 34

 reason, 34
FC_TURN_ON_cb
 bin_parser_cb.c, 84
FC_TURN_ON_s, 34
 FC, 35
 len, 35
FC_VERSION_ASK_cb
 bin_parser_cb.c, 84
FC_VERSION_s, 35
 fMajor, 36
 fMinor, 36
 FC, 36
 hMajor, 36
 hMinor, 36
 hash, 36
 len, 36
 role, 37
 serial, 37
FC_t
 bin_const.h, 74
FCTRL_ACK_REQ_MASK
 deca_device.c, 107
FCTRL_LEN_MAX
 deca_device.c, 107
fMajor
 FC_VERSION_s, 36
fMinor
 FC_VERSION_s, 36
FORCE_LDE
 deca_device.c, 107
FORCE OTP OFF
 deca_device.c, 107
FORCE OTP ON
 deca_device.c, 107
FORCE_SYS_PLL
 deca_device.c, 107
FORCE_SYS_XTI
 deca_device.c, 107
FORCE_TX_PLL
 deca_device.c, 108
FP_AMPL2_MASK
 deca_regs.h, 284
FP_AMPL2_SHIFT
 deca_regs.h, 284
FP_AMPL3_MASK
 deca_regs.h, 284
FP_AMPL3_SHIFT
 deca_regs.h, 284
FR_CR_ACK
 mac_const.h, 439
FR_CR_BEACON
 mac_const.h, 439
FR_CR_DATA
 mac_const.h, 439
FR_CR_MAC
 mac_const.h, 439
FR_CR_PAN
 mac_const.h, 440

FR_CR_PENDING
 mac_const.h, 440
FR_CR_RACK
 mac_const.h, 440
FR_CR_SECURE
 mac_const.h, 440
FR_CR_TYPE_MASK
 mac_const.h, 440
FR_CRH_DA_LONG
 mac_const.h, 440
FR_CRH_DA_NONE
 mac_const.h, 440
FR_CRH_DA_SHORT
 mac_const.h, 440
FR_CRH_FVER0
 mac_const.h, 441
FR_CRH_FVER1
 mac_const.h, 441
FR_CRH_SA_LONG
 mac_const.h, 441
FR_CRH_SA_NONE
 mac_const.h, 441
FR_CRH_SA_SHORT
 mac_const.h, 441
FREQ_OFFSET_MULTIPLIER_110KB
 deca_device_api.h, 172
FREQ_OFFSET_MULTIPLIER
 deca_device_api.h, 172
FS_CTRL_ID
 deca_regs.h, 284
FS_CTRL_LEN
 deca_regs.h, 284
FS_PLLCFG_CH1
 deca_regs.h, 284
FS_PLLCFG_CH2
 deca_regs.h, 285
FS_PLLCFG_CH3
 deca_regs.h, 285
FS_PLLCFG_CH4
 deca_regs.h, 285
FS_PLLCFG_CH5
 deca_regs.h, 285
FS_PLLCFG_CH7
 deca_regs.h, 285
FS_PLLCFG_LEN
 deca_regs.h, 285
FS_PLLCFG_OFFSET
 deca_regs.h, 285
FS_PLLTUNE_CH1
 deca_regs.h, 285
FS_PLLTUNE_CH2
 deca_regs.h, 286
FS_PLLTUNE_CH3
 deca_regs.h, 286
FS_PLLTUNE_CH4
 deca_regs.h, 286
FS_PLLTUNE_CH5
 deca_regs.h, 286

FS_PLLTUNE_CH7
 deca_regs.h, 286
FS_PLLTUNE_LEN
 deca_regs.h, 286
FS_PLLTUNE_OFFSET
 deca_regs.h, 286
FS_RES1_LEN
 deca_regs.h, 286
FS_RES1_OFFSET
 deca_regs.h, 287
FS_RES2_LEN
 deca_regs.h, 287
FS_RES2_OFFSET
 deca_regs.h, 287
FS_RES3_LEN
 deca_regs.h, 287
FS_RES3_OFFSET
 deca_regs.h, 287
FS_XTALT_LEN
 deca_regs.h, 287
FS_XTALT_MASK
 deca_regs.h, 287
FS_XTALT_MIDRANGE
 deca_regs.h, 288
FS_XTALT_OFFSET
 deca_regs.h, 288
FU_ASSERT
 fu.h, 387
FU_AcceptFirmware
 fu.c, 379
 fu.h, 392
FU_AcceptFirmwareVersion
 fu.c, 379
FU_AcceptHardwareVersion
 fu.c, 379
FU_BLOCK_SIZE
 fu.h, 388
FU_DESTINATION_1
 fu.h, 388
FU_DESTINATION_2
 fu.h, 388
FU_ERR_BAD_F_VERSION
 fu.h, 388
FU_ERR_BAD_FILE_SIZE
 fu.h, 388
FU_ERR_BAD_FLASH_CRC
 fu.h, 388
FU_ERR_BAD_FRAME_CRC
 fu.h, 388
FU_ERR_BAD_FRAME_HASH
 fu.h, 389
FU_ERR_BAD_FRAME_LEN
 fu.h, 389
FU_ERR_BAD_H_VERSION
 fu.h, 389
FU_ERR_BAD_OFFSET
 fu.h, 389
FU_ERR_BAD_OPCODE_SET

fu.h, 389
FU_ERR_BAD_PROT_VER
 fu.h, 389
FU_ERR_FIR_NOT_ACCEPTED_YET
 fu.h, 390
FU_ERR_FLASH_ERASING
 fu.h, 390
FU_ERR_FLASH_WRITING
 fu.h, 390
FU_ERR_VERSION_IN_PACKAGE
 fu.h, 390
FU_GetCurrentFlashBase
 fu.c, 380
 fu.h, 392
FU_HandleAsDevice
 fu.c, 380
 fu.h, 393
FU_Init
 fu.c, 381
 fu.h, 394
FU_MAX_DATA_SIZE
 fu.h, 390
FU_MAX_PROGRAM_SIZE
 fu.h, 391
FU_OPCODE_ABORT
 fu.h, 391
FU_OPCODE_ACK
 fu.h, 391
FU_OPCODE_DATA
 fu.h, 391
FU_OPCODE_EOT
 fu.h, 391
FU_OPCODE_SOT
 fu.h, 391
FU PROT_HEAD_SIZE
 fu.h, 392
FU PROT_VERSION
 fu.h, 392
FU_SOT_prot, 40
 FC, 40
 fileSize, 41
 firmwareCRC, 41
 frameCRC, 41
 frameLen, 41
 fversion, 41
 hash, 41
 opcode, 42
FU_VERSION_LOC
 fu.h, 392
FU_instance_t, 37
 fileSize, 37
 newCrc, 37
 newHash, 38
 newVer, 38
 sesionPacketCounter, 38
FU_prot, 38
 data, 39
 extra, 39
 FC, 39
 frameLen, 39
 hash, 39
 opcode, 39
fail_cnt
 carry_trace_t, 14
FC
 FC_BEACON_s, 26
 FC_STAT_s, 28
 FC_SYNC_FIN_s, 30
 FC_SYNC_POLL_s, 32
 FC_SYNC RESP_s, 33
 FC_TURN_OFF_s, 34
 FC_TURN_ON_s, 35
 FC_VERSION_s, 36
 FU_SOT_prot, 40
 FU_prot, 39
 prot_cb_t, 50
fctrl
 dwt_cb_data_t, 15
fileSize
 FU_SOT_prot, 41
 FU_instance_t, 37
fin_dly_us
 toa_settings_t, 63
firmwareCRC
 FU_SOT_prot, 41
firstPath
 dwt_rxdiag_t, 23
firstPathAmp1
 dwt_rxdiag_t, 24
firstPathAmp2
 dwt_rxdiag_t, 24
firstPathAmp3
 dwt_rxdiag_t, 24
flag_hops
 FC_CARRY_s, 27
frame
 mac_buf_t, 43
frame_under_tx_is_ranging
 mac_instance_t, 46
frameCRC
 FU_SOT_prot, 41
frameLen
 FU_SOT_prot, 41
 FU_prot, 39
fs_pll_cfg
 deca_param_types.h, 225
 deca_params_init.c, 228
fs_pll_tune
 deca_param_types.h, 225
 deca_params_init.c, 228
fu.c, 378
 FU_AcceptFirmware, 379
 FU_AcceptFirmwareVersion, 379
 FU_AcceptHardwareVersion, 379
 FU_GetCurrentFlashBase, 380
 FU_HandleAsDevice, 380

FU_Init, 381
fu.h, 381
 FU_ASSERT, 387
 FU_AcceptFirmware, 392
 FU_BLOCK_SIZE, 388
 FU_DESTINATION_1, 388
 FU_DESTINATION_2, 388
 FU_ERR_BAD_F_VERSION, 388
 FU_ERR_BAD_FILE_SIZE, 388
 FU_ERR_BAD_FLASH_CRC, 388
 FU_ERR_BAD_FRAME_CRC, 388
 FU_ERR_BAD_FRAME_HASH, 389
 FU_ERR_BAD_FRAME_LEN, 389
 FU_ERR_BAD_H_VERSION, 389
 FU_ERR_BAD_OFFSET, 389
 FU_ERR_BAD_OPCODE_SET, 389
 FU_ERR_BAD_PROT_VER, 389
 FU_ERR_FIR_NOT_ACCEPTED_YET, 390
 FU_ERR_FLASH_ERASING, 390
 FU_ERR_FLASH_WRITING, 390
 FU_ERR_VERSION_IN_PACKAGE, 390
 FU_GetCurrentFlashBase, 392
 FU_HandleAsDevice, 393
 FU_Init, 394
 FU_MAX_DATA_SIZE, 390
 FU_MAX_PROGRAM_SIZE, 391
 FU_OPCODE_ABORT, 391
 FU_OPCODE_ACK, 391
 FU_OPCODE_DATA, 391
 FU_OPCODE_EOT, 391
 FU_OPCODE_SOT, 391
 FU_PROT_HEAD_SIZE, 392
 FU_PROT_VERSION, 392
 FU_VERSION_LOC, 392
 PROG_DESTINATION1, 394
 PROG_DESTINATION2, 394
 PROG_SETTINGS1, 394
 PROG_START1, 395
fversion
 FU_SOT_prot, 41

GDM0
 deca_regs.h, 288
GDM1
 deca_regs.h, 288
GDM2
 deca_regs.h, 288
GDM3
 deca_regs.h, 288
GDM4
 deca_regs.h, 288
GDM5
 deca_regs.h, 289
GDM6
 deca_regs.h, 289
GDM7
 deca_regs.h, 289
GDM8
 deca_regs.h, 289

GDP0
 deca_regs.h, 289
GDP1
 deca_regs.h, 289
GDP2
 deca_regs.h, 289
GDP3
 deca_regs.h, 290
GDP4
 deca_regs.h, 290
GDP5
 deca_regs.h, 290
GDP6
 deca_regs.h, 290
GDP7
 deca_regs.h, 290
GDP8
 deca_regs.h, 290

GIBES0
 deca_regs.h, 290
GIBES1
 deca_regs.h, 290
GIBES2
 deca_regs.h, 291
GIBES3
 deca_regs.h, 291
GIBES4
 deca_regs.h, 291
GIBES5
 deca_regs.h, 291
GIBES6
 deca_regs.h, 291
GIBES7
 deca_regs.h, 291
GIBES8
 deca_regs.h, 291

GICLR0
 deca_regs.h, 291
GICLR1
 deca_regs.h, 292
GICLR2
 deca_regs.h, 292
GICLR3
 deca_regs.h, 292
GICLR4
 deca_regs.h, 292
GICLR5
 deca_regs.h, 292
GICLR6
 deca_regs.h, 292
GICLR7
 deca_regs.h, 292
GICLR8
 deca_regs.h, 293

GIDBE0
 deca_regs.h, 293
GIDBE1
 deca_regs.h, 293

GIDBE2
 deca_regs.h, 293
GIDBE3
 deca_regs.h, 293
GIDBE4
 deca_regs.h, 293
GIDBE5
 deca_regs.h, 293
GIDBE6
 deca_regs.h, 294
GIDBE7
 deca_regs.h, 294
GIDBE8
 deca_regs.h, 294
GIMOD0
 deca_regs.h, 294
GIMOD1
 deca_regs.h, 294
GIMOD2
 deca_regs.h, 294
GIMOD3
 deca_regs.h, 294
GIMOD4
 deca_regs.h, 295
GIMOD5
 deca_regs.h, 295
GIMOD6
 deca_regs.h, 295
GIMOD7
 deca_regs.h, 295
GIMOD8
 deca_regs.h, 295
GIRQE0
 deca_regs.h, 295
GIRQE1
 deca_regs.h, 295
GIRQE2
 deca_regs.h, 295
GIRQE3
 deca_regs.h, 296
GIRQE4
 deca_regs.h, 296
GIRQE5
 deca_regs.h, 296
GIRQE6
 deca_regs.h, 296
GIRQE7
 deca_regs.h, 296
GIRQE8
 deca_regs.h, 296
GIRQx0
 deca_regs.h, 296
GIRQx1
 deca_regs.h, 296
GIRQx2
 deca_regs.h, 297
GIRQx3
 deca_regs.h, 297
GIRQx4
 deca_regs.h, 297
GIRQx5
 deca_regs.h, 297
GIRQx6
 deca_regs.h, 297
GIRQx7
 deca_regs.h, 297
GIRQx8
 deca_regs.h, 297
GISEN0
 deca_regs.h, 297
GISEN1
 deca_regs.h, 298
GISEN2
 deca_regs.h, 298
GISEN3
 deca_regs.h, 298
GISEN4
 deca_regs.h, 298
GISEN5
 deca_regs.h, 298
GISEN6
 deca_regs.h, 298
GISEN7
 deca_regs.h, 298
GISEN8
 deca_regs.h, 299
GPIO_CTRL_ID
 deca_regs.h, 299
GPIO_CTRL_LEN
 deca_regs.h, 299
GPIO_DIR_LEN
 deca_regs.h, 299
GPIO_DIR_MASK
 deca_regs.h, 299
GPIO_DIR_OFFSET
 deca_regs.h, 299
GPIO_DOUT_LEN
 deca_regs.h, 299
GPIO_DOUT_MASK
 deca_regs.h, 300
GPIO_DOUT_OFFSET
 deca_regs.h, 300
GPIO_IBES_LEN
 deca_regs.h, 300
GPIO_IBES_MASK
 deca_regs.h, 300
GPIO_IBES_OFFSET
 deca_regs.h, 300
GPIO_ICLR_LEN
 deca_regs.h, 300
GPIO_ICLR_MASK
 deca_regs.h, 300
GPIO_ICLR_OFFSET
 deca_regs.h, 301
GPIO_IDBE_LEN
 deca_regs.h, 301

GPIO_IDBE_MASK
 deca_regs.h, 301
GPIO_IDBE_OFFSET
 deca_regs.h, 301
GPIO_IMODE_LEN
 deca_regs.h, 301
GPIO_IMODE_MASK
 deca_regs.h, 301
GPIO_IMODE_OFFSET
 deca_regs.h, 301
GPIO_IRQE_LEN
 deca_regs.h, 302
GPIO_IRQE_MASK
 deca_regs.h, 302
GPIO_IRQE_OFFSET
 deca_regs.h, 302
GPIO_ISEN_LEN
 deca_regs.h, 302
GPIO_ISEN_MASK
 deca_regs.h, 302
GPIO_ISEN_OFFSET
 deca_regs.h, 302
GPIO_MODE_LEN
 deca_regs.h, 302
GPIO_MODE_MASK
 deca_regs.h, 303
GPIO_MODE_OFFSET
 deca_regs.h, 303
GPIO_MSGP0_MASK
 deca_regs.h, 303
GPIO_MSGP1_MASK
 deca_regs.h, 303
GPIO_MSGP2_MASK
 deca_regs.h, 303
GPIO_MSGP3_MASK
 deca_regs.h, 303
GPIO_MSGP4_MASK
 deca_regs.h, 303
GPIO_MSGP5_MASK
 deca_regs.h, 303
GPIO_MSGP6_MASK
 deca_regs.h, 304
GPIO_MSGP7_MASK
 deca_regs.h, 304
GPIO_MSGP8_MASK
 deca_regs.h, 304
GPIO_PIN2_RXLED
 deca_regs.h, 304
GPIO_PIN3_TXLED
 deca_regs.h, 304
GPIO_PIN4_EXTPA
 deca_regs.h, 304
GPIO_PIN5_EXTTXE
 deca_regs.h, 304
GPIO_PIN6_EXTRXE
 deca_regs.h, 304
GPIO_RAW_LEN
 deca_regs.h, 305

GPIO_RAW_MASK
 deca_regs.h, 305
GPIO_RAW_OFFSET
 deca_regs.h, 305
GRAWP0
 deca_regs.h, 305
GRAWP1
 deca_regs.h, 305
GRAWP2
 deca_regs.h, 305
GRAWP3
 deca_regs.h, 305
GRAWP4
 deca_regs.h, 306
GRAWP5
 deca_regs.h, 306
GRAWP6
 deca_regs.h, 306
GRAWP7
 deca_regs.h, 306
GRAWP8
 deca_regs.h, 306
gitversion.h, 395
 __F_HASH__, 396
 __F_MAJOR__, 396
 __F_MINOR__, 396
guard_time_us
 toa_settings_t, 63
GxM0
 deca_regs.h, 306
GxM1
 deca_regs.h, 306
GxM2
 deca_regs.h, 306
GxM3
 deca_regs.h, 307
GxM4
 deca_regs.h, 307
GxM5
 deca_regs.h, 307
GxM6
 deca_regs.h, 307
GxM7
 deca_regs.h, 307
GxM8
 deca_regs.h, 307
GxP0
 deca_regs.h, 307
GxP1
 deca_regs.h, 307
GxP2
 deca_regs.h, 308
GxP3
 deca_regs.h, 308
GxP4
 deca_regs.h, 308
GxP5
 deca_regs.h, 308

GxP6
 deca_regs.h, 308
GxP7
 deca_regs.h, 308
GxP8
 deca_regs.h, 308

H_MAJOR_CALC
 settings.h, 476
H_VERSION_CALC
 settings.h, 476
h_major
 settings_otp_t, 52
h_minor
 settings_otp_t, 53
h_version
 settings_version_t, 56

HERTZ_TO_PPM_MULTIPLIER_CHAN_1
 deca_device_api.h, 173
HERTZ_TO_PPM_MULTIPLIER_CHAN_2
 deca_device_api.h, 173
HERTZ_TO_PPM_MULTIPLIER_CHAN_3
 deca_device_api.h, 173
HERTZ_TO_PPM_MULTIPLIER_CHAN_5
 deca_device_api.h, 173

hMajor
 FC_VERSION_s, 36
hMinor
 FC_VERSION_s, 36
HPW
 dwt_deviceentcnts_t, 19
hash
 FC_VERSION_s, 36
 FU_SOT_prot, 41
 FU_prot, 39
hops
 FC_CARRY_s, 27

IASSERT
 iassert.h, 397
INCREMENT_CYCLE
 tools.h, 527
INCREMENT_MOD
 tools.h, 527
iassert.h, 396
 IASSERT, 397
 PORT_iassert_fun, 397
init_xtrim
 dwt_local_data_t, 22
initiator
 toa_core_t, 61
int16
 deca_device_api.h, 173
 deca_types.h, 376
int32
 deca_device_api.h, 174
 deca_types.h, 376
int8
 deca_device_api.h, 174

 deca_types.h, 376
isConnectedToServer
 carry_instance_t, 10
isRangingFrame
 mac_buf_t, 43

 LDE_CFG1_LEN
 deca_regs.h, 308
 LDE_CFG1_NSTDEV_MASK
 deca_regs.h, 309
 LDE_CFG1_OFFSET
 deca_regs.h, 309
 LDE_CFG1_PMULT_MASK
 deca_regs.h, 309
 LDE_CFG2_LEN
 deca_regs.h, 309
 LDE_CFG2_OFFSET
 deca_regs.h, 309
 LDE_IF_ID
 deca_regs.h, 309
 LDE_IF_LEN
 deca_regs.h, 309
 LDE_PARAM1
 deca_param_types.h, 222
 LDE_PARAM3_16
 deca_param_types.h, 222
 LDE_PARAM3_64
 deca_param_types.h, 222
 LDE_PPAMPL_LEN
 deca_regs.h, 310
 LDE_PPAMPL_OFFSET
 deca_regs.h, 310
 LDE_PPINDEX_LEN
 deca_regs.h, 310
 LDE_PPINDEX_OFFSET
 deca_regs.h, 310
 LDE_REPC_LEN
 deca_regs.h, 310
 LDE_REPC_OFFSET
 deca_regs.h, 310
 LDE_REPCPCODE_1
 deca_regs.h, 310
 LDE_REPCPCODE_10
 deca_regs.h, 311
 LDE_REPCPCODE_11
 deca_regs.h, 311
 LDE_REPCPCODE_12
 deca_regs.h, 311
 LDE_REPCPCODE_13
 deca_regs.h, 311
 LDE_REPCPCODE_14
 deca_regs.h, 311
 LDE_REPCPCODE_15
 deca_regs.h, 311
 LDE_REPCPCODE_16
 deca_regs.h, 311
 LDE_REPCPCODE_17
 deca_regs.h, 311
 LDE_REPCPCODE_18

deca_regs.h, 312
 LDE_REPCPCODE_19
 deca_regs.h, 312
 LDE_REPCPCODE_20
 deca_regs.h, 312
 LDE_REPCPCODE_21
 deca_regs.h, 312
 LDE_REPCPCODE_22
 deca_regs.h, 312
 LDE_REPCPCODE_23
 deca_regs.h, 312
 LDE_REPCPCODE_24
 deca_regs.h, 312
 LDE_REPCPCODE_3
 deca_regs.h, 313
 LDE_REPCPCODE_4
 deca_regs.h, 313
 LDE_REPCPCODE_5
 deca_regs.h, 313
 LDE_REPCPCODE_6
 deca_regs.h, 313
 LDE_RXANTD_OFFSET
 deca_regs.h, 314
 LDE_THRESH_LEN
 deca_regs.h, 314
 LDE_THRESH_OFFSET
 deca_regs.h, 314
 LDOTUNE_ADDRESS
 deca_device.c, 108
 LISTENER_CASE
 listener_parser.c, 398
 LOG_BASE64
 logs.h, 402
 LOG_Bin
 logs.h, 403
 LOG_CRIT
 logs.h, 402
 LOG_DBG
 logs.h, 402
 LOG_ERR
 logs.h, 402
 LOG_INF
 logs.h, 402
 LOG_TEST
 logs.h, 403
 LOG_Text
 logs.h, 404
 LOG_WRN
 logs.h, 403
 logs.h, 403
 LOTID_ADDRESS
 deca_device.c, 108
 last_rx_ts
 mac_instance_t, 46
 last_update_time
 carry_target_t, 13
 carry_trace_t, 14
 mac_buf_t, 44
 lde_replicaCoeff
 deca_param_types.h, 225
 deca_params_init.c, 228
 len
 FC_BEACON_s, 26
 FC_STAT_s, 29
 FC_SYNC_FIN_s, 30
 FC_SYNC_POLL_s, 32
 FC_SYNC RESP_s, 33
 FC_TURN_OFF_s, 34
 FC_TURN_ON_s, 35
 FC_VERSION_s, 36
 listener_isr
 listener_parser.c, 398
 mac.h, 424
 listener_parse
 listener_parser.c, 399
 listener_parser.c, 397
 LISTENER_CASE, 398
 listener_isr, 398
 listener_parse, 399
 lo32
 agc_cfg_struct, 9
 local_obj
 sync_instance_t, 57
 logs.h, 400
 LOG_BASE64, 402
 LOG_Bin, 403
 LOG_CRIT, 402
 LOG_DBG, 402
 LOG_ERR, 402
 LOG_INF, 402
 LOG_TEST, 403
 LOG_Text, 404
 LOG_WRN, 403
 longFrames
 dwt_local_data_t, 22
 lotID
 dwt_local_data_t, 22
 low_power_mode
 transceiver_settings_t, 65
 MAC_ASSERT
 mac_port.h, 443
 MAC_AckFramelsr
 mac.c, 407
 mac.h, 425
 MAC_BUF_CNT
 mac_settings.h, 446
 MAC_BUF_LEN

mac_settings.h, 446
MAC_BeaconTimerGetMs
mac.c, 407
mac.h, 425
MAC_BeaconTimerReset
mac.c, 408
mac.h, 426
MAC_BufLen
mac.c, 410
mac.h, 429
MAC_Buffer
mac.c, 409
mac.h, 427
MAC_BufferPrepare
mac.c, 409
mac.h, 428
MAC_FillFrameTo
mac.c, 411
mac.h, 429
MAC_Free
mac.c, 412
mac.h, 430
MAC_HEAD_LENGTH
mac.h, 424
MAC_Init
mac.c, 412
mac.h, 431
MAC_LOG_ERR
mac_port.h, 443
MAC_Read
mac.c, 413
mac.h, 431
MAC_Read8
mac.c, 413
mac.h, 432
MAC_SETTINGS_DEF
mac_settings.h, 446
MAC_Send
mac.c, 414
mac.h, 432
MAC_SendRanging
mac.c, 415
mac.h, 433
MAC_SetFrameType
mac.c, 415
mac.h, 434
MAC_TRACE_ENABLED
mac.h, 424
MAC_TRACE
mac.h, 424
MAC_ToSlotsTime
mac.h, 435
MAC_ToSlotsTimeUs
mac.c, 416
MAC_TryTransmitFrameInSlot
mac.c, 416
MAC_USAGE_BUF_START
mac.h, 424
MAC_USAGE_BUF_STOP
mac.h, 424
MAC_UpdateSlotTimer
mac.c, 417
MAC_UsFromRx
mac.c, 418
mac.h, 435
MAC_Write
mac.c, 418
mac.h, 436
MAC_Write8
mac.c, 419
mac.h, 436
MAC_YourSlotIsr
mac.c, 420
mac.h, 437
MASK_40BIT
transceiver.h, 543
MIXER_GAIN_STEP
deca_param_types.h, 223
MOVE_ARRAY_ELEM
sync.c, 479
mac
mac.c, 420
settings_t, 54
sync.c, 492
mac.c, 404
_MAC_BufGetOldestToTx, 406
MAC_BufferReset, 406
MAC_AckFramesr, 407
MAC_BeaconTimerGetMs, 407
MAC_BeaconTimerReset, 408
MAC_BufLen, 410
MAC_Buffer, 409
MAC_BufferPrepare, 409
MAC_FillFrameTo, 411
MAC_Free, 412
MAC_Init, 412
MAC_Read, 413
MAC_Read8, 413
MAC_Send, 414
MAC_SendRanging, 415
MAC_SetFrameType, 415
MAC_ToSlotsTimeUs, 416
MAC_TryTransmitFrameInSlot, 416
MAC_UpdateSlotTimer, 417
MAC_UsFromRx, 418
MAC_Write, 418
MAC_Write8, 419
MAC_YourSlotIsr, 420
mac, 420
mac.h, 421
listener_isr, 424
MAC_AckFramesr, 425
MAC_BeaconTimerGetMs, 425
MAC_BeaconTimerReset, 426
MAC_BufLen, 429
MAC_Buffer, 427

MAC_BufferPrepare, 428
 MAC_FillFrameTo, 429
 MAC_Free, 430
 MAC_HEAD_LENGTH, 424
 MAC_Init, 431
 MAC_Read, 431
 MAC_Read8, 432
 MAC_Send, 432
 MAC_SendRanging, 433
 MAC_SetFrameType, 434
 MAC_TRACE_ENABLED, 424
 MAC_TRACE, 424
 MAC_ToSlotsTime, 435
 MAC_USAGE_BUF_START, 424
 MAC_USAGE_BUF_STOP, 424
 MAC_UsFromRx, 435
 MAC_Write, 436
 MAC_Write8, 436
 MAC_YourSlotIsr, 437
 mac_buf_state
 mac_const.h, 441
 mac_buf_t, 42
 buf, 43
 control, 43
 dPtr, 43
 data, 43
 dst, 43
 frame, 43
 isRangingFrame, 43
 last_update_time, 44
 pan, 44
 retransmit_fail_cnt, 44
 rx_len, 44
 seq_num, 44
 src, 44
 state, 44
 mac_buff_time_t
 mac_port.h, 444
 mac_const.h, 438
 ADDR_ANCHOR_FLAG, 439
 ADDR_BROADCAST, 439
 dev_addr_t, 441
 FR_CR_ACK, 439
 FR_CR_BEACON, 439
 FR_CR_DATA, 439
 FR_CR_MAC, 439
 FR_CR_PAN, 440
 FR_CR_PENDING, 440
 FR_CR_RACK, 440
 FR_CR_SECURE, 440
 FR_CR_TYPE_MASK, 440
 FR_CRH_DA_LONG, 440
 FR_CRH_DA_NONE, 440
 FR_CRH_DA_SHORT, 440
 FR_CRH_FVER0, 441
 FR_CRH_FVER1, 441
 FR_CRH_SA_LONG, 441
 FR_CRH_SA_NONE, 441
 FR_CRH_SA_SHORT, 441
 mac_buf_state, 441
 pan_dev_addr_t, 441
 mac_instance_t, 45
 beacon_timer_timestamp, 45
 buf, 46
 buf_get_ind, 46
 frame_under_tx_is_ranging, 46
 last_rx_ts, 46
 rx_buf, 46
 seq_num, 46
 slot_number, 46
 slot_time_offset, 46
 mac_port.h, 442
 MAC_ASSERT, 443
 MAC_LOG_ERR, 443
 mac_buff_time_t, 444
 mac_settings.h, 444
 _DEF_SLOT_CNT, 446
 _DEF_SLOT_SUM_TIME, 446
 _DEF_SLOT_TIME, 446
 MAC_BUF_CNT, 446
 MAC_BUF_LEN, 446
 MAC_SETTINGS_DEF, 446
 rtls_role, 447
 SYNC_MAC_NEIGHBOURS, 446
 TOA_MAX_DEV_IN_POLL, 447
 mac_settings_t, 47
 addr, 48
 max_buf_inactive_time, 48
 max_frame_fail_cnt, 48
 pan, 48
 raport_anchor_anchor_distance, 48
 role, 48
 slot_guard_time_us, 49
 slot_time_us, 49
 slots_sum_time_us, 49
 sync_dly, 49
 max_buf_inactive_time
 mac_settings_t, 48
 max_frame_fail_cnt
 mac_settings_t, 48
 maxGrowthCIR
 dwt_rxdiag_t, 24
 maxNoise
 dwt_rxdiag_t, 24
 N_STD_FACTOR
 deca_param_types.h, 223
 NULL
 deca_types.h, 376
 NUM_16M_OFFSETWB
 toa_calib.c, 520
 NUM_16M_OFFSET
 toa_calib.c, 520
 NUM_64M_OFFSETWB
 toa_calib.c, 521
 NUM_64M_OFFSET
 toa_calib.c, 521

NUM_BR
 deca_param_types.h, 223

NUM_BW
 deca_param_types.h, 223

NUM_CH_SUPPORTED
 deca_param_types.h, 223

NUM_CH
 deca_param_types.h, 223

NUM_PACS
 deca_param_types.h, 223

NUM_PRF
 deca_param_types.h, 223

NUM_SFD
 deca_param_types.h, 224

neighbour
 sync_instance_t, 57

newCrc
 FU_instance_t, 37

newHash
 FU_instance_t, 38

newVer
 FU_instance_t, 38

nsSFD
 dwt_config_t, 16

num_poll_anchor
 FC_SYNC_POLL_s, 32

OTP_ADDR_LEN
 deca_regs.h, 314

OTP_ADDR_MASK
 deca_regs.h, 314

OTP_ADDR
 deca_regs.h, 314

OTP_CTRL_LDELOAD
 deca_regs.h, 315

OTP_CTRL_LEN
 deca_regs.h, 315

OTP_CTRL_MASK
 deca_regs.h, 315

OTP_CTRL_OTPPROG
 deca_regs.h, 315

OTP_CTRL_OTPRDEN
 deca_regs.h, 315

OTP_CTRL_OTPREAD
 deca_regs.h, 315

OTP_CTRL
 deca_regs.h, 314

OTP_IF_ID
 deca_regs.h, 315

OTP_IF_LEN
 deca_regs.h, 316

OTP_RDAT_LEN
 deca_regs.h, 316

OTP_RDAT
 deca_regs.h, 316

OTP_SF_LDO_KICK
 deca_regs.h, 316

OTP_SF_LEN
 deca_regs.h, 316

OTP_SF_MASK
 deca_regs.h, 316

OTP_SF_OPS_KICK
 deca_regs.h, 317

OTP_SF_OPS_SEL_L64
 deca_regs.h, 317

OTP_SF_OPS_SEL_MASK
 deca_regs.h, 317

OTP_SF_OPS_SEL_SHFT
 deca_regs.h, 317

OTP_SF_OPS_SEL_TIGHT
 deca_regs.h, 317

OTP_SRDAT_LEN
 deca_regs.h, 317

OTP_SRDAT
 deca_regs.h, 317

OTP_STAT_LEN
 deca_regs.h, 318

OTP_STAT_MASK
 deca_regs.h, 318

OTP_STAT_OTPPRGD
 deca_regs.h, 318

OTP_STAT_OTPVPOK
 deca_regs.h, 318

OTP_STAT
 deca_regs.h, 318

OTP_SF
 deca_regs.h, 316

OTP_WDAT_LEN
 deca_regs.h, 318

OTP_WDAT
 deca_regs.h, 318

OVER
 dwt_deviceentcnts_t, 19

opcode
 FU_SOT_prot, 42
 FU_prot, 39

otprev
 dwt_local_data_t, 22

PANADR_ID
 deca_regs.h, 319

PANADR_LEN
 deca_regs.h, 319

PANADR_PAN_ID_MASK
 deca_regs.h, 319

PANADR_PAN_ID_OFFSET
 deca_regs.h, 319

PANADR_SHORT_ADDR_MASK
 deca_regs.h, 319

PANADR_SHORT_ADDR_OFFSET
 deca_regs.h, 319

PARTID_ADDRESS
 deca_device.c, 108

PCODES
 deca_param_types.h, 224

PEAK_MULTIPLIER
 deca_param_types.h, 224

PGdly

dwt_txconfig_t, 25
PHE
 dwt_deviceentcnts_t, 19
PMSC_CTRL0_FACE
 deca_regs.h, 319
PMSC_CTRL0_GPDCE
 deca_regs.h, 320
PMSC_CTRL0_KHZCLEN
 deca_regs.h, 320
PMSC_CTRL0_LEN
 deca_regs.h, 320
PMSC_CTRL0_MASK
 deca_regs.h, 320
PMSC_CTRL0_OFFSET
 deca_regs.h, 320
PMSC_CTRL0_PLL2_SEQ_EN
 deca_regs.h, 320
PMSC_CTRL0_RESET_ALL
 deca_regs.h, 320
PMSC_CTRL0_RESET_CLEAR
 deca_regs.h, 320
PMSC_CTRL0_RESET_RX
 deca_regs.h, 321
PMSC_CTRL0_RXCLKS_125M
 deca_regs.h, 321
PMSC_CTRL0_RXCLKS_19M
 deca_regs.h, 321
PMSC_CTRL0_RXCLKS_AUTO
 deca_regs.h, 321
PMSC_CTRL0_RXCLKS_OFF
 deca_regs.h, 321
PMSC_CTRL0_SOFTRESET_OFFSET
 deca_regs.h, 321
PMSC_CTRL0_SYSCLKS_125M
 deca_regs.h, 321
PMSC_CTRL0_SYSCLKS_19M
 deca_regs.h, 322
PMSC_CTRL0_SYSCLKS_AUTO
 deca_regs.h, 322
PMSC_CTRL0_TXCLKS_125M
 deca_regs.h, 322
PMSC_CTRL0_TXCLKS_19M
 deca_regs.h, 322
PMSC_CTRL0_TXCLKS_AUTO
 deca_regs.h, 322
PMSC_CTRL0_TXCLKS_OFF
 deca_regs.h, 322
PMSC_CTRL1_ARX2INIT
 deca_regs.h, 322
PMSC_CTRL1_ARXSLP
 deca_regs.h, 323
PMSC_CTRL1_ATXSLP
 deca_regs.h, 323
PMSC_CTRL1_KHZCLKDIV_MASK
 deca_regs.h, 323
PMSC_CTRL1_LDERUNE
 deca_regs.h, 323
PMSC_CTRL1_LEN
 deca_regs.h, 323
PMSC_CTRL1_MASK
 deca_regs.h, 323
PMSC_CTRL1_OFFSET
 deca_regs.h, 323
PMSC_CTRL1_PKTSEQ_DISABLE
 deca_regs.h, 324
PMSC_CTRL1_PKTSEQ_ENABLE
 deca_regs.h, 324
PMSC_CTRL1_PLLSYN
 deca_regs.h, 324
PMSC_CTRL1_SNOZE
 deca_regs.h, 324
PMSC_CTRL1_SNOZR
 deca_regs.h, 324
PMSC_ID
 deca_regs.h, 324
PMSC_LEDC_BLINK_NOW_ALL
 deca_regs.h, 324
PMSC_LEDC_BLINK_TIM_MASK
 deca_regs.h, 325
PMSC_LEDC_BLINK_TIME_DEF
 deca_regs.h, 325
PMSC_LEDC_BLNKEN
 deca_regs.h, 325
PMSC_LEDC_LEN
 deca_regs.h, 325
PMSC_LEDC_MASK
 deca_regs.h, 325
PMSC_LEDC_OFFSET
 deca_regs.h, 325
PMSC_LEN
 deca_regs.h, 325
PMSC_RES1_OFFSET
 deca_regs.h, 326
PMSC_RES2_OFFSET
 deca_regs.h, 326
PMSC_RES3_OFFSET
 deca_regs.h, 326
PMSC_SNOZT_LEN
 deca_regs.h, 326
PMSC_SNOZT_OFFSET
 deca_regs.h, 326
PMSC_TXFINESEQ_DISABLE
 deca_regs.h, 326
PMSC_TXFINESEQ_ENABLE
 deca_regs.h, 326
PMSC_TXFINESEQ_OFFSET
 deca_regs.h, 327
PORT_ASSERT
 port.h, 450
PORT_BatteryMeasure
 port.h, 451
PORT_BatteryVoltage
 port.h, 451
PORT_BkpRegisterRead
 port.h, 452
PORT_BkpRegisterWrite

port.h, 453
PORT_CrcFeed
 port.h, 454
PORT_CrcReset
 port.h, 454
PORT_EnterStopMode
 port.h, 454
PORT_FlashErase
 port.h, 455
PORT_FlashSave
 port.h, 455
PORT_lassert_fun
 port.h, 456
PORT_Init
 port.h, 456
PORT_LedOff
 port.h, 457
PORT_LedOn
 port.h, 458
PORT_Reboot
 port.h, 458
PORT_ResetTransceiver
 port.h, 459
PORT_SetSlotTimerPeriodUs
 port.h, 459
PORT_SleepMs
 port.h, 460
PORT_SlotTimerSetUsOffset
 port.h, 460
PORT_SlotTimerTickUs
 port.h, 461
PORT_SpiSpeedSlow
 port.h, 462
PORT_TickHr
 port.h, 462
PORT_TickHrToUs
 port.h, 463
PORT_TickMs
 port.h, 463
PORT_TimeStartTimers
 port.h, 464
PORT_WakeupTransceiver
 port.h, 464
PORT_WatchdogInit
 port.h, 464
PORT_WatchdogRefresh
 port.h, 465
PORT_iassert_fun
 iassert.h, 397
PRINT_Stat
 printer.c, 467
 printer.h, 470
PRINT_Version
 printer.c, 468
 printer.h, 470
PROG_DESTINATION1
 fu.h, 394
PROG_DESTINATION2
 fu.h, 394
PROG_SETTINGS1
 fu.h, 394
PROG_START1
 fu.h, 395
PROT_CHECK_LEN
 sync.c, 479
PTO
 dwt_deviceentcnts_t, 19
pan
 mac_buf_t, 44
 mac_settings_t, 48
pan_dev_addr_t
 mac_const.h, 441
partID
 dwt_local_data_t, 22
pass_cnt
 carry_trace_t, 14
path
 carry_trace_t, 14
path_len
 carry_trace_t, 14
phrMode
 dwt_config_t, 17
poll_addr
 FC_SYNC_POLL_s, 32
port.h, 447
 DBG, 450
 decamutexoff, 450
 decamutexon, 451
 PORT_ASSERT, 450
 PORT_BatteryMeasure, 451
 PORT_BatteryVoltage, 451
 PORT_BkpRegisterRead, 452
 PORT_BkpRegisterWrite, 453
 PORT_CrcFeed, 454
 PORT_CrcReset, 454
 PORT_EnterStopMode, 454
 PORT_FlashErase, 455
 PORT_FlashSave, 455
 PORT_lassert_fun, 456
 PORT_Init, 456
 PORT_LedOff, 457
 PORT_LedOn, 458
 PORT_Reboot, 458
 PORT_ResetTransceiver, 459
 PORT_SetSlotTimerPeriodUs, 459
 PORT_SleepMs, 460
 PORT_SlotTimerSetUsOffset, 460
 PORT_SlotTimerTickUs, 461
 PORT_SpiSpeedSlow, 462
 PORT_TickHr, 462
 PORT_TickHrToUs, 463
 PORT_TickMs, 463
 PORT_TimeStartTimers, 464
 PORT_WakeupTransceiver, 464
 PORT_WatchdogInit, 464
 PORT_WatchdogRefresh, 465

readfromspi, 465
 writetospi, 466
 power
 dwt_txconfig_t, 25
 prev_state
 toa_core_t, 61
 prf
 dwt_config_t, 17
 printer.c, 466
 PRINT_Stat, 467
 PRINT_Version, 468
 printer.h, 468
 PRINT_Stat, 470
 PRINT_Version, 470
 prot_cb_len
 bin_parser_cb.c, 84
 prot_cb_t, 50
 cb, 50
 FC, 50
 prot_cb_tab
 bin_parser_cb.c, 85
 prot_packet_info_t, 51
 carry, 51
 direct_src, 52
 prot_parser_cb
 bin_parser.h, 79
 READ_ACC_OFF
 deca_device.c, 108
 READ_ACC_ON
 deca_device.c, 108
 README.md, 471
 REG_05_ID_RESERVED
 deca_regs.h, 327
 REG_07_ID_RESERVED
 deca_regs.h, 327
 REG_0B_ID_RESERVED
 deca_regs.h, 327
 REG_16_ID_RESERVED
 deca_regs.h, 327
 REG_1B_ID_RESERVED
 deca_regs.h, 327
 REG_1C_ID_RESERVED
 deca_regs.h, 328
 REG_20_ID_RESERVED
 deca_regs.h, 328
 REG_22_ID_RESERVED
 deca_regs.h, 328
 REG_29_ID_RESERVED
 deca_regs.h, 328
 REG_30_ID_RESERVED
 deca_regs.h, 328
 REG_31_ID_RESERVED
 deca_regs.h, 328
 REG_32_ID_RESERVED
 deca_regs.h, 329
 REG_33_ID_RESERVED
 deca_regs.h, 329
 REG_34_ID_RESERVED
 deca_regs.h, 329
 REG_35_ID_RESERVED
 deca_regs.h, 329
 REG_37_ID_RESERVED
 deca_regs.h, 329
 REG_38_ID_RESERVED
 deca_regs.h, 329
 REG_39_ID_RESERVED
 deca_regs.h, 329
 REG_3A_ID_RESERVED
 deca_regs.h, 330
 REG_3B_ID_RESERVED
 deca_regs.h, 330
 REG_3D_ID_RESERVED
 deca_regs.h, 330
 REG_3E_ID_RESERVED
 deca_regs.h, 330
 REG_3F_ID_RESERVED
 deca_regs.h, 330
 RF_CONF_ID
 deca_regs.h, 330
 RF_CONF_LEN
 deca_regs.h, 331
 RF_CONF_PGMIXBIASEN_MASK
 deca_regs.h, 331
 RF_CONF_PLLEN_MASK
 deca_regs.h, 331
 RF_CONF_RXEN_MASK
 deca_regs.h, 331
 RF_CONF_TXALLEN_MASK
 deca_regs.h, 331
 RF_CONF_TXBLOCKSEN_MASK
 deca_regs.h, 331
 RF_CONF_TXEN_MASK
 deca_regs.h, 331
 RF_CONF_TXPLLPOWEN_MASK
 deca_regs.h, 332
 RF_CONF_TXPOW_MASK
 deca_regs.h, 332
 RF_RXCTRLH_LEN
 deca_regs.h, 332
 RF_RXCTRLH_NBW
 deca_regs.h, 332
 RF_RXCTRLH_OFFSET
 deca_regs.h, 332
 RF_RXCTRLH_WBW
 deca_regs.h, 332
 RF_STATUS_OFFSET
 deca_regs.h, 332
 RF_TXCTRL_CH1
 deca_regs.h, 332
 RF_TXCTRL_CH2
 deca_regs.h, 333
 RF_TXCTRL_CH3
 deca_regs.h, 333
 RF_TXCTRL_CH4

deca_regs.h, 333
RF_TXCTRL_CH5
 deca_regs.h, 333
RF_TXCTRL_CH7
 deca_regs.h, 333
RF_TXCTRL_LEN
 deca_regs.h, 333
RF_TXCTRL_OFFSET
 deca_regs.h, 333
RF_TXCTRL_TXMTUNE_MASK
 deca_regs.h, 334
RF_TXCTRL_TXTXMQ_MASK
 deca_regs.h, 334
RSL
 dwt_deviceentcnts_t, 19
RTO
 dwt_deviceentcnts_t, 19
RX_BUFFER_ID
 deca_regs.h, 334
RX_BUFFER_LEN
 deca_regs.h, 334
RX_EQUAL_CIR_MXG_MASK
 deca_regs.h, 334
RX_EQUAL_CIR_MXG_SHIFT
 deca_regs.h, 334
RX_EQUAL_FP_AMPL2_MASK
 deca_regs.h, 334
RX_EQUAL_FP_AMPL2_SHIFT
 deca_regs.h, 335
RX_EQUAL_FP_AMPL3_MASK
 deca_regs.h, 335
RX_EQUAL_FP_AMPL3_SHIFT
 deca_regs.h, 335
RX_EQUAL_STD_NOISE_MASK
 deca_regs.h, 335
RX_EQUAL_STD_NOISE_SHIFT
 deca_regs.h, 335
RX_FINFO_ID
 deca_regs.h, 335
RX_FINFO_LEN
 deca_regs.h, 335
RX_FINFO_MASK_32
 deca_regs.h, 336
RX_FINFO_OFFSET
 deca_regs.h, 336
RX_FINFO_RNG_SHIFT
 deca_regs.h, 336
RX_FINFO_RNG
 deca_regs.h, 336
RX_FINFO_RXBR_110k
 deca_regs.h, 336
RX_FINFO_RXBR_6M
 deca_regs.h, 336
RX_FINFO_RXBR_850k
 deca_regs.h, 336
RX_FINFO_RXBR_MASK
 deca_regs.h, 337
RX_FINFO_RXBR_SHIFT
 deca_regs.h, 337
RX_FINFO_RXFL_MASK_1023
 deca_regs.h, 337
RX_FINFO_RXFLE_MASK
 deca_regs.h, 337
RX_FINFO_RXFLEN_MASK
 deca_regs.h, 337
RX_FINFO_RXNSPL_MASK
 deca_regs.h, 337
RX_FINFO_RXPACC_MASK
 deca_regs.h, 337
RX_FINFO_RXPACC_SHIFT
 deca_regs.h, 338
RX_FINFO_RXPEL_1024
 deca_regs.h, 338
RX_FINFO_RXPEL_128
 deca_regs.h, 338
RX_FINFO_RXPEL_1536
 deca_regs.h, 338
RX_FINFO_RXPEL_2048
 deca_regs.h, 338
RX_FINFO_RXPEL_256
 deca_regs.h, 338
RX_FINFO_RXPEL_4096
 deca_regs.h, 338
RX_FINFO_RXPEL_512
 deca_regs.h, 338
RX_FINFO_RXPEL_64
 deca_regs.h, 339
RX_FINFO_RXPEL_MASK
 deca_regs.h, 339
RX_FINFO_RXPRF_16M
 deca_regs.h, 339
RX_FINFO_RXPRF_64M
 deca_regs.h, 339
RX_FINFO_RXPRF_MASK
 deca_regs.h, 339
RX_FINFO_RXPRF_SHIFT
 deca_regs.h, 339
RX_FINFO_RXPSR_MASK
 deca_regs.h, 339
RX_FQUAL_ID
 deca_regs.h, 339
RX_FQUAL_LEN
 deca_regs.h, 340
RX_FWTO_ID
 deca_regs.h, 340
RX_FWTO_LEN
 deca_regs.h, 340
RX_FWTO_MASK
 deca_regs.h, 340
RX_FWTO_OFFSET
 deca_regs.h, 340
RX_SNIFF_ID
 deca_regs.h, 340
RX_SNIFF_LEN
 deca_regs.h, 340
RX_SNIFF_MASK

deca_regs.h, 341
 RX_SNIFF_OFFSET
 deca_regs.h, 341
 RX_SNIFF_SNIFF_OFFFT_MASK
 deca_regs.h, 341
 RX_SNIFF_SNIFF_ONT_MASK
 deca_regs.h, 341
 RX_STAMP_LEN
 deca_regs.h, 341
 RX_TIME_FP_AMPL1_OFFSET
 deca_regs.h, 341
 RX_TIME_FP_INDEX_OFFSET
 deca_regs.h, 341
 RX_TIME_FP_RAWST_OFFSET
 deca_regs.h, 342
 RX_TIME_ID
 deca_regs.h, 342
 RX_TIME_LLEN
 deca_regs.h, 342
 RX_TIME_RX_STAMP_LEN
 deca_regs.h, 342
 RX_TIME_RX_STAMP_OFFSET
 deca_regs.h, 342
 RX_TTCKI_ID
 deca_regs.h, 342
 RX_TTCKI_LEN
 deca_regs.h, 342
 RX_TTCKO_ID
 deca_regs.h, 343
 RX_TTCKO_LEN
 deca_regs.h, 343
 RX_TTCKO_MASK_32
 deca_regs.h, 343
 RX_TTCKO_RCPHASE_MASK
 deca_regs.h, 343
 RX_TTCKO_RSMPDEL_MASK
 deca_regs.h, 343
 RX_TTCKO_RXTOFS_MASK
 deca_regs.h, 343
 range25cm16PRFnB
 toa_calib.c, 522
 range25cm16PRFwb
 toa_calib.c, 522
 range25cm64PRFnB
 toa_calib.c, 523
 range25cm64PRFwb
 toa_calib.c, 523
 RangingControl
 uwb_main.c, 574
 rapport_anchor_anchor_distance
 mac_settings_t, 48
 readfromspi
 deca_device_api.h, 218
 port.h, 465
 reason
 FC_TURN_OFF_s, 34
 resp_dly_us
 toa_settings_t, 63
 resp_ind
 toa_core_t, 61
 retransmit_fail_cnt
 mac_buf_t, 44
 role
 FC_VERSION_s, 37
 mac_settings_t, 48
 rtlS_role
 mac_settings.h, 447
 rx_after_tx_offset_us
 toa_settings_t, 63
 rx_buf
 mac_instance_t, 46
 rx_cnt
 FC_STAT_s, 29
 rx_config
 deca_param_types.h, 225
 deca_params_init.c, 229
 rx_flags
 dwt_cb_data_t, 15
 rx_len
 mac_buf_t, 44
 rxCode
 dwt_config_t, 17
 rxPAC
 dwt_config_t, 17
 rxPreamCount
 dwt_rxdiag_t, 24
 SETTINGS_Init
 settings.c, 472
 settings.h, 477
 SFDTO
 dwt_deviceentcnts_t, 19
 SNIFF_OFFT_MASK
 deca_regs.h, 343
 SNIFF_ONT_MASK
 deca_regs.h, 344
 SPEED_OF_LIGHT
 toa.h, 511
 STD_NOISE_MASK
 deca_regs.h, 344
 STD_NOISE_SHIFT
 deca_regs.h, 344
 SYNC_ASSERT
 sync.h, 495
 SYNC_CalcTimeCoeff
 sync.c, 482
 SYNC_FindOrCreateNeighbour
 sync.c, 482
 sync.h, 496
 SYNC_GlobTime
 sync.c, 483
 sync.h, 497
 SYNC_GlobTimeNeig
 sync.c, 484
 SYNC_Init
 sync.c, 485
 sync.h, 497

SYNC_InitNeighbour
sync.c, 485
SYNC_MAC_NEIGHBOURS
mac_settings.h, 446
SYNC_RxCb
sync.c, 485
sync.h, 498
SYNC_RxToCb
sync.c, 486
sync.h, 499
SYNC_SendFinal
sync.c, 487
SYNC_SendPoll
sync.c, 488
sync.h, 500
SYNC_SendResp
sync.c, 488
SYNC_Smooth
sync.c, 489
SYNC_TIME_DUMP_ENABLED
sync.h, 495
SYNC_TIME_DUMP
sync.h, 495
SYNC_TRACE_ENABLED
sync.h, 495
SYNC_TRACE_TOA_ENABLED
sync.h, 495
SYNC_TRACE_TOA
sync.h, 495
SYNC_TRACE
sync.h, 495
SYNC_TrimDrift
sync.c, 489
SYNC_TxB
sync.c, 489
sync.h, 500
SYNC_Update
sync.c, 490
SYNC_UpdateLocalTimeParams
sync.c, 491
SYNC_UpdateNeighbour
sync.c, 491
SYS_CFG_AACKPEND
deca_regs.h, 344
SYS_CFG_AUTOACK
deca_regs.h, 344
SYS_CFG_DIS_DRXB
deca_regs.h, 344
SYS_CFG_DIS_FCE
deca_regs.h, 344
SYS_CFG_DIS_PHE
deca_regs.h, 344
SYS_CFG_DIS_RSDE
deca_regs.h, 345
SYS_CFG_DIS_STXP
deca_regs.h, 345
SYS_CFG_FCS_INIT2F
deca_regs.h, 345
SYS_CFG_FF_ALL_EN
deca_regs.h, 345
SYS_CFG_FFA4
deca_regs.h, 345
SYS_CFG_FFA5
deca_regs.h, 345
SYS_CFG_FFAA
deca_regs.h, 345
SYS_CFG_FFAB
deca_regs.h, 346
SYS_CFG_FFAD
deca_regs.h, 346
SYS_CFG_FFAM
deca_regs.h, 346
SYS_CFG_FFAR
deca_regs.h, 346
SYS_CFG_FFBC
deca_regs.h, 346
SYS_CFG_HIRQ_POL
deca_regs.h, 346
SYS_CFG_ID
deca_regs.h, 347
SYS_CFG_LEN
deca_regs.h, 347
SYS_CFG_MASK
deca_regs.h, 347
SYS_CFG_PHR_MODE_00
deca_regs.h, 347
SYS_CFG_PHR_MODE_11
deca_regs.h, 347
SYS_CFG_PHR_MODE_SHFT
deca_regs.h, 347
SYS_CFG_RXAUTR
deca_regs.h, 347
SYS_CFG_RXM110K
deca_regs.h, 348
SYS_CFG_RXWTOE
deca_regs.h, 348
SYS_CFG_SPI_EDGE
deca_regs.h, 348
SYS_CTRL_CANSFCS
deca_regs.h, 348
SYS_CTRL_HRBT_OFFSET
deca_regs.h, 348
SYS_CTRL_HRBT
deca_regs.h, 348
SYS_CTRL_HSRBT_TOGGLE
deca_regs.h, 348
SYS_CTRL_ID
deca_regs.h, 349
SYS_CTRL_LEN
deca_regs.h, 349
SYS_CTRL_MASK_32
deca_regs.h, 349
SYS_CTRL_OFFSET
deca_regs.h, 349

SYS_CTRL_RXDLYE
 deca_regs.h, 349
SYS_CTRL_RXENAB
 deca_regs.h, 349
SYS_CTRL_SFCST
 deca_regs.h, 349
SYS_CTRL_TRXOFF
 deca_regs.h, 350
SYS_CTRL_TXDLYS
 deca_regs.h, 350
SYS_CTRL_TXSTRT
 deca_regs.h, 350
SYS_CTRL_WAIT4RESP
 deca_regs.h, 350
SYS_MASK_ID
 deca_regs.h, 350
SYS_MASK_LEN
 deca_regs.h, 350
SYS_MASK_MAAT
 deca_regs.h, 350
SYS_MASK_MAFFREJ
 deca_regs.h, 351
SYS_MASK_MASK_32
 deca_regs.h, 351
SYS_MASK_MCPLLLL
 deca_regs.h, 351
SYS_MASK_MCPLOCK
 deca_regs.h, 351
SYS_MASK_ME SYNCNR
 deca_regs.h, 351
SYS_MASK_MGPIOIRQ
 deca_regs.h, 351
SYS_MASK_MHPDWARN
 deca_regs.h, 351
SYS_MASK_MLDEDONE
 deca_regs.h, 352
SYS_MASK_MLDEERR
 deca_regs.h, 352
SYS_MASK_MRFPLLLL
 deca_regs.h, 352
SYS_MASK_MRXdFR
 deca_regs.h, 352
SYS_MASK_MRxFCE
 deca_regs.h, 352
SYS_MASK_MRxFCG
 deca_regs.h, 352
SYS_MASK_MRxoVRR
 deca_regs.h, 352
SYS_MASK_MRxPHD
 deca_regs.h, 352
SYS_MASK_MRXPHE
 deca_regs.h, 353
SYS_MASK_MRXP RD
 deca_regs.h, 353
SYS_MASK_MRxPTO
 deca_regs.h, 353
SYS_MASK_MRxRFSL
 deca_regs.h, 353
SYS_MASK_MRXRFTO
 deca_regs.h, 353
SYS_MASK_MRxsFDTO
 deca_regs.h, 353
SYS_MASK_MRxsFDD
 deca_regs.h, 353
SYS_MASK_MSLP2INIT
 deca_regs.h, 353
SYS_MASK_MTxBerr
 deca_regs.h, 354
SYS_MASK_MTxFRB
 deca_regs.h, 354
SYS_MASK_MTxFRS
 deca_regs.h, 354
SYS_MASK_MTxPhs
 deca_regs.h, 354
SYS_MASK_MTxPrs
 deca_regs.h, 354
SYS_STATE_ID
 deca_regs.h, 354
SYS_STATE_LEN
 deca_regs.h, 354
SYS_STATUS_AAT
 deca_regs.h, 355
SYS_STATUS_AFFREJ
 deca_regs.h, 355
SYS_STATUS_ALL_DBLBUFF
 deca_regs.h, 355
SYS_STATUS_ALL_RX_ERR
 deca_regs.h, 355
SYS_STATUS_ALL_RX_GOOD
 deca_regs.h, 355
SYS_STATUS_ALL_RX_TO
 deca_regs.h, 355
SYS_STATUS_ALL_TX
 deca_regs.h, 356
SYS_STATUS_CLKPLL_LL
 deca_regs.h, 356
SYS_STATUS_CPlOCK
 deca_regs.h, 356
SYS_STATUS_ESYNCR
 deca_regs.h, 356
SYS_STATUS_GPIOIRQ
 deca_regs.h, 356
SYS_STATUS_HPDWARN
 deca_regs.h, 356
SYS_STATUS_HSRBP
 deca_regs.h, 356
SYS_STATUS_ICRBP
 deca_regs.h, 357
SYS_STATUS IRQS
 deca_regs.h, 357
SYS_STATUS_ID
 deca_regs.h, 357
SYS_STATUS_LDEDONE
 deca_regs.h, 357
SYS_STATUS_LDEERR
 deca_regs.h, 357

SYS_STATUS_LEN
 deca_regs.h, 357

SYS_STATUS_MASK_32
 deca_regs.h, 357

SYS_STATUS_OFFSET
 deca_regs.h, 358

SYS_STATUS_RFPLL_LL
 deca_regs.h, 358

SYS_STATUS_RXDFR
 deca_regs.h, 358

SYS_STATUS_RXFCE
 deca_regs.h, 358

SYS_STATUS_RXFCG
 deca_regs.h, 358

SYS_STATUS_RXOVRR
 deca_regs.h, 358

SYS_STATUS_RXPHD
 deca_regs.h, 359

SYS_STATUS_RXPHE
 deca_regs.h, 359

SYS_STATUS_RXPREJ
 deca_regs.h, 359

SYS_STATUS_RXPRD
 deca_regs.h, 359

SYS_STATUS_RXPTO
 deca_regs.h, 359

SYS_STATUS_RXRFSL
 deca_regs.h, 359

SYS_STATUS_RXRFTO
 deca_regs.h, 359

SYS_STATUS_RXRSCS
 deca_regs.h, 359

SYS_STATUS_RXSFDTO
 deca_regs.h, 360

SYS_STATUS_RXSFDD
 deca_regs.h, 360

SYS_STATUS_SLP2INIT
 deca_regs.h, 360

SYS_STATUS_TXBERR
 deca_regs.h, 360

SYS_STATUS_TXERR
 deca_regs.h, 360

SYS_STATUS_TXFRB
 deca_regs.h, 360

SYS_STATUS_TXFRS
 deca_regs.h, 360

SYS_STATUS_TXPHS
 deca_regs.h, 361

SYS_STATUS_TXPRS
 deca_regs.h, 361

SYS_STATUS_TXPUTE
 deca_regs.h, 361

SYS_STATUS_reserved
 deca_regs.h, 358

SYS_TIME_ID
 deca_regs.h, 361

SYS_TIME_LEN
 deca_regs.h, 361

SYS_TIME_OFFSET
 deca_regs.h, 361

SendBeaconMessage
 uwb_main.c, 575

SendTurnOffMessage
 uwb_main.c, 576

SendTurnOnMessage
 uwb_main.c, 576

seq_num
 mac_buf_t, 44
 mac_instance_t, 46

serial
 FC_BEACON_s, 26
 FC_VERSION_s, 37
 settings_otp_t, 53

sesionPacketCounter
 FU_instance_t, 38

set_pac_from_settings
 transceiver_settings_t, 65

settings
 settings.c, 473
 settings.h, 478

settings.c, 471
 _settings_otp, 473

SETTINGS_Init, 472

settings, 473

settings_is_otp_erased, 472

settings_otp, 473

settings.h, 474
 __H_VERSION__, 475

DEF_SETTINGS, 475

H_MAJOR_CALC, 476

H_VERSION_CALC, 476

SETTINGS_Init, 477

settings, 478

settings_otp, 478

VERSION_SETTINGS_DEF, 476

settings_is_otp_erased
 settings.c, 472

settings_otp
 settings.c, 473
 settings.h, 478

settings_otp_t, 52
 h_major, 52
 h_minor, 53
 serial, 53

settings_t, 53
 carry, 54
 mac, 54
 transceiver, 54
 version, 54

settings_version_t, 55
 boot_reserved, 55
 f_hash, 55
 f_major, 55
 f_minor, 56
 h_version, 56

sfdTO

dwt_config_t, 17
 sftsh
 deca_param_types.h, 225
 deca_params_init.c, 229
 sleep_mode
 dwt_local_data_t, 22
 slot_guard_time_us
 mac_settings_t, 49
 slot_num
 FC_SYNC_FIN_s, 30
 slot_number
 mac_instance_t, 46
 slot_time_offset
 mac_instance_t, 46
 slot_time_us
 mac_settings_t, 49
 slots_sum_time_us
 mac_settings_t, 49
 src
 mac_buf_t, 44
 src_addr
 FC_CARRY_s, 27
 start
 txt_buf_t, 66
 state
 mac_buf_t, 44
 toa_core_t, 61
 status
 dwt_cb_data_t, 15
 stdNoise
 dwt_rxdiag_t, 24
 str_append
 uwb_main.c, 577
 sync
 sync.c, 492
 sync.c, 478
 bad_len_msg, 492
 FC_SYNC_FIN_cb, 480
 FC_SYNC_POLL_cb, 480
 FC_SYNC_RESP_cb, 481
 MOVE_ARRAY_ELEM, 479
 mac, 492
 PROT_CHECK_LEN, 479
 SYNC_CalcTimeCoeff, 482
 SYNC_FindOrCreateNeighbour, 482
 SYNC_GlobTime, 483
 SYNC_GlobTimeNeig, 484
 SYNC_Init, 485
 SYNC_InitNeighbour, 485
 SYNC_RxCb, 485
 SYNC_RxToCb, 486
 SYNC_SendFinal, 487
 SYNC_SendPoll, 488
 SYNC_SendResp, 488
 SYNC_Smooth, 489
 SYNC_TrimDrift, 489
 SYNC_TxCb, 489
 SYNC_Update, 490
 SYNC_UpdateLocalTimeParams, 491
 SYNC_UpdateNeighbour, 491
 sync, 492
 sync.h, 492
 SYNC_ASSERT, 495
 SYNC_FindOrCreateNeighbour, 496
 SYNC_GlobTime, 497
 SYNC_Init, 497
 SYNC_RxCb, 498
 SYNC_RxToCb, 499
 SYNC_SendPoll, 500
 SYNC_TIME_DUMP_ENABLED, 495
 SYNC_TIME_DUMP, 495
 SYNC_TRACE_ENABLED, 495
 SYNC_TRACE_TOA_ENABLED, 495
 SYNC_TRACE_TOA, 495
 SYNC_TRACE, 495
 SYNC_TxCb, 500
 sync_dly
 mac_settings_t, 49
 sync_instance_t, 56
 local_obj, 57
 neighbour, 57
 toa, 57
 toa_ts_poll_rx_raw, 57
 tree_level, 57
 sync_neighbour_t, 58
 addr, 58
 drift, 58
 sync_ready, 59
 time_coeffP_raw, 59
 time_coeffP, 59
 time_drift_sum, 59
 time_offset, 59
 tof_dw, 59
 tree_level, 60
 update_ts, 60
 sync_ready
 sync_neighbour_t, 59
 sysCFGreg
 dwt_local_data_t, 23
 TC_PGCAL_STATUS_DELAY_MASK
 deca_regs.h, 361
 TC_PGCAL_STATUS_LEN
 deca_regs.h, 362
 TC_PGCAL_STATUS_OFFSET
 deca_regs.h, 362
 TC_PGCCTRL_AUTOCAL
 deca_regs.h, 362
 TC_PGCCTRL_CALSTART
 deca_regs.h, 362
 TC_PGCCTRL_DIR_CONV
 deca_regs.h, 362
 TC_PGCCTRL_LEN
 deca_regs.h, 362
 TC_PGCCTRL_OFFSET
 deca_regs.h, 362
 TC_PGCCTRL_ON_TX

deca_regs.h, 362
TC_PGCCTRL_TMEAS_MASK
 deca_regs.h, 363
TC_PGDELAY_CH1
 deca_regs.h, 363
TC_PGDELAY_CH2
 deca_regs.h, 363
TC_PGDELAY_CH3
 deca_regs.h, 363
TC_PGDELAY_CH4
 deca_regs.h, 363
TC_PGDELAY_CH5
 deca_regs.h, 363
TC_PGDELAY_CH7
 deca_regs.h, 363
TC_PGDELAY_LEN
 deca_regs.h, 364
TC_PGDELAY_OFFSET
 deca_regs.h, 364
TC_PGTTEST_CW
 deca_regs.h, 364
TC_PGTTEST_LEN
 deca_regs.h, 364
TC_PGTTEST_NORMAL
 deca_regs.h, 364
TC_PGTTEST_OFFSET
 deca_regs.h, 364
TC_SARL_SAR_LTEMP_OFFSET
 deca_regs.h, 364
TC_SARL_SAR_LVBAT_OFFSET
 deca_regs.h, 365
TC_SARL_SAR_C
 deca_regs.h, 364
TC_SARW_SAR_WTEMP_OFFSET
 deca_regs.h, 365
TC_SARW_SAR_WVBAT_OFFSET
 deca_regs.h, 365
TOA_AddMeasure
 toa.c, 503
 toa.h, 512
TOA_CalcTof
 toa.h, 513
TOA_CalcTofDwTu
 toa.c, 503
 toa.h, 513
TOA_CalcTofconst
 toa.c, 503
TOA_EnableRxBeforeFin
 toa.c, 504
 toa.h, 514
TOA_FindAddrIndexInResp
 toa.c, 505
 toa.h, 515
TOA_GetRangeBias
 toa.c, 506
TOA_MAX_DEV_IN_POLL
 mac_settings.h, 447
TOA_Read40bValue
 toa.c, 506
 toa.h, 515
TOA_SetTxTime
 toa.c, 506
 toa.h, 516
TOA_State
 toa.c, 507
 toa.h, 517
TOA_TofToCm
 toa.c, 508
 toa.h, 517
TOA_Write40bValue
 toa.c, 509
 toa.h, 518
TRANSCEIVER_ASSERT
 transceiver_settings.h, 555
TRANSCEIVER_DefaultRx
 transceiver.c, 531
 transceiver.h, 543
TRANSCEIVER_EnterDeepSleep
 transceiver.c, 531
 transceiver.h, 544
TRANSCEIVER_EstimateTxTimeUs
 transceiver.c, 532
 transceiver.h, 545
TRANSCEIVER_GetRxTimestamp
 transceiver.c, 533
 transceiver.h, 545
TRANSCEIVER_GetTime
 transceiver.c, 533
 transceiver.h, 546
TRANSCEIVER_GetTxTimestamp
 transceiver.c, 534
 transceiver.h, 547
TRANSCEIVER_Init
 transceiver.c, 535
 transceiver.h, 547
TRANSCEIVER_Read
 transceiver.c, 535
 transceiver.h, 548
TRANSCEIVER_ReadDiagnostic
 transceiver.c, 536
 transceiver.h, 549
TRANSCEIVER_SETTINGS_DEF
 transceiver_settings.h, 555
TRANSCEIVER_Send
 transceiver.c, 537
 transceiver.h, 549
TRANSCEIVER_SendRanging
 transceiver.c, 537
 transceiver.h, 550
TRANSCEIVER_SetAddr
 transceiver.c, 538
 transceiver.h, 551
TRANSCEIVER_SetCb
 transceiver.c, 539
 transceiver.h, 551
TRANSCEIVER_WakeUp

transceiver.c, 540
TX_ANTD_ID
 deca_regs.h, 365
TX_ANTD_LEN
 deca_regs.h, 365
TX_ANTD_OFFSET
 deca_regs.h, 365
TX_BUFFER_ID
 deca_regs.h, 365
TX_BUFFER_LEN
 deca_regs.h, 366
TX_CAL_ID
 deca_regs.h, 366
TX_CAL_LEN
 deca_regs.h, 366
TX_FCTRL_FLE_MASK
 deca_regs.h, 366
TX_FCTRL_IFSDELAY_MASK
 deca_regs.h, 366
TX_FCTRL_ID
 deca_regs.h, 366
TX_FCTRL_LEN
 deca_regs.h, 366
TX_FCTRL_PE_MASK
 deca_regs.h, 367
TX_FCTRL_PE_SHFT
 deca_regs.h, 367
TX_FCTRL_SAFE_MASK_32
 deca_regs.h, 367
TX_FCTRL_TFLE_MASK
 deca_regs.h, 367
TX_FCTRL_TFLEN_MASK
 deca_regs.h, 367
TX_FCTRL_TR_SHFT
 deca_regs.h, 367
TX_FCTRL_TXBOFFS_MASK
 deca_regs.h, 368
TX_FCTRL_TXBOFFS_SHFT
 deca_regs.h, 368
TX_FCTRL_TXBR_110k
 deca_regs.h, 368
TX_FCTRL_TXBR_6M
 deca_regs.h, 368
TX_FCTRL_TXBR_850k
 deca_regs.h, 368
TX_FCTRL_TXBR_MASK
 deca_regs.h, 368
TX_FCTRL_TXBR_SHFT
 deca_regs.h, 368
TX_FCTRL_TXPRF_16M
 deca_regs.h, 369
TX_FCTRL_TXPRF_4M
 deca_regs.h, 369
TX_FCTRL_TXPRF_64M
 deca_regs.h, 369
TX_FCTRL_TXPRF_MASK
 deca_regs.h, 369
TX_FCTRL_TXPRF_SHFT
 deca_regs.h, 369
TX_FCTRL_TXPSR_MASK
 deca_regs.h, 369
TX_FCTRL_TXPSR_PE_1024
 deca_regs.h, 369
TX_FCTRL_TXPSR_PE_128
 deca_regs.h, 370
TX_FCTRL_TXPSR_PE_1536
 deca_regs.h, 370
TX_FCTRL_TXPSR_PE_16
 deca_regs.h, 370
TX_FCTRL_TXPSR_PE_2048
 deca_regs.h, 370
TX_FCTRL_TXPSR_PE_256
 deca_regs.h, 370
TX_FCTRL_TXPSR_PE_4096
 deca_regs.h, 370
TX_FCTRL_TXPSR_PE_512
 deca_regs.h, 370
TX_FCTRL_TXPSR_PE_64
 deca_regs.h, 370
TX_FCTRL_TXPSR_PE_MASK
 deca_regs.h, 371
TX_FCTRL_TXPSR_SHFT
 deca_regs.h, 371
TX_FCTRL_TR
 deca_regs.h, 367
TX_POWER_BOOSTNORM_MASK
 deca_regs.h, 371
TX_POWER_BOOSTNORM_SHIFT
 deca_regs.h, 371
TX_POWER_BOOSTP125_MASK
 deca_regs.h, 371
TX_POWER_BOOSTP125_SHIFT
 deca_regs.h, 371
TX_POWER_BOOSTP250_MASK
 deca_regs.h, 371
TX_POWER_BOOSTP250_SHIFT
 deca_regs.h, 372
TX_POWER_BOOSTP500_MASK
 deca_regs.h, 372
TX_POWER_BOOSTP500_SHIFT
 deca_regs.h, 372
TX_POWER_ID
 deca_regs.h, 372
TX_POWER_LEN
 deca_regs.h, 372
TX_POWER_MAN_DEFAULT
 deca_regs.h, 372
TX_POWER_TXPOWPHR_MASK
 deca_regs.h, 372
TX_POWER_TXPOWSD_MASK
 deca_regs.h, 373
TX_STAMP_LEN
 deca_regs.h, 373
TX_TIME_ID
 deca_regs.h, 373
TX_TIME_LLEN

deca_regs.h, 373
TX_TIME_TX_RAWST_OFFSET
 deca_regs.h, 373
TX_TIME_TX_STAMP_LEN
 deca_regs.h, 373
TX_TIME_TX_STAMP_OFFSET
 deca_regs.h, 373
TXT_Atol
 txt_parser.c, 557
 txt_parser.h, 563
TXT_Control
 txt_parser.c, 558
 txt_parser.h, 564
TXT_GetParam
 txt_parser.c, 558
 txt_parser.h, 564
TXT_HangCb
 txt_parser_cb.c, 569
TXT_Input
 txt_parser.c, 559
 txt_parser.h, 565
TXT_Parse
 txt_parser.c, 560
TXT_PointParamNumber
 txt_parser.c, 560
 txt_parser.h, 566
TXT_RFSet
 txt_parser_cb.c, 569
TXT_StartsWith
 txt_parser.c, 560
 txt_parser.h, 566
TXT_StatCb
 txt_parser_cb.c, 569
TXT_TestCb
 txt_parser_cb.c, 570
TXT_VersionCb
 txt_parser_cb.c, 570
TXF
 dwt_deviceentcnts_t, 20
TXW
 dwt_deviceentcnts_t, 20
target
 agc_cfg_struct, 9
 carry_instance_t, 10
time_coeffP_raw
 sync_neighbour_t, 59
time_coeffP
 sync_neighbour_t, 59
time_drift_sum
 sync_neighbour_t, 59
time_offset
 sync_neighbour_t, 59
to_cnt
 FC_STAT_s, 29
toSinkId
 carry_instance_t, 11
toa
 sync_instance_t, 57
 toa.c, 501
 _TOA_GetRangeBias, 502
 TOA_AddMeasure, 503
 TOA_CalcTofDwTu, 503
 TOA_CalcTofconst, 503
 TOA_EnableRxBeforeFin, 504
 TOA_FindAddrIndexInResp, 505
 TOA_GetRangeBias, 506
 TOA_Read40bValue, 506
 TOA_SetTxTime, 506
 TOA_State, 507
 TOA_TofToCm, 508
 TOA_Write40bValue, 509
 toa.h, 509
 SPEED_OF_LIGHT, 511
 TOA_AddMeasure, 512
 TOA_CalcTof, 513
 TOA_CalcTofDwTu, 513
 TOA_EnableRxBeforeFin, 514
 TOA_FindAddrIndexInResp, 515
 TOA_Read40bValue, 515
 TOA_SetTxTime, 516
 TOA_State, 517
 TOA_TofToCm, 517
 TOA_Write40bValue, 518
 toa_state_t, 512
 toa_calib.c, 519
 _TOA_GetRangeBias, 521
 CM_OFFSET_16M_NB, 520
 CM_OFFSET_16M_WB, 520
 CM_OFFSET_64M_NB, 520
 CM_OFFSET_64M_WB, 520
 chan_idxnb, 522
 chan_idxwb, 522
 NUM_16M_OFFSETWB, 520
 NUM_16M_OFFSET, 520
 NUM_64M_OFFSETWB, 521
 NUM_64M_OFFSET, 521
 range25cm16PRFn, 522
 range25cm16PRFwb, 522
 range25cm64PRFn, 523
 range25cm64PRFwb, 523
 toa_core_t, 60
 addr_tab, 61
 anc_in_poll_cnt, 61
 initiator, 61
 prev_state, 61
 resp_ind, 61
 state, 61
 TsFinRx, 61
 TsFinTx, 62
 TsPollRx, 62
 TsPollTx, 62
 TsRespRx, 62
 TsRespTx, 62
 toa_settings_t, 62
 fin_dly_us, 63
 guard_time_us, 63

resp_dly_us, 63
 rx_after_tx_offset_us, 63
 toa_state_t
 toa.h, 512
 toa_ts_poll_rx_raw
 sync_instance_t, 57
 tof_dw
 sync_neighbour_t, 59
 tools.c, 524
 tools.h, 524
 ALL_UNUSED_IMPL_, 526
 ALL_UNUSED_IMPL, 526
 ALL_UNUSED, 526
 DECREMENT_CYCLE, 526
 DECREMENT_MOD, 526
 INCREMENT_CYCLE, 527
 INCREMENT_MOD, 527
 UNUSED_1, 527
 UNUSED_2, 527
 UNUSED_3, 527
 UNUSED_4, 528
 UNUSED_5, 528
 UNUSED_6, 528
 UNUSED_7, 528
 UNUSED_8, 528
 VA_NUM_ARGS_IMPL, 529
 VA_NUM_ARGS, 529
 trace
 carry_target_t, 13
 trace_max_fail_cnt
 carry_settings_t, 11
 trace_max_inactive_time
 carry_settings_t, 11
 transceiver
 settings_t, 54
 transceiver.c, 529
 TRANSCEIVER_DefaultRx, 531
 TRANSCEIVER_EnterDeepSleep, 531
 TRANSCEIVER_EstimateTxTimeUs, 532
 TRANSCEIVER_GetRxTimestamp, 533
 TRANSCEIVER_GetTime, 533
 TRANSCEIVER_GetTxTimestamp, 534
 TRANSCEIVER_Init, 535
 TRANSCEIVER_Read, 535
 TRANSCEIVER_ReadDiagnostic, 536
 TRANSCEIVER_Send, 537
 TRANSCEIVER_SendRanging, 537
 TRANSCEIVER_SetAddr, 538
 TRANSCEIVER_SetCb, 539
 TRANSCEIVER_WakeUp, 540
 transceiver_br, 540
 transceiver_pac, 540
 transceiver_plen, 540
 transceiver_sfd, 541
 transceiver.h, 541
 MASK_40BIT, 543
 TRANSCEIVER_DefaultRx, 543
 TRANSCEIVER_EnterDeepSleep, 544
 TRANSCEIVER_EstimateTxTimeUs, 545
 TRANSCEIVER_GetRxTimestamp, 545
 TRANSCEIVER_GetTime, 546
 TRANSCEIVER_GetTxTimestamp, 547
 TRANSCEIVER_Init, 547
 TRANSCEIVER_Read, 548
 TRANSCEIVER_ReadDiagnostic, 549
 TRANSCEIVER_Send, 549
 TRANSCEIVER_SendRanging, 550
 TRANSCEIVER_SetAddr, 551
 TRANSCEIVER_SetCb, 551
 UUS_TO_DWT_TIME, 543
 transceiver_br
 transceiver.c, 540
 transceiver_pac
 transceiver.c, 540
 transceiver_plen
 transceiver.c, 540
 transceiver_settings.h, 552
 TRANSCEIVER_ASSERT, 555
 TRANSCEIVER_SETTINGS_DEF, 555
 transceiver_settings_t, 64
 ant_dly_rx, 64
 ant_dly_tx, 64
 dwt_config, 65
 dwt_txconfig, 65
 low_power_mode, 65
 set_pac_from_settings, 65
 transceiver_sfd
 transceiver.c, 541
 transceiver_translator.c, 555
 deca_sleep, 556
 tree_level
 FC_SYNC_FIN_s, 30
 sync_instance_t, 57
 sync_neighbour_t, 60
 TsFinRx
 toa_core_t, 61
 TsFinTx
 toa_core_t, 62
 TsFinTxBuf
 FC_SYNC_FIN_s, 30
 TsOffset
 FC_SYNC_FIN_s, 31
 TsPollRx
 FC_SYNC_RESP_s, 33
 toa_core_t, 62
 TsPollTx
 FC_SYNC_FIN_s, 31
 toa_core_t, 62
 TsRespRx
 FC_SYNC_FIN_s, 31
 toa_core_t, 62
 TsRespTx
 FC_SYNC_RESP_s, 33
 toa_core_t, 62
 TurnOff
 uwb_main.c, 577

tx_cnt
 FC_STAT_s, 29

tx_config
 deca_param_types.h, 225
 deca_params_init.c, 229

txCode
 dwt_config_t, 17

txFCTRL
 dwt_local_data_t, 23

txPreambLength
 dwt_config_t, 18

txpwr_compensation
 deca_param_types.h, 226
 deca_params_init.c, 230

txt_buf_t, 65
 cmd, 66
 cnt, 66
 end, 66
 start, 66

txt_cb_len
 txt_parser_cb.c, 571

txt_cb_t, 66
 cb, 67
 cmd, 67

txt_cb_tab
 txt_parser_cb.c, 571

txt_parser.c, 557
 _txt_buf_raw, 561
 TXT_Atol, 557
 TXT_Control, 558
 TXT_GetParam, 558
 TXT_Input, 559
 TXT_Parse, 560
 TXT_PointParamNumber, 560
 TXT_StartsWith, 560

txt_parser.h, 561
 cchar, 563
 TXT_Atol, 563
 TXT_Control, 564
 TXT_GetParam, 564
 TXT_Input, 565
 TXT_PointParamNumber, 566
 TXT_StartsWith, 566
 txt_parser_cb, 563

txt_parser_cb
 txt_parser.h, 563

txt_parser_cb.c, 567
 _TXT_Ask, 567
 _TXT_Finalize, 568
 TXT_HangCb, 569
 TXT_RFSet, 569
 TXT_StatCb, 569
 TXT_TestCb, 570
 TXT_VersionCb, 570
 txt_cb_len, 571
 txt_cb_tab, 571

UNUSED_1
 tools.h, 527

UNUSED_2
 tools.h, 527

UNUSED_3
 tools.h, 527

UNUSED_4
 tools.h, 528

UNUSED_5
 tools.h, 528

UNUSED_6
 tools.h, 528

UNUSED_7
 tools.h, 528

UNUSED_8
 tools.h, 528

USR_SFD_ID
 deca_regs.h, 374

USR_SFD_LEN
 deca_regs.h, 374

UUS_TO_DWT_TIME
 transceiver.h, 543

uint16
 deca_device_api.h, 174
 deca_types.h, 377

uint32
 deca_device_api.h, 174
 deca_types.h, 377

uint8
 deca_device_api.h, 174
 deca_types.h, 377

update_ts
 sync_neighbour_t, 60

uwb_main.c, 571
 BatteryControl, 572
 BeaconSender, 572
 CheckSleepMode, 573
 Desynchronize, 573
 diagnostic, 574
 RangingControl, 574
 SendBeaconMessage, 575
 SendTurnOffMessage, 576
 SendTurnOnMessage, 576
 str_append, 577
 TurnOff, 577
 UwbMain, 577

uwb_main.h, 578
 UwbMain, 580

UwbMain
 uwb_main.c, 577
 uwb_main.h, 580

VA_NUM_ARGS_IMPL
 tools.h, 529

VA_NUM_ARGS
 tools.h, 529

VBAT_ADDRESS
 deca_device.c, 108

VERSION_SETTINGS_DEF
 settings.h, 476

VTEMP_ADDRESS

deca_device.c, [108](#)
version
 settings_t, [54](#)

W4R_TIM_MASK
 deca_regs.h, [374](#)

wait4resp
 dwt_local_data_t, [23](#)

writetospi
 deca_device_api.h, [219](#)
 port.h, [466](#)

XMLPARAMS_VERSION
 deca_param_types.h, [224](#)

XTRIM_ADDRESS
 deca_device.c, [109](#)