

WildhornAV

Generated by Doxygen 1.9.1

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 control Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Field Documentation	5
3.1.2.1 state	5
3.2 debug_context Struct Reference	6
3.2.1 Detailed Description	6
3.3 debug_interface_context Struct Reference	6
3.3.1 Detailed Description	6
3.3.2 Field Documentation	6
3.3.2.1 msv2	7
3.4 device Struct Reference	7
3.4.1 Detailed Description	7
3.4.2 Field Documentation	7
3.4.2.1 context	8
3.4.2.2 id	8
3.4.2.3 interface	8
3.4.2.4 read_reg	8
3.4.2.5 write_reg	9
3.5 device_daemon Struct Reference	9
3.5.1 Detailed Description	9
3.5.2 Field Documentation	10
3.5.2.1 buffer	10
3.5.2.2 context	10
3.5.2.3 data_rdy	10
3.5.2.4 handle	10
3.5.2.5 id	10
3.5.2.6 interfaces	11
3.5.2.7 interfaces_count	11
3.5.2.8 stack	11
3.6 device_interface Struct Reference	11
3.6.1 Detailed Description	11
3.6.2 Field Documentation	12
3.6.2.1 context	12
3.6.2.2 handle_data	12
3.6.2.3 id	12
3.6.2.4 recv	12

3.6.2.5 send	13
3.7 dma_request Struct Reference	13
3.7.1 Detailed Description	13
3.7.2 Field Documentation	13
3.7.2.1 dst	13
3.7.2.2 dst_inc	13
3.7.2.3 src	14
3.7.2.4 src_inc	14
3.7.2.5 tranfser_len	14
3.8 dma_scheduler_dev Struct Reference	14
3.8.1 Detailed Description	15
3.8.2 Field Documentation	15
3.8.2.1 free_stream_count	15
3.8.2.2 stream_count	15
3.8.2.3 streams	15
3.9 dma_stream_config Struct Reference	16
3.9.1 Detailed Description	16
3.9.2 Field Documentation	16
3.9.2.1 direction	16
3.9.2.2 dmamux_request_number	16
3.9.2.3 m0_addr	17
3.9.2.4 m1_addr	17
3.9.2.5 p_addr	17
3.9.2.6 peripheral_flow_control	17
3.9.2.7 priority	17
3.9.2.8 stream_number	18
3.9.2.9 transfer_cplt	18
3.9.2.10 transfer_error	18
3.9.2.11 transfer_half	18
3.9.2.12 transfer_size	18
3.9.2.13 user_context	19
3.10 dma_stream_dev Struct Reference	19
3.10.1 Detailed Description	19
3.10.2 Field Documentation	19
3.10.2.1 dma	19
3.10.2.2 dma_stream	20
3.10.2.3 dmamux_channel	20
3.10.2.4 dmamux_channel_status	20
3.10.2.5 number	20
3.10.2.6 state	20
3.10.2.7 transfer_cplt	21
3.10.2.8 transfer_error	21

3.10.2.9 transfer_half	21
3.10.2.10 user_context	21
3.11 gpio_config Struct Reference	21
3.11.1 Detailed Description	22
3.11.2 Field Documentation	22
3.11.2.1 alternate	22
3.11.2.2 bias	22
3.11.2.3 drive	22
3.11.2.4 mode	23
3.11.2.5 speed	23
3.12 hostproc_interface_context Struct Reference	23
3.12.1 Detailed Description	23
3.12.2 Field Documentation	23
3.12.2.1 rx_once	23
3.12.2.2 uart	24
3.13 i2c_interface_context Struct Reference	24
3.13.1 Detailed Description	24
3.13.2 Field Documentation	24
3.13.2.1 i2c	24
3.14 i2c_sensor_context Struct Reference	24
3.14.1 Detailed Description	25
3.14.2 Field Documentation	25
3.14.2.1 device_address	25
3.15 led_color Struct Reference	25
3.15.1 Detailed Description	25
3.15.2 Field Documentation	25
3.15.2.1 b	25
3.15.2.2 g	26
3.15.2.3 r	26
3.16 MSV2_INST Struct Reference	26
3.16.1 Detailed Description	27
3.16.2 Field Documentation	27
3.16.2.1 id	27
3.16.2.2 rx	27
3.16.2.3 tx	27
3.17 MSV2_RX_DATA Struct Reference	27
3.17.1 Detailed Description	28
3.17.2 Field Documentation	28
3.17.2.1 counter	28
3.17.2.2 crc	28
3.17.2.3 crc_data	28
3.17.2.4 data	29

3.17.2.5 data_len	29
3.17.2.6 escape	29
3.17.2.7 length	29
3.17.2.8 opcode	29
3.17.2.9 state	30
3.18 MSV2_TX_DATA Struct Reference	30
3.18.1 Detailed Description	30
3.18.2 Field Documentation	30
3.18.2.1 crc	30
3.18.2.2 crc_data	30
3.18.2.3 data	31
3.18.2.4 data_len	31
3.18.2.5 opcode	31
3.19 note Struct Reference	31
3.19.1 Detailed Description	31
3.19.2 Field Documentation	32
3.19.2.1 freq	32
3.19.2.2 time	32
3.20 od_entry_t Struct Reference	32
3.20.1 Detailed Description	32
3.20.2 Field Documentation	32
3.20.2.1 data	32
3.20.2.2 data_id	33
3.20.2.3 size	33
3.21 od_frame_t Struct Reference	33
3.21.1 Detailed Description	33
3.21.2 Field Documentation	33
3.21.2.1 data	33
3.21.2.2 data_id	34
3.21.2.3 size	34
3.22 packet_def Struct Reference	34
3.22.1 Detailed Description	34
3.22.2 Field Documentation	34
3.22.2.1 len	34
3.22.2.2 opcode	35
3.23 serial_daemon_context Struct Reference	35
3.23.1 Detailed Description	35
3.23.2 Field Documentation	35
3.23.2.1 rx_sem	35
3.23.2.2 rx_sem_buffer	35
3.24 serial_interface_context Struct Reference	36
3.24.1 Detailed Description	36

3.24.2 Field Documentation	36
3.24.2.1 protocol	36
3.24.2.2 rx_buffer	37
3.24.2.3 rx_data	37
3.24.2.4 rx_data_len	37
3.24.2.5 rx_fragment	37
3.24.2.6 tx_data	37
3.24.2.7 uart	38
3.25 util_buffer_i16 Struct Reference	38
3.25.1 Detailed Description	38
3.25.2 Field Documentation	38
3.25.2.1 bfr_len	38
3.25.2.2 buffer	38
3.25.2.3 c_ix	39
3.25.2.4 l_ix	39
3.26 util_buffer_u16 Struct Reference	39
3.26.1 Detailed Description	39
3.26.2 Field Documentation	39
3.26.2.1 bfr_len	39
3.26.2.2 buffer	40
3.26.2.3 c_ix	40
3.26.2.4 l_ix	40
3.27 util_buffer_u8 Struct Reference	40
3.27.1 Detailed Description	40
3.27.2 Field Documentation	41
3.27.2.1 bfr_len	41
3.27.2.2 buffer	41
3.27.2.3 c_ix	41
3.27.2.4 l_ix	41
4 File Documentation	43
4.1 accelerometer.c File Reference	43
4.1.1 Function Documentation	43
4.1.1.1 accelerometer_init()	44
4.2 accelerometer.h File Reference	44
4.2.1 Function Documentation	45
4.2.1.1 accelerometer_init()	45
4.3 barometer.c File Reference	45
4.4 barometer.h File Reference	45
4.5 buzzer.c File Reference	46
4.5.1 Macro Definition Documentation	47
4.5.1.1 BUZZER_PIN	47

4.5.1.2 BUZZER_PORT	47
4.5.1.3 COMPUTE_NOTE	47
4.5.1.4 COMPUTE_RYTM	47
4.5.1.5 NOTE_PRESC	48
4.5.1.6 NOTE_TIMER	48
4.5.1.7 NOTE_TIMER_DEV	48
4.5.1.8 RYTM_MS	48
4.5.1.9 RYTM_PRESC	48
4.5.1.10 RYTM_TIMER	48
4.5.1.11 RYTM_TIMER_DEV	49
4.5.1.12 TIMER_FREQ	49
4.5.1.13 TIMER_TRIM	49
4.5.2 Function Documentation	49
4.5.2.1 buzzer_disable()	49
4.5.2.2 buzzer_enable()	49
4.5.2.3 buzzer_init()	50
4.5.2.4 buzzer_note_interrupt()	50
4.5.2.5 buzzer_rytm_interrupt()	51
4.5.3 Variable Documentation	51
4.5.3.1 melody_active	51
4.5.3.2 melody_state	51
4.5.3.3 state	52
4.6 buzzer.h File Reference	52
4.6.1 Function Documentation	53
4.6.1.1 buzzer_disable()	53
4.6.1.2 buzzer_enable()	53
4.6.1.3 buzzer_init()	53
4.6.1.4 buzzer_note_interrupt()	54
4.6.1.5 buzzer_rytm_interrupt()	54
4.7 control.c File Reference	55
4.7.1 Macro Definition Documentation	57
4.7.1.1 CONTROL_HEART_BEAT	57
4.7.2 Typedef Documentation	57
4.7.2.1 control_state_t	57
4.7.2.2 control_t	57
4.7.3 Enumeration Type Documentation	57
4.7.3.1 control_state	57
4.7.4 Function Documentation	58
4.7.4.1 control_abort_run()	58
4.7.4.2 control_abort_start()	58
4.7.4.3 control_apogee_run()	59
4.7.4.4 control_apogee_start()	59

4.7.4.5 control_armed_run()	59
4.7.4.6 control_armed_start()	59
4.7.4.7 control_ballistic_run()	60
4.7.4.8 control_ballistic_start()	60
4.7.4.9 control_calibration_run()	60
4.7.4.10 control_calibration_start()	60
4.7.4.11 control_coast_run()	61
4.7.4.12 control_coast_start()	61
4.7.4.13 control_drogue_run()	61
4.7.4.14 control_drogue_start()	61
4.7.4.15 control_error_run()	62
4.7.4.16 control_error_start()	62
4.7.4.17 control_event_run()	62
4.7.4.18 control_event_start()	62
4.7.4.19 control_idle_run()	63
4.7.4.20 control_idle_start()	63
4.7.4.21 control_main_run()	63
4.7.4.22 control_main_start()	63
4.7.4.23 control_powered_run()	64
4.7.4.24 control_powered_start()	64
4.7.4.25 control_supersonic_run()	64
4.7.4.26 control_supersonic_start()	64
4.7.4.27 control_thread()	64
4.7.4.28 control_touchdown_run()	65
4.7.4.29 control_touchdown_start()	66
4.7.5 Variable Documentation	66
4.7.5.1 control	66
4.8 control.h File Reference	66
4.8.1 Function Documentation	67
4.8.1.1 control_thread()	67
4.9 debug.c File Reference	67
4.9.1 Typedef Documentation	68
4.9.1.1 debug_context_t	68
4.9.1.2 debug_interface_context_t	68
4.9.2 Function Documentation	68
4.9.2.1 debug_init()	68
4.9.3 Variable Documentation	69
4.9.3.1 debug_context	69
4.9.3.2 debug_interface_context	69
4.9.3.3 feedback_interface_context	69
4.10 debug.h File Reference	70
4.11 device.c File Reference	70

4.11.1 Macro Definition Documentation	72
4.11.1.1 LEN_16	72
4.11.1.2 LEN_32	72
4.11.1.3 LEN_8	72
4.11.2 Function Documentation	72
4.11.2.1 device_create()	72
4.11.2.2 device_daemon_create()	73
4.11.2.3 device_daemon_thread()	74
4.11.2.4 device_interface_create()	75
4.11.2.5 device_interface_recv()	75
4.11.2.6 device_interface_send()	76
4.11.2.7 device_read_i16()	76
4.11.2.8 device_read_i32()	77
4.11.2.9 device_read_i8()	77
4.11.2.10 device_read_u16()	78
4.11.2.11 device_read_u32()	78
4.11.2.12 device_read_u8()	79
4.11.2.13 device_write_i16()	79
4.11.2.14 device_write_i32()	80
4.11.2.15 device_write_i8()	80
4.11.2.16 device_write_u16()	81
4.11.2.17 device_write_u32()	81
4.11.2.18 device_write_u8()	82
4.12 device.h File Reference	82
4.12.1 Macro Definition Documentation	84
4.12.1.1 DEAMON_STACK_SIZE	84
4.12.1.2 DEVICE_MAX_INTERFACES_PER_DEAMON	84
4.12.1.3 DEVICE_NAME_LEN	84
4.12.2 Typedef Documentation	84
4.12.2.1 device_daemon_t	84
4.12.2.2 device_interface_t	84
4.12.2.3 device_t	84
4.12.3 Function Documentation	85
4.12.3.1 device_create()	85
4.12.3.2 device_daemon_create()	86
4.12.3.3 device_interface_create()	87
4.12.3.4 device_interface_recv()	87
4.12.3.5 device_interface_send()	88
4.12.3.6 device_read_i16()	88
4.12.3.7 device_read_i32()	89
4.12.3.8 device_read_i8()	89
4.12.3.9 device_read_u16()	90

4.12.3.10 device_read_u32()	90
4.12.3.11 device_read_u8()	91
4.12.3.12 device_write_i16()	91
4.12.3.13 device_write_i32()	92
4.12.3.14 device_write_i8()	92
4.12.3.15 device_write_u16()	93
4.12.3.16 device_write_u32()	93
4.12.3.17 device_write_u8()	94
4.13 dma.c File Reference	94
4.13.1 Function Documentation	95
4.13.1.1 dma2_get_scheduler()	95
4.13.1.2 dma2_get_streams()	95
4.13.1.3 dma2_init_scheduler()	96
4.13.1.4 dma_handle_interrupt()	96
4.13.1.5 dma_scheduler_init()	96
4.13.1.6 dma_scheduler_release_stream()	97
4.13.1.7 dma_scheduler_request_stream()	97
4.13.1.8 dma_start_stream()	97
4.13.2 Variable Documentation	97
4.13.2.1 dma2_scheduler	97
4.13.2.2 dma2_streams	98
4.14 dma.h File Reference	98
4.14.1 Macro Definition Documentation	102
4.14.1.1 DMA_STATUS_TC	102
4.14.1.2 DMA_STATUS_TE	102
4.14.1.3 DMA_STATUS_TH	102
4.14.1.4 DMA_STREAMS_MAX_LEN	102
4.14.1.5 STM32_DMAMUX1_ADC1	103
4.14.1.6 STM32_DMAMUX1_ADC2	103
4.14.1.7 STM32_DMAMUX1_CRY2_IN	103
4.14.1.8 STM32_DMAMUX1_CRY2_OUT	103
4.14.1.9 STM32_DMAMUX1_DAC1_CH1	103
4.14.1.10 STM32_DMAMUX1_DAC1_CH2	103
4.14.1.11 STM32_DMAMUX1_DCMI	104
4.14.1.12 STM32_DMAMUX1_DFSDM1_FLT0	104
4.14.1.13 STM32_DMAMUX1_DFSDM1_FLT1	104
4.14.1.14 STM32_DMAMUX1_DFSDM1_FLT2	104
4.14.1.15 STM32_DMAMUX1_DFSDM1_FLT3	104
4.14.1.16 STM32_DMAMUX1_DFSDM1_FLT4	104
4.14.1.17 STM32_DMAMUX1_DFSDM1_FLT5	105
4.14.1.18 STM32_DMAMUX1_HASH2_IN	105
4.14.1.19 STM32_DMAMUX1_I2C1_RX	105

4.14.1.20 STM32_DMAMUX1_I2C1_TX	105
4.14.1.21 STM32_DMAMUX1_I2C2_RX	105
4.14.1.22 STM32_DMAMUX1_I2C2_TX	105
4.14.1.23 STM32_DMAMUX1_I2C3_RX	106
4.14.1.24 STM32_DMAMUX1_I2C3_TX	106
4.14.1.25 STM32_DMAMUX1_I2C5_RX	106
4.14.1.26 STM32_DMAMUX1_I2C5_TX	106
4.14.1.27 STM32_DMAMUX1_REQ_GEN0	106
4.14.1.28 STM32_DMAMUX1_REQ_GEN1	106
4.14.1.29 STM32_DMAMUX1_REQ_GEN2	107
4.14.1.30 STM32_DMAMUX1_REQ_GEN3	107
4.14.1.31 STM32_DMAMUX1_REQ_GEN4	107
4.14.1.32 STM32_DMAMUX1_REQ_GEN5	107
4.14.1.33 STM32_DMAMUX1_REQ_GEN6	107
4.14.1.34 STM32_DMAMUX1_REQ_GEN7	107
4.14.1.35 STM32_DMAMUX1_RSVD117	108
4.14.1.36 STM32_DMAMUX1_RSVD118	108
4.14.1.37 STM32_DMAMUX1_RSVD119	108
4.14.1.38 STM32_DMAMUX1_RSVD120	108
4.14.1.39 STM32_DMAMUX1_RSVD121	108
4.14.1.40 STM32_DMAMUX1_RSVD122	108
4.14.1.41 STM32_DMAMUX1_RSVD123	109
4.14.1.42 STM32_DMAMUX1_RSVD124	109
4.14.1.43 STM32_DMAMUX1_RSVD125	109
4.14.1.44 STM32_DMAMUX1_RSVD126	109
4.14.1.45 STM32_DMAMUX1_RSVD127	109
4.14.1.46 STM32_DMAMUX1_RSVD41	109
4.14.1.47 STM32_DMAMUX1_RSVD42	110
4.14.1.48 STM32_DMAMUX1_RSVD54	110
4.14.1.49 STM32_DMAMUX1_RSVD95	110
4.14.1.50 STM32_DMAMUX1_RSVD96	110
4.14.1.51 STM32_DMAMUX1_RSVD97	110
4.14.1.52 STM32_DMAMUX1_RSVD98	110
4.14.1.53 STM32_DMAMUX1_SAI1_A	111
4.14.1.54 STM32_DMAMUX1_SAI1_B	111
4.14.1.55 STM32_DMAMUX1_SAI2_A	111
4.14.1.56 STM32_DMAMUX1_SAI2_B	111
4.14.1.57 STM32_DMAMUX1_SAI3_A	111
4.14.1.58 STM32_DMAMUX1_SAI3_B	111
4.14.1.59 STM32_DMAMUX1_SAI4_A	112
4.14.1.60 STM32_DMAMUX1_SAI4_B	112
4.14.1.61 STM32_DMAMUX1_SPDIFRX_CS	112

4.14.1.62 STM32_DMAMUX1_SPDIFRX_DT	112
4.14.1.63 STM32_DMAMUX1_SPI1_RX	112
4.14.1.64 STM32_DMAMUX1_SPI1_TX	112
4.14.1.65 STM32_DMAMUX1_SPI2_RX	113
4.14.1.66 STM32_DMAMUX1_SPI2_TX	113
4.14.1.67 STM32_DMAMUX1_SPI3_RX	113
4.14.1.68 STM32_DMAMUX1_SPI3_TX	113
4.14.1.69 STM32_DMAMUX1_SPI4_RX	113
4.14.1.70 STM32_DMAMUX1_SPI4_TX	113
4.14.1.71 STM32_DMAMUX1_SPI5_RX	114
4.14.1.72 STM32_DMAMUX1_SPI5_TX	114
4.14.1.73 STM32_DMAMUX1_TIM15_CH1	114
4.14.1.74 STM32_DMAMUX1_TIM15_COM	114
4.14.1.75 STM32_DMAMUX1_TIM15_TRIG	114
4.14.1.76 STM32_DMAMUX1_TIM15_UP	114
4.14.1.77 STM32_DMAMUX1_TIM16_CH1	115
4.14.1.78 STM32_DMAMUX1_TIM16_UP	115
4.14.1.79 STM32_DMAMUX1_TIM17_CH1	115
4.14.1.80 STM32_DMAMUX1_TIM17_UP	115
4.14.1.81 STM32_DMAMUX1_TIM1_CH1	115
4.14.1.82 STM32_DMAMUX1_TIM1_CH2	115
4.14.1.83 STM32_DMAMUX1_TIM1_CH3	116
4.14.1.84 STM32_DMAMUX1_TIM1_CH4	116
4.14.1.85 STM32_DMAMUX1_TIM1_COM	116
4.14.1.86 STM32_DMAMUX1_TIM1_TRIG	116
4.14.1.87 STM32_DMAMUX1_TIM1_UP	116
4.14.1.88 STM32_DMAMUX1_TIM2_CH1	116
4.14.1.89 STM32_DMAMUX1_TIM2_CH2	117
4.14.1.90 STM32_DMAMUX1_TIM2_CH3	117
4.14.1.91 STM32_DMAMUX1_TIM2_CH4	117
4.14.1.92 STM32_DMAMUX1_TIM2_UP	117
4.14.1.93 STM32_DMAMUX1_TIM3_CH1	117
4.14.1.94 STM32_DMAMUX1_TIM3_CH2	117
4.14.1.95 STM32_DMAMUX1_TIM3_CH3	118
4.14.1.96 STM32_DMAMUX1_TIM3_CH4	118
4.14.1.97 STM32_DMAMUX1_TIM3_TRIG	118
4.14.1.98 STM32_DMAMUX1_TIM3_UP	118
4.14.1.99 STM32_DMAMUX1_TIM4_CH1	118
4.14.1.100 STM32_DMAMUX1_TIM4_CH2	118
4.14.1.101 STM32_DMAMUX1_TIM4_CH3	119
4.14.1.102 STM32_DMAMUX1_TIM4_UP	119
4.14.1.103 STM32_DMAMUX1_TIM5_CH1	119

4.14.1.104 STM32_DMAMUX1_TIM5_CH2	119
4.14.1.105 STM32_DMAMUX1_TIM5_CH3	119
4.14.1.106 STM32_DMAMUX1_TIM5_CH4	119
4.14.1.107 STM32_DMAMUX1_TIM5_TRIG	120
4.14.1.108 STM32_DMAMUX1_TIM5_UP	120
4.14.1.109 STM32_DMAMUX1_TIM6_UP	120
4.14.1.110 STM32_DMAMUX1_TIM7_UP	120
4.14.1.111 STM32_DMAMUX1_TIM8_CH1	120
4.14.1.112 STM32_DMAMUX1_TIM8_CH2	120
4.14.1.113 STM32_DMAMUX1_TIM8_CH3	121
4.14.1.114 STM32_DMAMUX1_TIM8_CH4	121
4.14.1.115 STM32_DMAMUX1_TIM8_COM	121
4.14.1.116 STM32_DMAMUX1_TIM8_TRIG	121
4.14.1.117 STM32_DMAMUX1_TIM8_UP	121
4.14.1.118 STM32_DMAMUX1_UART4_RX	121
4.14.1.119 STM32_DMAMUX1_UART4_TX	122
4.14.1.120 STM32_DMAMUX1_UART5_RX	122
4.14.1.121 STM32_DMAMUX1_UART5_TX	122
4.14.1.122 STM32_DMAMUX1_UART7_RX	122
4.14.1.123 STM32_DMAMUX1_UART7_TX	122
4.14.1.124 STM32_DMAMUX1_UART8_RX	122
4.14.1.125 STM32_DMAMUX1_UART8_TX	123
4.14.1.126 STM32_DMAMUX1_USART2_RX	123
4.14.1.127 STM32_DMAMUX1_USART2_TX	123
4.14.1.128 STM32_DMAMUX1_USART3_RX	123
4.14.1.129 STM32_DMAMUX1_USART3_TX	123
4.14.1.130 STM32_DMAMUX1_USART6_RX	123
4.14.1.131 STM32_DMAMUX1_USART6_TX	124
4.14.2 Typedef Documentation	124
4.14.2.1 dma_request_t	124
4.14.2.2 dma_scheduler_dev_t	124
4.14.2.3 dma_stream_config_t	124
4.14.2.4 dma_stream_dev_t	124
4.14.2.5 dma_stream_dir_t	124
4.14.2.6 dma_stream_state_t	124
4.14.3 Enumeration Type Documentation	124
4.14.3.1 dma_stream_dir	124
4.14.3.2 dma_stream_state	125
4.14.4 Function Documentation	125
4.14.4.1 dma2_get_scheduler()	125
4.14.4.2 dma2_get_streams()	125
4.14.4.3 dma2_init_scheduler()	126

4.14.4.4 dma_copy()	126
4.14.4.5 dma_scheduler_init()	126
4.14.4.6 dma_scheduler_release_stream()	127
4.14.4.7 dma_scheduler_request_stream()	127
4.14.4.8 dma_start_stream()	127
4.14.4.9 dma_stop_stream()	127
4.15 gnss.c File Reference	128
4.16 gnss.h File Reference	128
4.17 gpio.c File Reference	129
4.17.1 Function Documentation	129
4.17.1.1 gpio_cfg()	130
4.17.1.2 gpio_clr()	130
4.17.1.3 gpio_get()	130
4.17.1.4 gpio_set()	131
4.18 gpio.h File Reference	131
4.18.1 Typedef Documentation	132
4.18.1.1 gpio_bias_t	132
4.18.1.2 gpio_config_t	133
4.18.1.3 gpio_drive_t	133
4.18.1.4 gpio_mode_t	133
4.18.2 Enumeration Type Documentation	133
4.18.2.1 gpio_bias	133
4.18.2.2 gpio_drive	133
4.18.2.3 gpio_mode	134
4.18.3 Function Documentation	134
4.18.3.1 gpio_cfg()	134
4.18.3.2 gpio_clr()	134
4.18.3.3 gpio_get()	135
4.18.3.4 gpio_set()	135
4.19 gyroscope.c File Reference	135
4.20 gyroscope.h File Reference	136
4.21 hostproc.c File Reference	136
4.21.1 Typedef Documentation	137
4.21.1.1 hostproc_interface_context_t	138
4.21.2 Function Documentation	138
4.21.2.1 host_rcv()	138
4.21.2.2 host_send()	138
4.21.2.3 host_UART0_RX()	139
4.21.2.4 hostproc_get_device()	139
4.21.2.5 hostproc_get_interface()	140
4.21.2.6 hostproc_init()	140
4.21.3 Variable Documentation	141

4.21.3.1 host_UART0	141
4.21.3.2 hostproc_device	141
4.21.3.3 hostproc_interface	141
4.21.3.4 hostproc_interface_context	141
4.22 hostproc.h File Reference	142
4.22.1 Function Documentation	143
4.22.1.1 hostproc_get_device()	143
4.22.1.2 hostproc_get_interface()	143
4.22.1.3 hostproc_init()	144
4.23 i2c.c File Reference	144
4.23.1 Macro Definition Documentation	145
4.23.1.1 S1_I2C	146
4.23.1.2 S2_I2C	146
4.23.1.3 S3_I2C	146
4.23.2 Function Documentation	146
4.23.2.1 i2c_get_sensor_interface()	146
4.23.2.2 i2c_init()	147
4.23.2.3 i2c_spi_guard()	147
4.23.3 Variable Documentation	148
4.23.3.1 sensor_interface	148
4.23.3.2 sensor_interface_context	148
4.24 i2c.h File Reference	148
4.24.1 Typedef Documentation	149
4.24.1.1 i2c_interface_context_t	149
4.24.2 Function Documentation	149
4.24.2.1 i2c_get_sensor_interface()	150
4.24.2.2 i2c_init()	150
4.24.2.3 i2c_spi_guard()	151
4.25 i2c_sensor.c File Reference	151
4.25.1 Typedef Documentation	152
4.25.1.1 i2c_sensor_context_t	152
4.25.2 Function Documentation	152
4.25.2.1 i2c_get_accelerometer()	152
4.25.2.2 i2c_sensor_init()	153
4.25.2.3 read_reg()	153
4.25.2.4 write_reg()	154
4.25.3 Variable Documentation	154
4.25.3.1 i2c_accelerometer_device	154
4.25.3.2 i2c_accelerometer_device_context	154
4.25.3.3 i2c_barometer_device	155
4.25.3.4 i2c_barometer_device_context	155
4.25.3.5 i2c_gyroscope_device	155

4.25.3.6 i2c_gyroscope_device_context	155
4.26 i2c_sensor.h File Reference	156
4.26.1 Function Documentation	156
4.26.1.1 i2c_sensor_init()	156
4.27 led.c File Reference	157
4.27.1 Macro Definition Documentation	158
4.27.1.1 LED_MAX	158
4.27.1.2 LED_TIM	158
4.27.2 Typedef Documentation	158
4.27.2.1 led_blink_state_t	158
4.27.3 Enumeration Type Documentation	158
4.27.3.1 led_blick_state	158
4.27.4 Function Documentation	159
4.27.4.1 led_feedback_init()	159
4.27.4.2 led_rgb_init()	159
4.27.4.3 led_rgb_set_color()	160
4.27.4.4 led_rgb_set_rgb()	160
4.27.4.5 led_rgb_thread()	161
4.27.5 Variable Documentation	161
4.27.5.1 blink_sequence	162
4.27.5.2 blink_sequence_len	162
4.27.5.3 color_sequence	162
4.27.5.4 color_sequence_len	162
4.28 led.h File Reference	163
4.28.1 Macro Definition Documentation	164
4.28.1.1 LED_BLACK	164
4.28.1.2 LED_BLUE	164
4.28.1.3 LED_GREEN	164
4.28.1.4 LED_LILA	165
4.28.1.5 LED_ORANGE	165
4.28.1.6 LED_PINK	165
4.28.1.7 LED_RED	165
4.28.1.8 LED_TEAL	165
4.28.1.9 LED_WHITE	165
4.28.1.10 LED_YELLOW	166
4.28.2 Typedef Documentation	166
4.28.2.1 led_color_t	166
4.28.3 Function Documentation	166
4.28.3.1 led_feedback_init()	166
4.28.3.2 led_rgb_init()	167
4.28.3.3 led_rgb_set_color()	167
4.28.3.4 led_rgb_set_rgb()	168

4.28.3.5 led_rgb_thread()	168
4.28.4 Variable Documentation	168
4.28.4.1 led_black	168
4.28.4.2 led_blue	169
4.28.4.3 led_green	169
4.28.4.4 led_red	169
4.29 msv2.c File Reference	170
4.29.1 Macro Definition Documentation	170
4.29.1.1 DLE	170
4.29.1.2 STX	171
4.29.2 Function Documentation	171
4.29.2.1 calc_field_CRC()	171
4.29.2.2 msv2_create_frame()	171
4.29.2.3 msv2_decode_fragment()	172
4.29.2.4 msv2_init()	172
4.29.2.5 msv2_rx_data()	173
4.29.2.6 msv2_tx_data()	173
4.30 msv2.h File Reference	173
4.30.1 Macro Definition Documentation	175
4.30.1.1 MSV2_MAX_DATA_LEN	175
4.30.1.2 MSV2_MAX_FRAME_LEN	175
4.30.2 Typedef Documentation	175
4.30.2.1 MSV2_DECODE_STATE_t	175
4.30.2.2 MSV2_ERROR_t	175
4.30.2.3 MSV2_INST_t	175
4.30.2.4 MSV2_RX_DATA_t	175
4.30.2.5 MSV2_TX_DATA_t	175
4.30.3 Enumeration Type Documentation	175
4.30.3.1 MSV2_DECODE_STATE	175
4.30.3.2 MSV2_ERROR	176
4.30.4 Function Documentation	176
4.30.4.1 msv2_create_frame()	176
4.30.4.2 msv2_decode_fragment()	177
4.30.4.3 msv2_init()	177
4.30.4.4 msv2_rx_data()	178
4.30.4.5 msv2_tx_data()	178
4.31 note.h File Reference	178
4.31.1 Macro Definition Documentation	181
4.31.1.1 A0	181
4.31.1.2 A0H	181
4.31.1.3 A1	181
4.31.1.4 A1H	182

4.31.1.5 A2	182
4.31.1.6 A2H	182
4.31.1.7 A3	182
4.31.1.8 A3H	182
4.31.1.9 A4	182
4.31.1.10 A4H	183
4.31.1.11 A5	183
4.31.1.12 A5H	183
4.31.1.13 A6	183
4.31.1.14 A6H	183
4.31.1.15 A7	183
4.31.1.16 A7H	184
4.31.1.17 A8	184
4.31.1.18 A8H	184
4.31.1.19 B0	184
4.31.1.20 B1	184
4.31.1.21 B2	184
4.31.1.22 B3	185
4.31.1.23 B4	185
4.31.1.24 B5	185
4.31.1.25 B6	185
4.31.1.26 B7	185
4.31.1.27 B8	185
4.31.1.28 C0	186
4.31.1.29 C0H	186
4.31.1.30 C1	186
4.31.1.31 C1H	186
4.31.1.32 C2	186
4.31.1.33 C2H	186
4.31.1.34 C3	187
4.31.1.35 C3H	187
4.31.1.36 C4	187
4.31.1.37 C4H	187
4.31.1.38 C5	187
4.31.1.39 C5H	187
4.31.1.40 C6	188
4.31.1.41 C6H	188
4.31.1.42 C7	188
4.31.1.43 C7H	188
4.31.1.44 C8	188
4.31.1.45 C8H	188
4.31.1.46 D0	189

4.31.1.47 D0H	189
4.31.1.48 D1	189
4.31.1.49 D1H	189
4.31.1.50 D2	189
4.31.1.51 D2H	189
4.31.1.52 D3	190
4.31.1.53 D3H	190
4.31.1.54 D4	190
4.31.1.55 D4H	190
4.31.1.56 D5	190
4.31.1.57 D5H	190
4.31.1.58 D6	191
4.31.1.59 D6H	191
4.31.1.60 D7	191
4.31.1.61 D7H	191
4.31.1.62 D8	191
4.31.1.63 D8H	191
4.31.1.64 E0	192
4.31.1.65 E1	192
4.31.1.66 E2	192
4.31.1.67 E3	192
4.31.1.68 E4	192
4.31.1.69 E5	192
4.31.1.70 E6	193
4.31.1.71 E7	193
4.31.1.72 E8	193
4.31.1.73 F0	193
4.31.1.74 F0H	193
4.31.1.75 F1	193
4.31.1.76 F1H	194
4.31.1.77 F2	194
4.31.1.78 F2H	194
4.31.1.79 F3	194
4.31.1.80 F3H	194
4.31.1.81 F4	194
4.31.1.82 F4H	195
4.31.1.83 F5	195
4.31.1.84 F5H	195
4.31.1.85 F6	195
4.31.1.86 F6H	195
4.31.1.87 F7	195
4.31.1.88 F7H	196

4.31.1.89 F8	196
4.31.1.90 F8H	196
4.31.1.91 G0	196
4.31.1.92 G0H	196
4.31.1.93 G1	196
4.31.1.94 G1H	197
4.31.1.95 G2	197
4.31.1.96 G2H	197
4.31.1.97 G3	197
4.31.1.98 G3H	197
4.31.1.99 G4	197
4.31.1.100 G4H	198
4.31.1.101 G5	198
4.31.1.102 G5H	198
4.31.1.103 G6	198
4.31.1.104 G6H	198
4.31.1.105 G7	198
4.31.1.106 G7H	199
4.31.1.107 G8	199
4.31.1.108 G8H	199
4.31.1.109 T1	199
4.31.1.110 T1_1_2	199
4.31.1.111 T1_2	199
4.31.1.112 T1_4	200
4.31.1.113 T2	200
4.31.1.114 T4	200
4.31.2 Typedef Documentation	200
4.31.2.1 note_t	200
4.32 od.c File Reference	200
4.32.1 Macro Definition Documentation	201
4.32.1.1 ALLOCATE_OD_ENTRY	201
4.32.1.2 DEBUG_NO_CAN	202
4.32.1.3 LINK_OD_ENTRY	202
4.32.1.4 OD_MSGQ_SIZE	202
4.32.2 Function Documentation	202
4.32.2.1 od_init()	202
4.32.2.2 od_unsafe_read()	203
4.32.2.3 od_unsafe_write()	203
4.32.2.4 od_update_task()	203
4.32.3 Variable Documentation	204
4.32.3.1 in_q	204
4.32.3.2 od_entries	204

4.32.3.3 out_q	204
4.33 od.h File Reference	205
4.33.1 Macro Definition Documentation	205
4.33.1.1 DECLARE_OD_ENTRY	206
4.33.1.2 OD_FRAME_MAX_SIZE	206
4.33.1.3 OD_MAX_DATAID	206
4.33.2 Function Documentation	206
4.33.2.1 od_init()	206
4.33.2.2 od_update_task()	207
4.34 packet.h File Reference	207
4.34.1 Typedef Documentation	207
4.34.1.1 packet_def_t	207
4.34.2 Variable Documentation	208
4.34.2.1 ping	208
4.35 serial.c File Reference	208
4.35.1 Macro Definition Documentation	209
4.35.1.1 S1_UART	209
4.35.1.2 S2_UART	209
4.35.1.3 S3_UART	209
4.35.1.4 SERIAL_DMA_LEN	210
4.35.2 Function Documentation	210
4.35.2.1 HAL_UART_RxCpltCallback()	210
4.35.2.2 serial_data_ready()	210
4.35.2.3 serial_feedback_init()	211
4.35.2.4 serial_get_deamon()	211
4.35.2.5 serial_get_feedback_interface()	211
4.35.2.6 serial_handle_data()	212
4.35.2.7 serial_init()	212
4.35.2.8 serial_recv()	213
4.35.2.9 serial_send()	214
4.35.2.10 serial_setup_reception()	215
4.35.3 Variable Documentation	215
4.35.3.1 feedback_interface	215
4.35.3.2 feedback_interface_context	216
4.35.3.3 serial_deamon	216
4.35.3.4 serial_deamon_context	216
4.36 serial.h File Reference	216
4.36.1 Macro Definition Documentation	218
4.36.1.1 SERIAL_BUFFER_LEN	218
4.36.2 Typedef Documentation	218
4.36.2.1 serial_deamon_context_t	218
4.36.2.2 serial_interface_context_t	218

4.36.2.3 serial_interrupt_source_t	219
4.36.2.4 serial_transfer_mode_t	219
4.36.3 Enumeration Type Documentation	219
4.36.3.1 serial_interrupt_source	219
4.36.3.2 serial_transfer_mode	219
4.36.4 Function Documentation	219
4.36.4.1 serial_feedback_init()	220
4.36.4.2 serial_get_deamon()	220
4.36.4.3 serial_get_feedback_interface()	220
4.36.4.4 serial_init()	221
4.36.4.5 serial_rcv()	222
4.36.4.6 serial_send()	222
4.37 still_alive.h File Reference	223
4.37.1 Variable Documentation	224
4.37.1.1 still_alive	224
4.37.1.2 still_alive_len	224
4.38 still_alive_bak.h File Reference	224
4.38.1 Variable Documentation	225
4.38.1.1 still_alive	225
4.38.1.2 still_alive_len	225
4.39 template.c File Reference	225
4.40 template.h File Reference	226
4.41 threads.c File Reference	226
4.41.1 Macro Definition Documentation	227
4.41.1.1 CONTROL_PRIO	227
4.41.1.2 CONTROL_SZ	228
4.41.1.3 CREATE_THREAD	228
4.41.1.4 DEFAULT_SZ	228
4.41.1.5 LED_RGB_PRIO	229
4.41.1.6 LED_RGB_SZ	229
4.41.1.7 OD_PRIO	229
4.41.1.8 OD_SZ	229
4.41.2 Function Documentation	229
4.41.2.1 threads_init()	230
4.41.3 Variable Documentation	230
4.41.3.1 control_handle	230
4.41.3.2 led_rgb_handle	231
4.41.3.3 od_handle	231
4.42 threads.h File Reference	231
4.42.1 Function Documentation	232
4.42.1.1 threads_init()	232
4.43 uart.c File Reference	233

4.44 uart.h File Reference	233
4.45 util.h File Reference	234
4.45.1 Macro Definition Documentation	236
4.45.1.1 ENTER_CRITICAL	236
4.45.1.2 EXIT_CRITICAL	236
4.45.1.3 util_abs	236
4.45.1.4 UTIL_GENERATE_BUFFER	236
4.45.1.5 WRITE_IN_REG	237
4.45.2 Typedef Documentation	237
4.45.2.1 util_buffer_i16_t	237
4.45.2.2 util_buffer_u16_t	238
4.45.2.3 util_buffer_u8_t	238
4.45.2.4 util_error_t	238
4.45.3 Enumeration Type Documentation	238
4.45.3.1 util_error	238
4.45.4 Function Documentation	239
4.45.4.1 util_buffer_i16_add()	239
4.45.4.2 util_buffer_i16_get()	239
4.45.4.3 util_buffer_i16_init()	239
4.45.4.4 util_buffer_i16_isempty()	239
4.45.4.5 util_buffer_u16_add()	240
4.45.4.6 util_buffer_u16_get()	240
4.45.4.7 util_buffer_u16_init()	240
4.45.4.8 util_buffer_u16_isempty()	240
4.45.4.9 util_buffer_u8_access()	241
4.45.4.10 util_buffer_u8_add()	241
4.45.4.11 util_buffer_u8_get()	241
4.45.4.12 util_buffer_u8_init()	242
4.45.4.13 util_buffer_u8_isempty()	242
4.45.4.14 util_decode_i16()	243
4.45.4.15 util_decode_i32()	243
4.45.4.16 util_decode_i8()	244
4.45.4.17 util_decode_u16()	244
4.45.4.18 util_decode_u32()	245
4.45.4.19 util_decode_u8()	245
4.45.4.20 util_encode_i16()	246
4.45.4.21 util_encode_i32()	246
4.45.4.22 util_encode_i8()	247
4.45.4.23 util_encode_u16()	247
4.45.4.24 util_encode_u32()	248
4.45.4.25 util_encode_u8()	248
4.46 wildhorn.h File Reference	249

4.46.1 Macro Definition Documentation	249
4.46.1.1 WH_FALSE	250
4.46.1.2 WH_HAS_FEEDBACK	250
4.46.1.3 WH_HAS_GNSS	250
4.46.1.4 WH_HAS_KRTEK	250
4.46.1.5 WH_HAS_RADIO	250
4.46.1.6 WH_HAS_SENSORS	250
4.46.1.7 WH_TRUE	250
4.46.1.8 WH_USE_BUZZER	250
Index	251

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

control	5
debug_context	6
debug_interface_context	6
device	7
device_deamon	9
device_interface	11
dma_request	13
dma_scheduler_dev	14
dma_stream_config	16
dma_stream_dev	19
gpio_config	21
hostproc_interface_context	23
i2c_interface_context	24
i2c_sensor_context	24
led_color	25
MSV2_INST	26
MSV2_RX_DATA	27
MSV2_TX_DATA	30
note	31
od_entry_t	32
od_frame_t	33
packet_def	34
serial_deamon_context	35
serial_interface_context	36
util_buffer_i16	38
util_buffer_u16	39
util_buffer_u8	40

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

accelerometer.c	43
accelerometer.h	44
barometer.c	45
barometer.h	45
buzzer.c	46
buzzer.h	52
control.c	55
control.h	66
debug.c	67
debug.h	70
device.c	70
device.h	82
dma.c	94
dma.h	98
gnss.c	128
gnss.h	128
gpio.c	129
gpio.h	131
gyroscope.c	135
gyroscope.h	136
hostproc.c	136
hostproc.h	142
i2c.c	144
i2c.h	148
i2c_sensor.c	151
i2c_sensor.h	156
led.c	157
led.h	163
msv2.c	170
msv2.h	173
note.h	178
od.c	200
od.h	205
packet.h	207
serial.c	208

serial.h	216
still_alive.h	223
still_alive_bak.h	224
template.c	225
template.h	226
threads.c	226
threads.h	231
uart.c	233
uart.h	233
util.h	234
wildhorn.h	249

Chapter 3

Data Structure Documentation

3.1 control Struct Reference

Data Fields

- [control_state_t state](#)

3.1.1 Detailed Description

Definition at line 80 of file control.c.

3.1.2 Field Documentation

3.1.2.1 state

```
control_state_t control::state
```

Definition at line 81 of file control.c.

Referenced by `control_abort_start()`, `control_apogee_start()`, `control_armed_start()`, `control_ballistic_start()`, `control_calibration_start()`, `control_coast_start()`, `control_drogue_start()`, `control_error_start()`, `control_event_start()`, `control_idle_start()`, `control_main_start()`, `control_powered_start()`, `control_supersonic_start()`, and `control_touchdown_start()`.

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

- [control.c](#)

3.2 debug_context Struct Reference

3.2.1 Detailed Description

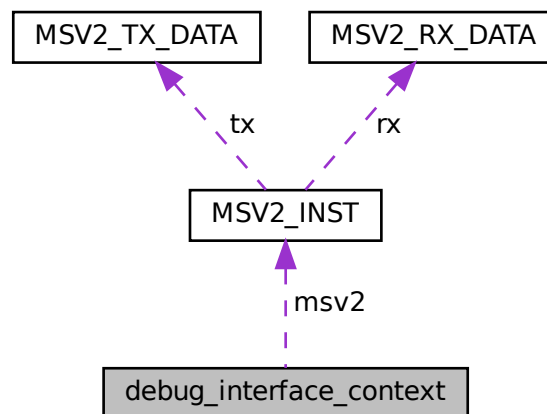
Definition at line 42 of file debug.c.

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

- [debug.c](#)

3.3 debug_interface_context Struct Reference

Collaboration diagram for debug_interface_context:



Data Fields

- [MSV2_INST_t msv2](#)

3.3.1 Detailed Description

Definition at line 34 of file debug.c.

3.3.2 Field Documentation

3.3.2.1 msv2

`MSV2_INST_t` `debug_interface_context::msv2`

Definition at line 35 of file `debug.c`.

Referenced by `debug_init()`.

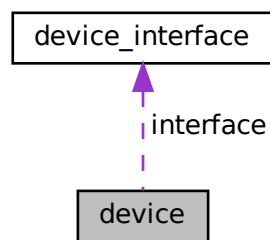
The documentation for this struct was generated from the following file:

- `debug.c`

3.4 device Struct Reference

```
#include <device.h>
```

Collaboration diagram for `device`:



Data Fields

- `uint32_t` `id`
- `device_interface_t` * `interface`
- `void` * `context`
- `util_error_t`(* `read_reg`)(`void` *, `device_interface_t` *, `uint32_t`, `uint8_t` *, `uint32_t`)
- `util_error_t`(* `write_reg`)(`void` *, `device_interface_t` *, `uint32_t`, `uint8_t` *, `uint32_t`)

3.4.1 Detailed Description

Definition at line 65 of file `device.h`.

3.4.2 Field Documentation

3.4.2.1 context

```
void* device::context
```

Definition at line 68 of file device.h.

Referenced by `device_create()`, `device_read_i16()`, `device_read_i32()`, `device_read_i8()`, `device_read_u16()`, `device_read_u32()`, `device_read_u8()`, `device_write_i16()`, `device_write_i32()`, `device_write_i8()`, `device_write_u16()`, `device_write_u32()`, and `device_write_u8()`.

3.4.2.2 id

```
uint32_t device::id
```

Definition at line 66 of file device.h.

Referenced by `device_create()`.

3.4.2.3 interface

```
device_interface_t* device::interface
```

Definition at line 67 of file device.h.

Referenced by `device_create()`, `device_read_i16()`, `device_read_i32()`, `device_read_i8()`, `device_read_u16()`, `device_read_u32()`, `device_read_u8()`, `device_write_i16()`, `device_write_i32()`, `device_write_i8()`, `device_write_u16()`, `device_write_u32()`, and `device_write_u8()`.

3.4.2.4 read_reg

```
util_error_t(* device::read_reg) (void *, device_interface_t *, uint32_t, uint8_t *, uint32_t)
```

Definition at line 70 of file device.h.

Referenced by `device_create()`, `device_read_i16()`, `device_read_i32()`, `device_read_i8()`, `device_read_u16()`, `device_read_u32()`, and `device_read_u8()`.

3.4.2.5 write_reg

```
util_error_t(* device::write_reg) (void *, device_interface_t *, uint32_t, uint8_t *, uint32_t)
```

Definition at line 72 of file device.h.

Referenced by device_create(), device_write_i16(), device_write_i32(), device_write_i8(), device_write_u16(), device_write_u32(), and device_write_u8().

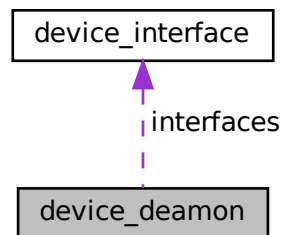
The documentation for this struct was generated from the following file:

- [device.h](#)

3.5 device_daemon Struct Reference

```
#include <device.h>
```

Collaboration diagram for device_daemon:



Data Fields

- uint32_t id
- StaticTask_t buffer
- StackType_t stack [DEAMON_STACK_SIZE]
- TaskHandle_t handle
- uint32_t interfaces_count
- device_interface_t * interfaces [DEVICE_MAX_INTERFACES_PER_DEAMON]
- void * context
- util_error_t(* data_rdy)(void *)

3.5.1 Detailed Description

Definition at line 54 of file device.h.

3.5.2 Field Documentation

3.5.2.1 buffer

`StaticTask_t device_deamon::buffer`

Definition at line 56 of file device.h.

Referenced by `device_deamon_create()`.

3.5.2.2 context

`void* device_deamon::context`

Definition at line 61 of file device.h.

Referenced by `device_deamon_create()`, `device_deamon_thread()`, and `HAL_UART_RxCpltCallback()`.

3.5.2.3 data_rdy

`util_error_t(* device_deamon::data_rdy) (void *)`

Definition at line 62 of file device.h.

Referenced by `device_deamon_create()`, and `device_deamon_thread()`.

3.5.2.4 handle

`TaskHandle_t device_deamon::handle`

Definition at line 58 of file device.h.

Referenced by `device_deamon_create()`.

3.5.2.5 id

`uint32_t device_deamon::id`

Definition at line 55 of file device.h.

Referenced by `device_deamon_create()`.

3.5.2.6 interfaces

```
device_interface_t* device_deamon::interfaces[DEVICE_MAX_INTERFACES_PER_DEAMON]
```

Definition at line 60 of file device.h.

Referenced by device_interface_create(), and HAL_UART_RxCpltCallback().

3.5.2.7 interfaces_count

```
uint32_t device_deamon::interfaces_count
```

Definition at line 59 of file device.h.

Referenced by device_deamon_create(), device_deamon_thread(), device_interface_create(), and HAL_UART_RxCpltCallback().

3.5.2.8 stack

```
StackType_t device_deamon::stack[DEAMON_STACK_SIZE]
```

Definition at line 57 of file device.h.

Referenced by device_deamon_create().

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

- [device.h](#)

3.6 device_interface Struct Reference

```
#include <device.h>
```

Data Fields

- uint32_t [id](#)
- void * [context](#)
- util_error_t(* [send](#))(void *, uint8_t *, uint32_t)
- util_error_t(* [recv](#))(void *, uint8_t *, uint32_t *)
- util_error_t(* [handle_data](#))(void *, void *)

3.6.1 Detailed Description

Definition at line 42 of file device.h.

3.6.2 Field Documentation

3.6.2.1 context

```
void* device_interface::context
```

Definition at line 44 of file device.h.

Referenced by `device_deamon_thread()`, `device_interface_create()`, `device_interface_rcv()`, `device_interface_send()`, `HAL_UART_RxCpltCallback()`, `read_reg()`, and `write_reg()`.

3.6.2.2 handle_data

```
util_error_t(* device_interface::handle_data) (void *, void *)
```

Definition at line 50 of file device.h.

Referenced by `device_deamon_thread()`, and `device_interface_create()`.

3.6.2.3 id

```
uint32_t device_interface::id
```

Definition at line 43 of file device.h.

Referenced by `device_interface_create()`.

3.6.2.4 rcv

```
util_error_t(* device_interface::rcv) (void *, uint8_t *, uint32_t *)
```

Definition at line 48 of file device.h.

Referenced by `device_interface_create()`, and `device_interface_rcv()`.

3.6.2.5 send

```
util_error_t(* device_interface::send) (void *, uint8_t *, uint32_t)
```

Definition at line 46 of file device.h.

Referenced by device_interface_create(), device_interface_rcv(), and device_interface_send().

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

- [device.h](#)

3.7 dma_request Struct Reference

```
#include <dma.h>
```

Data Fields

- uint32_t [src](#)
- uint32_t [dst](#)
- uint32_t [transfer_len](#)
- uint8_t [dst_inc](#)
- uint8_t [src_inc](#)

3.7.1 Detailed Description

Definition at line 193 of file dma.h.

3.7.2 Field Documentation

3.7.2.1 dst

```
uint32_t dma_request::dst
```

Definition at line 195 of file dma.h.

3.7.2.2 dst_inc

```
uint8_t dma_request::dst_inc
```

Definition at line 197 of file dma.h.

3.7.2.3 src

```
uint32_t dma_request::src
```

Definition at line 194 of file dma.h.

3.7.2.4 src_inc

```
uint8_t dma_request::src_inc
```

Definition at line 198 of file dma.h.

3.7.2.5 tranfser_len

```
uint32_t dma_request::tranfser_len
```

Definition at line 196 of file dma.h.

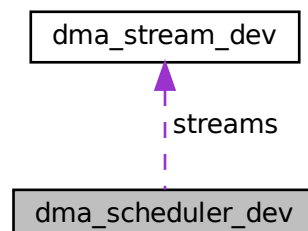
The documentation for this struct was generated from the following file:

- [dma.h](#)

3.8 dma_scheduler_dev Struct Reference

```
#include <dma.h>
```

Collaboration diagram for dma_scheduler_dev:



Data Fields

- uint16_t [stream_count](#)
- uint16_t [free_stream_count](#)
- [dma_stream_dev_t](#) * [streams](#) [[DMA_STREAMS_MAX_LEN](#)]

3.8.1 Detailed Description

Definition at line 233 of file dma.h.

3.8.2 Field Documentation

3.8.2.1 free_stream_count

```
uint16_t dma_scheduler_dev::free_stream_count
```

Definition at line 235 of file dma.h.

Referenced by [dma_scheduler_release_stream\(\)](#), and [dma_scheduler_request_stream\(\)](#).

3.8.2.2 stream_count

```
uint16_t dma_scheduler_dev::stream_count
```

Definition at line 234 of file dma.h.

Referenced by [dma_scheduler_request_stream\(\)](#).

3.8.2.3 streams

```
dma\_stream\_dev\_t* dma_scheduler_dev::streams [DMA\_STREAMS\_MAX\_LEN]
```

Definition at line 236 of file dma.h.

Referenced by [dma_scheduler_init\(\)](#), and [dma_scheduler_request_stream\(\)](#).

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

- [dma.h](#)

3.9 dma_stream_config Struct Reference

```
#include <dma.h>
```

Data Fields

- uint32_t [stream_number](#)
- uint32_t [p_addr](#)
- uint32_t [m0_addr](#)
- uint32_t [m1_addr](#)
- uint32_t [transfer_size](#)
- uint16_t [dmamux_request_number](#)
- uint8_t [priority](#)
- [dma_stream_dir_t](#) [direction](#)
- uint8_t [peripheral_flow_control](#)
- void * [user_context](#)
- void(* [transfer_cplt](#))(void *)
- void(* [transfer_half](#))(void *)
- void(* [transfer_error](#))(void *)

3.9.1 Detailed Description

Definition at line 202 of file dma.h.

3.9.2 Field Documentation

3.9.2.1 direction

```
dma\_stream\_dir\_t dma_stream_config::direction
```

Definition at line 210 of file dma.h.

Referenced by [dma_start_stream\(\)](#).

3.9.2.2 dmamux_request_number

```
uint16_t dma_stream_config::dmamux_request_number
```

Definition at line 208 of file dma.h.

Referenced by [dma_start_stream\(\)](#).

3.9.2.3 m0_addr

```
uint32_t dma_stream_config::m0_addr
```

Definition at line 205 of file dma.h.

Referenced by dma_start_stream().

3.9.2.4 m1_addr

```
uint32_t dma_stream_config::m1_addr
```

Definition at line 206 of file dma.h.

Referenced by dma_start_stream().

3.9.2.5 p_addr

```
uint32_t dma_stream_config::p_addr
```

Definition at line 204 of file dma.h.

Referenced by dma_start_stream().

3.9.2.6 peripheral_flow_control

```
uint8_t dma_stream_config::peripheral_flow_control
```

Definition at line 211 of file dma.h.

Referenced by dma_start_stream().

3.9.2.7 priority

```
uint8_t dma_stream_config::priority
```

Definition at line 209 of file dma.h.

Referenced by dma_start_stream().

3.9.2.8 stream_number

```
uint32_t dma_stream_config::stream_number
```

Definition at line 203 of file dma.h.

3.9.2.9 transfer_cplt

```
void(* dma_stream_config::transfer_cplt) (void *)
```

Definition at line 213 of file dma.h.

Referenced by dma_start_stream().

3.9.2.10 transfer_error

```
void(* dma_stream_config::transfer_error) (void *)
```

Definition at line 215 of file dma.h.

Referenced by dma_start_stream().

3.9.2.11 transfer_half

```
void(* dma_stream_config::transfer_half) (void *)
```

Definition at line 214 of file dma.h.

Referenced by dma_start_stream().

3.9.2.12 transfer_size

```
uint32_t dma_stream_config::transfer_size
```

Definition at line 207 of file dma.h.

Referenced by dma_start_stream().

3.9.2.13 user_context

```
void* dma_stream_config::user_context
```

Definition at line 212 of file dma.h.

Referenced by dma_start_stream().

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

- [dma.h](#)

3.10 dma_stream_dev Struct Reference

```
#include <dma.h>
```

Data Fields

- DMA_TypeDef * [dma](#)
- DMA_Stream_TypeDef * [dma_stream](#)
- DMAMUX_Channel_TypeDef * [dmamux_channel](#)
- DMAMUX_ChannelStatus_TypeDef * [dmamux_channel_status](#)
- [dma_stream_state_t](#) state
- uint16_t number
- void * [user_context](#)
- void(* [transfer_cplt](#))(void *)
- void(* [transfer_half](#))(void *)
- void(* [transfer_error](#))(void *)

3.10.1 Detailed Description

Definition at line 219 of file dma.h.

3.10.2 Field Documentation

3.10.2.1 dma

```
DMA_TypeDef* dma_stream_dev::dma
```

Definition at line 220 of file dma.h.

Referenced by dma_handle_interrupt(), and dma_start_stream().

3.10.2.2 dma_stream

```
DMA_Stream_TypeDef* dma_stream_dev::dma_stream
```

Definition at line 221 of file dma.h.

Referenced by dma_start_stream().

3.10.2.3 dmamux_channel

```
DMAMUX_Channel_TypeDef* dma_stream_dev::dmamux_channel
```

Definition at line 222 of file dma.h.

Referenced by dma_start_stream().

3.10.2.4 dmamux_channel_status

```
DMAMUX_ChannelStatus_TypeDef* dma_stream_dev::dmamux_channel_status
```

Definition at line 223 of file dma.h.

3.10.2.5 number

```
uint16_t dma_stream_dev::number
```

Definition at line 225 of file dma.h.

Referenced by dma_handle_interrupt(), and dma_start_stream().

3.10.2.6 state

```
dma_stream_state_t dma_stream_dev::state
```

Definition at line 224 of file dma.h.

Referenced by dma_scheduler_init(), dma_scheduler_release_stream(), and dma_scheduler_request_stream().

3.10.2.7 transfer_cplt

```
void(* dma_stream_dev::transfer_cplt) (void *)
```

Definition at line 227 of file dma.h.

Referenced by dma_handle_interrupt(), and dma_start_stream().

3.10.2.8 transfer_error

```
void(* dma_stream_dev::transfer_error) (void *)
```

Definition at line 229 of file dma.h.

Referenced by dma_handle_interrupt(), and dma_start_stream().

3.10.2.9 transfer_half

```
void(* dma_stream_dev::transfer_half) (void *)
```

Definition at line 228 of file dma.h.

Referenced by dma_handle_interrupt(), and dma_start_stream().

3.10.2.10 user_context

```
void* dma_stream_dev::user_context
```

Definition at line 226 of file dma.h.

Referenced by dma_handle_interrupt(), and dma_start_stream().

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

- [dma.h](#)

3.11 gpio_config Struct Reference

```
#include <gpio.h>
```

Data Fields

- [gpio_drive_t](#) drive
- [gpio_mode_t](#) mode
- [gpio_bias_t](#) bias
- [uint8_t](#) speed
- [uint8_t](#) alternate

3.11.1 Detailed Description

Definition at line 56 of file gpio.h.

3.11.2 Field Documentation

3.11.2.1 alternate

```
uint8_t gpio_config::alternate
```

Definition at line 61 of file gpio.h.

Referenced by `gpio_cfg()`.

3.11.2.2 bias

```
gpio_bias_t gpio_config::bias
```

Definition at line 59 of file gpio.h.

3.11.2.3 drive

```
gpio_drive_t gpio_config::drive
```

Definition at line 57 of file gpio.h.

Referenced by `gpio_cfg()`.

3.11.2.4 mode

`gpio_mode_t` `gpio_config::mode`

Definition at line 58 of file `gpio.h`.

Referenced by `gpio_cfg()`.

3.11.2.5 speed

`uint8_t` `gpio_config::speed`

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

Referenced by `gpio_cfg()`.

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

- [gpio.h](#)

3.12 hostproc_interface_context Struct Reference

Data Fields

- `VIRT_UART_HandleTypeDef * uart`
- `uint8_t rx_once`

3.12.1 Detailed Description

Definition at line 40 of file `hostproc.c`.

3.12.2 Field Documentation

3.12.2.1 rx_once

`uint8_t` `hostproc_interface_context::rx_once`

Definition at line 42 of file `hostproc.c`.

Referenced by `host_send()`, `host_UART0_RX()`, and `hostproc_init()`.

3.12.2.2 uart

```
VIRT_UART_HandleTypeDef* hostproc_interface_context::uart
```

Definition at line 41 of file hostproc.c.

Referenced by `host_send()`, and `hostproc_init()`.

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

- [hostproc.c](#)

3.13 i2c_interface_context Struct Reference

```
#include <i2c.h>
```

Data Fields

- I2C_HandleTypeDef * [i2c](#)

3.13.1 Detailed Description

Definition at line 36 of file i2c.h.

3.13.2 Field Documentation

3.13.2.1 i2c

```
I2C_HandleTypeDef* i2c_interface_context::i2c
```

Definition at line 37 of file i2c.h.

Referenced by `read_reg()`, and `write_reg()`.

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

- [i2c.h](#)

3.14 i2c_sensor_context Struct Reference

Data Fields

- uint8_t [device_address](#)

3.14.1 Detailed Description

Definition at line 31 of file i2c_sensor.c.

3.14.2 Field Documentation

3.14.2.1 device_address

```
uint8_t i2c_sensor_context::device_address
```

Definition at line 32 of file i2c_sensor.c.

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

- [i2c_sensor.c](#)

3.15 led_color Struct Reference

```
#include <led.h>
```

Data Fields

- [uint8_t r](#)
- [uint8_t g](#)
- [uint8_t b](#)

3.15.1 Detailed Description

Definition at line 50 of file led.h.

3.15.2 Field Documentation

3.15.2.1 b

```
uint8_t led_color::b
```

Definition at line 53 of file led.h.

Referenced by [led_rgb_set_color\(\)](#).

3.15.2.2 g

```
uint8_t led_color::g
```

Definition at line 52 of file led.h.

Referenced by led_rgb_set_color().

3.15.2.3 r

```
uint8_t led_color::r
```

Definition at line 51 of file led.h.

Referenced by led_rgb_set_color().

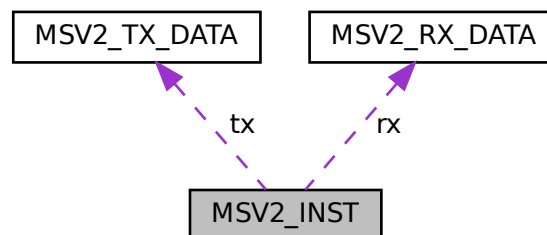
The documentation for this struct was generated from the following file:

- [led.h](#)

3.16 MSV2_INST Struct Reference

```
#include <msv2.h>
```

Collaboration diagram for MSV2_INST:



Data Fields

- `uint32_t id`
- [MSV2_RX_DATA_t rx](#)
- [MSV2_TX_DATA_t tx](#)

3.16.1 Detailed Description

Definition at line 74 of file msv2.h.

3.16.2 Field Documentation

3.16.2.1 id

```
uint32_t MSV2_INST::id
```

Definition at line 75 of file msv2.h.

Referenced by `msv2_init()`.

3.16.2.2 rx

```
MSV2_RX_DATA_t MSV2_INST::rx
```

Definition at line 76 of file msv2.h.

Referenced by `msv2_decode_fragment()`, and `msv2_rx_data()`.

3.16.2.3 tx

```
MSV2_TX_DATA_t MSV2_INST::tx
```

Definition at line 77 of file msv2.h.

Referenced by `msv2_create_frame()`, and `msv2_tx_data()`.

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

- [msv2.h](#)

3.17 MSV2_RX_DATA Struct Reference

```
#include <msv2.h>
```

Data Fields

- `uint8_t opcode`
- `uint8_t data_len`
- `uint16_t crc`
- `MSV2_DECODE_STATE_t state`
- `uint8_t escape`
- `uint16_t length`
- `uint16_t counter`
- `uint8_t data` [`MSV2_MAX_FRAME_LEN`]
- `uint16_t crc_data` [`MSV2_MAX_FRAME_LEN/sizeof(uint16_t)`]

3.17.1 Detailed Description

Definition at line 54 of file `msv2.h`.

3.17.2 Field Documentation

3.17.2.1 counter

```
uint16_t MSV2_RX_DATA::counter
```

Definition at line 61 of file `msv2.h`.

Referenced by `msv2_decode_fragment()`.

3.17.2.2 crc

```
uint16_t MSV2_RX_DATA::crc
```

Definition at line 57 of file `msv2.h`.

Referenced by `msv2_decode_fragment()`.

3.17.2.3 crc_data

```
uint16_t MSV2_RX_DATA::crc_data[MSV2_MAX_FRAME_LEN/sizeof(uint16_t)]
```

Definition at line 63 of file `msv2.h`.

Referenced by `msv2_decode_fragment()`.

3.17.2.4 data

```
uint8_t MSV2_RX_DATA::data[MSV2_MAX_FRAME_LEN]
```

Definition at line 62 of file msv2.h.

Referenced by `msv2_decode_fragment()`, and `msv2_rx_data()`.

3.17.2.5 data_len

```
uint8_t MSV2_RX_DATA::data_len
```

Definition at line 56 of file msv2.h.

Referenced by `msv2_decode_fragment()`.

3.17.2.6 escape

```
uint8_t MSV2_RX_DATA::escape
```

Definition at line 59 of file msv2.h.

Referenced by `msv2_decode_fragment()`.

3.17.2.7 length

```
uint16_t MSV2_RX_DATA::length
```

Definition at line 60 of file msv2.h.

Referenced by `msv2_decode_fragment()`.

3.17.2.8 opcode

```
uint8_t MSV2_RX_DATA::opcode
```

Definition at line 55 of file msv2.h.

Referenced by `msv2_decode_fragment()`.

3.17.2.9 state

`MSV2_DECODE_STATE_t MSV2_RX_DATA::state`

Definition at line 58 of file `msv2.h`.

Referenced by `msv2_decode_fragment()`.

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

- [msv2.h](#)

3.18 MSV2_TX_DATA Struct Reference

```
#include <msv2.h>
```

Data Fields

- `uint8_t opcode`
- `uint8_t data_len`
- `uint16_t crc`
- `uint8_t data [MSV2_MAX_FRAME_LEN]`
- `uint16_t crc_data [MSV2_MAX_FRAME_LEN/sizeof(uint16_t)]`

3.18.1 Detailed Description

Definition at line 66 of file `msv2.h`.

3.18.2 Field Documentation

3.18.2.1 crc

`uint16_t MSV2_TX_DATA::crc`

Definition at line 69 of file `msv2.h`.

3.18.2.2 crc_data

`uint16_t MSV2_TX_DATA::crc_data [MSV2_MAX_FRAME_LEN/sizeof(uint16_t)]`

Definition at line 71 of file `msv2.h`.

Referenced by `msv2_create_frame()`.

3.18.2.3 data

```
uint8_t MSV2_TX_DATA::data[MSV2_MAX_FRAME_LEN]
```

Definition at line 70 of file msv2.h.

Referenced by `msv2_create_frame()`, and `msv2_tx_data()`.

3.18.2.4 data_len

```
uint8_t MSV2_TX_DATA::data_len
```

Definition at line 68 of file msv2.h.

Referenced by `msv2_create_frame()`.

3.18.2.5 opcode

```
uint8_t MSV2_TX_DATA::opcode
```

Definition at line 67 of file msv2.h.

Referenced by `msv2_create_frame()`.

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

- [msv2.h](#)

3.19 note Struct Reference

```
#include <note.h>
```

Data Fields

- `uint16_t` [freq](#)
- `uint16_t` [time](#)

3.19.1 Detailed Description

Definition at line 5 of file note.h.

3.19.2 Field Documentation

3.19.2.1 freq

```
uint16_t note::freq
```

Definition at line 6 of file note.h.

3.19.2.2 time

```
uint16_t note::time
```

Definition at line 7 of file note.h.

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

- [note.h](#)

3.20 od_entry_t Struct Reference

Data Fields

- uint8_t [data_id](#)
- uint8_t [size](#)
- uint8_t * [data](#)

3.20.1 Detailed Description

Definition at line 49 of file od.c.

3.20.2 Field Documentation

3.20.2.1 data

```
uint8_t* od_entry_t::data
```

Definition at line 52 of file od.c.

Referenced by `od_unsafe_read()`, and `od_update_task()`.

3.20.2.2 data_id

```
uint8_t od_entry_t::data_id
```

Definition at line 50 of file od.c.

Referenced by od_unsafe_write().

3.20.2.3 size

```
uint8_t od_entry_t::size
```

Definition at line 51 of file od.c.

Referenced by od_unsafe_read(), od_unsafe_write(), and od_update_task().

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

- [od.c](#)

3.21 od_frame_t Struct Reference

Data Fields

- uint8_t [data_id](#)
- uint8_t [size](#)
- uint8_t [data](#) [[OD_FRAME_MAX_SIZE](#)]

3.21.1 Detailed Description

Definition at line 55 of file od.c.

3.21.2 Field Documentation

3.21.2.1 data

```
uint8_t od_frame_t::data [OD\_FRAME\_MAX\_SIZE]
```

Definition at line 58 of file od.c.

Referenced by od_unsafe_write(), and od_update_task().

3.21.2.2 data_id

```
uint8_t od_frame_t::data_id
```

Definition at line 56 of file od.c.

Referenced by `od_unsafe_write()`, and `od_update_task()`.

3.21.2.3 size

```
uint8_t od_frame_t::size
```

Definition at line 57 of file od.c.

Referenced by `od_unsafe_write()`.

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

- [od.c](#)

3.22 packet_def Struct Reference

```
#include <packet.h>
```

Data Fields

- uint8_t [opcode](#)
- uint8_t [len](#)

3.22.1 Detailed Description

Definition at line 25 of file packet.h.

3.22.2 Field Documentation

3.22.2.1 len

```
uint8_t packet_def::len
```

Definition at line 27 of file packet.h.

3.22.2.2 opcode

```
uint8_t packet_def::opcode
```

Definition at line 26 of file packet.h.

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

- [packet.h](#)

3.23 serial_deamon_context Struct Reference

```
#include <serial.h>
```

Data Fields

- SemaphoreHandle_t [rx_sem](#)
- StaticSemaphore_t [rx_sem_buffer](#)

3.23.1 Detailed Description

Definition at line 50 of file serial.h.

3.23.2 Field Documentation

3.23.2.1 rx_sem

```
SemaphoreHandle_t serial_deamon_context::rx_sem
```

Definition at line 51 of file serial.h.

Referenced by HAL_UART_RxCpltCallback(), serial_data_ready(), and serial_init().

3.23.2.2 rx_sem_buffer

```
StaticSemaphore_t serial_deamon_context::rx_sem_buffer
```

Definition at line 52 of file serial.h.

Referenced by serial_init().

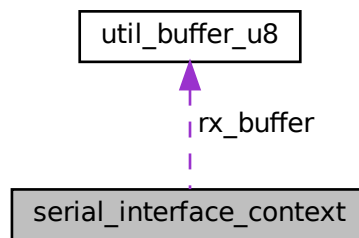
The documentation for this struct was generated from the following file:

- [serial.h](#)

3.24 serial_interface_context Struct Reference

```
#include <serial.h>
```

Collaboration diagram for serial_interface_context:



Data Fields

- UART_HandleTypeDef * [uart](#)
- [util_buffer_u8_t](#) [rx_buffer](#)
- uint8_t [rx_data](#) [SERIAL_BUFFER_LEN]
- uint32_t [rx_data_len](#)
- uint8_t [rx_fragment](#)
- uint8_t [tx_data](#) [SERIAL_BUFFER_LEN]
- void * [protocol](#)

3.24.1 Detailed Description

Definition at line 56 of file serial.h.

3.24.2 Field Documentation

3.24.2.1 protocol

```
void* serial_interface_context::protocol
```

Definition at line 63 of file serial.h.

3.24.2.2 rx_buffer

```
util_buffer_u8_t serial_interface_context::rx_buffer
```

Definition at line 58 of file serial.h.

Referenced by HAL_UART_RxCpltCallback(), serial_recv(), and serial_setup_reception().

3.24.2.3 rx_data

```
uint8_t serial_interface_context::rx_data[SERIAL\_BUFFER\_LEN]
```

Definition at line 59 of file serial.h.

Referenced by serial_setup_reception().

3.24.2.4 rx_data_len

```
uint32_t serial_interface_context::rx_data_len
```

Definition at line 60 of file serial.h.

3.24.2.5 rx_fragment

```
uint8_t serial_interface_context::rx_fragment
```

Definition at line 61 of file serial.h.

Referenced by HAL_UART_RxCpltCallback(), and serial_setup_reception().

3.24.2.6 tx_data

```
uint8_t serial_interface_context::tx_data[SERIAL\_BUFFER\_LEN]
```

Definition at line 62 of file serial.h.

3.24.2.7 uart

```
UART_HandleTypeDef* serial_interface_context::uart
```

Definition at line 57 of file serial.h.

Referenced by HAL_UART_RxCpltCallback(), serial_send(), and serial_setup_reception().

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

- [serial.h](#)

3.25 util_buffer_i16 Struct Reference

```
#include <util.h>
```

Data Fields

- uint16_t [c_ix](#)
- uint16_t [l_ix](#)
- uint16_t [bfr_len](#)
- int16_t * [buffer](#)

3.25.1 Detailed Description

Definition at line 104 of file util.h.

3.25.2 Field Documentation

3.25.2.1 bfr_len

```
uint16_t util_buffer_i16::bfr_len
```

Definition at line 107 of file util.h.

Referenced by util_buffer_i16_add(), util_buffer_i16_get(), and util_buffer_i16_init().

3.25.2.2 buffer

```
int16_t* util_buffer_i16::buffer
```

Definition at line 108 of file util.h.

Referenced by util_buffer_i16_add(), util_buffer_i16_get(), and util_buffer_i16_init().

3.25.2.3 c_ix

```
uint16_t util_buffer_i16::c_ix
```

Definition at line 105 of file util.h.

Referenced by util_buffer_i16_add(), util_buffer_i16_init(), and util_buffer_i16_isempty().

3.25.2.4 l_ix

```
uint16_t util_buffer_i16::l_ix
```

Definition at line 106 of file util.h.

Referenced by util_buffer_i16_get(), util_buffer_i16_init(), and util_buffer_i16_isempty().

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

- [util.h](#)

3.26 util_buffer_u16 Struct Reference

```
#include <util.h>
```

Data Fields

- uint16_t [c_ix](#)
- uint16_t [l_ix](#)
- uint16_t [bfr_len](#)
- uint16_t * [buffer](#)

3.26.1 Detailed Description

Definition at line 97 of file util.h.

3.26.2 Field Documentation

3.26.2.1 bfr_len

```
uint16_t util_buffer_u16::bfr_len
```

Definition at line 100 of file util.h.

Referenced by util_buffer_u16_add(), util_buffer_u16_get(), and util_buffer_u16_init().

3.26.2.2 buffer

```
uint16_t* util_buffer_u16::buffer
```

Definition at line 101 of file util.h.

Referenced by util_buffer_u16_add(), util_buffer_u16_get(), and util_buffer_u16_init().

3.26.2.3 c_ix

```
uint16_t util_buffer_u16::c_ix
```

Definition at line 98 of file util.h.

Referenced by util_buffer_u16_add(), util_buffer_u16_init(), and util_buffer_u16_isempty().

3.26.2.4 l_ix

```
uint16_t util_buffer_u16::l_ix
```

Definition at line 99 of file util.h.

Referenced by util_buffer_u16_get(), util_buffer_u16_init(), and util_buffer_u16_isempty().

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

- [util.h](#)

3.27 util_buffer_u8 Struct Reference

```
#include <util.h>
```

Data Fields

- uint16_t [c_ix](#)
- uint16_t [l_ix](#)
- uint16_t [bfr_len](#)
- uint8_t* [buffer](#)

3.27.1 Detailed Description

Definition at line 90 of file util.h.

3.27.2 Field Documentation

3.27.2.1 bfr_len

```
uint16_t util_buffer_u8::bfr_len
```

Definition at line 93 of file util.h.

Referenced by util_buffer_u8_access(), util_buffer_u8_add(), util_buffer_u8_get(), and util_buffer_u8_init().

3.27.2.2 buffer

```
uint8_t* util_buffer_u8::buffer
```

Definition at line 94 of file util.h.

Referenced by util_buffer_u8_access(), util_buffer_u8_add(), util_buffer_u8_get(), and util_buffer_u8_init().

3.27.2.3 c_ix

```
uint16_t util_buffer_u8::c_ix
```

Definition at line 91 of file util.h.

Referenced by util_buffer_u8_access(), util_buffer_u8_add(), util_buffer_u8_init(), and util_buffer_u8_isempty().

3.27.2.4 l_ix

```
uint16_t util_buffer_u8::l_ix
```

Definition at line 92 of file util.h.

Referenced by util_buffer_u8_get(), util_buffer_u8_init(), and util_buffer_u8_isempty().

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

- [util.h](#)

Chapter 4

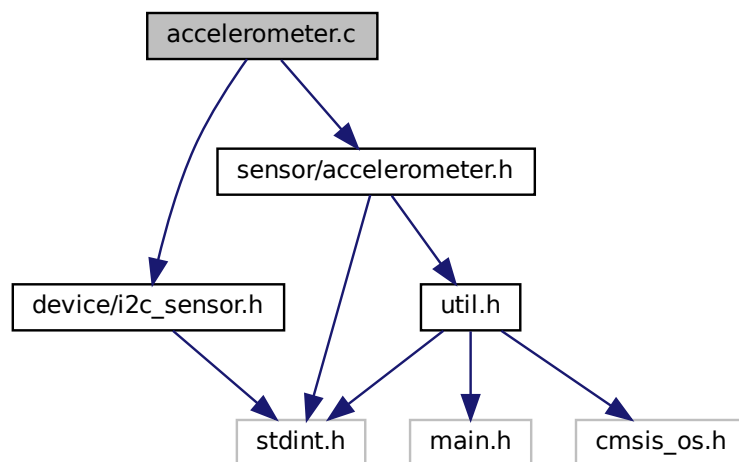
File Documentation

4.1 accelerometer.c File Reference

```
#include <sensor/accelerometer.h>
```

```
#include <device/i2c_sensor.h>
```

Include dependency graph for accelerometer.c:



Functions

- [util_error_t accelerometer_init](#) (void)
Initialize accelerometers.

4.1.1 Function Documentation

4.1.1.1 accelerometer_init()

```
util_error_t accelerometer_init (
    void )
```

Initialize accelerometers.

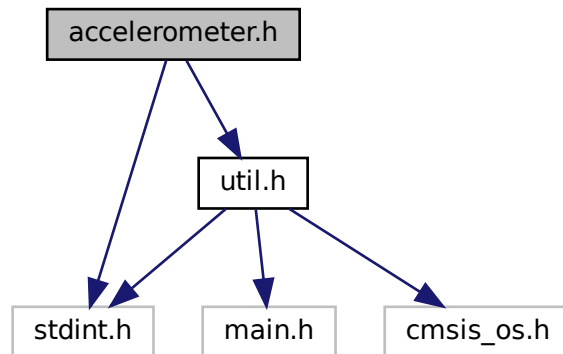
Definition at line 48 of file accelerometer.c.

4.2 accelerometer.h File Reference

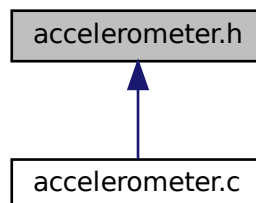
```
#include <stdint.h>
```

```
#include <util.h>
```

Include dependency graph for accelerometer.h:



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



Functions

- `util_error_t accelerometer_init (void)`
Initialize accelerometers.

4.2.1 Function Documentation

4.2.1.1 accelerometer_init()

```
util_error_t accelerometer_init (  
    void )
```

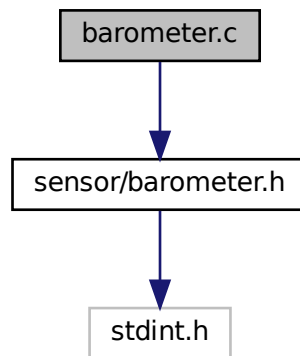
Initialize accelerometers.

Definition at line 48 of file accelerometer.c.

4.3 barometer.c File Reference

```
#include <sensor/barometer.h>
```

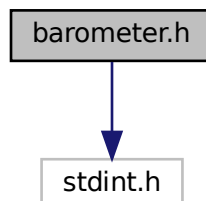
Include dependency graph for barometer.c:



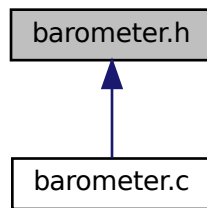
4.4 barometer.h File Reference

```
#include <stdint.h>
```

Include dependency graph for barometer.h:



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



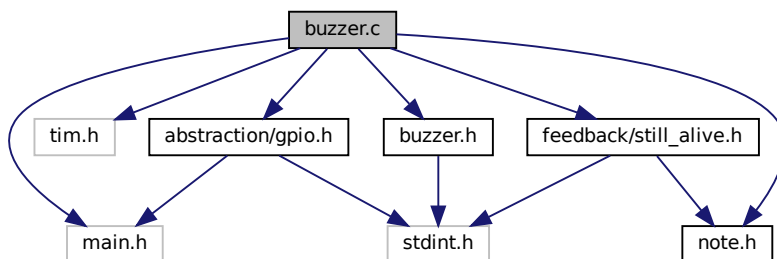
4.5 buzzer.c File Reference

```

#include <main.h>
#include <tim.h>
#include "buzzer.h"
#include <abstraction/gpio.h>
#include <feedback/still_alive.h>
#include "note.h"

```

Include dependency graph for buzzer.c:



Macros

- `#define NOTE_TIMER_DEV` htim14
- `#define RYTM_TIMER_DEV` htim16
- `#define NOTE_TIMER` NOTE_TIMER_DEV.Instance
- `#define RYTM_TIMER` RYTM_TIMER_DEV.Instance
- `#define BUZZER_PIN` GPIO_PIN_3
- `#define BUZZER_PORT` GPIOC
- `#define TIMER_FREQ` 200e6
- `#define NOTE_PRESC` 10
- `#define RYTM_PRESC` 20000
- `#define TIMER_TRIM` -1e6
- `#define COMPUTE_NOTE(note)` (((((TIMER_FREQ)+(TIMER_TRIM))*10)/(NOTE_PRESC))/(note))/2
- `#define RYTM_MS(ms)` (ms)*(((TIMER_FREQ)+(TIMER_TRIM))/(RYTM_PRESC))/(1000)
- `#define COMPUTE_RYTM(time)` RYTM_MS((time)*100)

Functions

- void `buzzer_note_interrupt` (void)
- void `buzzer_rytm_interrupt` (void)
- void `buzzer_enable` (void)
- void `buzzer_disable` (void)
- void `buzzer_init` (void)

Variables

- static uint16_t `melody_state` = 0
- static uint8_t `state` = 0
- static uint8_t `melody_active` = 1

4.5.1 Macro Definition Documentation

4.5.1.1 BUZZER_PIN

```
#define BUZZER_PIN GPIO_PIN_3
```

Definition at line 30 of file buzzer.c.

4.5.1.2 BUZZER_PORT

```
#define BUZZER_PORT GPIOC
```

Definition at line 31 of file buzzer.c.

4.5.1.3 COMPUTE_NOTE

```
#define COMPUTE_NOTE(  
    note ) ((( (TIMER_FREQ) + (TIMER_TRIM) ) * 10) / (NOTE_PRESC) / (note)) / 2
```

Definition at line 44 of file buzzer.c.

4.5.1.4 COMPUTE_RYTM

```
#define COMPUTE_RYTM(  
    time ) RYTM_MS ((time) * 100)
```

Definition at line 46 of file buzzer.c.

4.5.1.5 NOTE_PRESC

```
#define NOTE_PRESC 10
```

Definition at line 39 of file buzzer.c.

4.5.1.6 NOTE_TIMER

```
#define NOTE_TIMER NOTE_TIMER_DEV.Instance
```

Definition at line 27 of file buzzer.c.

4.5.1.7 NOTE_TIMER_DEV

```
#define NOTE_TIMER_DEV htim14
```

Definition at line 24 of file buzzer.c.

4.5.1.8 RYTM_MS

```
#define RYTM_MS(  
    ms ) (ms) * ( ( (TIMER_FREQ) + (TIMER_TRIM) ) / (RYTM_PRESC) ) / (1000)
```

Definition at line 45 of file buzzer.c.

4.5.1.9 RYTM_PRESC

```
#define RYTM_PRESC 20000
```

Definition at line 40 of file buzzer.c.

4.5.1.10 RYTM_TIMER

```
#define RYTM_TIMER RYTM_TIMER_DEV.Instance
```

Definition at line 28 of file buzzer.c.

4.5.1.11 RYTM_TIMER_DEV

```
#define RYTM_TIMER_DEV htim16
```

Definition at line 25 of file buzzer.c.

4.5.1.12 TIMER_FREQ

```
#define TIMER_FREQ 200e6
```

Definition at line 38 of file buzzer.c.

4.5.1.13 TIMER_TRIM

```
#define TIMER_TRIM -1e6
```

Definition at line 41 of file buzzer.c.

4.5.2 Function Documentation

4.5.2.1 buzzer_disable()

```
void buzzer_disable (  
    void )
```

Definition at line 103 of file buzzer.c.

References NOTE_TIMER_DEV, and RYTM_TIMER_DEV.

4.5.2.2 buzzer_enable()

```
void buzzer_enable (  
    void )
```

Definition at line 98 of file buzzer.c.

References NOTE_TIMER_DEV, and RYTM_TIMER_DEV.

4.5.2.3 buzzer_init()

```
void buzzer_init (  
    void )
```

Definition at line 108 of file buzzer.c.

References A4, BUZZER_PIN, BUZZER_PORT, COMPUTE_NOTE, COMPUTE_RYTM, melody_active, NOTE_TIMER, and RYTM_TIMER.

Referenced by threads_init().

Here is the caller graph for this function:



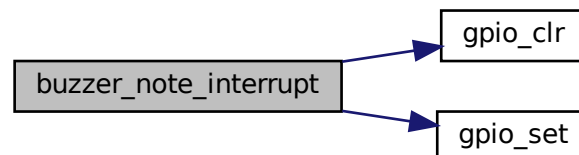
4.5.2.4 buzzer_note_interrupt()

```
void buzzer_note_interrupt (  
    void )
```

Definition at line 70 of file buzzer.c.

References BUZZER_PIN, BUZZER_PORT, gpio_clr(), gpio_set(), melody_active, and state.

Here is the call graph for this function:



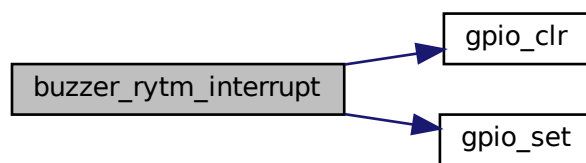
4.5.2.5 buzzer_rytm_interrupt()

```
void buzzer_rytm_interrupt (  
    void )
```

Definition at line 82 of file buzzer.c.

References `COMPUTE_NOTE`, `gpio_clr()`, `gpio_set()`, `melody_active`, `melody_state`, `NOTE_TIMER`, `still_alive`, and `still_alive_len`.

Here is the call graph for this function:



4.5.3 Variable Documentation

4.5.3.1 melody_active

```
uint8_t melody_active = 1 [static]
```

Definition at line 59 of file buzzer.c.

Referenced by `buzzer_init()`, `buzzer_note_interrupt()`, and `buzzer_rytm_interrupt()`.

4.5.3.2 melody_state

```
uint16_t melody_state = 0 [static]
```

Definition at line 57 of file buzzer.c.

Referenced by `buzzer_rytm_interrupt()`.

4.5.3.3 state

```
uint8_t state = 0 [static]
```

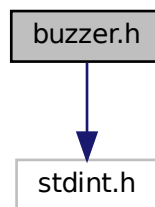
Definition at line 58 of file buzzer.c.

Referenced by buzzer_note_interrupt().

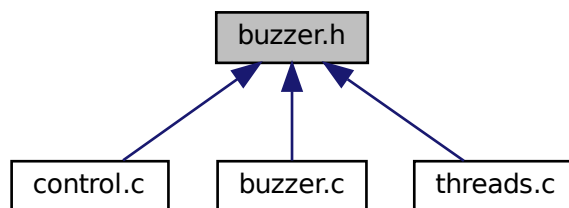
4.6 buzzer.h File Reference

```
#include <stdint.h>
```

Include dependency graph for buzzer.h:



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



Functions

- void [buzzer_note_interrupt](#) (void)
- void [buzzer_rytm_interrupt](#) (void)
- void [buzzer_enable](#) (void)
- void [buzzer_disable](#) (void)
- void [buzzer_init](#) (void)

4.6.1 Function Documentation

4.6.1.1 buzzer_disable()

```
void buzzer_disable (
    void )
```

Definition at line 103 of file buzzer.c.

References NOTE_TIMER_DEV, and RYTM_TIMER_DEV.

4.6.1.2 buzzer_enable()

```
void buzzer_enable (
    void )
```

Definition at line 98 of file buzzer.c.

References NOTE_TIMER_DEV, and RYTM_TIMER_DEV.

4.6.1.3 buzzer_init()

```
void buzzer_init (
    void )
```

Definition at line 108 of file buzzer.c.

References A4, BUZZER_PIN, BUZZER_PORT, COMPUTE_NOTE, COMPUTE_RYTM, melody_active, NOTE_TIMER, and RYTM_TIMER.

Referenced by threads_init().

Here is the caller graph for this function:



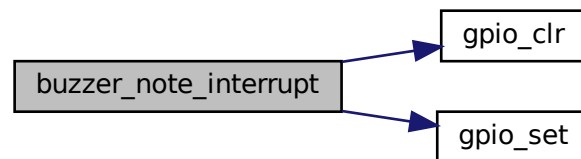
4.6.1.4 buzzer_note_interrupt()

```
void buzzer_note_interrupt (  
    void )
```

Definition at line 70 of file buzzer.c.

References BUZZER_PIN, BUZZER_PORT, gpio_clr(), gpio_set(), melody_active, and state.

Here is the call graph for this function:



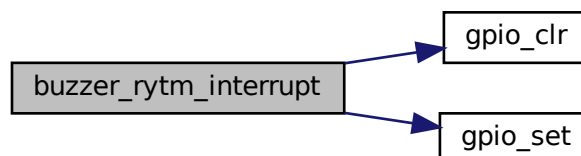
4.6.1.5 buzzer_rytm_interrupt()

```
void buzzer_rytm_interrupt (  
    void )
```

Definition at line 82 of file buzzer.c.

References COMPUTE_NOTE, gpio_clr(), gpio_set(), melody_active, melody_state, NOTE_TIMER, still_alive, and still_alive_len.

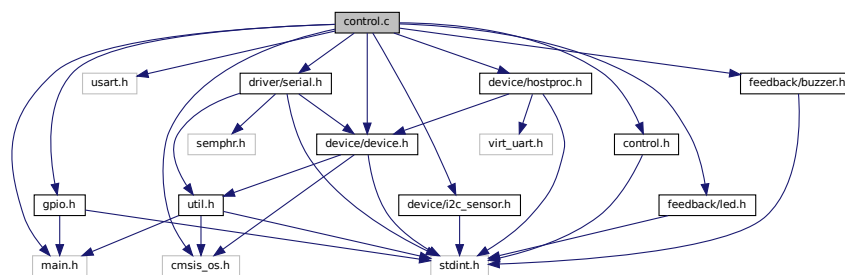
Here is the call graph for this function:



4.7 control.c File Reference

```
#include <main.h>
#include <gpio.h>
#include <usart.h>
#include <cmsis_os.h>
#include <driver/serial.h>
#include <device/device.h>
#include <device/i2c_sensor.h>
#include <device/hostproc.h>
#include <control.h>
#include <feedback/led.h>
#include <feedback/buzzer.h>
```

Include dependency graph for control.c:



Data Structures

- struct [control](#)

Macros

- `#define` [CONTROL_HEART_BEAT](#) 200

Typedefs

- typedef enum [control_state](#) [control_state_t](#)
State of the control FSM.
- typedef struct [control](#) [control_t](#)

Enumerations

- enum [control_state](#) {
[CONTROL_IDLE](#) , [CONTROL_CALIBRATION](#) , [CONTROL_ARMED](#) , [CONTROL_POWERED](#) ,
[CONTROL_SUPERSONIC](#) , [CONTROL_COAST](#) , [CONTROL_APOGEE](#) , [CONTROL_DROGUE](#) ,
[CONTROL_EVENT](#) , [CONTROL_MAIN](#) , [CONTROL_TOUCHDOWN](#) , [CONTROL BALLISTIC](#) ,
[CONTROL_ERROR](#) , [CONTROL_ABORT](#) }
State of the control FSM.

Functions

- void [control_idle_start](#) (void)
Idle state entry.
- void [control_idle_run](#) (void)
Idle state runtime.
- void [control_calibration_start](#) (void)
Calibration state entry.
- void [control_calibration_run](#) (void)
Calibration state runtime.
- void [control_armed_start](#) (void)
Armed state entry.
- void [control_armed_run](#) (void)
Armed state runtime.
- void [control_powered_start](#) (void)
Powered state entry.
- void [control_powered_run](#) (void)
Powered state runtime.
- void [control_supersonic_start](#) (void)
Supersonic state entry.
- void [control_supersonic_run](#) (void)
Supersonic state runtime.
- void [control_coast_start](#) (void)
Coast state entry.
- void [control_coast_run](#) (void)
Coast state runtime.
- void [control_apogee_start](#) (void)
Apogee state entry.
- void [control_apogee_run](#) (void)
Apogee state runtime.
- void [control_drogue_start](#) (void)
Drogue state entry.
- void [control_drogue_run](#) (void)
Drogue state runtime.
- void [control_event_start](#) (void)
Event state entry.
- void [control_event_run](#) (void)
Event state runtime.
- void [control_main_start](#) (void)
Main state entry.
- void [control_main_run](#) (void)
Main state runtime.
- void [control_touchdown_start](#) (void)
Touchdown state entry.
- void [control_touchdown_run](#) (void)
Touchdown state runtime.
- void [control_ballistic_start](#) (void)
Ballistic state entry.
- void [control_ballistic_run](#) (void)
Ballistic state runtime.
- void [control_error_start](#) (void)

- Error state entry.*
 - void [control_error_run](#) (void)
 - Error state runtime.*
 - void [control_abort_start](#) (void)
 - Abort state entry.*
 - void [control_abort_run](#) (void)
 - Abort state runtime.*
 - void [control_thread](#) ([__attribute__\(\(unused\)\)](#) void *arg)
- Control thread entry point.*

Variables

- [control_t](#) [control](#)

4.7.1 Macro Definition Documentation

4.7.1.1 CONTROL_HEART_BEAT

```
#define CONTROL_HEART_BEAT 200
```

Definition at line 33 of file control.c.

4.7.2 Typedef Documentation

4.7.2.1 control_state_t

```
typedef enum control\_state control\_state\_t
```

State of the control FSM.

4.7.2.2 control_t

```
typedef struct control control\_t
```

4.7.3 Enumeration Type Documentation

4.7.3.1 control_state

```
enum control\_state
```

State of the control FSM.

Enumerator

CONTROL_IDLE	Wait for arming or calibration
CONTROL_CALIBRATION	Calibrate sensors and actuators
CONTROL_ARMED	Armed, wait for liftoff
CONTROL_POWERED	Powered ascent
CONTROL_SUPERSONIC	Supersonic flight
CONTROL_COAST	Subsonic, coast flight
CONTROL_APOGEE	Apogee reached, trigger first event
CONTROL_DROGUE	Drogue chute descent, wait for second event
CONTROL_EVENT	Low alt reached, trigger second event
CONTROL_MAIN	Main chute descent, wait for touchdown
CONTROL_TOUCHDOWN	Touchdown detected, end of the flight
CONTROL_BALLISTIC	Ballistic flight detected
CONTROL_ERROR	Auto triggered error
CONTROL_ABORT	User triggered error

Definition at line 48 of file control.c.

4.7.4 Function Documentation

4.7.4.1 control_abort_run()

```
void control_abort_run (  
    void )
```

Abort state runtime.

Definition at line 407 of file control.c.

4.7.4.2 control_abort_start()

```
void control_abort_start (  
    void )
```

Abort state entry.

Definition at line 398 of file control.c.

References CONTROL_ABORT, and control::state.

4.7.4.3 control_apogee_run()

```
void control_apogee_run (  
    void )
```

Apogee state runtime.

Definition at line 288 of file control.c.

4.7.4.4 control_apogee_start()

```
void control_apogee_start (  
    void )
```

Apogee state entry.

Definition at line 279 of file control.c.

References CONTROL_APOGEE, and control::state.

4.7.4.5 control_armed_run()

```
void control_armed_run (  
    void )
```

Armed state runtime.

Definition at line 220 of file control.c.

4.7.4.6 control_armed_start()

```
void control_armed_start (  
    void )
```

Armed state entry.

Definition at line 211 of file control.c.

References CONTROL_ARMED, and control::state.

4.7.4.7 control_ballistic_run()

```
void control_ballistic_run (  
    void )
```

Ballistic state runtime.

Definition at line 373 of file control.c.

4.7.4.8 control_ballistic_start()

```
void control_ballistic_start (  
    void )
```

Ballistic state entry.

Definition at line 364 of file control.c.

References CONTROL_BALLISTIC, and control::state.

4.7.4.9 control_calibration_run()

```
void control_calibration_run (  
    void )
```

Calibration state runtime.

Definition at line 203 of file control.c.

4.7.4.10 control_calibration_start()

```
void control_calibration_start (  
    void )
```

Calibration state entry.

Definition at line 195 of file control.c.

References CONTROL_CALIBRATION, and control::state.

4.7.4.11 control_coast_run()

```
void control_coast_run (  
    void )
```

Coast state runtime.

Definition at line 271 of file control.c.

4.7.4.12 control_coast_start()

```
void control_coast_start (  
    void )
```

Coast state entry.

Definition at line 262 of file control.c.

References CONTROL_COAST, and control::state.

4.7.4.13 control_drogue_run()

```
void control_drogue_run (  
    void )
```

Drogue state runtime.

Definition at line 305 of file control.c.

4.7.4.14 control_drogue_start()

```
void control_drogue_start (  
    void )
```

Drogue state entry.

Definition at line 296 of file control.c.

References CONTROL_DROGUE, and control::state.

4.7.4.15 control_error_run()

```
void control_error_run (  
    void )
```

Error state runtime.

Definition at line 390 of file control.c.

4.7.4.16 control_error_start()

```
void control_error_start (  
    void )
```

Error state entry.

Definition at line 381 of file control.c.

References CONTROL_ERROR, and control::state.

4.7.4.17 control_event_run()

```
void control_event_run (  
    void )
```

Event state runtime.

Definition at line 322 of file control.c.

4.7.4.18 control_event_start()

```
void control_event_start (  
    void )
```

Event state entry.

Definition at line 313 of file control.c.

References CONTROL_EVENT, and control::state.

4.7.4.19 control_idle_run()

```
void control_idle_run (  
    void )
```

Idle state runtime.

Definition at line 187 of file control.c.

4.7.4.20 control_idle_start()

```
void control_idle_start (  
    void )
```

Idle state entry.

Definition at line 178 of file control.c.

References CONTROL_IDLE, and control::state.

4.7.4.21 control_main_run()

```
void control_main_run (  
    void )
```

Main state runtime.

Definition at line 339 of file control.c.

4.7.4.22 control_main_start()

```
void control_main_start (  
    void )
```

Main state entry.

Definition at line 330 of file control.c.

References CONTROL_MAIN, and control::state.

4.7.4.23 control_powered_run()

```
void control_powered_run (
    void )
```

Powered state runtime.

Definition at line 237 of file control.c.

4.7.4.24 control_powered_start()

```
void control_powered_start (
    void )
```

Powered state entry.

Definition at line 228 of file control.c.

References CONTROL_POWERED, and control::state.

4.7.4.25 control_supersonic_run()

```
void control_supersonic_run (
    void )
```

Supersonic state runtime.

Definition at line 254 of file control.c.

4.7.4.26 control_supersonic_start()

```
void control_supersonic_start (
    void )
```

Supersonic state entry.

Definition at line 245 of file control.c.

References CONTROL_SUPERSONIC, and control::state.

4.7.4.27 control_thread()

```
void control_thread (
    __attribute__((unused)) void * arg )
```

Control thread entry point.

This thread holds the main state machine of the WildhornAV software. It will be the main decision point for actions to be taken with respect to real world events.

Parameters

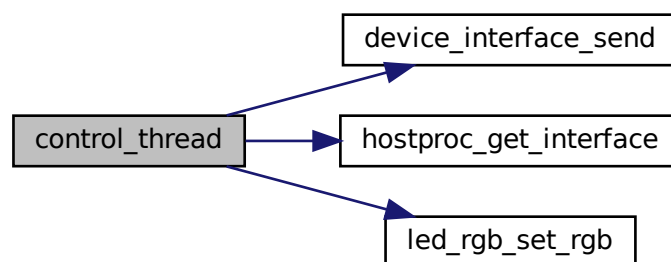
<i>arg</i>	freertos thread entry point context (unused)
------------	--

Definition at line 153 of file control.c.

References `CONTROL_HEART_BEAT`, `device_interface_send()`, `hostproc_get_interface()`, `hostproc_interface`, and `led_rgb_set_rgb()`.

Referenced by `threads_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.4.28 control_touchdown_run()

```
void control_touchdown_run (  
    void )
```

Touchdown state runtime.

Definition at line 356 of file control.c.

4.7.4.29 control_touchdown_start()

```
void control_touchdown_start (  
    void )
```

Touchdown state entry.

Definition at line 347 of file control.c.

References CONTROL_TOUCHDOWN, and control::state.

4.7.5 Variable Documentation

4.7.5.1 control

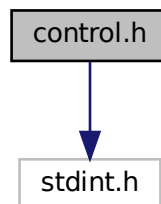
```
control_t control
```

Definition at line 89 of file control.c.

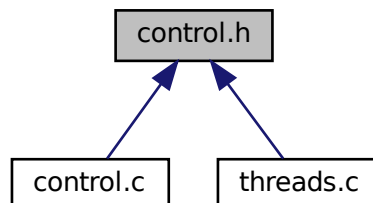
4.8 control.h File Reference

```
#include <stdint.h>
```

Include dependency graph for control.h:



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



Functions

- void [control_thread](#) (void *arg)

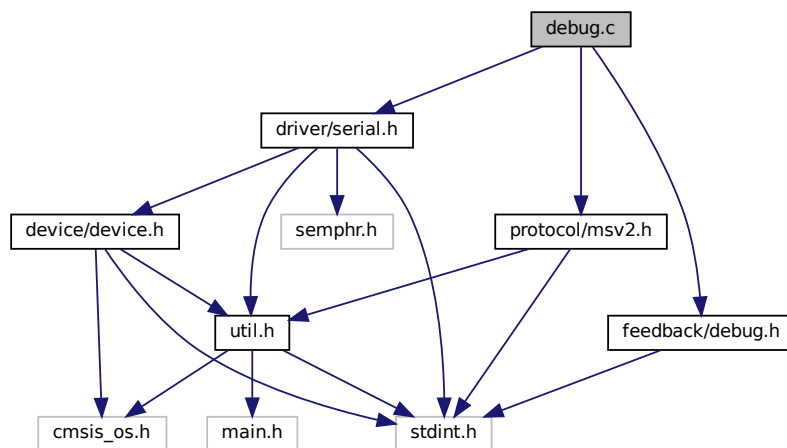
4.8.1 Function Documentation

4.8.1.1 control_thread()

```
void control_thread (
    void * arg )
```

4.9 debug.c File Reference

```
#include <protocol/msv2.h>
#include <feedback/debug.h>
#include <driver/serial.h>
Include dependency graph for debug.c:
```



Data Structures

- struct [debug_interface_context](#)
- struct [debug_context](#)

Typedefs

- typedef struct [debug_interface_context](#) [debug_interface_context_t](#)
- typedef struct [debug_context](#) [debug_context_t](#)

Functions

- [util_error_t debug_init](#) (void)

Variables

- [debug_context_t debug_context](#)
- [serial_interface_context_t feedback_interface_context](#)
- [debug_interface_context_t debug_interface_context](#)

4.9.1 Typedef Documentation

4.9.1.1 debug_context_t

```
typedef struct debug\_context debug\_context\_t
```

4.9.1.2 debug_interface_context_t

```
typedef struct debug\_interface\_context debug\_interface\_context\_t
```

4.9.2 Function Documentation

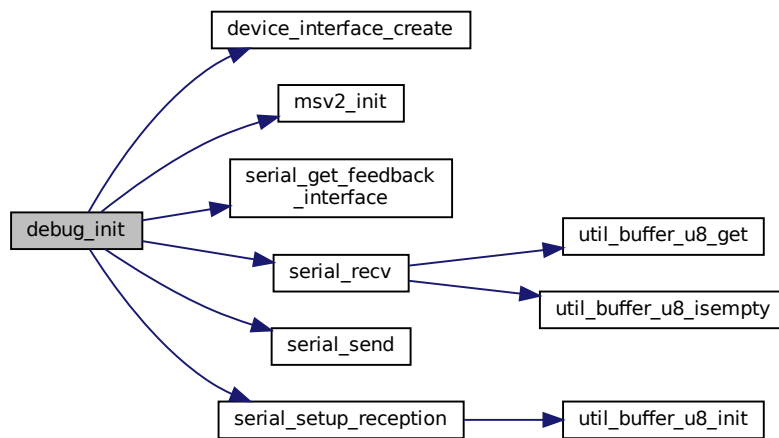
4.9.2.1 debug_init()

```
util\_error\_t debug\_init (  
    void )
```

Definition at line 69 of file `debug.c`.

References `device_interface_create()`, `feedback_interface`, `feedback_interface_context`, `debug_interface_context`, `msv2`, `msv2_init()`, `serial_daemon`, `serial_get_feedback_interface()`, `serial_rcv()`, `serial_send()`, `serial_setup_reception()`, and `SERIAL_TRANSFER_IT`.

Here is the call graph for this function:



4.9.3 Variable Documentation

4.9.3.1 debug_context

`debug_context_t debug_context`

Definition at line 50 of file `debug.c`.

4.9.3.2 debug_interface_context

`debug_interface_context_t debug_interface_context`

Definition at line 52 of file `debug.c`.

4.9.3.3 feedback_interface_context

`serial_interface_context_t feedback_interface_context`

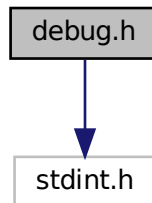
Definition at line 51 of file `debug.c`.

Referenced by `debug_init()`.

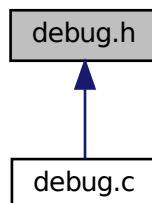
4.10 debug.h File Reference

```
#include <stdint.h>
```

Include dependency graph for debug.h:



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



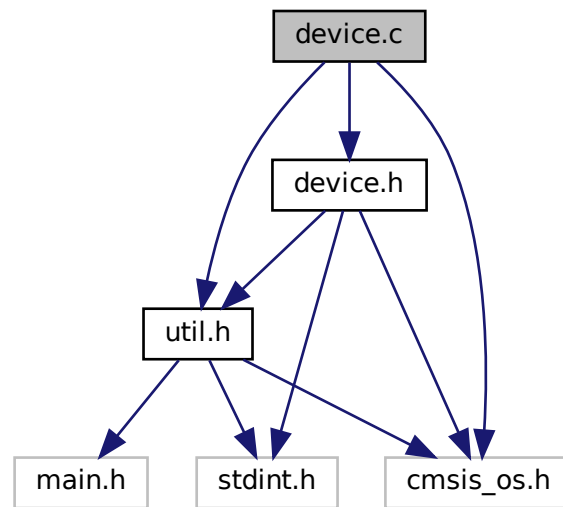
4.11 device.c File Reference

```
#include <cmsis_os.h>
```

```
#include <util.h>
```

```
#include "device.h"
```


Include dependency graph for device.c:



Macros

- `#define LEN_32 4`
- `#define LEN_16 2`
- `#define LEN_8 1`

Functions

- `void device_daemon_thread (void *arg)`
- `util_error_t device_create (device_t *dev, void *context, device_interface_t *interface, util_error_t(*read_reg)(void *, device_interface_t *, uint32_t, uint8_t *, uint32_t), util_error_t(*write_reg)(void *, device_interface_t *, uint32_t, uint8_t *, uint32_t))`
Initialize a device instance.
- `util_error_t device_interface_create (device_interface_t *interface, void *context, device_daemon_t *daemon, util_error_t(*send)(void *, uint8_t *, uint32_t), util_error_t(*recv)(void *, uint8_t *, uint32_t *), util_error_t(*handle_data)(void *, void *))`
- `util_error_t device_daemon_create (device_daemon_t *daemon, const char *name, uint32_t prio, void *inst, util_error_t(*data_rdy)(void *))`
- `util_error_t device_interface_send (device_interface_t *interface, uint8_t *data, uint32_t len)`
- `util_error_t device_interface_recv (device_interface_t *interface, uint8_t *data, uint32_t *len)`
- `util_error_t device_write_i32 (device_t *dev, uint32_t addr, int32_t data)`
- `util_error_t device_write_u32 (device_t *dev, uint32_t addr, uint32_t data)`
- `util_error_t device_write_i16 (device_t *dev, uint32_t addr, int16_t data)`
- `util_error_t device_write_u16 (device_t *dev, uint32_t addr, uint16_t data)`
- `util_error_t device_write_i8 (device_t *dev, uint32_t addr, int8_t data)`
- `util_error_t device_write_u8 (device_t *dev, uint32_t addr, uint8_t data)`
- `util_error_t device_read_i32 (device_t *dev, uint32_t addr, int32_t *data)`
- `util_error_t device_read_u32 (device_t *dev, uint32_t addr, uint32_t *data)`
- `util_error_t device_read_i16 (device_t *dev, uint32_t addr, int16_t *data)`
- `util_error_t device_read_u16 (device_t *dev, uint32_t addr, uint16_t *data)`
- `util_error_t device_read_i8 (device_t *dev, uint32_t addr, int8_t *data)`
- `util_error_t device_read_u8 (device_t *dev, uint32_t addr, uint8_t *data)`

4.11.1 Macro Definition Documentation

4.11.1.1 LEN_16

```
#define LEN_16 2
```

Definition at line 24 of file device.c.

4.11.1.2 LEN_32

```
#define LEN_32 4
```

Definition at line 23 of file device.c.

4.11.1.3 LEN_8

```
#define LEN_8 1
```

Definition at line 25 of file device.c.

4.11.2 Function Documentation

4.11.2.1 device_create()

```
util_error_t device_create (
    device_t * dev,
    void * context,
    device_interface_t * interface,
    util_error_t(*) (void *, device_interface_t *, uint32_t, uint8_t *, uint32_t) read←
_reg,
    util_error_t(*) (void *, device_interface_t *, uint32_t, uint8_t *, uint32_t) write←
_reg )
```

Initialize a device instance.

Parameters

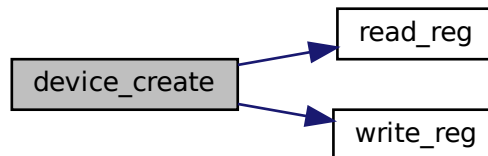
<i>dev</i>	Pointer to the <code>device_t</code> structure describing this device.
<i>context</i>	Generic pointer to a device context.
<i>interface</i>	Pointer to the <code>device_interface_t</code> associated with this device.
<i>read_reg</i>	Pointer to a read register function for this device.
<i>write_reg</i>	Pointer to a write register function for this device.

Definition at line 66 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::id`, `device::interface`, `read_reg()`, `device::read_reg`, `write_reg()`, and `device::write_reg`.

Referenced by `i2c_sensor_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.11.2.2 device_daemon_create()

```

util_error_t device_daemon_create (
    device_daemon_t * daemon,
    const char * name,
    uint32_t prio,
    void * inst,
    util_error_t(*) (void *) data_rdy )
  
```

Definition at line 103 of file device.c.

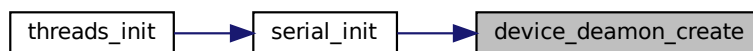
References `device_daemon::buffer`, `device_daemon::context`, `device_daemon::data_rdy`, `DEAMON_STACK_SIZE`, `device_daemon_thread()`, `ER_RESSOURCE_ERROR`, `ER_SUCCESS`, `device_daemon::handle`, `device_daemon::id`, `device_daemon::interfaces_count`, and `device_daemon::stack`.

Referenced by `serial_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.11.2.3 device_deamon_thread()

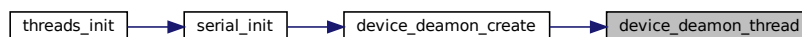
```
void device_deamon_thread (
    void * arg )
```

Definition at line 122 of file device.c.

References `device_interface::context`, `device_deamon::context`, `device_deamon::data_rdy`, `ER_SUCCESS`, `device_interface::handle_data`, and `device_deamon::interfaces_count`.

Referenced by `device_deamon_create()`.

Here is the caller graph for this function:



4.11.2.4 device_interface_create()

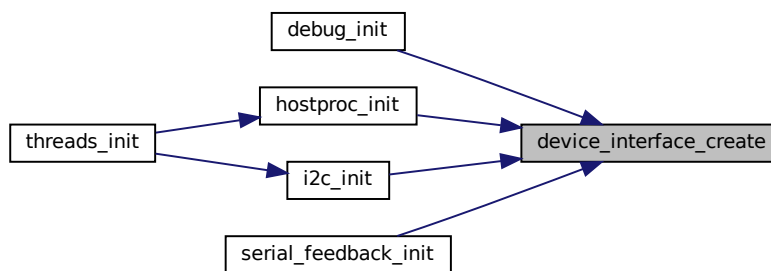
```
util_error_t device_interface_create (
    device_interface_t * interface,
    void * context,
    device_deamon_t * deamon,
    util_error_t(*) (void *, uint8_t *, uint32_t) send,
    util_error_t(*) (void *, uint8_t *, uint32_t *) recv,
    util_error_t(*) (void *, void *) handle_data )
```

Definition at line 83 of file device.c.

References `device_interface::context`, `ER_SUCCESS`, `device_interface::handle_data`, `device_interface::id`, `device_deamon::interfaces`, `device_deamon::interfaces_count`, `device_interface::recv`, and `device_interface::send`.

Referenced by `debug_init()`, `hostproc_init()`, `i2c_init()`, and `serial_feedback_init()`.

Here is the caller graph for this function:



4.11.2.5 device_interface_recv()

```
util_error_t device_interface_recv (
    device_interface_t * interface,
    uint8_t * data,
    uint32_t * len )
```

Definition at line 150 of file device.c.

References `device_interface::context`, `ER_RESSOURCE_ERROR`, `device_interface::recv`, and `device_interface::send`.

4.11.2.6 device_interface_send()

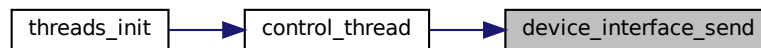
```
util_error_t device_interface_send (  
    device_interface_t * interface,  
    uint8_t * data,  
    uint32_t len )
```

Definition at line 141 of file device.c.

References device_interface::context, ER_RESSOURCE_ERROR, and device_interface::send.

Referenced by control_thread().

Here is the caller graph for this function:



4.11.2.7 device_read_i16()

```
util_error_t device_read_i16 (  
    device_t * dev,  
    uint32_t addr,  
    int16_t * data )
```

Definition at line 224 of file device.c.

References device::context, ER_SUCCESS, device::interface, LEN_16, device::read_reg, and util_decode_i16().

Here is the call graph for this function:



4.11.2.8 device_read_i32()

```
util_error_t device_read_i32 (  
    device_t * dev,  
    uint32_t addr,  
    int32_t * data )
```

Definition at line 209 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::interface`, `LEN_32`, `device::read_reg`, and `util_decode_i32()`.

Here is the call graph for this function:



4.11.2.9 device_read_i8()

```
util_error_t device_read_i8 (  
    device_t * dev,  
    uint32_t addr,  
    int8_t * data )
```

Definition at line 240 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::interface`, `LEN_8`, `device::read_reg`, and `util_decode_i8()`.

Here is the call graph for this function:



4.11.2.10 device_read_u16()

```
util_error_t device_read_u16 (  
    device_t * dev,  
    uint32_t addr,  
    uint16_t * data )
```

Definition at line 232 of file device.c.

References device::context, ER_SUCCESS, device::interface, LEN_16, device::read_reg, and util_decode_u16().

Here is the call graph for this function:



4.11.2.11 device_read_u32()

```
util_error_t device_read_u32 (  
    device_t * dev,  
    uint32_t addr,  
    uint32_t * data )
```

Definition at line 216 of file device.c.

References device::context, ER_SUCCESS, device::interface, LEN_32, device::read_reg, and util_decode_u32().

Here is the call graph for this function:



4.11.2.12 device_read_u8()

```
util_error_t device_read_u8 (  
    device_t * dev,  
    uint32_t addr,  
    uint8_t * data )
```

Definition at line 248 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::interface`, `LEN_8`, `device::read_reg`, and `util_decode_u8()`.

Here is the call graph for this function:



4.11.2.13 device_write_i16()

```
util_error_t device_write_i16 (  
    device_t * dev,  
    uint32_t addr,  
    int16_t data )
```

Definition at line 178 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::interface`, `LEN_16`, `util_encode_i16()`, and `device::write_reg`.

Here is the call graph for this function:



4.11.2.14 device_write_i32()

```
util_error_t device_write_i32 (  
    device_t * dev,  
    uint32_t addr,  
    int32_t data )
```

Definition at line 162 of file device.c.

References device::context, ER_SUCCESS, device::interface, LEN_32, util_encode_i32(), and device::write_reg.

Here is the call graph for this function:



4.11.2.15 device_write_i8()

```
util_error_t device_write_i8 (  
    device_t * dev,  
    uint32_t addr,  
    int8_t data )
```

Definition at line 194 of file device.c.

References device::context, ER_SUCCESS, device::interface, LEN_8, util_encode_i8(), and device::write_reg.

Here is the call graph for this function:



4.11.2.16 device_write_u16()

```
util_error_t device_write_u16 (  
    device_t * dev,  
    uint32_t addr,  
    uint16_t data )
```

Definition at line 186 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::interface`, `LEN_16`, `util_encode_u16()`, and `device::write_reg`.

Here is the call graph for this function:



4.11.2.17 device_write_u32()

```
util_error_t device_write_u32 (  
    device_t * dev,  
    uint32_t addr,  
    uint32_t data )
```

Definition at line 170 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::interface`, `LEN_32`, `util_encode_u32()`, and `device::write_reg`.

Here is the call graph for this function:



4.11.2.18 device_write_u8()

```
util_error_t device_write_u8 (  
    device_t * dev,  
    uint32_t addr,  
    uint8_t data )
```

Definition at line 201 of file device.c.

References device::context, ER_SUCCESS, device::interface, LEN_8, util_encode_u8(), and device::write_reg.

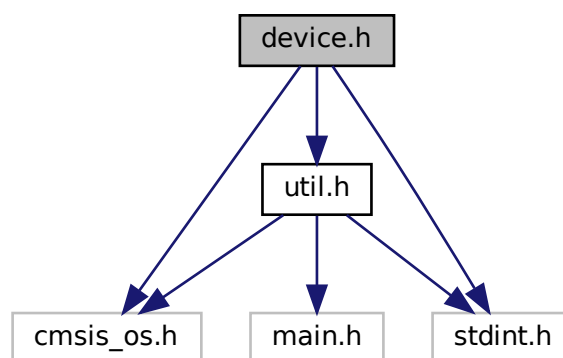
Here is the call graph for this function:



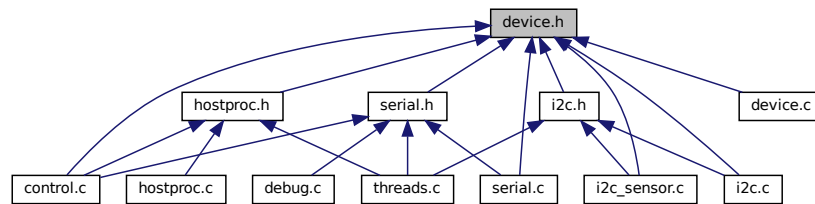
4.12 device.h File Reference

```
#include <cmsis_os.h>  
#include <stdint.h>  
#include <util.h>
```

Include dependency graph for device.h:



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



Data Structures

- struct [device_interface](#)
- struct [device_daemon](#)
- struct [device](#)

Macros

- #define [DEVICE_NAME_LEN](#) 16
- #define [DEVICE_MAX_INTERFACES_PER_DEAMON](#) 16
- #define [DEAMON_STACK_SIZE](#) 1024

Typedefs

- typedef struct [device_interface](#) [device_interface_t](#)
- typedef struct [device_daemon](#) [device_daemon_t](#)
- typedef struct [device](#) [device_t](#)

Functions

- [util_error_t device_create](#) ([device_t](#) *dev, void *context, [device_interface_t](#) *interface, [util_error_t\(*read_reg\)](#)(void *, [device_interface_t](#) *, [uint32_t](#), [uint8_t](#) *, [uint32_t](#)), [util_error_t\(*write_reg\)](#)(void *, [device_interface_t](#) *, [uint32_t](#), [uint8_t](#) *, [uint32_t](#)))
Initialize a device instance.
- [util_error_t device_daemon_create](#) ([device_daemon_t](#) *daemon, const char *name, [uint32_t](#) prio, void *inst, [util_error_t\(*data_rdy\)](#)(void *))
- [util_error_t device_interface_create](#) ([device_interface_t](#) *interface, void *inst, [device_daemon_t](#) *daemon, [util_error_t\(*send\)](#)(void *, [uint8_t](#) *, [uint32_t](#)), [util_error_t\(*recv\)](#)(void *, [uint8_t](#) *, [uint32_t](#) *), [util_error_t\(*handle_data\)](#)(void *, void *))
- [util_error_t device_interface_send](#) ([device_interface_t](#) *interface, [uint8_t](#) *data, [uint32_t](#) len)
- [util_error_t device_interface_rcv](#) ([device_interface_t](#) *interface, [uint8_t](#) *data, [uint32_t](#) *len)
- [util_error_t device_write_i32](#) ([device_t](#) *dev, [uint32_t](#) addr, [int32_t](#) data)
- [util_error_t device_write_u32](#) ([device_t](#) *dev, [uint32_t](#) addr, [uint32_t](#) data)
- [util_error_t device_write_i16](#) ([device_t](#) *dev, [uint32_t](#) addr, [int16_t](#) data)
- [util_error_t device_write_u16](#) ([device_t](#) *dev, [uint32_t](#) addr, [uint16_t](#) data)
- [util_error_t device_write_i8](#) ([device_t](#) *dev, [uint32_t](#) addr, [int8_t](#) data)
- [util_error_t device_write_u8](#) ([device_t](#) *dev, [uint32_t](#) addr, [uint8_t](#) data)
- [util_error_t device_read_i32](#) ([device_t](#) *dev, [uint32_t](#) addr, [int32_t](#) *data)
- [util_error_t device_read_u32](#) ([device_t](#) *dev, [uint32_t](#) addr, [uint32_t](#) *data)
- [util_error_t device_read_i16](#) ([device_t](#) *dev, [uint32_t](#) addr, [int16_t](#) *data)
- [util_error_t device_read_u16](#) ([device_t](#) *dev, [uint32_t](#) addr, [uint16_t](#) *data)
- [util_error_t device_read_i8](#) ([device_t](#) *dev, [uint32_t](#) addr, [int8_t](#) *data)
- [util_error_t device_read_u8](#) ([device_t](#) *dev, [uint32_t](#) addr, [uint8_t](#) *data)

4.12.1 Macro Definition Documentation

4.12.1.1 DEAMON_STACK_SIZE

```
#define DEAMON_STACK_SIZE 1024
```

Definition at line 30 of file device.h.

4.12.1.2 DEVICE_MAX_INTERFACES_PER_DEAMON

```
#define DEVICE_MAX_INTERFACES_PER_DEAMON 16
```

Definition at line 28 of file device.h.

4.12.1.3 DEVICE_NAME_LEN

```
#define DEVICE_NAME_LEN 16
```

Definition at line 26 of file device.h.

4.12.2 Typedef Documentation

4.12.2.1 device_daemon_t

```
typedef struct device_daemon device_daemon_t
```

4.12.2.2 device_interface_t

```
typedef struct device_interface device_interface_t
```

4.12.2.3 device_t

```
typedef struct device device_t
```

4.12.3 Function Documentation

4.12.3.1 device_create()

```
util_error_t device_create (
    device_t * dev,
    void * context,
    device_interface_t * interface,
    util_error_t(*) (void *, device_interface_t *, uint32_t, uint8_t *, uint32_t) read←
_reg,
    util_error_t(*) (void *, device_interface_t *, uint32_t, uint8_t *, uint32_t) write←
_reg )
```

Initialize a device instance.

Parameters

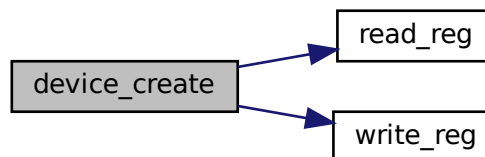
<i>dev</i>	Pointer to the <code>device_t</code> structure describing this device.
<i>context</i>	Generic pointer to a device context.
<i>interface</i>	Pointer to the <code>device_interface_t</code> associated with this device.
<i>read_reg</i>	Pointer to a read register function for this device.
<i>write_reg</i>	Pointer to a write register function for this device.

Definition at line 66 of file `device.c`.

References `device::context`, `ER_SUCCESS`, `device::id`, `device::interface`, `read_reg()`, `device::read_reg`, `write_reg()`, and `device::write_reg`.

Referenced by `i2c_sensor_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.12.3.2 device_daemon_create()

```
util_error_t device_daemon_create (
    device_daemon_t * daemon,
    const char * name,
    uint32_t prio,
    void * inst,
    util_error_t (*) (void *) data_rdy )
```

Definition at line 103 of file device.c.

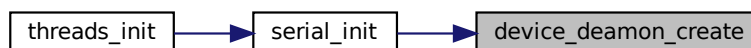
References `device_daemon::buffer`, `device_daemon::context`, `device_daemon::data_rdy`, `DEAMON_STACK_SIZE`, `device_daemon_thread()`, `ER_RESSOURCE_ERROR`, `ER_SUCCESS`, `device_daemon::handle`, `device_daemon::id`, `device_daemon::interfaces_count`, and `device_daemon::stack`.

Referenced by `serial_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.12.3.3 device_interface_create()

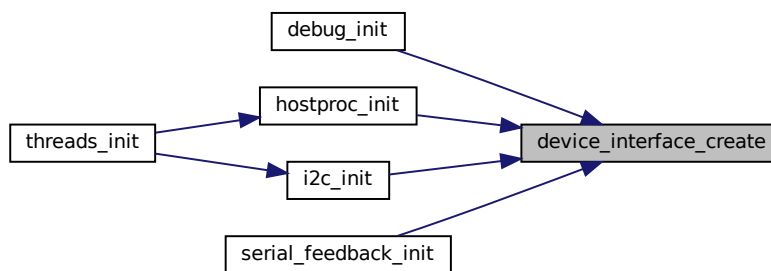
```
util_error_t device_interface_create (
    device_interface_t * interface,
    void * inst,
    device_deamon_t * deamon,
    util_error_t(*) (void *, uint8_t *, uint32_t) send,
    util_error_t(*) (void *, uint8_t *, uint32_t *) recv,
    util_error_t(*) (void *, void *) handle_data )
```

Definition at line 83 of file device.c.

References `device_interface::context`, `ER_SUCCESS`, `device_interface::handle_data`, `device_interface::id`, `device_deamon::interfaces`, `device_deamon::interfaces_count`, `device_interface::recv`, and `device_interface::send`.

Referenced by `debug_init()`, `hostproc_init()`, `i2c_init()`, and `serial_feedback_init()`.

Here is the caller graph for this function:



4.12.3.4 device_interface_recv()

```
util_error_t device_interface_recv (
    device_interface_t * interface,
    uint8_t * data,
    uint32_t * len )
```

Definition at line 150 of file device.c.

References `device_interface::context`, `ER_RESSOURCE_ERROR`, `device_interface::recv`, and `device_interface::send`.

4.12.3.5 device_interface_send()

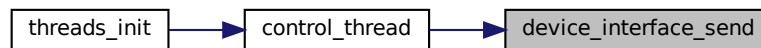
```
util_error_t device_interface_send (  
    device_interface_t * interface,  
    uint8_t * data,  
    uint32_t len )
```

Definition at line 141 of file device.c.

References device_interface::context, ER_RESSOURCE_ERROR, and device_interface::send.

Referenced by control_thread().

Here is the caller graph for this function:



4.12.3.6 device_read_i16()

```
util_error_t device_read_i16 (  
    device_t * dev,  
    uint32_t addr,  
    int16_t * data )
```

Definition at line 224 of file device.c.

References device::context, ER_SUCCESS, device::interface, LEN_16, device::read_reg, and util_decode_i16().

Here is the call graph for this function:



4.12.3.7 device_read_i32()

```
util_error_t device_read_i32 (  
    device_t * dev,  
    uint32_t addr,  
    int32_t * data )
```

Definition at line 209 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::interface`, `LEN_32`, `device::read_reg`, and `util_decode_i32()`.

Here is the call graph for this function:



4.12.3.8 device_read_i8()

```
util_error_t device_read_i8 (  
    device_t * dev,  
    uint32_t addr,  
    int8_t * data )
```

Definition at line 240 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::interface`, `LEN_8`, `device::read_reg`, and `util_decode_i8()`.

Here is the call graph for this function:



4.12.3.9 device_read_u16()

```
util_error_t device_read_u16 (  
    device_t * dev,  
    uint32_t addr,  
    uint16_t * data )
```

Definition at line 232 of file device.c.

References device::context, ER_SUCCESS, device::interface, LEN_16, device::read_reg, and util_decode_u16().

Here is the call graph for this function:



4.12.3.10 device_read_u32()

```
util_error_t device_read_u32 (  
    device_t * dev,  
    uint32_t addr,  
    uint32_t * data )
```

Definition at line 216 of file device.c.

References device::context, ER_SUCCESS, device::interface, LEN_32, device::read_reg, and util_decode_u32().

Here is the call graph for this function:



4.12.3.11 device_read_u8()

```
util_error_t device_read_u8 (  
    device_t * dev,  
    uint32_t addr,  
    uint8_t * data )
```

Definition at line 248 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::interface`, `LEN_8`, `device::read_reg`, and `util_decode_u8()`.

Here is the call graph for this function:



4.12.3.12 device_write_i16()

```
util_error_t device_write_i16 (  
    device_t * dev,  
    uint32_t addr,  
    int16_t data )
```

Definition at line 178 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::interface`, `LEN_16`, `util_encode_i16()`, and `device::write_reg`.

Here is the call graph for this function:



4.12.3.13 device_write_i32()

```
util_error_t device_write_i32 (  
    device_t * dev,  
    uint32_t addr,  
    int32_t data )
```

Definition at line 162 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::interface`, `LEN_32`, `util_encode_i32()`, and `device::write_reg`.

Here is the call graph for this function:



4.12.3.14 device_write_i8()

```
util_error_t device_write_i8 (  
    device_t * dev,  
    uint32_t addr,  
    int8_t data )
```

Definition at line 194 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::interface`, `LEN_8`, `util_encode_i8()`, and `device::write_reg`.

Here is the call graph for this function:



4.12.3.15 device_write_u16()

```
util_error_t device_write_u16 (  
    device_t * dev,  
    uint32_t addr,  
    uint16_t data )
```

Definition at line 186 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::interface`, `LEN_16`, `util_encode_u16()`, and `device::write_reg`.

Here is the call graph for this function:



4.12.3.16 device_write_u32()

```
util_error_t device_write_u32 (  
    device_t * dev,  
    uint32_t addr,  
    uint32_t data )
```

Definition at line 170 of file device.c.

References `device::context`, `ER_SUCCESS`, `device::interface`, `LEN_32`, `util_encode_u32()`, and `device::write_reg`.

Here is the call graph for this function:



4.12.3.17 device_write_u8()

```
util_error_t device_write_u8 (  
    device_t * dev,  
    uint32_t addr,  
    uint8_t data )
```

Definition at line 201 of file device.c.

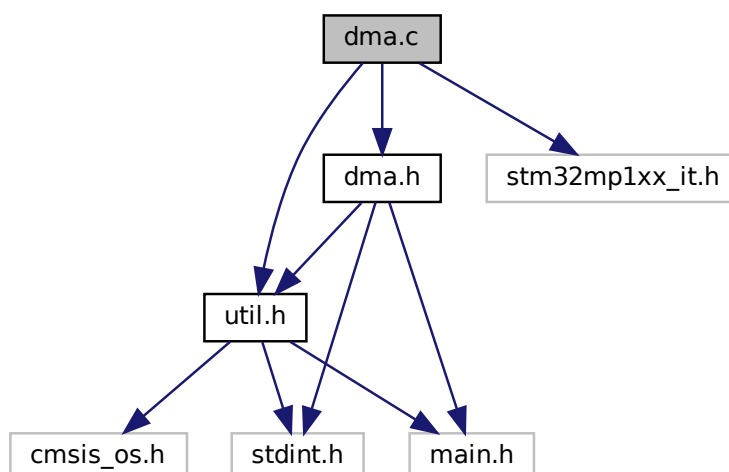
References device::context, ER_SUCCESS, device::interface, LEN_8, util_encode_u8(), and device::write_reg.

Here is the call graph for this function:



4.13 dma.c File Reference

```
#include "dma.h"  
#include <util.h>  
#include "stm32mp1xx_it.h"  
Include dependency graph for dma.c:
```



Functions

- void [dma_handle_interrupt](#) ([dma_stream_dev_t](#) *stream)
- [util_error_t](#) [dma2_init_scheduler](#) (void)
- [dma_scheduler_dev_t](#) * [dma2_get_scheduler](#) (void)
- [dma_stream_dev_t](#) * [dma2_get_streams](#) (void)
- [util_error_t](#) [dma_scheduler_init](#) ([dma_scheduler_dev_t](#) *dma_scheduler, [dma_stream_dev_t](#) *dma_streams, [uint16_t](#) nb_dma_streams)
- [dma_stream_dev_t](#) * [dma_scheduler_request_stream](#) ([dma_scheduler_dev_t](#) *dma_scheduler)
- [util_error_t](#) [dma_scheduler_release_stream](#) ([dma_scheduler_dev_t](#) *dma_scheduler, [dma_stream_dev_t](#) *dma_stream)
- [util_error_t](#) [dma_start_stream](#) ([dma_stream_dev_t](#) *stream, [dma_stream_config_t](#) config)

Variables

- [dma_scheduler_dev_t](#) [dma2_scheduler](#)
- [dma_stream_dev_t](#) [dma2_streams](#) []

4.13.1 Function Documentation

4.13.1.1 dma2_get_scheduler()

```
dma\_scheduler\_dev\_t* dma2_get_scheduler (  
    void )
```

Definition at line 182 of file dma.c.

References [dma2_scheduler](#).

4.13.1.2 dma2_get_streams()

```
dma\_stream\_dev\_t* dma2_get_streams (  
    void )
```

Definition at line 186 of file dma.c.

References [dma2_streams](#).

4.13.1.3 dma2_init_scheduler()

```
util_error_t dma2_init_scheduler (
    void )
```

Definition at line 176 of file dma.c.

References dma2_scheduler, dma2_streams, dma_scheduler_init(), and ER_SUCCESS.

Here is the call graph for this function:



4.13.1.4 dma_handle_interrupt()

```
void dma_handle_interrupt (
    dma_stream_dev_t * stream )
```

Definition at line 151 of file dma.c.

References dma_stream_dev::dma, DMA_STATUS_TC, DMA_STATUS_TE, DMA_STATUS_TH, dma_stream_dev::number, dma_stream_dev::transfer_cplt, dma_stream_dev::transfer_error, dma_stream_dev::transfer_half, and dma_stream_dev::user_context.

4.13.1.5 dma_scheduler_init()

```
util_error_t dma_scheduler_init (
    dma_scheduler_dev_t * dma_scheduler,
    dma_stream_dev_t * dma_streams,
    uint16_t nb_dma_streams )
```

Definition at line 195 of file dma.c.

References DMA_STREAM_FREE, DMA_STREAMS_MAX_LEN, ER_OUT_OF_RANGE, ER_SUCCESS, dma_stream_dev::state, and dma_scheduler_dev::streams.

Referenced by dma2_init_scheduler().

Here is the caller graph for this function:



4.13.1.6 dma_scheduler_release_stream()

```
util_error_t dma_scheduler_release_stream (
    dma_scheduler_dev_t * dma_scheduler,
    dma_stream_dev_t * dma_stream )
```

Definition at line 250 of file dma.c.

References DMA_STREAM_FREE, ENTER_CRITICAL, ER_SUCCESS, EXIT_CRITICAL, dma_scheduler_dev::free_stream_count, and dma_stream_dev::state.

4.13.1.7 dma_scheduler_request_stream()

```
dma_stream_dev_t* dma_scheduler_request_stream (
    dma_scheduler_dev_t * dma_scheduler )
```

Definition at line 224 of file dma.c.

References DMA_STREAM_BUSY, DMA_STREAM_FREE, ENTER_CRITICAL, EXIT_CRITICAL, dma_scheduler_dev::free_stream_count, dma_stream_dev::state, dma_scheduler_dev::stream_count, and dma_scheduler_dev::streams.

4.13.1.8 dma_start_stream()

```
util_error_t dma_start_stream (
    dma_stream_dev_t * stream,
    dma_stream_config_t config )
```

Definition at line 263 of file dma.c.

References dma_stream_config::direction, dma_stream_dev::dma, dma_stream_dev::dma_stream, dma_stream_dev::dmamux_channel, dma_stream_config::dmamux_request_number, ER_OUT_OF_RANGE, dma_stream_config::m0_addr, dma_stream_config::m1_addr, dma_stream_dev::number, dma_stream_config::p_addr, dma_stream_config::peripheral_flow_control, dma_stream_config::priority, dma_stream_config::transfer_cplt, dma_stream_dev::transfer_cplt, dma_stream_config::transfer_error, dma_stream_dev::transfer_error, dma_stream_config::transfer_half, dma_stream_dev::transfer_half, dma_stream_config::transfer_size, dma_stream_config::user_context, dma_stream_dev::user_context, and WRITE_IN_REG.

4.13.2 Variable Documentation

4.13.2.1 dma2_scheduler

```
dma_scheduler_dev_t dma2_scheduler
```

Definition at line 37 of file dma.c.

Referenced by dma2_get_scheduler(), and dma2_init_scheduler().

4.13.2.2 dma2_streams

```
dma_stream_dev_t dma2_streams[ ]
```

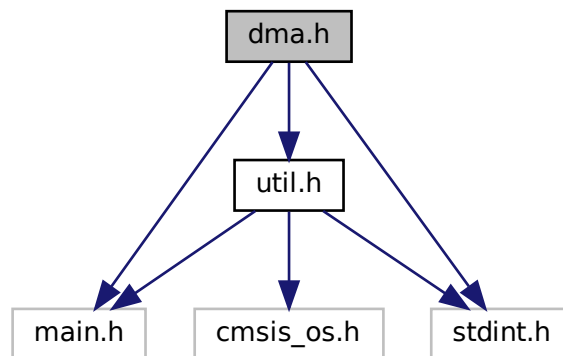
Definition at line 39 of file dma.c.

Referenced by dma2_get_streams(), and dma2_init_scheduler().

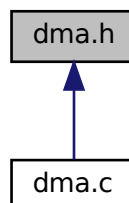
4.14 dma.h File Reference

```
#include <main.h>
#include <stdint.h>
#include <util.h>
```

Include dependency graph for dma.h:



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



Data Structures

- struct [dma_request](#)
- struct [dma_stream_config](#)
- struct [dma_stream_dev](#)
- struct [dma_scheduler_dev](#)

Macros

- #define [DMA_STREAMS_MAX_LEN](#) 8
- #define [STM32_DMAMUX1_REQ_GEN0](#) 1
- #define [STM32_DMAMUX1_REQ_GEN1](#) 2
- #define [STM32_DMAMUX1_REQ_GEN2](#) 3
- #define [STM32_DMAMUX1_REQ_GEN3](#) 4
- #define [STM32_DMAMUX1_REQ_GEN4](#) 5
- #define [STM32_DMAMUX1_REQ_GEN5](#) 6
- #define [STM32_DMAMUX1_REQ_GEN6](#) 7
- #define [STM32_DMAMUX1_REQ_GEN7](#) 8
- #define [STM32_DMAMUX1_ADC1](#) 9
- #define [STM32_DMAMUX1_ADC2](#) 10
- #define [STM32_DMAMUX1_TIM1_CH1](#) 11
- #define [STM32_DMAMUX1_TIM1_CH2](#) 12
- #define [STM32_DMAMUX1_TIM1_CH3](#) 13
- #define [STM32_DMAMUX1_TIM1_CH4](#) 14
- #define [STM32_DMAMUX1_TIM1_UP](#) 15
- #define [STM32_DMAMUX1_TIM1_TRIG](#) 16
- #define [STM32_DMAMUX1_TIM1_COM](#) 17
- #define [STM32_DMAMUX1_TIM2_CH1](#) 18
- #define [STM32_DMAMUX1_TIM2_CH2](#) 19
- #define [STM32_DMAMUX1_TIM2_CH3](#) 20
- #define [STM32_DMAMUX1_TIM2_CH4](#) 21
- #define [STM32_DMAMUX1_TIM2_UP](#) 22
- #define [STM32_DMAMUX1_TIM3_CH1](#) 23
- #define [STM32_DMAMUX1_TIM3_CH2](#) 24
- #define [STM32_DMAMUX1_TIM3_CH3](#) 25
- #define [STM32_DMAMUX1_TIM3_CH4](#) 26
- #define [STM32_DMAMUX1_TIM3_UP](#) 27
- #define [STM32_DMAMUX1_TIM3_TRIG](#) 28
- #define [STM32_DMAMUX1_TIM4_CH1](#) 29
- #define [STM32_DMAMUX1_TIM4_CH2](#) 30
- #define [STM32_DMAMUX1_TIM4_CH3](#) 31
- #define [STM32_DMAMUX1_TIM4_UP](#) 32
- #define [STM32_DMAMUX1_I2C1_RX](#) 33
- #define [STM32_DMAMUX1_I2C1_TX](#) 34
- #define [STM32_DMAMUX1_I2C2_RX](#) 35
- #define [STM32_DMAMUX1_I2C2_TX](#) 36
- #define [STM32_DMAMUX1_SPI1_RX](#) 37
- #define [STM32_DMAMUX1_SPI1_TX](#) 38
- #define [STM32_DMAMUX1_SPI2_RX](#) 39
- #define [STM32_DMAMUX1_SPI2_TX](#) 40
- #define [STM32_DMAMUX1_RSVD41](#) 41
- #define [STM32_DMAMUX1_RSVD42](#) 42
- #define [STM32_DMAMUX1_USART2_RX](#) 43
- #define [STM32_DMAMUX1_USART2_TX](#) 44

- #define STM32_DMAMUX1_USART3_RX 45
- #define STM32_DMAMUX1_USART3_TX 46
- #define STM32_DMAMUX1_TIM8_CH1 47
- #define STM32_DMAMUX1_TIM8_CH2 48
- #define STM32_DMAMUX1_TIM8_CH3 49
- #define STM32_DMAMUX1_TIM8_CH4 50
- #define STM32_DMAMUX1_TIM8_UP 51
- #define STM32_DMAMUX1_TIM8_TRIG 52
- #define STM32_DMAMUX1_TIM8_COM 53
- #define STM32_DMAMUX1_RSVD54 54
- #define STM32_DMAMUX1_TIM5_CH1 55
- #define STM32_DMAMUX1_TIM5_CH2 56
- #define STM32_DMAMUX1_TIM5_CH3 57
- #define STM32_DMAMUX1_TIM5_CH4 58
- #define STM32_DMAMUX1_TIM5_UP 59
- #define STM32_DMAMUX1_TIM5_TRIG 60
- #define STM32_DMAMUX1_SPI3_RX 61
- #define STM32_DMAMUX1_SPI3_TX 62
- #define STM32_DMAMUX1_UART4_RX 63
- #define STM32_DMAMUX1_UART4_TX 64
- #define STM32_DMAMUX1_UART5_RX 65
- #define STM32_DMAMUX1_UART5_TX 66
- #define STM32_DMAMUX1_DAC1_CH1 67
- #define STM32_DMAMUX1_DAC1_CH2 68
- #define STM32_DMAMUX1_TIM6_UP 69
- #define STM32_DMAMUX1_TIM7_UP 70
- #define STM32_DMAMUX1_USART6_RX 71
- #define STM32_DMAMUX1_USART6_TX 72
- #define STM32_DMAMUX1_I2C3_RX 73
- #define STM32_DMAMUX1_I2C3_TX 74
- #define STM32_DMAMUX1_DCM1 75
- #define STM32_DMAMUX1_Cryp2_IN 76
- #define STM32_DMAMUX1_Cryp2_OUT 77
- #define STM32_DMAMUX1_HASH2_IN 78
- #define STM32_DMAMUX1_UART7_RX 79
- #define STM32_DMAMUX1_UART7_TX 80
- #define STM32_DMAMUX1_UART8_RX 81
- #define STM32_DMAMUX1_UART8_TX 82
- #define STM32_DMAMUX1_SPI4_RX 83
- #define STM32_DMAMUX1_SPI4_TX 84
- #define STM32_DMAMUX1_SPI5_RX 85
- #define STM32_DMAMUX1_SPI5_TX 86
- #define STM32_DMAMUX1_SAI1_A 87
- #define STM32_DMAMUX1_SAI1_B 88
- #define STM32_DMAMUX1_SAI2_A 89
- #define STM32_DMAMUX1_SAI2_B 90
- #define STM32_DMAMUX1_DFSDM1_FLT4 91
- #define STM32_DMAMUX1_DFSDM1_FLT5 92
- #define STM32_DMAMUX1_SPDIFRX_DT 93
- #define STM32_DMAMUX1_SPDIFRX_CS 94
- #define STM32_DMAMUX1_RSVD95 95
- #define STM32_DMAMUX1_RSVD96 96
- #define STM32_DMAMUX1_RSVD97 97
- #define STM32_DMAMUX1_RSVD98 98
- #define STM32_DMAMUX1_SAI4_A 99

- #define [STM32_DMAMUX1_SAI4_B](#) 100
- #define [STM32_DMAMUX1_DFSDM1_FLT0](#) 101
- #define [STM32_DMAMUX1_DFSDM1_FLT1](#) 102
- #define [STM32_DMAMUX1_DFSDM1_FLT2](#) 103
- #define [STM32_DMAMUX1_DFSDM1_FLT3](#) 104
- #define [STM32_DMAMUX1_TIM15_CH1](#) 105
- #define [STM32_DMAMUX1_TIM15_UP](#) 106
- #define [STM32_DMAMUX1_TIM15_TRIG](#) 107
- #define [STM32_DMAMUX1_TIM15_COM](#) 108
- #define [STM32_DMAMUX1_TIM16_CH1](#) 109
- #define [STM32_DMAMUX1_TIM16_UP](#) 110
- #define [STM32_DMAMUX1_TIM17_CH1](#) 111
- #define [STM32_DMAMUX1_TIM17_UP](#) 112
- #define [STM32_DMAMUX1_SAI3_A](#) 113
- #define [STM32_DMAMUX1_SAI3_B](#) 114
- #define [STM32_DMAMUX1_I2C5_RX](#) 115
- #define [STM32_DMAMUX1_I2C5_TX](#) 116
- #define [STM32_DMAMUX1_RSVD117](#) 117
- #define [STM32_DMAMUX1_RSVD118](#) 118
- #define [STM32_DMAMUX1_RSVD119](#) 119
- #define [STM32_DMAMUX1_RSVD120](#) 120
- #define [STM32_DMAMUX1_RSVD121](#) 121
- #define [STM32_DMAMUX1_RSVD122](#) 122
- #define [STM32_DMAMUX1_RSVD123](#) 123
- #define [STM32_DMAMUX1_RSVD124](#) 124
- #define [STM32_DMAMUX1_RSVD125](#) 125
- #define [STM32_DMAMUX1_RSVD126](#) 126
- #define [STM32_DMAMUX1_RSVD127](#) 127
- #define [DMA_STATUS_TC](#) (0b1<<5)
- #define [DMA_STATUS_TH](#) (0b1<<4)
- #define [DMA_STATUS_TE](#) (0b1<<3)

Typedefs

- typedef enum [dma_stream_state](#) [dma_stream_state_t](#)
- typedef enum [dma_stream_dir](#) [dma_stream_dir_t](#)
- typedef struct [dma_request](#) [dma_request_t](#)
- typedef struct [dma_stream_config](#) [dma_stream_config_t](#)
- typedef struct [dma_stream_dev](#) [dma_stream_dev_t](#)
- typedef struct [dma_scheduler_dev](#) [dma_scheduler_dev_t](#)

Enumerations

- enum [dma_stream_state](#) { [DMA_STREAM_BUSY](#) , [DMA_STREAM_FREE](#) }
- enum [dma_stream_dir](#) { [DMA_STREAM_P2M](#) = 0b00 , [DMA_STREAM_M2P](#) = 0b01 , [DMA_STREAM_M2M](#) = 0b10 }

Functions

- [util_error_t dma2_init_scheduler](#) (void)
- [dma_scheduler_dev_t * dma2_get_scheduler](#) (void)
- [dma_stream_dev_t * dma2_get_streams](#) (void)
- [util_error_t dma_scheduler_init](#) ([dma_scheduler_dev_t](#) *dma_scheduler, [dma_stream_dev_t](#) *dma_streams, [uint16_t](#) nb_dma_streams)
- [dma_stream_dev_t * dma_scheduler_request_stream](#) ([dma_scheduler_dev_t](#) *dma_scheduler)
- [util_error_t dma_scheduler_release_stream](#) ([dma_scheduler_dev_t](#) *dma_scheduler, [dma_stream_dev_t](#) *dma_stream)
- [util_error_t dma_start_stream](#) ([dma_stream_dev_t](#) *stream, [dma_stream_config_t](#) config)
- [util_error_t dma_stop_stream](#) ([dma_stream_dev_t](#) *stream)
- [util_error_t dma_copy](#) (void *dst, void *src, [uint32_t](#) len)

4.14.1 Macro Definition Documentation

4.14.1.1 DMA_STATUS_TC

```
#define DMA_STATUS_TC (0b1<<5)
```

Definition at line 162 of file dma.h.

4.14.1.2 DMA_STATUS_TE

```
#define DMA_STATUS_TE (0b1<<3)
```

Definition at line 164 of file dma.h.

4.14.1.3 DMA_STATUS_TH

```
#define DMA_STATUS_TH (0b1<<4)
```

Definition at line 163 of file dma.h.

4.14.1.4 DMA_STREAMS_MAX_LEN

```
#define DMA_STREAMS_MAX_LEN 8
```

Definition at line 25 of file dma.h.

4.14.1.5 STM32_DMAMUX1_ADC1

```
#define STM32_DMAMUX1_ADC1 9
```

Definition at line 41 of file dma.h.

4.14.1.6 STM32_DMAMUX1_ADC2

```
#define STM32_DMAMUX1_ADC2 10
```

Definition at line 42 of file dma.h.

4.14.1.7 STM32_DMAMUX1_Cryp2_IN

```
#define STM32_DMAMUX1_Cryp2_IN 76
```

Definition at line 108 of file dma.h.

4.14.1.8 STM32_DMAMUX1_Cryp2_OUT

```
#define STM32_DMAMUX1_Cryp2_OUT 77
```

Definition at line 109 of file dma.h.

4.14.1.9 STM32_DMAMUX1_DAC1_CH1

```
#define STM32_DMAMUX1_DAC1_CH1 67
```

Definition at line 99 of file dma.h.

4.14.1.10 STM32_DMAMUX1_DAC1_CH2

```
#define STM32_DMAMUX1_DAC1_CH2 68
```

Definition at line 100 of file dma.h.

4.14.1.11 STM32_DMAMUX1_DCMI

```
#define STM32_DMAMUX1_DCMI 75
```

Definition at line 107 of file dma.h.

4.14.1.12 STM32_DMAMUX1_DFSDM1_FLT0

```
#define STM32_DMAMUX1_DFSDM1_FLT0 101
```

Definition at line 133 of file dma.h.

4.14.1.13 STM32_DMAMUX1_DFSDM1_FLT1

```
#define STM32_DMAMUX1_DFSDM1_FLT1 102
```

Definition at line 134 of file dma.h.

4.14.1.14 STM32_DMAMUX1_DFSDM1_FLT2

```
#define STM32_DMAMUX1_DFSDM1_FLT2 103
```

Definition at line 135 of file dma.h.

4.14.1.15 STM32_DMAMUX1_DFSDM1_FLT3

```
#define STM32_DMAMUX1_DFSDM1_FLT3 104
```

Definition at line 136 of file dma.h.

4.14.1.16 STM32_DMAMUX1_DFSDM1_FLT4

```
#define STM32_DMAMUX1_DFSDM1_FLT4 91
```

Definition at line 123 of file dma.h.

4.14.1.17 STM32_DMAMUX1_DFSDM1_FLT5

```
#define STM32_DMAMUX1_DFSDM1_FLT5 92
```

Definition at line 124 of file dma.h.

4.14.1.18 STM32_DMAMUX1_HASH2_IN

```
#define STM32_DMAMUX1_HASH2_IN 78
```

Definition at line 110 of file dma.h.

4.14.1.19 STM32_DMAMUX1_I2C1_RX

```
#define STM32_DMAMUX1_I2C1_RX 33
```

Definition at line 65 of file dma.h.

4.14.1.20 STM32_DMAMUX1_I2C1_TX

```
#define STM32_DMAMUX1_I2C1_TX 34
```

Definition at line 66 of file dma.h.

4.14.1.21 STM32_DMAMUX1_I2C2_RX

```
#define STM32_DMAMUX1_I2C2_RX 35
```

Definition at line 67 of file dma.h.

4.14.1.22 STM32_DMAMUX1_I2C2_TX

```
#define STM32_DMAMUX1_I2C2_TX 36
```

Definition at line 68 of file dma.h.

4.14.1.23 STM32_DMAMUX1_I2C3_RX

```
#define STM32_DMAMUX1_I2C3_RX 73
```

Definition at line 105 of file dma.h.

4.14.1.24 STM32_DMAMUX1_I2C3_TX

```
#define STM32_DMAMUX1_I2C3_TX 74
```

Definition at line 106 of file dma.h.

4.14.1.25 STM32_DMAMUX1_I2C5_RX

```
#define STM32_DMAMUX1_I2C5_RX 115
```

Definition at line 147 of file dma.h.

4.14.1.26 STM32_DMAMUX1_I2C5_TX

```
#define STM32_DMAMUX1_I2C5_TX 116
```

Definition at line 148 of file dma.h.

4.14.1.27 STM32_DMAMUX1_REQ_GEN0

```
#define STM32_DMAMUX1_REQ_GEN0 1
```

Definition at line 33 of file dma.h.

4.14.1.28 STM32_DMAMUX1_REQ_GEN1

```
#define STM32_DMAMUX1_REQ_GEN1 2
```

Definition at line 34 of file dma.h.

4.14.1.29 STM32_DMAMUX1_REQ_GEN2

```
#define STM32_DMAMUX1_REQ_GEN2 3
```

Definition at line 35 of file dma.h.

4.14.1.30 STM32_DMAMUX1_REQ_GEN3

```
#define STM32_DMAMUX1_REQ_GEN3 4
```

Definition at line 36 of file dma.h.

4.14.1.31 STM32_DMAMUX1_REQ_GEN4

```
#define STM32_DMAMUX1_REQ_GEN4 5
```

Definition at line 37 of file dma.h.

4.14.1.32 STM32_DMAMUX1_REQ_GEN5

```
#define STM32_DMAMUX1_REQ_GEN5 6
```

Definition at line 38 of file dma.h.

4.14.1.33 STM32_DMAMUX1_REQ_GEN6

```
#define STM32_DMAMUX1_REQ_GEN6 7
```

Definition at line 39 of file dma.h.

4.14.1.34 STM32_DMAMUX1_REQ_GEN7

```
#define STM32_DMAMUX1_REQ_GEN7 8
```

Definition at line 40 of file dma.h.

4.14.1.35 STM32_DMAMUX1_RSVD117

```
#define STM32_DMAMUX1_RSVD117 117
```

Definition at line 149 of file dma.h.

4.14.1.36 STM32_DMAMUX1_RSVD118

```
#define STM32_DMAMUX1_RSVD118 118
```

Definition at line 150 of file dma.h.

4.14.1.37 STM32_DMAMUX1_RSVD119

```
#define STM32_DMAMUX1_RSVD119 119
```

Definition at line 151 of file dma.h.

4.14.1.38 STM32_DMAMUX1_RSVD120

```
#define STM32_DMAMUX1_RSVD120 120
```

Definition at line 152 of file dma.h.

4.14.1.39 STM32_DMAMUX1_RSVD121

```
#define STM32_DMAMUX1_RSVD121 121
```

Definition at line 153 of file dma.h.

4.14.1.40 STM32_DMAMUX1_RSVD122

```
#define STM32_DMAMUX1_RSVD122 122
```

Definition at line 154 of file dma.h.

4.14.1.41 STM32_DMAMUX1_RSVD123

```
#define STM32_DMAMUX1_RSVD123 123
```

Definition at line 155 of file dma.h.

4.14.1.42 STM32_DMAMUX1_RSVD124

```
#define STM32_DMAMUX1_RSVD124 124
```

Definition at line 156 of file dma.h.

4.14.1.43 STM32_DMAMUX1_RSVD125

```
#define STM32_DMAMUX1_RSVD125 125
```

Definition at line 157 of file dma.h.

4.14.1.44 STM32_DMAMUX1_RSVD126

```
#define STM32_DMAMUX1_RSVD126 126
```

Definition at line 158 of file dma.h.

4.14.1.45 STM32_DMAMUX1_RSVD127

```
#define STM32_DMAMUX1_RSVD127 127
```

Definition at line 159 of file dma.h.

4.14.1.46 STM32_DMAMUX1_RSVD41

```
#define STM32_DMAMUX1_RSVD41 41
```

Definition at line 73 of file dma.h.

4.14.1.47 STM32_DMAMUX1_RSVD42

```
#define STM32_DMAMUX1_RSVD42 42
```

Definition at line 74 of file dma.h.

4.14.1.48 STM32_DMAMUX1_RSVD54

```
#define STM32_DMAMUX1_RSVD54 54
```

Definition at line 86 of file dma.h.

4.14.1.49 STM32_DMAMUX1_RSVD95

```
#define STM32_DMAMUX1_RSVD95 95
```

Definition at line 127 of file dma.h.

4.14.1.50 STM32_DMAMUX1_RSVD96

```
#define STM32_DMAMUX1_RSVD96 96
```

Definition at line 128 of file dma.h.

4.14.1.51 STM32_DMAMUX1_RSVD97

```
#define STM32_DMAMUX1_RSVD97 97
```

Definition at line 129 of file dma.h.

4.14.1.52 STM32_DMAMUX1_RSVD98

```
#define STM32_DMAMUX1_RSVD98 98
```

Definition at line 130 of file dma.h.

4.14.1.53 STM32_DMAMUX1_SAI1_A

```
#define STM32_DMAMUX1_SAI1_A 87
```

Definition at line 119 of file dma.h.

4.14.1.54 STM32_DMAMUX1_SAI1_B

```
#define STM32_DMAMUX1_SAI1_B 88
```

Definition at line 120 of file dma.h.

4.14.1.55 STM32_DMAMUX1_SAI2_A

```
#define STM32_DMAMUX1_SAI2_A 89
```

Definition at line 121 of file dma.h.

4.14.1.56 STM32_DMAMUX1_SAI2_B

```
#define STM32_DMAMUX1_SAI2_B 90
```

Definition at line 122 of file dma.h.

4.14.1.57 STM32_DMAMUX1_SAI3_A

```
#define STM32_DMAMUX1_SAI3_A 113
```

Definition at line 145 of file dma.h.

4.14.1.58 STM32_DMAMUX1_SAI3_B

```
#define STM32_DMAMUX1_SAI3_B 114
```

Definition at line 146 of file dma.h.

4.14.1.59 STM32_DMAMUX1_SAI4_A

```
#define STM32_DMAMUX1_SAI4_A 99
```

Definition at line 131 of file dma.h.

4.14.1.60 STM32_DMAMUX1_SAI4_B

```
#define STM32_DMAMUX1_SAI4_B 100
```

Definition at line 132 of file dma.h.

4.14.1.61 STM32_DMAMUX1_SPDIFRX_CS

```
#define STM32_DMAMUX1_SPDIFRX_CS 94
```

Definition at line 126 of file dma.h.

4.14.1.62 STM32_DMAMUX1_SPDIFRX_DT

```
#define STM32_DMAMUX1_SPDIFRX_DT 93
```

Definition at line 125 of file dma.h.

4.14.1.63 STM32_DMAMUX1_SPI1_RX

```
#define STM32_DMAMUX1_SPI1_RX 37
```

Definition at line 69 of file dma.h.

4.14.1.64 STM32_DMAMUX1_SPI1_TX

```
#define STM32_DMAMUX1_SPI1_TX 38
```

Definition at line 70 of file dma.h.

4.14.1.65 STM32_DMAMUX1_SPI2_RX

```
#define STM32_DMAMUX1_SPI2_RX 39
```

Definition at line 71 of file dma.h.

4.14.1.66 STM32_DMAMUX1_SPI2_TX

```
#define STM32_DMAMUX1_SPI2_TX 40
```

Definition at line 72 of file dma.h.

4.14.1.67 STM32_DMAMUX1_SPI3_RX

```
#define STM32_DMAMUX1_SPI3_RX 61
```

Definition at line 93 of file dma.h.

4.14.1.68 STM32_DMAMUX1_SPI3_TX

```
#define STM32_DMAMUX1_SPI3_TX 62
```

Definition at line 94 of file dma.h.

4.14.1.69 STM32_DMAMUX1_SPI4_RX

```
#define STM32_DMAMUX1_SPI4_RX 83
```

Definition at line 115 of file dma.h.

4.14.1.70 STM32_DMAMUX1_SPI4_TX

```
#define STM32_DMAMUX1_SPI4_TX 84
```

Definition at line 116 of file dma.h.

4.14.1.71 STM32_DMAMUX1_SPI5_RX

```
#define STM32_DMAMUX1_SPI5_RX 85
```

Definition at line 117 of file dma.h.

4.14.1.72 STM32_DMAMUX1_SPI5_TX

```
#define STM32_DMAMUX1_SPI5_TX 86
```

Definition at line 118 of file dma.h.

4.14.1.73 STM32_DMAMUX1_TIM15_CH1

```
#define STM32_DMAMUX1_TIM15_CH1 105
```

Definition at line 137 of file dma.h.

4.14.1.74 STM32_DMAMUX1_TIM15_COM

```
#define STM32_DMAMUX1_TIM15_COM 108
```

Definition at line 140 of file dma.h.

4.14.1.75 STM32_DMAMUX1_TIM15_TRIG

```
#define STM32_DMAMUX1_TIM15_TRIG 107
```

Definition at line 139 of file dma.h.

4.14.1.76 STM32_DMAMUX1_TIM15_UP

```
#define STM32_DMAMUX1_TIM15_UP 106
```

Definition at line 138 of file dma.h.

4.14.1.77 STM32_DMAMUX1_TIM16_CH1

```
#define STM32_DMAMUX1_TIM16_CH1 109
```

Definition at line 141 of file dma.h.

4.14.1.78 STM32_DMAMUX1_TIM16_UP

```
#define STM32_DMAMUX1_TIM16_UP 110
```

Definition at line 142 of file dma.h.

4.14.1.79 STM32_DMAMUX1_TIM17_CH1

```
#define STM32_DMAMUX1_TIM17_CH1 111
```

Definition at line 143 of file dma.h.

4.14.1.80 STM32_DMAMUX1_TIM17_UP

```
#define STM32_DMAMUX1_TIM17_UP 112
```

Definition at line 144 of file dma.h.

4.14.1.81 STM32_DMAMUX1_TIM1_CH1

```
#define STM32_DMAMUX1_TIM1_CH1 11
```

Definition at line 43 of file dma.h.

4.14.1.82 STM32_DMAMUX1_TIM1_CH2

```
#define STM32_DMAMUX1_TIM1_CH2 12
```

Definition at line 44 of file dma.h.

4.14.1.83 STM32_DMAMUX1_TIM1_CH3

```
#define STM32_DMAMUX1_TIM1_CH3 13
```

Definition at line 45 of file dma.h.

4.14.1.84 STM32_DMAMUX1_TIM1_CH4

```
#define STM32_DMAMUX1_TIM1_CH4 14
```

Definition at line 46 of file dma.h.

4.14.1.85 STM32_DMAMUX1_TIM1_COM

```
#define STM32_DMAMUX1_TIM1_COM 17
```

Definition at line 49 of file dma.h.

4.14.1.86 STM32_DMAMUX1_TIM1_TRIG

```
#define STM32_DMAMUX1_TIM1_TRIG 16
```

Definition at line 48 of file dma.h.

4.14.1.87 STM32_DMAMUX1_TIM1_UP

```
#define STM32_DMAMUX1_TIM1_UP 15
```

Definition at line 47 of file dma.h.

4.14.1.88 STM32_DMAMUX1_TIM2_CH1

```
#define STM32_DMAMUX1_TIM2_CH1 18
```

Definition at line 50 of file dma.h.

4.14.1.89 STM32_DMAMUX1_TIM2_CH2

```
#define STM32_DMAMUX1_TIM2_CH2 19
```

Definition at line 51 of file dma.h.

4.14.1.90 STM32_DMAMUX1_TIM2_CH3

```
#define STM32_DMAMUX1_TIM2_CH3 20
```

Definition at line 52 of file dma.h.

4.14.1.91 STM32_DMAMUX1_TIM2_CH4

```
#define STM32_DMAMUX1_TIM2_CH4 21
```

Definition at line 53 of file dma.h.

4.14.1.92 STM32_DMAMUX1_TIM2_UP

```
#define STM32_DMAMUX1_TIM2_UP 22
```

Definition at line 54 of file dma.h.

4.14.1.93 STM32_DMAMUX1_TIM3_CH1

```
#define STM32_DMAMUX1_TIM3_CH1 23
```

Definition at line 55 of file dma.h.

4.14.1.94 STM32_DMAMUX1_TIM3_CH2

```
#define STM32_DMAMUX1_TIM3_CH2 24
```

Definition at line 56 of file dma.h.

4.14.1.95 STM32_DMAMUX1_TIM3_CH3

```
#define STM32_DMAMUX1_TIM3_CH3 25
```

Definition at line 57 of file dma.h.

4.14.1.96 STM32_DMAMUX1_TIM3_CH4

```
#define STM32_DMAMUX1_TIM3_CH4 26
```

Definition at line 58 of file dma.h.

4.14.1.97 STM32_DMAMUX1_TIM3_TRIG

```
#define STM32_DMAMUX1_TIM3_TRIG 28
```

Definition at line 60 of file dma.h.

4.14.1.98 STM32_DMAMUX1_TIM3_UP

```
#define STM32_DMAMUX1_TIM3_UP 27
```

Definition at line 59 of file dma.h.

4.14.1.99 STM32_DMAMUX1_TIM4_CH1

```
#define STM32_DMAMUX1_TIM4_CH1 29
```

Definition at line 61 of file dma.h.

4.14.1.100 STM32_DMAMUX1_TIM4_CH2

```
#define STM32_DMAMUX1_TIM4_CH2 30
```

Definition at line 62 of file dma.h.

4.14.1.101 STM32_DMAMUX1_TIM4_CH3

```
#define STM32_DMAMUX1_TIM4_CH3 31
```

Definition at line 63 of file dma.h.

4.14.1.102 STM32_DMAMUX1_TIM4_UP

```
#define STM32_DMAMUX1_TIM4_UP 32
```

Definition at line 64 of file dma.h.

4.14.1.103 STM32_DMAMUX1_TIM5_CH1

```
#define STM32_DMAMUX1_TIM5_CH1 55
```

Definition at line 87 of file dma.h.

4.14.1.104 STM32_DMAMUX1_TIM5_CH2

```
#define STM32_DMAMUX1_TIM5_CH2 56
```

Definition at line 88 of file dma.h.

4.14.1.105 STM32_DMAMUX1_TIM5_CH3

```
#define STM32_DMAMUX1_TIM5_CH3 57
```

Definition at line 89 of file dma.h.

4.14.1.106 STM32_DMAMUX1_TIM5_CH4

```
#define STM32_DMAMUX1_TIM5_CH4 58
```

Definition at line 90 of file dma.h.

4.14.1.107 STM32_DMAMUX1_TIM5_TRIG

```
#define STM32_DMAMUX1_TIM5_TRIG 60
```

Definition at line 92 of file dma.h.

4.14.1.108 STM32_DMAMUX1_TIM5_UP

```
#define STM32_DMAMUX1_TIM5_UP 59
```

Definition at line 91 of file dma.h.

4.14.1.109 STM32_DMAMUX1_TIM6_UP

```
#define STM32_DMAMUX1_TIM6_UP 69
```

Definition at line 101 of file dma.h.

4.14.1.110 STM32_DMAMUX1_TIM7_UP

```
#define STM32_DMAMUX1_TIM7_UP 70
```

Definition at line 102 of file dma.h.

4.14.1.111 STM32_DMAMUX1_TIM8_CH1

```
#define STM32_DMAMUX1_TIM8_CH1 47
```

Definition at line 79 of file dma.h.

4.14.1.112 STM32_DMAMUX1_TIM8_CH2

```
#define STM32_DMAMUX1_TIM8_CH2 48
```

Definition at line 80 of file dma.h.

4.14.1.113 STM32_DMAMUX1_TIM8_CH3

```
#define STM32_DMAMUX1_TIM8_CH3 49
```

Definition at line 81 of file dma.h.

4.14.1.114 STM32_DMAMUX1_TIM8_CH4

```
#define STM32_DMAMUX1_TIM8_CH4 50
```

Definition at line 82 of file dma.h.

4.14.1.115 STM32_DMAMUX1_TIM8_COM

```
#define STM32_DMAMUX1_TIM8_COM 53
```

Definition at line 85 of file dma.h.

4.14.1.116 STM32_DMAMUX1_TIM8_TRIG

```
#define STM32_DMAMUX1_TIM8_TRIG 52
```

Definition at line 84 of file dma.h.

4.14.1.117 STM32_DMAMUX1_TIM8_UP

```
#define STM32_DMAMUX1_TIM8_UP 51
```

Definition at line 83 of file dma.h.

4.14.1.118 STM32_DMAMUX1_UART4_RX

```
#define STM32_DMAMUX1_UART4_RX 63
```

Definition at line 95 of file dma.h.

4.14.1.119 STM32_DMAMUX1_UART4_TX

```
#define STM32_DMAMUX1_UART4_TX 64
```

Definition at line 96 of file dma.h.

4.14.1.120 STM32_DMAMUX1_UART5_RX

```
#define STM32_DMAMUX1_UART5_RX 65
```

Definition at line 97 of file dma.h.

4.14.1.121 STM32_DMAMUX1_UART5_TX

```
#define STM32_DMAMUX1_UART5_TX 66
```

Definition at line 98 of file dma.h.

4.14.1.122 STM32_DMAMUX1_UART7_RX

```
#define STM32_DMAMUX1_UART7_RX 79
```

Definition at line 111 of file dma.h.

4.14.1.123 STM32_DMAMUX1_UART7_TX

```
#define STM32_DMAMUX1_UART7_TX 80
```

Definition at line 112 of file dma.h.

4.14.1.124 STM32_DMAMUX1_UART8_RX

```
#define STM32_DMAMUX1_UART8_RX 81
```

Definition at line 113 of file dma.h.

4.14.1.125 STM32_DMAMUX1_UART8_TX

```
#define STM32_DMAMUX1_UART8_TX 82
```

Definition at line 114 of file dma.h.

4.14.1.126 STM32_DMAMUX1_USART2_RX

```
#define STM32_DMAMUX1_USART2_RX 43
```

Definition at line 75 of file dma.h.

4.14.1.127 STM32_DMAMUX1_USART2_TX

```
#define STM32_DMAMUX1_USART2_TX 44
```

Definition at line 76 of file dma.h.

4.14.1.128 STM32_DMAMUX1_USART3_RX

```
#define STM32_DMAMUX1_USART3_RX 45
```

Definition at line 77 of file dma.h.

4.14.1.129 STM32_DMAMUX1_USART3_TX

```
#define STM32_DMAMUX1_USART3_TX 46
```

Definition at line 78 of file dma.h.

4.14.1.130 STM32_DMAMUX1_USART6_RX

```
#define STM32_DMAMUX1_USART6_RX 71
```

Definition at line 103 of file dma.h.

4.14.1.131 STM32_DMAMUX1_USART6_TX

```
#define STM32_DMAMUX1_USART6_TX 72
```

Definition at line 104 of file dma.h.

4.14.2 Typedef Documentation

4.14.2.1 dma_request_t

```
typedef struct dma_request dma_request_t
```

4.14.2.2 dma_scheduler_dev_t

```
typedef struct dma_scheduler_dev dma_scheduler_dev_t
```

4.14.2.3 dma_stream_config_t

```
typedef struct dma_stream_config dma_stream_config_t
```

4.14.2.4 dma_stream_dev_t

```
typedef struct dma_stream_dev dma_stream_dev_t
```

4.14.2.5 dma_stream_dir_t

```
typedef enum dma_stream_dir dma_stream_dir_t
```

4.14.2.6 dma_stream_state_t

```
typedef enum dma_stream_state dma_stream_state_t
```

4.14.3 Enumeration Type Documentation

4.14.3.1 dma_stream_dir

```
enum dma_stream_dir
```

Enumerator

DMA_STREAM_P2M	
DMA_STREAM_M2P	
DMA_STREAM_M2M	

Definition at line 181 of file dma.h.

4.14.3.2 dma_stream_state

```
enum dma_stream_state
```

Enumerator

DMA_STREAM_BUSY	
DMA_STREAM_FREE	

Definition at line 176 of file dma.h.

4.14.4 Function Documentation

4.14.4.1 dma2_get_scheduler()

```
dma_scheduler_dev_t* dma2_get_scheduler (  
    void )
```

Definition at line 182 of file dma.c.

References dma2_scheduler.

4.14.4.2 dma2_get_streams()

```
dma_stream_dev_t* dma2_get_streams (  
    void )
```

Definition at line 186 of file dma.c.

References dma2_streams.

4.14.4.3 dma2_init_scheduler()

```
util_error_t dma2_init_scheduler (
    void )
```

Definition at line 176 of file dma.c.

References dma2_scheduler, dma2_streams, dma_scheduler_init(), and ER_SUCCESS.

Here is the call graph for this function:



4.14.4.4 dma_copy()

```
util_error_t dma_copy (
    void * dst,
    void * src,
    uint32_t len )
```

4.14.4.5 dma_scheduler_init()

```
util_error_t dma_scheduler_init (
    dma_scheduler_dev_t * dma_scheduler,
    dma_stream_dev_t * dma_streams,
    uint16_t nb_dma_streams )
```

Definition at line 195 of file dma.c.

References DMA_STREAM_FREE, DMA_STREAMS_MAX_LEN, ER_OUT_OF_RANGE, ER_SUCCESS, dma_stream_dev::state, and dma_scheduler_dev::streams.

Referenced by dma2_init_scheduler().

Here is the caller graph for this function:



4.14.4.6 dma_scheduler_release_stream()

```
util_error_t dma_scheduler_release_stream (
    dma_scheduler_dev_t * dma_scheduler,
    dma_stream_dev_t * dma_stream )
```

Definition at line 250 of file dma.c.

References DMA_STREAM_FREE, ENTER_CRITICAL, ER_SUCCESS, EXIT_CRITICAL, dma_scheduler_dev↵::free_stream_count, and dma_stream_dev::state.

4.14.4.7 dma_scheduler_request_stream()

```
dma_stream_dev_t* dma_scheduler_request_stream (
    dma_scheduler_dev_t * dma_scheduler )
```

Definition at line 224 of file dma.c.

References DMA_STREAM_BUSY, DMA_STREAM_FREE, ENTER_CRITICAL, EXIT_CRITICAL, dma↵scheduler_dev::free_stream_count, dma_stream_dev::state, dma_scheduler_dev::stream_count, and dma↵scheduler_dev::streams.

4.14.4.8 dma_start_stream()

```
util_error_t dma_start_stream (
    dma_stream_dev_t * stream,
    dma_stream_config_t config )
```

Definition at line 263 of file dma.c.

References dma_stream_config::direction, dma_stream_dev::dma, dma_stream_dev::dma_stream, dma_stream↵_dev::dmamux_channel, dma_stream_config::dmamux_request_number, ER_OUT_OF_RANGE, dma_stream↵_config::m0_addr, dma_stream_config::m1_addr, dma_stream_dev::number, dma_stream_config::p_addr, dma↵_stream_config::peripheral_flow_control, dma_stream_config::priority, dma_stream_config::transfer_cplt, dma↵_stream_dev::transfer_cplt, dma_stream_config::transfer_error, dma_stream_dev::transfer_error, dma_stream↵_config::transfer_half, dma_stream_dev::transfer_half, dma_stream_config::transfer_size, dma_stream_config↵::user_context, dma_stream_dev::user_context, and WRITE_IN_REG.

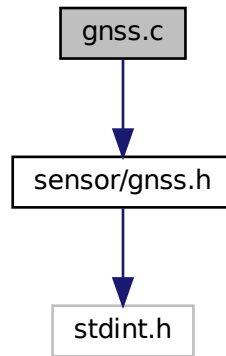
4.14.4.9 dma_stop_stream()

```
util_error_t dma_stop_stream (
    dma_stream_dev_t * stream )
```

4.15 gnss.c File Reference

```
#include <sensor/gnss.h>
```

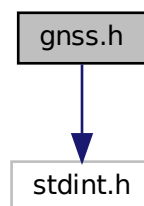
Include dependency graph for gnss.c:



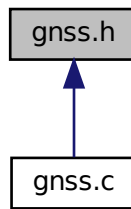
4.16 gnss.h File Reference

```
#include <stdint.h>
```

Include dependency graph for gnss.h:

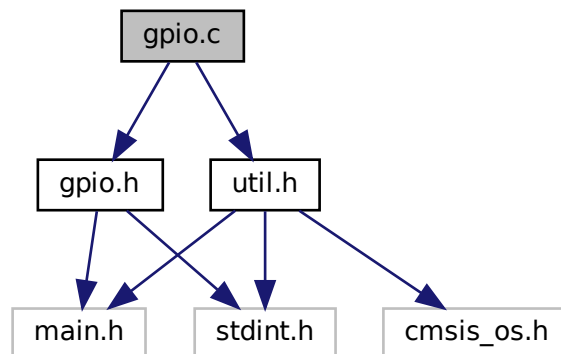


This graph shows which files directly or indirectly include this file:



4.17 gpio.c File Reference

```
#include "gpio.h"  
#include <util.h>  
Include dependency graph for gpio.c:
```



Functions

- uint8_t [gpio_get](#) (GPIO_TypeDef *gpio, uint16_t pin)
- void [gpio_set](#) (GPIO_TypeDef *gpio, uint16_t pin)
- void [gpio_clr](#) (GPIO_TypeDef *gpio, uint16_t pin)
- void [gpio_cfg](#) (GPIO_TypeDef *gpio, uint16_t pins, [gpio_config_t](#) cfg)

4.17.1 Function Documentation

4.17.1.1 gpio_cfg()

```
void gpio_cfg (
    GPIO_TypeDef * gpio,
    uint16_t pins,
    gpio_config_t cfg )
```

Definition at line 59 of file gpio.c.

References `gpio_config::alternate`, `gpio_config::drive`, `gpio_config::mode`, `gpio_config::speed`, and `WRITE_IN_REG`.

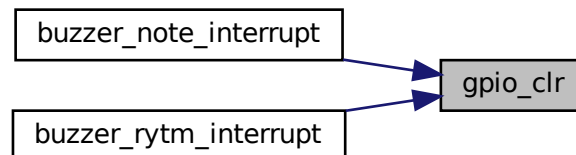
4.17.1.2 gpio_clr()

```
void gpio_clr (
    GPIO_TypeDef * gpio,
    uint16_t pin )
```

Definition at line 55 of file gpio.c.

Referenced by `buzzer_note_interrupt()`, and `buzzer_rytm_interrupt()`.

Here is the caller graph for this function:



4.17.1.3 gpio_get()

```
uint8_t gpio_get (
    GPIO_TypeDef * gpio,
    uint16_t pin )
```

Definition at line 47 of file gpio.c.

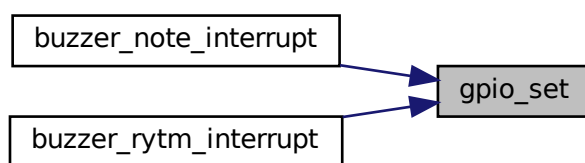
4.17.1.4 gpio_set()

```
void gpio_set (
    GPIO_TypeDef * gpio,
    uint16_t pin )
```

Definition at line 51 of file gpio.c.

Referenced by `buzzer_note_interrupt()`, and `buzzer_rytm_interrupt()`.

Here is the caller graph for this function:

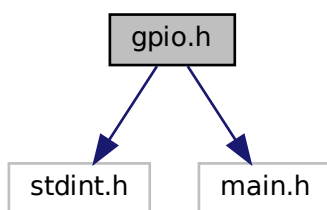


4.18 gpio.h File Reference

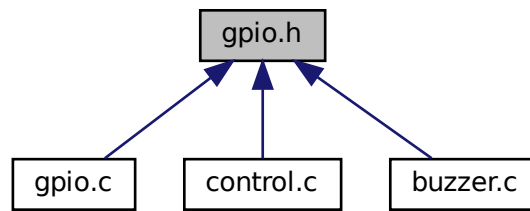
```
#include <stdint.h>
```

```
#include <main.h>
```

Include dependency graph for gpio.h:



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



Data Structures

- struct [gpio_config](#)

Typedefs

- typedef enum [gpio_drive](#) [gpio_drive_t](#)
- typedef enum [gpio_mode](#) [gpio_mode_t](#)
- typedef enum [gpio_bias](#) [gpio_bias_t](#)
- typedef struct [gpio_config](#) [gpio_config_t](#)

Enumerations

- enum [gpio_drive](#) { [GPIO_DRIVE_PP](#) = 0b0 , [GPIO_DRIVE_OD](#) = 0b1 }
- enum [gpio_mode](#) { [GPIO_MODE_IN](#) = 0b00 , [GPIO_MODE_OUT](#) = 0b01 , [GPIO_MODE_ALT](#) = 0b10 , [GPIO_MODE_ANA](#) = 0b11 }
- enum [gpio_bias](#) { [GPIO_BIAS_NONE](#) = 0b00 , [GPIO_BIAS_HIGH](#) = 0b01 , [GPIO_BIAS_LOW](#) = 0b10 }

Functions

- void [gpio_set](#) (GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
- void [gpio_clr](#) (GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
- uint8_t [gpio_get](#) (GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
- void [gpio_cfg](#) (GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, [gpio_config_t](#) cfg)

4.18.1 Typedef Documentation

4.18.1.1 [gpio_bias_t](#)

```
typedef enum gpio\_bias gpio\_bias\_t
```

4.18.1.2 gpio_config_t

```
typedef struct gpio_config gpio_config_t
```

4.18.1.3 gpio_drive_t

```
typedef enum gpio_drive gpio_drive_t
```

4.18.1.4 gpio_mode_t

```
typedef enum gpio_mode gpio_mode_t
```

4.18.2 Enumeration Type Documentation

4.18.2.1 gpio_bias

```
enum gpio_bias
```

Enumerator

GPIO_BIAS_NONE	
GPIO_BIAS_HIGH	
GPIO_BIAS_LOW	

Definition at line 48 of file gpio.h.

4.18.2.2 gpio_drive

```
enum gpio_drive
```

Enumerator

GPIO_DRIVE_PP	
GPIO_DRIVE_OD	

Definition at line 36 of file gpio.h.

4.18.2.3 gpio_mode

enum `gpio_mode`

Enumerator

GPIO_MODE_IN	
GPIO_MODE_OUT	
GPIO_MODE_ALT	
GPIO_MODE_ANA	

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

4.18.3 Function Documentation

4.18.3.1 gpio_cfg()

```
void gpio_cfg (  
    GPIO_TypeDef * GPIOx,  
    uint16_t GPIO_Pin,  
    gpio_config_t cfg )
```

Definition at line 59 of file `gpio.c`.

References `gpio_config::alternate`, `gpio_config::drive`, `gpio_config::mode`, `gpio_config::speed`, and `WRITE_IN_REG`.

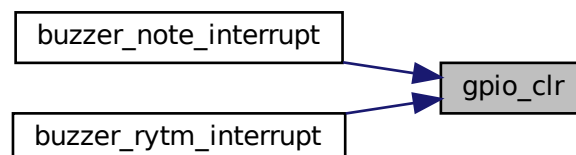
4.18.3.2 gpio_clr()

```
void gpio_clr (  
    GPIO_TypeDef * GPIOx,  
    uint16_t GPIO_Pin )
```

Definition at line 55 of file `gpio.c`.

Referenced by `buzzer_note_interrupt()`, and `buzzer_rytm_interrupt()`.

Here is the caller graph for this function:



4.18.3.3 gpio_get()

```
uint8_t gpio_get (
    GPIO_TypeDef * GPIOx,
    uint16_t GPIO_Pin )
```

Definition at line 47 of file gpio.c.

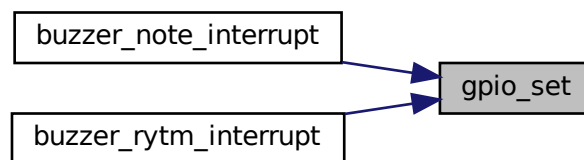
4.18.3.4 gpio_set()

```
void gpio_set (
    GPIO_TypeDef * GPIOx,
    uint16_t GPIO_Pin )
```

Definition at line 51 of file gpio.c.

Referenced by buzzer_note_interrupt(), and buzzer_rytm_interrupt().

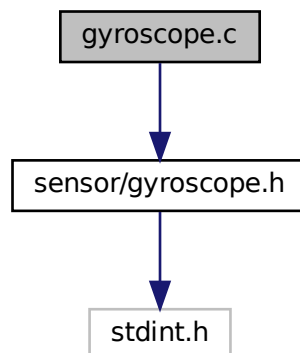
Here is the caller graph for this function:



4.19 gyroscope.c File Reference

```
#include <sensor/gyroscope.h>
```

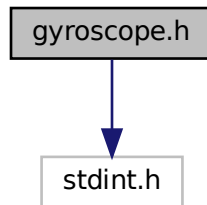
Include dependency graph for gyroscope.c:



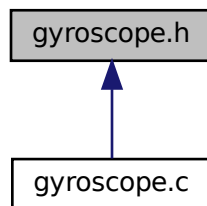
4.20 gyroscope.h File Reference

```
#include <stdint.h>
```

Include dependency graph for gyroscope.h:



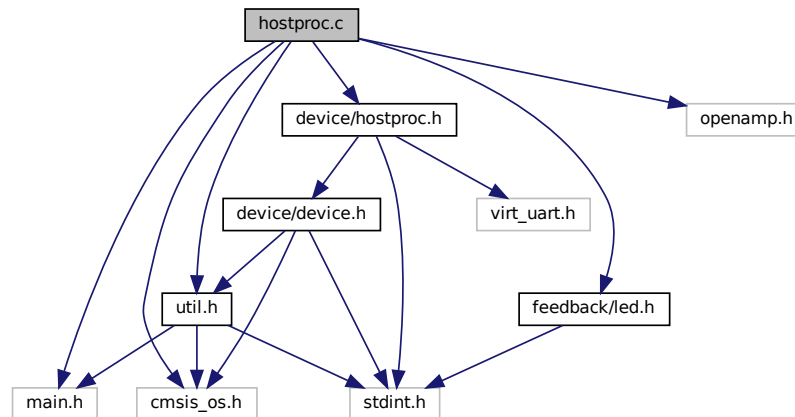
This graph shows which files directly or indirectly include this file:



4.21 hostproc.c File Reference

```
#include <main.h>
#include <cmsis_os.h>
#include <device/hostproc.h>
#include <feedback/led.h>
#include <openamp.h>
#include <util.h>
```

Include dependency graph for hostproc.c:



Data Structures

- struct [hostproc_interface_context](#)

Typedefs

- typedef struct [hostproc_interface_context](#) [hostproc_interface_context_t](#)

Functions

- void [host_UART0_RX](#) (VIRT_UART_HandleTypeDef *huart)
- [util_error_t](#) [host_send](#) (void *context, uint8_t *data, uint32_t len)
- [util_error_t](#) [host_rcv](#) (void *context, uint8_t *data, uint32_t *len)
- [device_interface_t](#) * [hostproc_get_interface](#) (void)
- [device_t](#) * [hostproc_get_device](#) (void)
- [util_error_t](#) [hostproc_init](#) (void)

Variables

- static VIRT_UART_HandleTypeDef [host_UART0](#)
- static [device_t](#) [hostproc_device](#)
- static [device_interface_t](#) [hostproc_interface](#)
- static [hostproc_interface_context_t](#) [hostproc_interface_context](#)

4.21.1 Typedef Documentation

4.21.1.1 `hostproc_interface_context_t`

```
typedef struct hostproc_interface_context hostproc_interface_context_t
```

4.21.2 Function Documentation

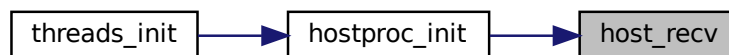
4.21.2.1 `host_rcv()`

```
util_error_t host_rcv (  
    void * context,  
    uint8_t * data,  
    uint32_t * len )
```

Definition at line 96 of file `hostproc.c`.

Referenced by `hostproc_init()`.

Here is the caller graph for this function:



4.21.2.2 `host_send()`

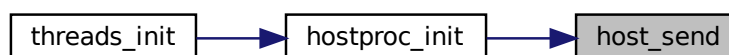
```
util_error_t host_send (  
    void * context,  
    uint8_t * data,  
    uint32_t len )
```

Definition at line 86 of file `hostproc.c`.

References `ER_SUCCESS`, `hostproc_interface_context::rx_once`, and `hostproc_interface_context::uart`.

Referenced by `hostproc_init()`.

Here is the caller graph for this function:



4.21.2.3 host_UART0_RX()

```
void host_UART0_RX (
    VIRT_UART_HandleTypeDef * huart )
```

Definition at line 81 of file hostproc.c.

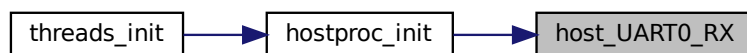
References `led_rgb_set_rgb()`, and `hostproc_interface_context::rx_once`.

Referenced by `hostproc_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.21.2.4 hostproc_get_device()

```
device_t* hostproc_get_device (
    void )
```

Definition at line 76 of file hostproc.c.

References `hostproc_device`.

4.21.2.5 hostproc_get_interface()

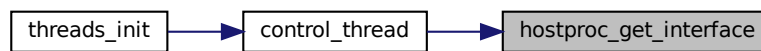
```
device_interface_t* hostproc_get_interface (
    void )
```

Definition at line 72 of file hostproc.c.

References hostproc_interface.

Referenced by control_thread().

Here is the caller graph for this function:



4.21.2.6 hostproc_init()

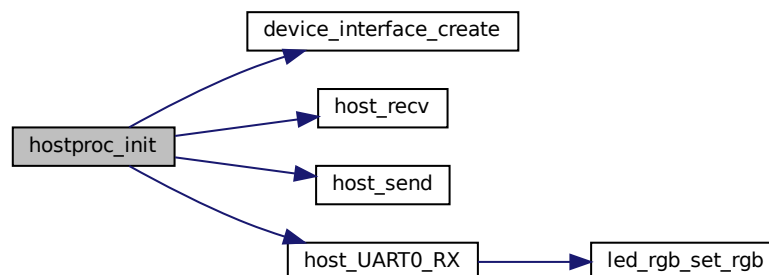
```
util_error_t hostproc_init (
    void )
```

Definition at line 101 of file hostproc.c.

References device_interface_create(), ER_FAILURE, ER_SUCCESS, host_rcv(), host_send(), host_UART0, host_UART0_RX(), hostproc_interface, hostproc_interface_context::rx_once, and hostproc_interface_context::uart.

Referenced by threads_init().

Here is the call graph for this function:



Here is the caller graph for this function:



4.21.3 Variable Documentation

4.21.3.1 host_UART0

```
VIRT_UART_HandleTypeDef host_UART0 [static]
```

Definition at line 50 of file `hostproc.c`.

Referenced by `hostproc_init()`.

4.21.3.2 hostproc_device

```
device_t hostproc_device [static]
```

Definition at line 53 of file `hostproc.c`.

Referenced by `hostproc_get_device()`.

4.21.3.3 hostproc_interface

```
device_interface_t hostproc_interface [static]
```

Definition at line 54 of file `hostproc.c`.

Referenced by `control_thread()`, `hostproc_get_interface()`, and `hostproc_init()`.

4.21.3.4 hostproc_interface_context

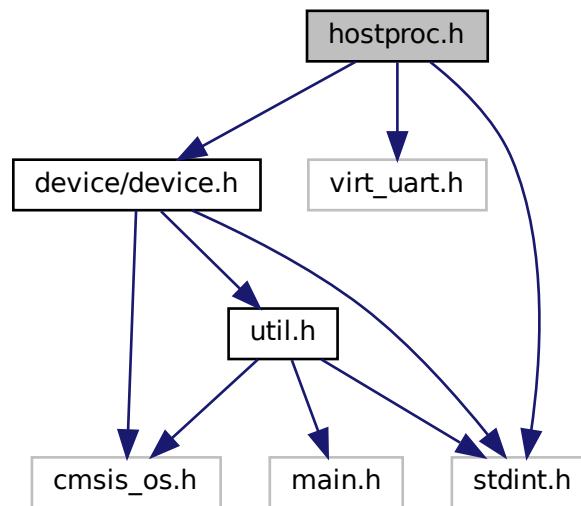
```
hostproc_interface_context_t hostproc_interface_context [static]
```

Definition at line 55 of file `hostproc.c`.

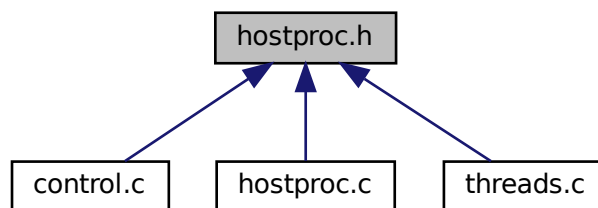
4.22 hostproc.h File Reference

```
#include <stdint.h>
#include <virt_uart.h>
#include <device/device.h>
```

Include dependency graph for hostproc.h:



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



Functions

- `device_interface_t * hostproc_get_interface` (void)
- `device_t * hostproc_get_device` (void)
- `util_error_t hostproc_init` (void)

4.22.1 Function Documentation

4.22.1.1 hostproc_get_device()

```
device_t* hostproc_get_device (  
    void )
```

Definition at line 76 of file hostproc.c.

References hostproc_device.

4.22.1.2 hostproc_get_interface()

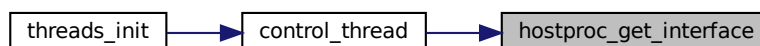
```
device_interface_t* hostproc_get_interface (  
    void )
```

Definition at line 72 of file hostproc.c.

References hostproc_interface.

Referenced by control_thread().

Here is the caller graph for this function:



4.22.1.3 hostproc_init()

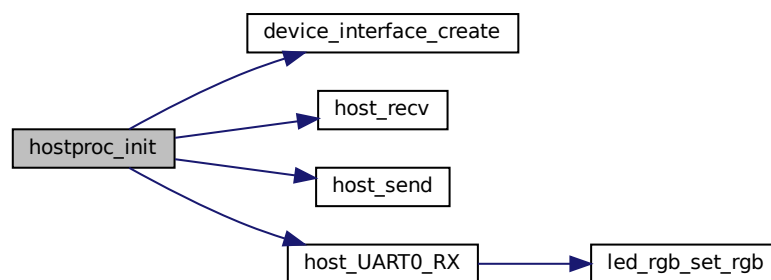
```
util_error_t hostproc_init (
    void )
```

Definition at line 101 of file hostproc.c.

References `device_interface_create()`, `ER_FAILURE`, `ER_SUCCESS`, `host_rcv()`, `host_send()`, `host_UART0`, `host_UART0_RX()`, `hostproc_interface`, `hostproc_interface_context::rx_once`, and `hostproc_interface_context::uart`.

Referenced by `threads_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:

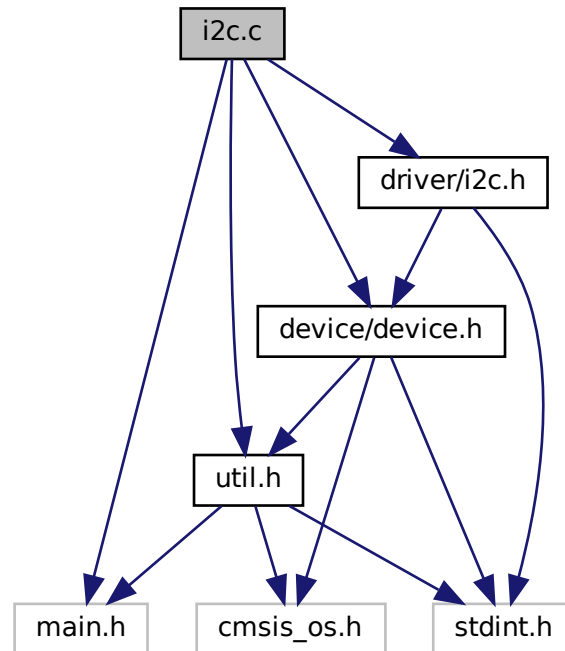


4.23 i2c.c File Reference

```
#include <main.h>
#include <driver/i2c.h>
#include <device/device.h>
```

```
#include <util.h>
```

Include dependency graph for i2c.c:



Macros

- `#define S1_I2C` hi2c1
- `#define S2_I2C` hi2c2
- `#define S3_I2C` hi2c5

Functions

- `void i2c_spi_guard` (void)
- `device_interface_t * i2c_get_sensor_interface` (void)
- `void i2c_init` (void)

Variables

- `device_interface_t sensor_interface`
- `i2c_interface_context_t sensor_interface_context`

4.23.1 Macro Definition Documentation

4.23.1.1 S1_I2C

```
#define S1_I2C hi2c1
```

Definition at line 23 of file i2c.c.

4.23.1.2 S2_I2C

```
#define S2_I2C hi2c2
```

Definition at line 24 of file i2c.c.

4.23.1.3 S3_I2C

```
#define S3_I2C hi2c5
```

Definition at line 25 of file i2c.c.

4.23.2 Function Documentation

4.23.2.1 i2c_get_sensor_interface()

```
device_interface_t* i2c_get_sensor_interface (  
    void )
```

Definition at line 81 of file i2c.c.

References sensor_interface.

Referenced by i2c_sensor_init().

Here is the caller graph for this function:



4.23.2.2 i2c_init()

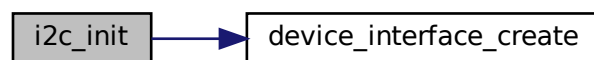
```
void i2c_init (  
    void )
```

Definition at line 85 of file i2c.c.

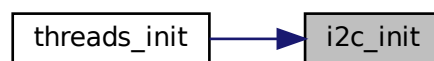
References `device_interface_create()`, `ER_SUCCESS`, `sensor_interface`, and `sensor_interface_context`.

Referenced by `threads_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.23.2.3 i2c_spi_guard()

```
void i2c_spi_guard (  
    void )
```

Definition at line 58 of file i2c.c.

Referenced by `threads_init()`.

Here is the caller graph for this function:



4.23.3 Variable Documentation

4.23.3.1 sensor_interface

`device_interface_t` sensor_interface

Definition at line 42 of file i2c.c.

Referenced by `i2c_get_sensor_interface()`, and `i2c_init()`.

4.23.3.2 sensor_interface_context

`i2c_interface_context_t` sensor_interface_context

Initial value:

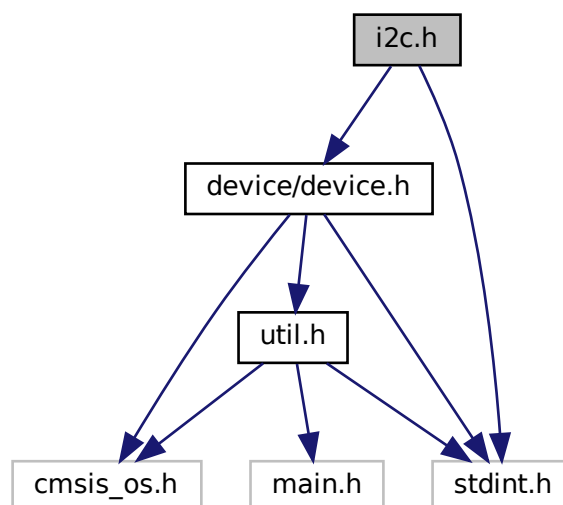
```
= {  
    .i2c = &S2_I2C  
}
```

Definition at line 44 of file i2c.c.

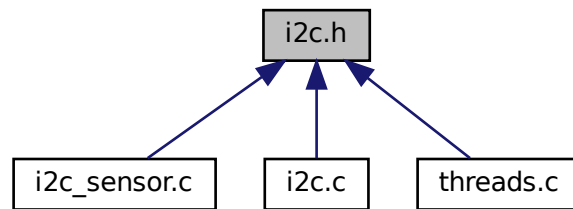
Referenced by `i2c_init()`.

4.24 i2c.h File Reference

```
#include <stdint.h>  
#include <device/device.h>  
Include dependency graph for i2c.h:
```



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



Data Structures

- struct [i2c_interface_context](#)

Typedefs

- typedef struct [i2c_interface_context](#) [i2c_interface_context_t](#)

Functions

- void [i2c_init](#) (void)
- void [i2c_spi_guard](#) (void)
- [device_interface_t](#) * [i2c_get_sensor_interface](#) (void)

4.24.1 Typedef Documentation

4.24.1.1 i2c_interface_context_t

```
typedef struct i2c\_interface\_context i2c\_interface\_context\_t
```

4.24.2 Function Documentation

4.24.2.1 i2c_get_sensor_interface()

```
device_interface_t* i2c_get_sensor_interface (  
    void )
```

Definition at line 81 of file i2c.c.

References sensor_interface.

Referenced by i2c_sensor_init().

Here is the caller graph for this function:



4.24.2.2 i2c_init()

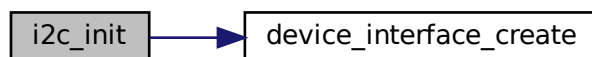
```
void i2c_init (  
    void )
```

Definition at line 85 of file i2c.c.

References device_interface_create(), ER_SUCCESS, sensor_interface, and sensor_interface_context.

Referenced by threads_init().

Here is the call graph for this function:



Here is the caller graph for this function:



4.24.2.3 i2c_spi_guard()

```
void i2c_spi_guard (  
    void )
```

Definition at line 58 of file i2c.c.

Referenced by threads_init().

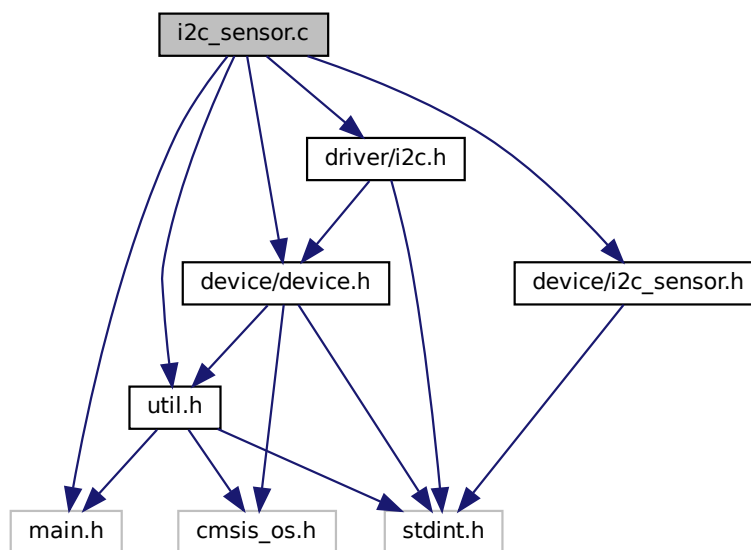
Here is the caller graph for this function:



4.25 i2c_sensor.c File Reference

```
#include <main.h>  
#include <device/device.h>  
#include <device/i2c_sensor.h>  
#include <driver/i2c.h>  
#include <util.h>
```

Include dependency graph for i2c_sensor.c:



Data Structures

- struct [i2c_sensor_context](#)

Typedefs

- typedef struct [i2c_sensor_context](#) [i2c_sensor_context_t](#)

Functions

- [util_error_t read_reg](#) (void *context, [device_interface_t](#) *dev, uint32_t address, uint8_t *data, uint32_t data↵_len)
- [util_error_t write_reg](#) (void *context, [device_interface_t](#) *dev, uint32_t address, uint8_t *data, uint32_t data↵_len)
- [device_t * i2c_get_accelerometer](#) (void)
- [util_error_t i2c_sensor_init](#) (void)

Variables

- static [device_t i2c_accelerometer_device](#)
- static [device_t i2c_gyroscope_device](#)
- static [device_t i2c_barometer_device](#)
- static [i2c_sensor_context_t i2c_accelerometer_device_context](#)
- static [i2c_sensor_context_t i2c_gyroscope_device_context](#)
- static [i2c_sensor_context_t i2c_barometer_device_context](#)

4.25.1 Typedef Documentation

4.25.1.1 [i2c_sensor_context_t](#)

```
typedef struct i2c\_sensor\_context i2c\_sensor\_context\_t
```

4.25.2 Function Documentation

4.25.2.1 [i2c_get_accelerometer\(\)](#)

```
device\_t\* i2c\_get\_accelerometer (  
    void )
```

Definition at line 72 of file [i2c_sensor.c](#).

References [i2c_accelerometer_device](#).

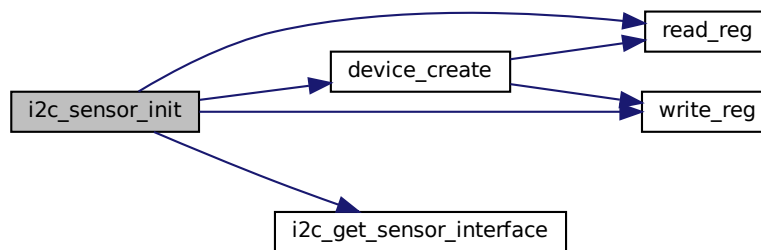
4.25.2.2 i2c_sensor_init()

```
util_error_t i2c_sensor_init (
    void )
```

Definition at line 76 of file i2c_sensor.c.

References `device_create()`, `ER_SUCCESS`, `i2c_accelerometer_device`, `i2c_accelerometer_device_context`, `i2c_get_sensor_interface()`, `read_reg()`, and `write_reg()`.

Here is the call graph for this function:



4.25.2.3 read_reg()

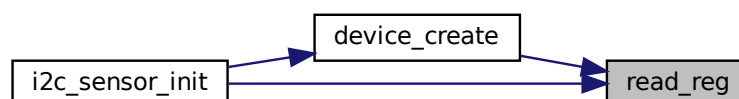
```
util_error_t read_reg (
    void * context,
    device_interface_t * dev,
    uint32_t address,
    uint8_t * data,
    uint32_t data_len )
```

Definition at line 87 of file i2c_sensor.c.

References `device_interface::context`, `ER_SUCCESS`, and `i2c_interface_context::i2c`.

Referenced by `device_create()`, and `i2c_sensor_init()`.

Here is the caller graph for this function:



4.25.2.4 write_reg()

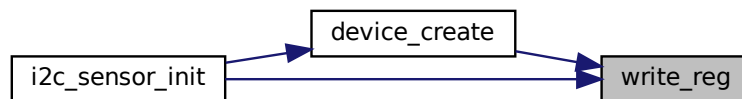
```
util_error_t write_reg (
    void * context,
    device_interface_t * dev,
    uint32_t address,
    uint8_t * data,
    uint32_t data_len )
```

Definition at line 95 of file i2c_sensor.c.

References device_interface::context, ER_SUCCESS, and i2c_interface_context::i2c.

Referenced by device_create(), and i2c_sensor_init().

Here is the caller graph for this function:



4.25.3 Variable Documentation

4.25.3.1 i2c_accelerometer_device

```
device_t i2c_accelerometer_device [static]
```

Definition at line 43 of file i2c_sensor.c.

Referenced by i2c_get_accelerometer(), and i2c_sensor_init().

4.25.3.2 i2c_accelerometer_device_context

```
i2c_sensor_context_t i2c_accelerometer_device_context [static]
```

Initial value:

```
= {
    .device_address = 0x68
}
```

Definition at line 47 of file i2c_sensor.c.

Referenced by i2c_sensor_init().

4.25.3.3 i2c_barometer_device

```
device_t i2c_barometer_device [static]
```

Definition at line 45 of file i2c_sensor.c.

4.25.3.4 i2c_barometer_device_context

```
i2c_sensor_context_t i2c_barometer_device_context [static]
```

Initial value:

```
= {  
    .device_address = 0x18  
}
```

Definition at line 55 of file i2c_sensor.c.

4.25.3.5 i2c_gyroscope_device

```
device_t i2c_gyroscope_device [static]
```

Definition at line 44 of file i2c_sensor.c.

4.25.3.6 i2c_gyroscope_device_context

```
i2c_sensor_context_t i2c_gyroscope_device_context [static]
```

Initial value:

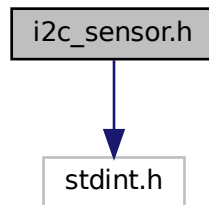
```
= {  
    .device_address = 0x68  
}
```

Definition at line 51 of file i2c_sensor.c.

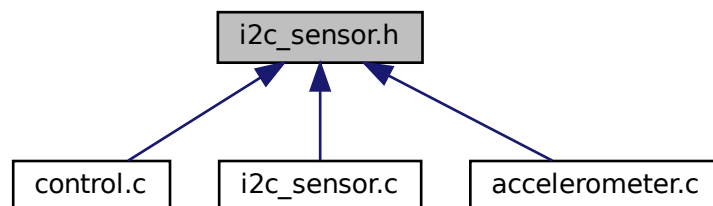
4.26 i2c_sensor.h File Reference

```
#include <stdint.h>
```

Include dependency graph for i2c_sensor.h:



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



Functions

- [util_error_t i2c_sensor_init](#) (void)

4.26.1 Function Documentation

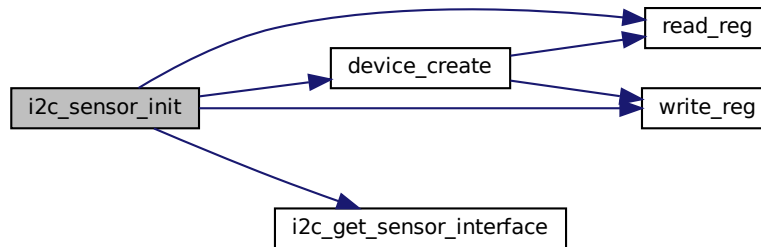
4.26.1.1 i2c_sensor_init()

```
util_error_t i2c_sensor_init (  
    void )
```

Definition at line 76 of file i2c_sensor.c.

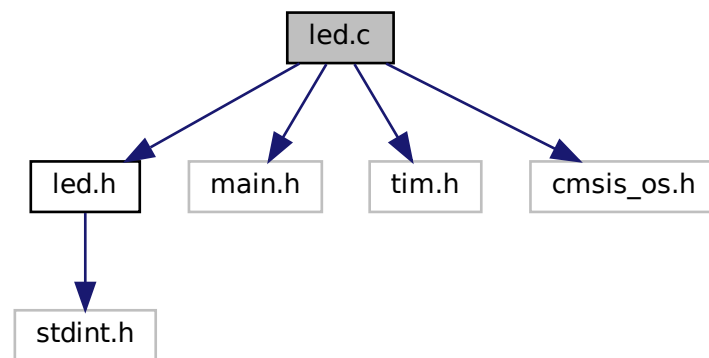
References `device_create()`, `ER_SUCCESS`, `i2c_accelerometer_device`, `i2c_accelerometer_device_context`, `i2c_get_sensor_interface()`, `read_reg()`, and `write_reg()`.

Here is the call graph for this function:



4.27 led.c File Reference

```
#include "led.h"
#include <main.h>
#include <tim.h>
#include <cmsis_os.h>
Include dependency graph for led.c:
```



Macros

- `#define LED_TIM` `htim3`
- `#define LED_MAX` `(0xff)`

Typedefs

- typedef enum [led_blick_state](#) [led_blink_state_t](#)

Enumerations

- enum [led_blick_state](#) { [LED_ON](#) , [LED_FAINT](#) , [LED_OFF](#) }

Functions

- void [led_feedback_init](#) (void)
- void [led_rgb_init](#) (void)
- void [led_rgb_set_rgb](#) (uint8_t r, uint8_t g, uint8_t b)
- void [led_rgb_set_color](#) ([led_color_t](#) color)
- void [led_rgb_thread](#) (__attribute__((unused)) void *arg)

Variables

- static [led_blink_state_t](#) [blink_sequence](#) []
- static const int [blink_sequence_len](#) = sizeof([blink_sequence](#))/sizeof([led_blink_state_t](#))
- static [led_color_t](#) [color_sequence](#) []
- static const int [color_sequence_len](#) = sizeof([color_sequence](#))/sizeof([led_color_t](#))

4.27.1 Macro Definition Documentation

4.27.1.1 LED_MAX

```
#define LED_MAX (0xff)
```

Definition at line 31 of file [led.c](#).

4.27.1.2 LED_TIM

```
#define LED_TIM htim3
```

Definition at line 22 of file [led.c](#).

4.27.2 Typedef Documentation

4.27.2.1 led_blink_state_t

```
typedef enum led\_blick\_state led\_blink\_state\_t
```

4.27.3 Enumeration Type Documentation

4.27.3.1 led_blick_state

```
enum led\_blick\_state
```


Enumerator

LED_ON	
LED_FAINT	
LED_OFF	

Definition at line 44 of file led.c.

4.27.4 Function Documentation

4.27.4.1 led_feedback_init()

```
void led_feedback_init (  
    void )
```

Definition at line 88 of file led.c.

Referenced by threads_init().

Here is the caller graph for this function:



4.27.4.2 led_rgb_init()

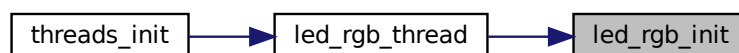
```
void led_rgb_init (  
    void )
```

Definition at line 104 of file led.c.

References LED_MAX, and LED_TIM.

Referenced by led_rgb_thread().

Here is the caller graph for this function:



4.27.4.3 led_rgb_set_color()

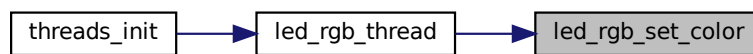
```
void led_rgb_set_color (
    led_color_t color )
```

Definition at line 139 of file led.c.

References `led_color::b`, `led_color::g`, `LED_TIM`, and `led_color::r`.

Referenced by `led_rgb_thread()`.

Here is the caller graph for this function:



4.27.4.4 led_rgb_set_rgb()

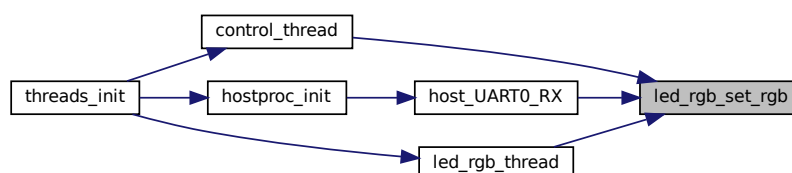
```
void led_rgb_set_rgb (
    uint8_t r,
    uint8_t g,
    uint8_t b )
```

Definition at line 133 of file led.c.

References `LED_TIM`.

Referenced by `control_thread()`, `host_UART0_RX()`, and `led_rgb_thread()`.

Here is the caller graph for this function:



4.27.4.5 led_rgb_thread()

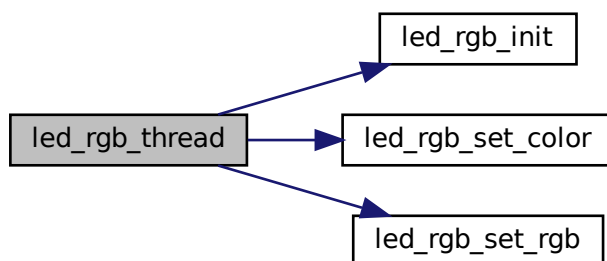
```
void led_rgb_thread (
    __attribute__((unused)) void * arg )
```

Definition at line 145 of file led.c.

References `blink_sequence`, `blink_sequence_len`, `color_sequence`, `color_sequence_len`, `LED_BLACK`, `led_blue`, `LED_FAINT`, `LED_OFF`, `LED_ON`, `led_rgb_init()`, `led_rgb_set_color()`, and `led_rgb_set_rgb()`.

Referenced by `threads_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.5 Variable Documentation

4.27.5.1 blink_sequence

```
led_blink_state_t blink_sequence[] [static]
```

Initial value:

```
= {  
    LED_ON,  
    LED_FAINT,  
    LED_ON,  
    LED_FAINT,  
    LED_ON,  
    LED_OFF  
}
```

Definition at line 57 of file led.c.

Referenced by led_rgb_thread().

4.27.5.2 blink_sequence_len

```
const int blink_sequence_len = sizeof(blink_sequence)/sizeof(led_blink_state_t) [static]
```

Definition at line 66 of file led.c.

Referenced by led_rgb_thread().

4.27.5.3 color_sequence

```
led_color_t color_sequence[] [static]
```

Initial value:

```
= {  
    led_green,  
    led_red,  
    led_blue,  
    led_red,  
}
```

Definition at line 68 of file led.c.

Referenced by led_rgb_thread().

4.27.5.4 color_sequence_len

```
const int color_sequence_len = sizeof(color_sequence)/sizeof(led_color_t) [static]
```

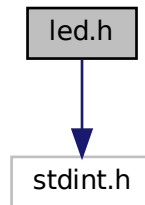
Definition at line 75 of file led.c.

Referenced by led_rgb_thread().

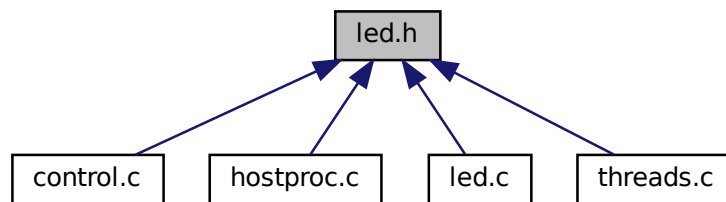
4.28 led.h File Reference

```
#include <stdint.h>
```

Include dependency graph for led.h:



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



Data Structures

- struct [led_color](#)

Macros

- #define [LED_RED](#) 0xff, 0x00, 0x00
- #define [LED_GREEN](#) 0x00, 0xff, 0x00
- #define [LED_BLUE](#) 0x00, 0x00, 0xff
- #define [LED_ORANGE](#) 0x7f, 0x0f, 0x07
- #define [LED_YELLOW](#) 0xff, 0x1f, 0x07
- #define [LED_TEAL](#) 0x00, 0x7f, 0x7f
- #define [LED_PINK](#) 0x7f, 0x00, 0x7f
- #define [LED_LILA](#) 0xff, 0x03, 0x4f
- #define [LED_BLACK](#) 0x00, 0x00, 0x00
- #define [LED_WHITE](#) 0xff, 0xff, 0xff

Typedefs

- typedef struct [led_color](#) [led_color_t](#)

Functions

- void [led_rgb_init](#) (void)
- void [led_rgb_set_color](#) ([led_color_t](#) color)
- void [led_rgb_set_rgb](#) (uint8_t r, uint8_t g, uint8_t b)
- void [led_feedback_init](#) (void)
- void [led_rgb_thread](#) (void *arg)

Variables

- static const [led_color_t](#) [led_red](#)
- static const [led_color_t](#) [led_green](#)
- static const [led_color_t](#) [led_blue](#)
- static const [led_color_t](#) [led_black](#)

4.28.1 Macro Definition Documentation

4.28.1.1 LED_BLACK

```
#define LED_BLACK 0x00, 0x00, 0x00
```

Definition at line 34 of file led.h.

4.28.1.2 LED_BLUE

```
#define LED_BLUE 0x00, 0x00, 0xff
```

Definition at line 26 of file led.h.

4.28.1.3 LED_GREEN

```
#define LED_GREEN 0x00, 0xff, 0x00
```

Definition at line 25 of file led.h.

4.28.1.4 LED_LILA

```
#define LED_LILA 0xff, 0x03, 0x4f
```

Definition at line 32 of file led.h.

4.28.1.5 LED_ORANGE

```
#define LED_ORANGE 0x7f, 0x0f, 0x07
```

Definition at line 28 of file led.h.

4.28.1.6 LED_PINK

```
#define LED_PINK 0x7f, 0x00, 0x7f
```

Definition at line 31 of file led.h.

4.28.1.7 LED_RED

```
#define LED_RED 0xff, 0x00, 0x00
```

Definition at line 24 of file led.h.

4.28.1.8 LED_TEAL

```
#define LED_TEAL 0x00, 0x7f, 0x7f
```

Definition at line 30 of file led.h.

4.28.1.9 LED_WHITE

```
#define LED_WHITE 0xff, 0xff, 0xff
```

Definition at line 35 of file led.h.

4.28.1.10 LED_YELLOW

```
#define LED_YELLOW 0xff, 0x1f, 0x07
```

Definition at line 29 of file led.h.

4.28.2 Typedef Documentation

4.28.2.1 led_color_t

```
typedef struct led_color led_color_t
```

4.28.3 Function Documentation

4.28.3.1 led_feedback_init()

```
void led_feedback_init (  
    void )
```

Definition at line 88 of file led.c.

Referenced by threads_init().

Here is the caller graph for this function:



4.28.3.2 led_rgb_init()

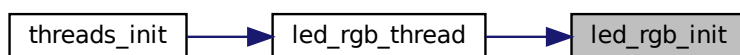
```
void led_rgb_init (
    void )
```

Definition at line 104 of file led.c.

References LED_MAX, and LED_TIM.

Referenced by led_rgb_thread().

Here is the caller graph for this function:



4.28.3.3 led_rgb_set_color()

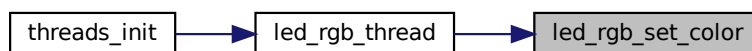
```
void led_rgb_set_color (
    led_color_t color )
```

Definition at line 139 of file led.c.

References led_color::b, led_color::g, LED_TIM, and led_color::r.

Referenced by led_rgb_thread().

Here is the caller graph for this function:



4.28.3.4 led_rgb_set_rgb()

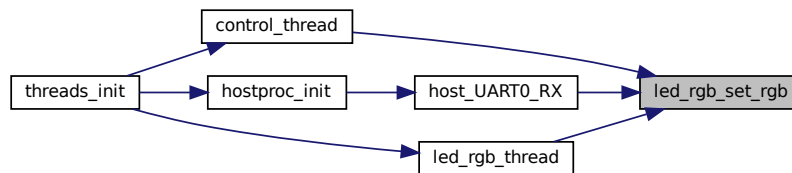
```
void led_rgb_set_rgb (
    uint8_t r,
    uint8_t g,
    uint8_t b )
```

Definition at line 133 of file led.c.

References LED_TIM.

Referenced by control_thread(), host_UART0_RX(), and led_rgb_thread().

Here is the caller graph for this function:



4.28.3.5 led_rgb_thread()

```
void led_rgb_thread (
    void * arg )
```

4.28.4 Variable Documentation

4.28.4.1 led_black

```
const led_color_t led_black [static]
```

Initial value:

```
= {
    .r = 0x00,
    .g = 0x00,
    .b = 0x00
}
```

Definition at line 80 of file led.h.

4.28.4.2 led_blue

```
const led_color_t led_blue [static]
```

Initial value:

```
= {  
    .r = 0x00,  
    .g = 0x00,  
    .b = 0xff  
}
```

Definition at line 74 of file led.h.

Referenced by led_rgb_thread().

4.28.4.3 led_green

```
const led_color_t led_green [static]
```

Initial value:

```
= {  
    .r = 0x00,  
    .g = 0xff,  
    .b = 0x00  
}
```

Definition at line 68 of file led.h.

4.28.4.4 led_red

```
const led_color_t led_red [static]
```

Initial value:

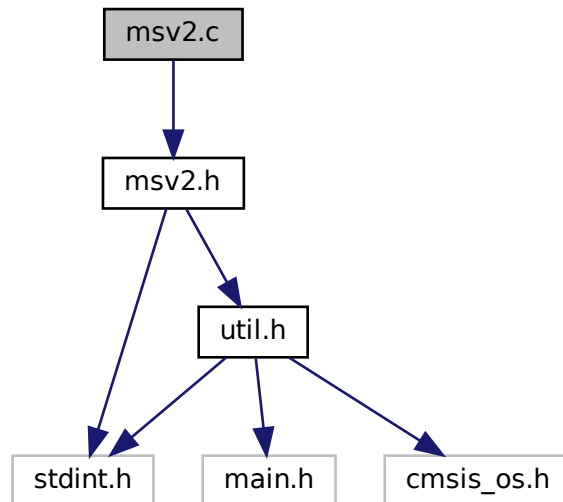
```
= {  
    .r = 0xff,  
    .g = 0x00,  
    .b = 0x00  
}
```

Definition at line 62 of file led.h.

4.29 msv2.c File Reference

```
#include "msv2.h"
```

Include dependency graph for msv2.c:



Macros

- `#define DLE` (0x90)
- `#define STX` (0x02)

Functions

- static uint16_t `calc_field_CRC` (uint16_t *p_data_array, uint16_t length)
- void `msv2_init` (MSV2_INST_t *msv2)
- uint16_t `msv2_create_frame` (MSV2_INST_t *msv2, uint8_t opcode, uint8_t data_len, uint8_t *data)
- MSV2_ERROR_t `msv2_decode_fragment` (MSV2_INST_t *msv2, uint8_t d)
- uint8_t * `msv2_rx_data` (MSV2_INST_t *msv2)
- uint8_t * `msv2_tx_data` (MSV2_INST_t *msv2)

4.29.1 Macro Definition Documentation

4.29.1.1 DLE

```
#define DLE (0x90)
```

Definition at line 26 of file msv2.c.

4.29.1.2 STX

```
#define STX (0x02)
```

Definition at line 27 of file msv2.c.

4.29.2 Function Documentation

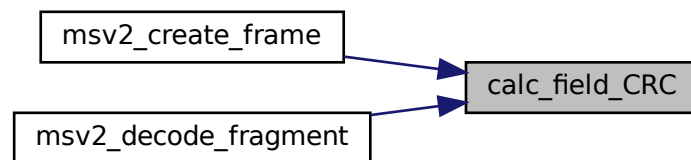
4.29.2.1 calc_field_CRC()

```
uint16_t calc_field_CRC (  
    uint16_t * p_data_array,  
    uint16_t length ) [static]
```

Definition at line 59 of file msv2.c.

Referenced by msv2_create_frame(), and msv2_decode_fragment().

Here is the caller graph for this function:



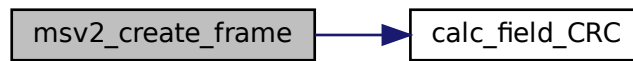
4.29.2.2 msv2_create_frame()

```
uint16_t msv2_create_frame (  
    MSV2_INST_t * msv2,  
    uint8_t opcode,  
    uint8_t data_len,  
    uint8_t * data )
```

Definition at line 84 of file msv2.c.

References `calc_field_CRC()`, `MSV2_TX_DATA::crc_data`, `MSV2_TX_DATA::data`, `MSV2_TX_DATA::data_len`, `DLE`, `MSV2_TX_DATA::opcode`, `STX`, and `MSV2_INST::tx`.

Here is the call graph for this function:



4.29.2.3 msv2_decode_fragment()

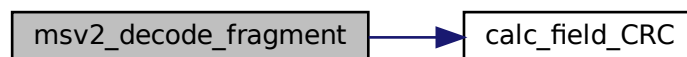
```

MSV2_ERROR_t msv2_decode_fragment (
    MSV2_INST_t * msv2,
    uint8_t d )
  
```

Definition at line 121 of file msv2.c.

References `calc_field_CRC()`, `MSV2_RX_DATA::counter`, `MSV2_RX_DATA::crc`, `MSV2_RX_DATA::crc_data`, `MSV2_RX_DATA::data`, `MSV2_RX_DATA::data_len`, `DLE`, `MSV2_RX_DATA::escape`, `MSV2_RX_DATA::length`, `MSV2_PROGRESS`, `MSV2_SUCCESS`, `MSV2_WRONG_CRC`, `MSV2_RX_DATA::opcode`, `MSV2_INST::rx`, `MSV2_RX_DATA::state`, `STX`, `WAITING_CRC1`, `WAITING_CRC2`, `WAITING_DATA`, `WAITING_DLE`, `WAITING_LEN`, `WAITING_OPCODE`, and `WAITING_STX`.

Here is the call graph for this function:



4.29.2.4 msv2_init()

```

void msv2_init (
    MSV2_INST_t * msv2 )
  
```

Definition at line 79 of file msv2.c.

References `MSV2_INST::id`.

Referenced by `debug_init()`.

Here is the caller graph for this function:



4.29.2.5 msv2_rx_data()

```
uint8_t* msv2_rx_data (
    MSV2_INST_t * msv2 )
```

Definition at line 199 of file msv2.c.

References MSV2_RX_DATA::data, and MSV2_INST::rx.

4.29.2.6 msv2_tx_data()

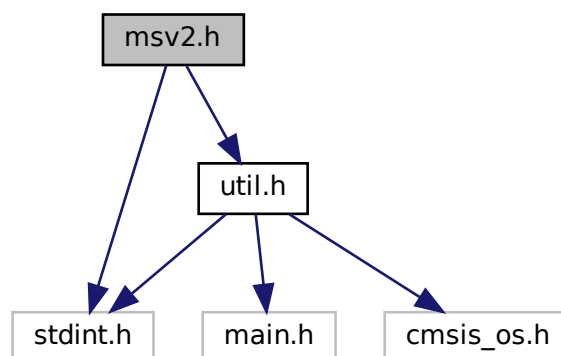
```
uint8_t* msv2_tx_data (
    MSV2_INST_t * msv2 )
```

Definition at line 203 of file msv2.c.

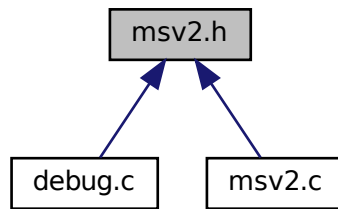
References MSV2_TX_DATA::data, and MSV2_INST::tx.

4.30 msv2.h File Reference

```
#include <util.h>
#include <stdint.h>
Include dependency graph for msv2.h:
```



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



Data Structures

- struct [MSV2_RX_DATA](#)
- struct [MSV2_TX_DATA](#)
- struct [MSV2_INST](#)

Macros

- `#define` [MSV2_MAX_FRAME_LEN](#) (1024)
- `#define` [MSV2_MAX_DATA_LEN](#) (512)

Typedefs

- typedef enum [MSV2_ERROR](#) [MSV2_ERROR_t](#)
- typedef enum [MSV2_DECODE_STATE](#) [MSV2_DECODE_STATE_t](#)
- typedef struct [MSV2_RX_DATA](#) [MSV2_RX_DATA_t](#)
- typedef struct [MSV2_TX_DATA](#) [MSV2_TX_DATA_t](#)
- typedef struct [MSV2_INST](#) [MSV2_INST_t](#)

Enumerations

- enum [MSV2_ERROR](#) { [MSV2_SUCCESS](#) = 0 , [MSV2_PROGRESS](#) , [MSV2_WRONG_CRC](#) , [MSV2_ERROR](#) }
- enum [MSV2_DECODE_STATE](#) { [WAITING_DLE](#) , [WAITING_STX](#) , [WAITING_OPCODE](#) , [WAITING_LEN](#) , [WAITING_DATA](#) , [WAITING_CRC1](#) , [WAITING_CRC2](#) }

Functions

- [MSV2_ERROR_t](#) [msv2_decode_fragment](#) ([MSV2_INST_t](#) *msv2, [uint8_t](#) d)
- void [msv2_init](#) ([MSV2_INST_t](#) *msv2)
- [uint16_t](#) [msv2_create_frame](#) ([MSV2_INST_t](#) *msv2, [uint8_t](#) opcode, [uint8_t](#) data_len, [uint8_t](#) *data)
- [uint8_t](#) * [msv2_rx_data](#) ([MSV2_INST_t](#) *msv2)
- [uint8_t](#) * [msv2_tx_data](#) ([MSV2_INST_t](#) *msv2)

4.30.1 Macro Definition Documentation

4.30.1.1 MSV2_MAX_DATA_LEN

```
#define MSV2_MAX_DATA_LEN (512)
```

Definition at line 25 of file msv2.h.

4.30.1.2 MSV2_MAX_FRAME_LEN

```
#define MSV2_MAX_FRAME_LEN (1024)
```

Definition at line 23 of file msv2.h.

4.30.2 Typedef Documentation

4.30.2.1 MSV2_DECODE_STATE_t

```
typedef enum MSV2_DECODE_STATE MSV2_DECODE_STATE_t
```

4.30.2.2 MSV2_ERROR_t

```
typedef enum MSV2_ERROR MSV2_ERROR_t
```

4.30.2.3 MSV2_INST_t

```
typedef struct MSV2_INST MSV2_INST_t
```

4.30.2.4 MSV2_RX_DATA_t

```
typedef struct MSV2_RX_DATA MSV2_RX_DATA_t
```

4.30.2.5 MSV2_TX_DATA_t

```
typedef struct MSV2_TX_DATA MSV2_TX_DATA_t
```

4.30.3 Enumeration Type Documentation

4.30.3.1 MSV2_DECODE_STATE

```
enum MSV2_DECODE_STATE
```

Enumerator

WAITING_DLE	
WAITING_STX	
WAITING_OPCODE	
WAITING_LEN	
WAITING_DATA	
WAITING_CRC1	
WAITING_CRC2	

Definition at line 44 of file msv2.h.

4.30.3.2 MSV2_ERROR

enum [MSV2_ERROR](#)

Enumerator

MSV2_SUCCESS	
MSV2_PROGRESS	
MSV2_WRONG_CRC	
MSV2_ERROR	

Definition at line 37 of file msv2.h.

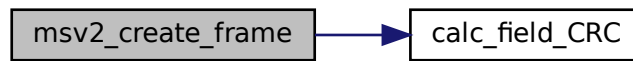
4.30.4 Function Documentation**4.30.4.1 msv2_create_frame()**

```
uint16_t msv2_create_frame (
    MSV2_INST_t * msv2,
    uint8_t opcode,
    uint8_t data_len,
    uint8_t * data )
```

Definition at line 84 of file msv2.c.

References [calc_field_CRC\(\)](#), [MSV2_TX_DATA::crc_data](#), [MSV2_TX_DATA::data](#), [MSV2_TX_DATA::data_len](#), [DLE](#), [MSV2_TX_DATA::opcode](#), [STX](#), and [MSV2_INST::tx](#).

Here is the call graph for this function:



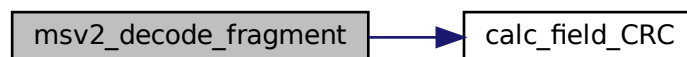
4.30.4.2 msv2_decode_fragment()

```
MSV2_ERROR_t msv2_decode_fragment (
    MSV2_INST_t * msv2,
    uint8_t d )
```

Definition at line 121 of file msv2.c.

References `calc_field_CRC()`, `MSV2_RX_DATA::counter`, `MSV2_RX_DATA::crc`, `MSV2_RX_DATA::crc_data`, `MSV2_RX_DATA::data`, `MSV2_RX_DATA::data_len`, `DLE`, `MSV2_RX_DATA::escape`, `MSV2_RX_DATA::length`, `MSV2_PROGRESS`, `MSV2_SUCCESS`, `MSV2_WRONG_CRC`, `MSV2_RX_DATA::opcode`, `MSV2_INST::rx`, `MSV2_RX_DATA::state`, `STX`, `WAITING_CRC1`, `WAITING_CRC2`, `WAITING_DATA`, `WAITING_DLE`, `WAITING_LEN`, `WAITING_OPCODE`, and `WAITING_STX`.

Here is the call graph for this function:



4.30.4.3 msv2_init()

```
void msv2_init (
    MSV2_INST_t * msv2 )
```

Definition at line 79 of file msv2.c.

References `MSV2_INST::id`.

Referenced by `debug_init()`.

Here is the caller graph for this function:



4.30.4.4 msv2_rx_data()

```
uint8_t* msv2_rx_data (
    MSV2_INST_t * msv2 )
```

Definition at line 199 of file msv2.c.

References MSV2_RX_DATA::data, and MSV2_INST::rx.

4.30.4.5 msv2_tx_data()

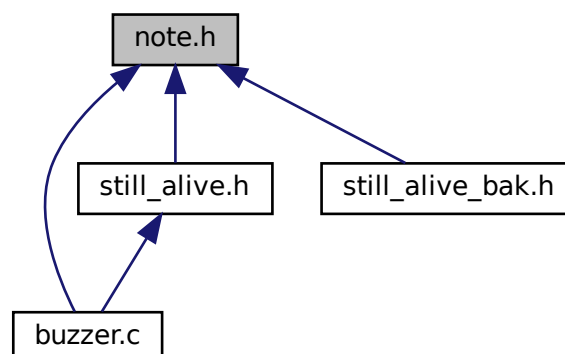
```
uint8_t* msv2_tx_data (
    MSV2_INST_t * msv2 )
```

Definition at line 203 of file msv2.c.

References MSV2_TX_DATA::data, and MSV2_INST::tx.

4.31 note.h File Reference

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



Data Structures

- struct [note](#)

Macros

- #define [T1_4](#) 1
- #define [T1_2](#) 2
- #define [T1](#) 4
- #define [T1_1_2](#) 6
- #define [T2](#) 8
- #define [T4](#) 16
- #define [C0](#) 163
- #define [C0H](#) 173
- #define [D0](#) 183
- #define [D0H](#) 194
- #define [E0](#) 206
- #define [F0](#) 218
- #define [F0H](#) 231
- #define [G0](#) 245
- #define [G0H](#) 259
- #define [A0](#) 275
- #define [A0H](#) 291
- #define [B0](#) 308
- #define [C1](#) 327
- #define [C1H](#) 346
- #define [D1](#) 367
- #define [D1H](#) 388
- #define [E1](#) 412
- #define [F1](#) 436
- #define [F1H](#) 462
- #define [G1](#) 490
- #define [G1H](#) 519
- #define [A1](#) 550
- #define [A1H](#) 582
- #define [B1](#) 617
- #define [C2](#) 654
- #define [C2H](#) 693
- #define [D2](#) 734
- #define [D2H](#) 777
- #define [E2](#) 824
- #define [F2](#) 873
- #define [F2H](#) 925
- #define [G2](#) 980
- #define [G2H](#) 1038
- #define [A2](#) 1100
- #define [A2H](#) 1165
- #define [B2](#) 1234
- #define [C3](#) 1308
- #define [C3H](#) 1385
- #define [D3](#) 1468
- #define [D3H](#) 1555
- #define [E3](#) 1648
- #define [F3](#) 1746

- #define [F3H](#) 1850
- #define [G3](#) 1960
- #define [G3H](#) 2076
- #define [A3](#) 2200
- #define [A3H](#) 2330
- #define [B3](#) 2469
- #define [C4](#) 2616
- #define [C4H](#) 2771
- #define [D4](#) 2936
- #define [D4H](#) 3111
- #define [E4](#) 3296
- #define [F4](#) 3492
- #define [F4H](#) 3699
- #define [G4](#) 3920
- #define [G4H](#) 4153
- #define [A4](#) 4400
- #define [A4H](#) 4661
- #define [B4](#) 4938
- #define [C5](#) 5232
- #define [C5H](#) 5543
- #define [D5](#) 5873
- #define [D5H](#) 6222
- #define [E5](#) 6592
- #define [F5](#) 6984
- #define [F5H](#) 7399
- #define [G5](#) 7839
- #define [G5H](#) 8306
- #define [A5](#) 8800
- #define [A5H](#) 9323
- #define [B5](#) 9877
- #define [C6](#) 10465
- #define [C6H](#) 11087
- #define [D6](#) 11746
- #define [D6H](#) 12445
- #define [E6](#) 13185
- #define [F6](#) 13969
- #define [F6H](#) 14799
- #define [G6](#) 15679
- #define [G6H](#) 16612
- #define [A6](#) 17600
- #define [A6H](#) 18646
- #define [B6](#) 19755
- #define [C7](#) 20930
- #define [C7H](#) 22174
- #define [D7](#) 23493
- #define [D7H](#) 24890
- #define [E7](#) 26370
- #define [F7](#) 27938
- #define [F7H](#) 29599
- #define [G7](#) 31359
- #define [G7H](#) 33224
- #define [A7](#) 35200
- #define [A7H](#) 37293
- #define [B7](#) 39510
- #define [C8](#) 41860

- `#define C8H` 44349
- `#define D8` 46986
- `#define D8H` 49780
- `#define E8` 52740
- `#define F8` 55876
- `#define F8H` 59199
- `#define G8` 62719
- `#define G8H` 66448
- `#define A8` 70400
- `#define A8H` 74586
- `#define B8` 79021

Typedefs

- `typedef struct note note_t`

4.31.1 Macro Definition Documentation

4.31.1.1 A0

```
#define A0 275
```

Definition at line 28 of file note.h.

4.31.1.2 A0H

```
#define A0H 291
```

Definition at line 29 of file note.h.

4.31.1.3 A1

```
#define A1 550
```

Definition at line 40 of file note.h.

4.31.1.4 A1H

```
#define A1H 582
```

Definition at line 41 of file note.h.

4.31.1.5 A2

```
#define A2 1100
```

Definition at line 52 of file note.h.

4.31.1.6 A2H

```
#define A2H 1165
```

Definition at line 53 of file note.h.

4.31.1.7 A3

```
#define A3 2200
```

Definition at line 64 of file note.h.

4.31.1.8 A3H

```
#define A3H 2330
```

Definition at line 65 of file note.h.

4.31.1.9 A4

```
#define A4 4400
```

Definition at line 76 of file note.h.

4.31.1.10 A4H

```
#define A4H 4661
```

Definition at line 77 of file note.h.

4.31.1.11 A5

```
#define A5 8800
```

Definition at line 88 of file note.h.

4.31.1.12 A5H

```
#define A5H 9323
```

Definition at line 89 of file note.h.

4.31.1.13 A6

```
#define A6 17600
```

Definition at line 100 of file note.h.

4.31.1.14 A6H

```
#define A6H 18646
```

Definition at line 101 of file note.h.

4.31.1.15 A7

```
#define A7 35200
```

Definition at line 112 of file note.h.

4.31.1.16 A7H

```
#define A7H 37293
```

Definition at line 113 of file note.h.

4.31.1.17 A8

```
#define A8 70400
```

Definition at line 124 of file note.h.

4.31.1.18 A8H

```
#define A8H 74586
```

Definition at line 125 of file note.h.

4.31.1.19 B0

```
#define B0 308
```

Definition at line 30 of file note.h.

4.31.1.20 B1

```
#define B1 617
```

Definition at line 42 of file note.h.

4.31.1.21 B2

```
#define B2 1234
```

Definition at line 54 of file note.h.

4.31.1.22 B3

```
#define B3 2469
```

Definition at line 66 of file note.h.

4.31.1.23 B4

```
#define B4 4938
```

Definition at line 78 of file note.h.

4.31.1.24 B5

```
#define B5 9877
```

Definition at line 90 of file note.h.

4.31.1.25 B6

```
#define B6 19755
```

Definition at line 102 of file note.h.

4.31.1.26 B7

```
#define B7 39510
```

Definition at line 114 of file note.h.

4.31.1.27 B8

```
#define B8 79021
```

Definition at line 126 of file note.h.

4.31.1.28 C0

```
#define C0 163
```

Definition at line 19 of file note.h.

4.31.1.29 C0H

```
#define C0H 173
```

Definition at line 20 of file note.h.

4.31.1.30 C1

```
#define C1 327
```

Definition at line 31 of file note.h.

4.31.1.31 C1H

```
#define C1H 346
```

Definition at line 32 of file note.h.

4.31.1.32 C2

```
#define C2 654
```

Definition at line 43 of file note.h.

4.31.1.33 C2H

```
#define C2H 693
```

Definition at line 44 of file note.h.

4.31.1.34 C3

```
#define C3 1308
```

Definition at line 55 of file note.h.

4.31.1.35 C3H

```
#define C3H 1385
```

Definition at line 56 of file note.h.

4.31.1.36 C4

```
#define C4 2616
```

Definition at line 67 of file note.h.

4.31.1.37 C4H

```
#define C4H 2771
```

Definition at line 68 of file note.h.

4.31.1.38 C5

```
#define C5 5232
```

Definition at line 79 of file note.h.

4.31.1.39 C5H

```
#define C5H 5543
```

Definition at line 80 of file note.h.

4.31.1.40 C6

```
#define C6 10465
```

Definition at line 91 of file note.h.

4.31.1.41 C6H

```
#define C6H 11087
```

Definition at line 92 of file note.h.

4.31.1.42 C7

```
#define C7 20930
```

Definition at line 103 of file note.h.

4.31.1.43 C7H

```
#define C7H 22174
```

Definition at line 104 of file note.h.

4.31.1.44 C8

```
#define C8 41860
```

Definition at line 115 of file note.h.

4.31.1.45 C8H

```
#define C8H 44349
```

Definition at line 116 of file note.h.

4.31.1.46 D0

```
#define D0 183
```

Definition at line 21 of file note.h.

4.31.1.47 D0H

```
#define D0H 194
```

Definition at line 22 of file note.h.

4.31.1.48 D1

```
#define D1 367
```

Definition at line 33 of file note.h.

4.31.1.49 D1H

```
#define D1H 388
```

Definition at line 34 of file note.h.

4.31.1.50 D2

```
#define D2 734
```

Definition at line 45 of file note.h.

4.31.1.51 D2H

```
#define D2H 777
```

Definition at line 46 of file note.h.

4.31.1.52 D3

```
#define D3 1468
```

Definition at line 57 of file note.h.

4.31.1.53 D3H

```
#define D3H 1555
```

Definition at line 58 of file note.h.

4.31.1.54 D4

```
#define D4 2936
```

Definition at line 69 of file note.h.

4.31.1.55 D4H

```
#define D4H 3111
```

Definition at line 70 of file note.h.

4.31.1.56 D5

```
#define D5 5873
```

Definition at line 81 of file note.h.

4.31.1.57 D5H

```
#define D5H 6222
```

Definition at line 82 of file note.h.

4.31.1.58 D6

```
#define D6 11746
```

Definition at line 93 of file note.h.

4.31.1.59 D6H

```
#define D6H 12445
```

Definition at line 94 of file note.h.

4.31.1.60 D7

```
#define D7 23493
```

Definition at line 105 of file note.h.

4.31.1.61 D7H

```
#define D7H 24890
```

Definition at line 106 of file note.h.

4.31.1.62 D8

```
#define D8 46986
```

Definition at line 117 of file note.h.

4.31.1.63 D8H

```
#define D8H 49780
```

Definition at line 118 of file note.h.

4.31.1.64 E0

```
#define E0 206
```

Definition at line 23 of file note.h.

4.31.1.65 E1

```
#define E1 412
```

Definition at line 35 of file note.h.

4.31.1.66 E2

```
#define E2 824
```

Definition at line 47 of file note.h.

4.31.1.67 E3

```
#define E3 1648
```

Definition at line 59 of file note.h.

4.31.1.68 E4

```
#define E4 3296
```

Definition at line 71 of file note.h.

4.31.1.69 E5

```
#define E5 6592
```

Definition at line 83 of file note.h.

4.31.1.70 E6

```
#define E6 13185
```

Definition at line 95 of file note.h.

4.31.1.71 E7

```
#define E7 26370
```

Definition at line 107 of file note.h.

4.31.1.72 E8

```
#define E8 52740
```

Definition at line 119 of file note.h.

4.31.1.73 F0

```
#define F0 218
```

Definition at line 24 of file note.h.

4.31.1.74 F0H

```
#define F0H 231
```

Definition at line 25 of file note.h.

4.31.1.75 F1

```
#define F1 436
```

Definition at line 36 of file note.h.

4.31.1.76 F1H

```
#define F1H 462
```

Definition at line 37 of file note.h.

4.31.1.77 F2

```
#define F2 873
```

Definition at line 48 of file note.h.

4.31.1.78 F2H

```
#define F2H 925
```

Definition at line 49 of file note.h.

4.31.1.79 F3

```
#define F3 1746
```

Definition at line 60 of file note.h.

4.31.1.80 F3H

```
#define F3H 1850
```

Definition at line 61 of file note.h.

4.31.1.81 F4

```
#define F4 3492
```

Definition at line 72 of file note.h.

4.31.1.82 F4H

```
#define F4H 3699
```

Definition at line 73 of file note.h.

4.31.1.83 F5

```
#define F5 6984
```

Definition at line 84 of file note.h.

4.31.1.84 F5H

```
#define F5H 7399
```

Definition at line 85 of file note.h.

4.31.1.85 F6

```
#define F6 13969
```

Definition at line 96 of file note.h.

4.31.1.86 F6H

```
#define F6H 14799
```

Definition at line 97 of file note.h.

4.31.1.87 F7

```
#define F7 27938
```

Definition at line 108 of file note.h.

4.31.1.88 F7H

```
#define F7H 29599
```

Definition at line 109 of file note.h.

4.31.1.89 F8

```
#define F8 55876
```

Definition at line 120 of file note.h.

4.31.1.90 F8H

```
#define F8H 59199
```

Definition at line 121 of file note.h.

4.31.1.91 G0

```
#define G0 245
```

Definition at line 26 of file note.h.

4.31.1.92 G0H

```
#define G0H 259
```

Definition at line 27 of file note.h.

4.31.1.93 G1

```
#define G1 490
```

Definition at line 38 of file note.h.

4.31.1.94 G1H

```
#define G1H 519
```

Definition at line 39 of file note.h.

4.31.1.95 G2

```
#define G2 980
```

Definition at line 50 of file note.h.

4.31.1.96 G2H

```
#define G2H 1038
```

Definition at line 51 of file note.h.

4.31.1.97 G3

```
#define G3 1960
```

Definition at line 62 of file note.h.

4.31.1.98 G3H

```
#define G3H 2076
```

Definition at line 63 of file note.h.

4.31.1.99 G4

```
#define G4 3920
```

Definition at line 74 of file note.h.

4.31.1.100 G4H

```
#define G4H 4153
```

Definition at line 75 of file note.h.

4.31.1.101 G5

```
#define G5 7839
```

Definition at line 86 of file note.h.

4.31.1.102 G5H

```
#define G5H 8306
```

Definition at line 87 of file note.h.

4.31.1.103 G6

```
#define G6 15679
```

Definition at line 98 of file note.h.

4.31.1.104 G6H

```
#define G6H 16612
```

Definition at line 99 of file note.h.

4.31.1.105 G7

```
#define G7 31359
```

Definition at line 110 of file note.h.

4.31.1.106 G7H

```
#define G7H 33224
```

Definition at line 111 of file note.h.

4.31.1.107 G8

```
#define G8 62719
```

Definition at line 122 of file note.h.

4.31.1.108 G8H

```
#define G8H 66448
```

Definition at line 123 of file note.h.

4.31.1.109 T1

```
#define T1 4
```

Definition at line 12 of file note.h.

4.31.1.110 T1_1_2

```
#define T1_1_2 6
```

Definition at line 13 of file note.h.

4.31.1.111 T1_2

```
#define T1_2 2
```

Definition at line 11 of file note.h.

4.31.1.112 T1_4

```
#define T1_4 1
```

Definition at line 10 of file note.h.

4.31.1.113 T2

```
#define T2 8
```

Definition at line 14 of file note.h.

4.31.1.114 T4

```
#define T4 16
```

Definition at line 15 of file note.h.

4.31.2 Typedef Documentation

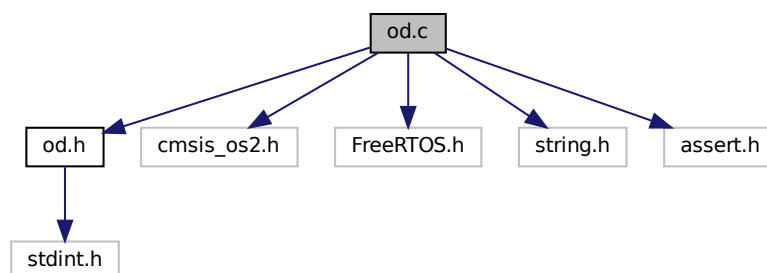
4.31.2.1 note_t

```
typedef struct note note_t
```

4.32 od.c File Reference

```
#include "od.h"  
#include <cmsis_os2.h>  
#include <FreeRTOS.h>  
#include <string.h>  
#include <assert.h>
```

Include dependency graph for od.c:



Data Structures

- struct [od_entry_t](#)
- struct [od_frame_t](#)

Macros

- `#define` [OD_MSGQ_SIZE](#) (16)
- `#define` [DEBUG_NO_CAN](#) 0
- `#define` [ALLOCATE_OD_ENTRY](#)(NAME, ID, TYPE)
- `#define` [LINK_OD_ENTRY](#)(NAME) [(NAME)] = (NAME ## _entry)

Functions

- static void [od_unsafe_read](#) (uint8_t data_id, uint8_t *dst)
- static void [od_unsafe_write](#) (uint8_t data_id, uint8_t *src)
- void [od_init](#) ()
- void [od_update_task](#) (__attribute__((unused)) void *argument)

Variables

- static const [od_entry_t](#) [od_entries](#) [[OD_MAX_DATAID](#)]
- [osMessageQueueId_t](#) [out_q](#)
- [osMessageQueueId_t](#) [in_q](#)

4.32.1 Macro Definition Documentation

4.32.1.1 ALLOCATE_OD_ENTRY

```
#define ALLOCATE_OD_ENTRY(  
    NAME,  
    ID,  
    TYPE )
```

Value:

```
enum { NAME = ID }; \
static_assert((NAME) < OD_MAX_DATAID); \
static_assert(sizeof(TYPE) < OD_FRAME_MAX_SIZE); \
static TYPE (NAME ## _var); \
static const od_entry_t (NAME ## _entry) = { .data_id=(NAME), .size=sizeof(TYPE), .data=(uint8_t*)&(NAME ## _var) }; \
\
void od_read_ ## NAME (TYPE *dst) { od_unsafe_read ((NAME), (uint8_t*) dst); } \
void od_write_ ## NAME (TYPE *src) { od_unsafe_write((NAME), (uint8_t*) src); }
```

Definition at line 32 of file od.c.

4.32.1.2 DEBUG_NO_CAN

```
#define DEBUG_NO_CAN 0
```

Definition at line 26 of file od.c.

4.32.1.3 LINK_OD_ENTRY

```
#define LINK_OD_ENTRY(  
    NAME ) [ (NAME) ] = (NAME ## _entry)
```

Definition at line 43 of file od.c.

4.32.1.4 OD_MSGQ_SIZE

```
#define OD_MSGQ_SIZE (16)
```

Definition at line 25 of file od.c.

4.32.2 Function Documentation

4.32.2.1 od_init()

```
void od_init ( )
```

Definition at line 94 of file od.c.

References in_q, OD_MSGQ_SIZE, and out_q.

Referenced by threads_init().

Here is the caller graph for this function:



4.32.2.2 od_unsafe_read()

```
static void od_unsafe_read (
    uint8_t data_id,
    uint8_t * dst ) [static]
```

Read/write interface

Definition at line 122 of file od.c.

References `od_entry_t::data`, `od_entries`, and `od_entry_t::size`.

4.32.2.3 od_unsafe_write()

```
static void od_unsafe_write (
    uint8_t data_id,
    uint8_t * src ) [static]
```

Definition at line 131 of file od.c.

References `od_frame_t::data`, `od_entry_t::data_id`, `od_frame_t::data_id`, `od_entries`, `out_q`, `od_entry_t::size`, and `od_frame_t::size`.

4.32.2.4 od_update_task()

```
void od_update_task (
    __attribute__((unused)) void * argument )
```

Task definition

Definition at line 143 of file od.c.

References `od_entry_t::data`, `od_frame_t::data`, `od_frame_t::data_id`, `in_q`, `od_entries`, `out_q`, and `od_entry_t::size`.

Referenced by `threads_init()`.

Here is the caller graph for this function:



4.32.3 Variable Documentation

4.32.3.1 in_q

```
osMessageQueueId_t in_q
```

Definition at line 88 of file od.c.

Referenced by `od_init()`, and `od_update_task()`.

4.32.3.2 od_entries

```
const od_entry_t od_entries[OD_MAX_DATAID] [static]
```

Initial value:

```
= {  
    LINK_OD_ENTRY(TEMPERATURE),  
}
```

Object dictionary entries The object dictionary

Definition at line 80 of file od.c.

Referenced by `od_unsafe_read()`, `od_unsafe_write()`, and `od_update_task()`.

4.32.3.3 out_q

```
osMessageQueueId_t out_q
```

Synchronization primitives

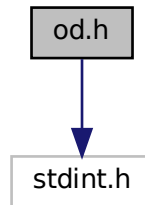
Definition at line 87 of file od.c.

Referenced by `od_init()`, `od_unsafe_write()`, and `od_update_task()`.

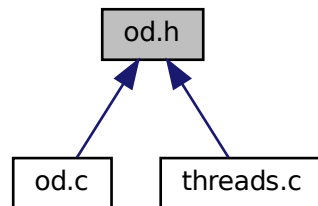
4.33 od.h File Reference

```
#include <stdint.h>
```

Include dependency graph for od.h:



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



Macros

- `#define OD_FRAME_MAX_SIZE` (64)
- `#define OD_MAX_DATAID` (256U)
- `#define DECLARE_OD_ENTRY`(NAME, TYPE)

Functions

- void `od_init` ()
- void `od_update_task` (void *argument)

4.33.1 Macro Definition Documentation

4.33.1.1 DECLARE_OD_ENTRY

```
#define DECLARE_OD_ENTRY(  
    NAME,  
    TYPE )
```

Value:

```
void od_read_ ## NAME (TYPE *dst); \  
void od_write_ ## NAME (TYPE *src);
```

Definition at line 31 of file od.h.

4.33.1.2 OD_FRAME_MAX_SIZE

```
#define OD_FRAME_MAX_SIZE (64)
```

Definition at line 24 of file od.h.

4.33.1.3 OD_MAX_DATAID

```
#define OD_MAX_DATAID (256U)
```

Definition at line 25 of file od.h.

4.33.2 Function Documentation

4.33.2.1 od_init()

```
void od_init ( )
```

Definition at line 94 of file od.c.

References `in_q`, `OD_MSGQ_SIZE`, and `out_q`.

Referenced by `threads_init()`.

Here is the caller graph for this function:

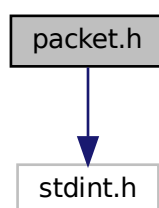


4.33.2.2 od_update_task()

```
void od_update_task (
    void * argument )
```

4.34 packet.h File Reference

```
#include <stdint.h>
Include dependency graph for packet.h:
```



Data Structures

- struct [packet_def](#)

Typedefs

- typedef struct [packet_def](#) [packet_def_t](#)

Variables

- const [packet_def_t](#) [ping](#) = {0x00, 0x02}

4.34.1 Typedef Documentation

4.34.1.1 packet_def_t

```
typedef struct packet\_def packet\_def\_t
```

4.34.2 Variable Documentation

4.34.2.1 ping

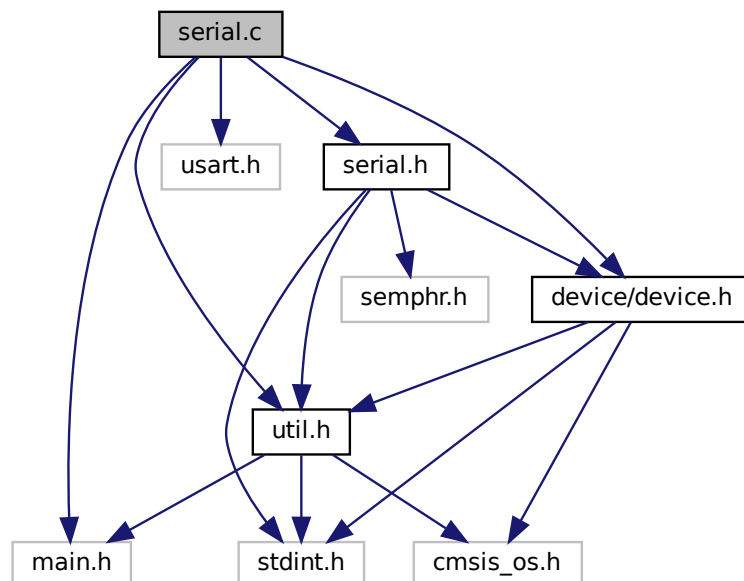
```
const packet_def_t ping = {0x00, 0x02}
```

Definition at line 30 of file packet.h.

4.35 serial.c File Reference

```
#include <util.h>
#include <usart.h>
#include <main.h>
#include "serial.h"
#include <device/device.h>
```

Include dependency graph for serial.c:



Macros

- #define S1_UART huart2
- #define S2_UART huart3
- #define S3_UART huart6
- #define SERIAL_DMA_LEN 32

Functions

- [util_error_t serial_data_ready](#) (void *context)
- [util_error_t serial_send](#) (void *context, uint8_t *data, uint32_t len)
- [util_error_t serial_recv](#) (void *context, uint8_t *data, uint32_t *len)
- [util_error_t serial_handle_data](#) (void *if_context, void *dem_context)
- [util_error_t serial_setup_reception](#) ([serial_interface_context_t](#) *interface_context, [serial_transfer_mode_t](#) mode)
- void [HAL_UART_RxCpltCallback](#) (UART_HandleTypeDef *huart)
- [device_deamon_t](#) * [serial_get_deamon](#) (void)
- [device_interface_t](#) * [serial_get_feedback_interface](#) (void)
- [util_error_t serial_init](#) (void)
- [util_error_t serial_feedback_init](#) (void)

Variables

- static [device_deamon_t](#) [serial_deamon](#)
- static [device_interface_t](#) [feedback_interface](#)
- static [serial_deamon_context_t](#) [serial_deamon_context](#)
- static [serial_interface_context_t](#) [feedback_interface_context](#)

4.35.1 Macro Definition Documentation

4.35.1.1 S1_UART

```
#define S1_UART huart2
```

Definition at line 25 of file serial.c.

4.35.1.2 S2_UART

```
#define S2_UART huart3
```

Definition at line 26 of file serial.c.

4.35.1.3 S3_UART

```
#define S3_UART huart6
```

Definition at line 27 of file serial.c.

4.35.1.4 SERIAL_DMA_LEN

```
#define SERIAL_DMA_LEN 32
```

Definition at line 29 of file serial.c.

4.35.2 Function Documentation

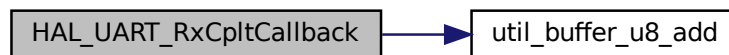
4.35.2.1 HAL_UART_RxCpltCallback()

```
void HAL_UART_RxCpltCallback (
    UART_HandleTypeDef * huart )
```

Definition at line 80 of file serial.c.

References `device_interface::context`, `device_daemon::context`, `device_daemon::interfaces`, `device_daemon::interfaces_count`, `serial_interface_context::rx_buffer`, `serial_interface_context::rx_fragment`, `serial_daemon_context::rx_sem`, `serial_daemon`, `serial_interface_context::uart`, and `util_buffer_u8_add()`.

Here is the call graph for this function:



4.35.2.2 serial_data_ready()

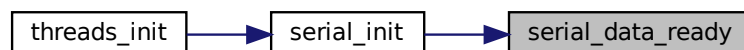
```
util_error_t serial_data_ready (
    void * context )
```

Definition at line 133 of file serial.c.

References `ER_SUCCESS`, `ER_TIMEOUT`, and `serial_daemon_context::rx_sem`.

Referenced by `serial_init()`.

Here is the caller graph for this function:



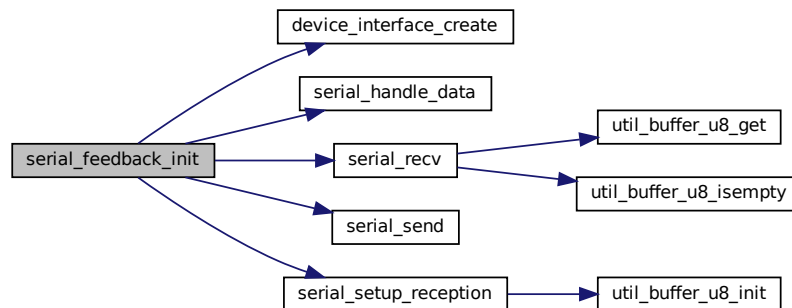
4.35.2.3 serial_feedback_init()

```
util_error_t serial_feedback_init (
    void )
```

Definition at line 123 of file serial.c.

References `device_interface_create()`, `ER_SUCCESS`, `feedback_interface`, `feedback_interface_context`, `serial_daemon`, `serial_handle_data()`, `serial_rcv()`, `serial_send()`, `serial_setup_reception()`, and `SERIAL_TRANSFER_IT`.

Here is the call graph for this function:



4.35.2.4 serial_get_daemon()

```
device_daemon_t* serial_get_daemon (
    void )
```

Definition at line 99 of file serial.c.

References `serial_daemon`.

4.35.2.5 serial_get_feedback_interface()

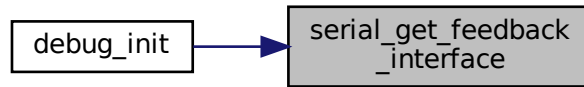
```
device_interface_t* serial_get_feedback_interface (
    void )
```

Definition at line 104 of file serial.c.

References `feedback_interface`.

Referenced by `debug_init()`.

Here is the caller graph for this function:



4.35.2.6 serial_handle_data()

```

util_error_t serial_handle_data (
    void * if_context,
    void * dem_context )
  
```

Definition at line 184 of file serial.c.

References `ER_SUCCESS`.

Referenced by `serial_feedback_init()`.

Here is the caller graph for this function:



4.35.2.7 serial_init()

```

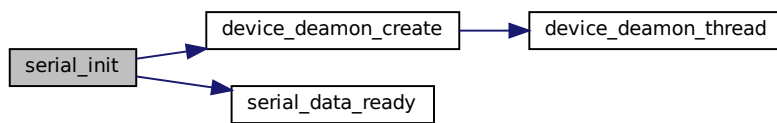
util_error_t serial_init (
    void )
  
```

Definition at line 110 of file serial.c.

References `device_daemon_create()`, `ER_SUCCESS`, `serial_daemon_context::rx_sem`, `serial_daemon_context::rx_sem_buffer`, `serial_data_ready()`, and `serial_daemon`.

Referenced by `threads_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.35.2.8 serial_recv()

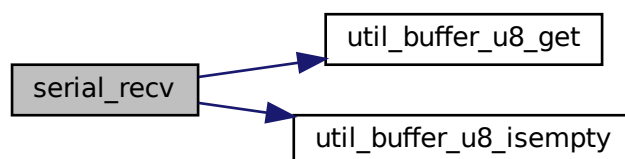
```
util_error_t serial_recv (  
    void * context,  
    uint8_t * data,  
    uint32_t * len )
```

Definition at line 172 of file serial.c.

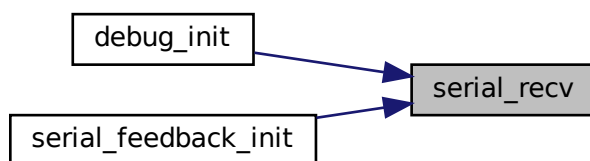
References `ER_SUCCESS`, `serial_interface_context::rx_buffer`, `util_buffer_u8_get()`, and `util_buffer_u8_isempty()`.

Referenced by `debug_init()`, and `serial_feedback_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.35.2.9 serial_send()

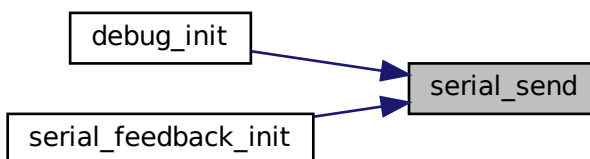
```
util_error_t serial_send (  
    void * context,  
    uint8_t * data,  
    uint32_t len )
```

Definition at line 163 of file `serial.c`.

References `ER_SUCCESS`, and `serial_interface_context::uart`.

Referenced by `debug_init()`, and `serial_feedback_init()`.

Here is the caller graph for this function:



4.35.2.10 serial_setup_reception()

```
util_error_t serial_setup_reception (
    serial_interface_context_t * interface_context,
    serial_transfer_mode_t mode )
```

Definition at line 144 of file serial.c.

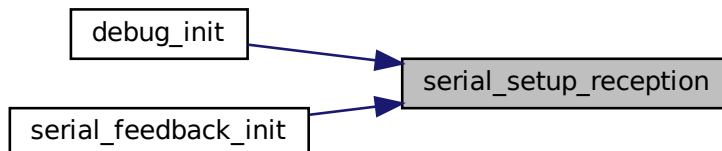
References `ER_FAILURE`, `ER_RESSOURCE_ERROR`, `serial_interface_context::rx_buffer`, `serial_interface_context::rx_data`, `serial_interface_context::rx_fragment`, `SERIAL_BUFFER_LEN`, `SERIAL_TRANSFER_DMA`, `SERIAL_TRANSFER_IT`, `serial_interface_context::uart`, and `util_buffer_u8_init()`.

Referenced by `debug_init()`, and `serial_feedback_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.35.3 Variable Documentation

4.35.3.1 feedback_interface

```
device_interface_t feedback_interface [static]
```

Definition at line 50 of file serial.c.

Referenced by `debug_init()`, `serial_feedback_init()`, and `serial_get_feedback_interface()`.

4.35.3.2 feedback_interface_context

```
serial_interface_context_t feedback_interface_context [static]
```

Initial value:

```
= {  
    .uart = &S3_UART  
}
```

Definition at line 54 of file serial.c.

Referenced by serial_feedback_init().

4.35.3.3 serial_deamon

```
device_deamon_t serial_deamon [static]
```

Definition at line 48 of file serial.c.

Referenced by debug_init(), HAL_UART_RxCpltCallback(), serial_feedback_init(), serial_get_deamon(), and serial_init().

4.35.3.4 serial_deamon_context

```
serial_deamon_context_t serial_deamon_context [static]
```

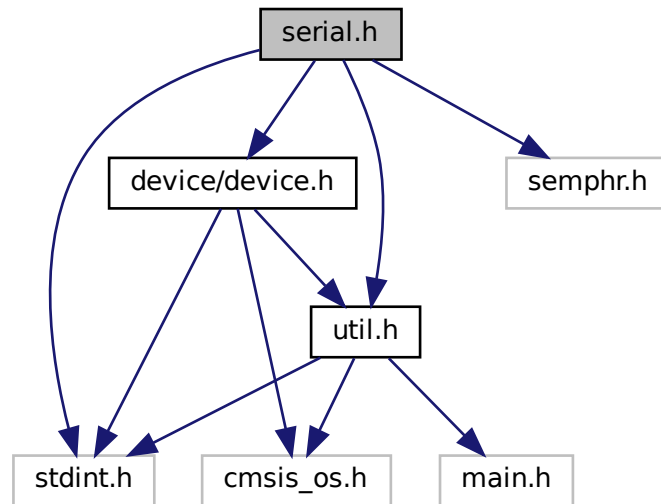
Definition at line 52 of file serial.c.

4.36 serial.h File Reference

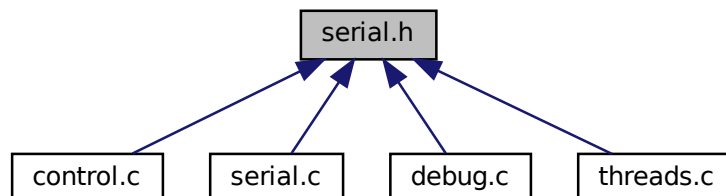
```
#include <stdint.h>  
#include <device/device.h>  
#include <util.h>
```

```
#include <semphr.h>
```

Include dependency graph for serial.h:



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



Data Structures

- struct [serial_daemon_context](#)
- struct [serial_interface_context](#)

Macros

- #define [SERIAL_BUFFER_LEN](#) 256

Typedefs

- typedef enum [serial_interrupt_source](#) [serial_interrupt_source_t](#)
- typedef enum [serial_transfer_mode](#) [serial_transfer_mode_t](#)
- typedef struct [serial_deamon_context](#) [serial_deamon_context_t](#)
- typedef struct [serial_interface_context](#) [serial_interface_context_t](#)

Enumerations

- enum [serial_interrupt_source](#) { [SERIAL_SOURCE_DMA_FIRST_HALF](#) , [SERIAL_SOURCE_DMA_SECOND_HALF](#) , [SERIAL_SOURCE_IDLE](#) }
- enum [serial_transfer_mode](#) { [SERIAL_TRANSFER_DMA](#) , [SERIAL_TRANSFER_IT](#) }

Functions

- [util_error_t serial_init](#) (void)
- [util_error_t serial_feedback_init](#) (void)
- [device_deamon_t * serial_get_deamon](#) (void)
- [device_interface_t * serial_get_feedback_interface](#) (void)
- [util_error_t serial_send](#) (void *context, [uint8_t](#) *data, [uint32_t](#) len)
- [util_error_t serial_recv](#) (void *context, [uint8_t](#) *data, [uint32_t](#) *len)

4.36.1 Macro Definition Documentation

4.36.1.1 SERIAL_BUFFER_LEN

```
#define SERIAL_BUFFER_LEN 256
```

Definition at line 27 of file serial.h.

4.36.2 Typedef Documentation

4.36.2.1 serial_deamon_context_t

```
typedef struct serial\_deamon\_context serial\_deamon\_context\_t
```

4.36.2.2 serial_interface_context_t

```
typedef struct serial\_interface\_context serial\_interface\_context\_t
```

4.36.2.3 serial_interrupt_source_t

```
typedef enum serial_interrupt_source serial_interrupt_source_t
```

4.36.2.4 serial_transfer_mode_t

```
typedef enum serial_transfer_mode serial_transfer_mode_t
```

4.36.3 Enumeration Type Documentation

4.36.3.1 serial_interrupt_source

```
enum serial_interrupt_source
```

Enumerator

SERIAL_SOURCE_DMA_FIRST_HALF	
SERIAL_SOURCE_DMA_SECOND_HALF	
SERIAL_SOURCE_IDLE	

Definition at line 39 of file serial.h.

4.36.3.2 serial_transfer_mode

```
enum serial_transfer_mode
```

Enumerator

SERIAL_TRANSFER_DMA	
SERIAL_TRANSFER_IT	

Definition at line 45 of file serial.h.

4.36.4 Function Documentation

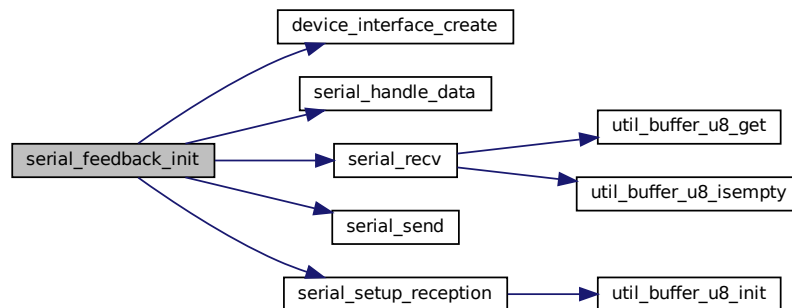
4.36.4.1 serial_feedback_init()

```
util_error_t serial_feedback_init (
    void )
```

Definition at line 123 of file serial.c.

References `device_interface_create()`, `ER_SUCCESS`, `feedback_interface`, `feedback_interface_context`, `serial_↵
daemon`, `serial_handle_data()`, `serial_rcv()`, `serial_send()`, `serial_setup_reception()`, and `SERIAL_TRANSFER_IT`.

Here is the call graph for this function:



4.36.4.2 serial_get_daemon()

```
device_daemon_t* serial_get_daemon (
    void )
```

Definition at line 99 of file serial.c.

References `serial_daemon`.

4.36.4.3 serial_get_feedback_interface()

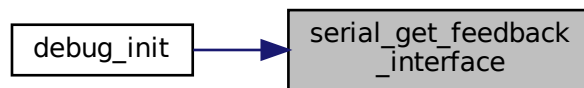
```
device_interface_t* serial_get_feedback_interface (
    void )
```

Definition at line 104 of file serial.c.

References `feedback_interface`.

Referenced by `debug_init()`.

Here is the caller graph for this function:



4.36.4.4 serial_init()

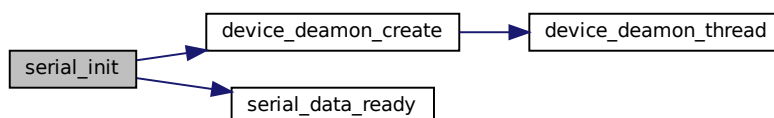
```
util_error_t serial_init (  
    void )
```

Definition at line 110 of file serial.c.

References `device_daemon_create()`, `ER_SUCCESS`, `serial_daemon_context::rx_sem`, `serial_daemon_context::rx_sem_buffer`, `serial_data_ready()`, and `serial_daemon`.

Referenced by `threads_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.36.4.5 serial_recv()

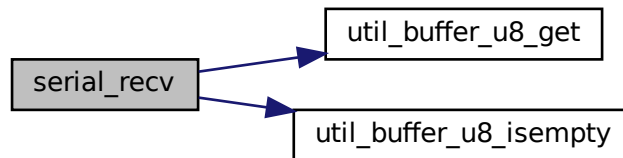
```
util_error_t serial_recv (  
    void * context,  
    uint8_t * data,  
    uint32_t * len )
```

Definition at line 172 of file serial.c.

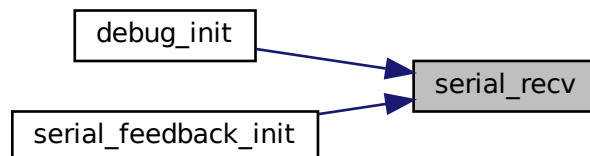
References `ER_SUCCESS`, `serial_interface_context::rx_buffer`, `util_buffer_u8_get()`, and `util_buffer_u8_isempty()`.

Referenced by `debug_init()`, and `serial_feedback_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.36.4.6 serial_send()

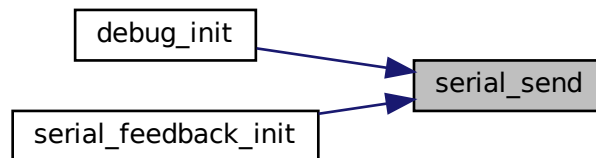
```
util_error_t serial_send (  
    void * context,  
    uint8_t * data,  
    uint32_t len )
```

Definition at line 163 of file serial.c.

References ER_SUCCESS, and serial_interface_context::uart.

Referenced by debug_init(), and serial_feedback_init().

Here is the caller graph for this function:

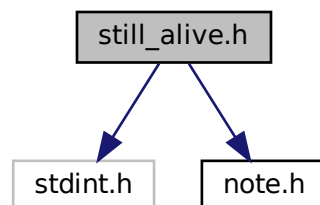


4.37 still_alive.h File Reference

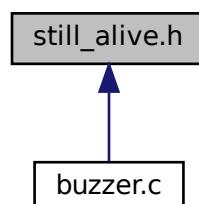
```
#include <stdint.h>
```

```
#include "note.h"
```

Include dependency graph for still_alive.h:



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



Variables

- uint16_t [still_alive](#) []
- uint32_t [still_alive_len](#) = sizeof([still_alive](#))/sizeof(uint16_t)

4.37.1 Variable Documentation

4.37.1.1 still_alive

```
uint16_t still_alive[]
```

Definition at line 14 of file `still_alive.h`.

Referenced by `buzzer_rytm_interrupt()`.

4.37.1.2 still_alive_len

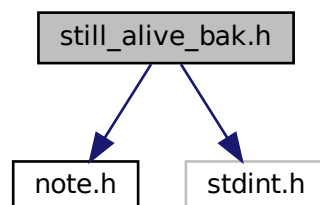
```
uint32_t still_alive_len = sizeof(still\_alive)/sizeof(uint16_t)
```

Definition at line 373 of file `still_alive.h`.

Referenced by `buzzer_rytm_interrupt()`.

4.38 still_alive_bak.h File Reference

```
#include "note.h"  
#include <stdint.h>  
Include dependency graph for still_alive_bak.h:
```



Variables

- `note_t still_alive []`
- `uint32_t still_alive_len = sizeof(still_alive)/sizeof(note_t)`

4.38.1 Variable Documentation

4.38.1.1 still_alive

```
note_t still_alive[]
```

Definition at line 16 of file still_alive_bak.h.

4.38.1.2 still_alive_len

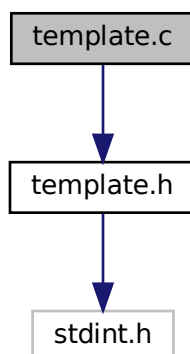
```
uint32_t still_alive_len = sizeof(still_alive)/sizeof(note_t)
```

Definition at line 82 of file still_alive_bak.h.

4.39 template.c File Reference

```
#include "template.h"
```

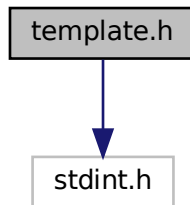
Include dependency graph for template.c:



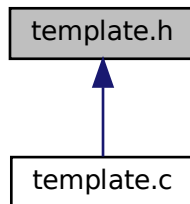
4.40 template.h File Reference

```
#include <stdint.h>
```

Include dependency graph for template.h:



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

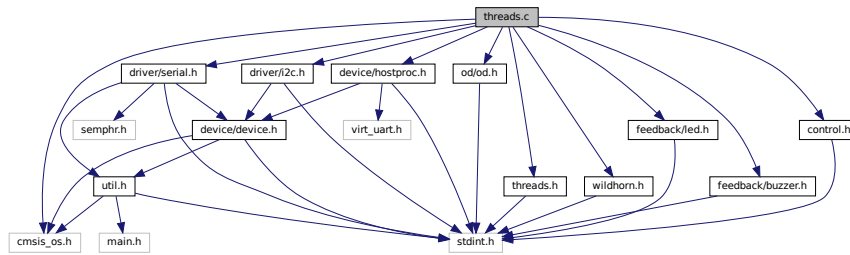


4.41 threads.c File Reference

```
#include <cmsis_os.h>
#include <threads.h>
#include <wildhorn.h>
#include <feedback/led.h>
#include <feedback/buzzer.h>
#include <driver/serial.h>
#include <driver/i2c.h>
#include <control.h>
#include <device/hostproc.h>
```

```
#include <od/od.h>
```

Include dependency graph for threads.c:



Macros

- `#define DEFAULT_SZ (1024)`
- `#define OD_SZ DEFAULT_SZ`
- `#define OD_PRIO (6)`
- `#define CONTROL_SZ DEFAULT_SZ`
- `#define CONTROL_PRIO (6)`
- `#define LED_RGB_SZ DEFAULT_SZ`
- `#define LED_RGB_PRIO (0)`
- `#define CREATE_THREAD(handle, name, func, cont, sz, prio)`

macro to declare a static thread in FreeRTOS

Functions

- `void threads_init (void)`
Initialize all the threads of Wildhorn AV.

Variables

- `static TaskHandle_t od_handle = NULL`
- `static TaskHandle_t control_handle = NULL`
- `static TaskHandle_t led_rgb_handle = NULL`

4.41.1 Macro Definition Documentation

4.41.1.1 CONTROL_PRIO

```
#define CONTROL_PRIO (6)
```

Definition at line 36 of file threads.c.

4.41.1.2 CONTROL_SZ

```
#define CONTROL_SZ DEFAULT_SZ
```

Definition at line 35 of file threads.c.

4.41.1.3 CREATE_THREAD

```
#define CREATE_THREAD(  
    handle,  
    name,  
    func,  
    cont,  
    sz,  
    prio )
```

Value:

```
static StaticTask_t name##_buffer; \  
static StackType_t name##_stack[ sz ]; \  
handle = xTaskCreateStatic( \  
    func, \  
    #name, \  
    sz, \  
    ( void * ) cont, \  
    prio, \  
    name##_stack, \  
    &name##_buffer)
```

macro to declare a static thread in FreeRTOS

This macros make the necessary funtion calls to setup a stack and working area for the declaration of a static FreeRTOS thread.

Parameters

<i>handle</i>	A TaskHandle_t object to reference the created Thread.
<i>name</i>	A name for thread.
<i>func</i>	The entry point for the thread.
<i>cont</i>	The context for the thread.
<i>sz</i>	The desired size for the thread stack.
<i>prio</i>	The priority for the thread.

Definition at line 58 of file threads.c.

4.41.1.4 DEFAULT_SZ

```
#define DEFAULT_SZ (1024)
```

Definition at line 30 of file threads.c.

4.41.1.5 LED_RGB_PRIO

```
#define LED_RGB_PRIO (0)
```

Definition at line 39 of file threads.c.

4.41.1.6 LED_RGB_SZ

```
#define LED_RGB_SZ DEFAULT_SZ
```

Definition at line 38 of file threads.c.

4.41.1.7 OD_PRIO

```
#define OD_PRIO (6)
```

Definition at line 33 of file threads.c.

4.41.1.8 OD_SZ

```
#define OD_SZ DEFAULT_SZ
```

Definition at line 32 of file threads.c.

4.41.2 Function Documentation

4.41.2.1 threads_init()

```
void threads_init (
    void )
```

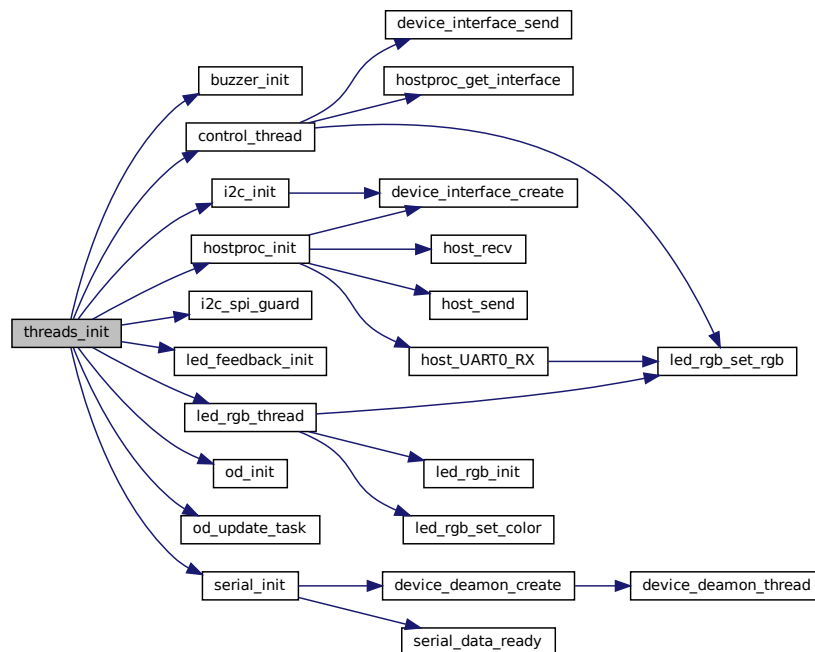
Initialize all the threads of Wildhorn AV.

This is the only function that needs to be called from the ST Auto-generated files. This is clever in case the autogeneration fails. This will minimize the code to be rewritten.

Definition at line 99 of file threads.c.

References buzzer_init(), control_handle, CONTROL_PRIO, CONTROL_SZ, control_thread(), CREATE_THREAD, ER_SUCCESS, hostproc_init(), i2c_init(), i2c_spi_guard(), led_feedback_init(), led_rgb_handle, LED_RGB_PRIO, LED_RGB_SZ, led_rgb_thread(), od_handle, od_init(), OD_PRIO, OD_SZ, od_update_task(), and serial_init().

Here is the call graph for this function:



4.41.3 Variable Documentation

4.41.3.1 control_handle

```
TaskHandle_t control_handle = NULL [static]
```

Definition at line 80 of file threads.c.

Referenced by threads_init().

4.41.3.2 led_rgb_handle

```
TaskHandle_t led_rgb_handle = NULL [static]
```

Definition at line 81 of file threads.c.

Referenced by threads_init().

4.41.3.3 od_handle

```
TaskHandle_t od_handle = NULL [static]
```

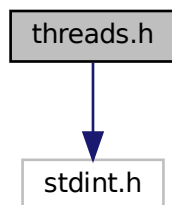
Definition at line 79 of file threads.c.

Referenced by threads_init().

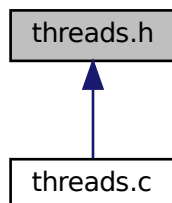
4.42 threads.h File Reference

```
#include <stdint.h>
```

Include dependency graph for threads.h:



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



Functions

- void [threads_init](#) (void)

Initialize all the threads of Wildhorn AV.

4.42.1 Function Documentation

4.42.1.1 threads_init()

```
void threads_init (
    void )
```

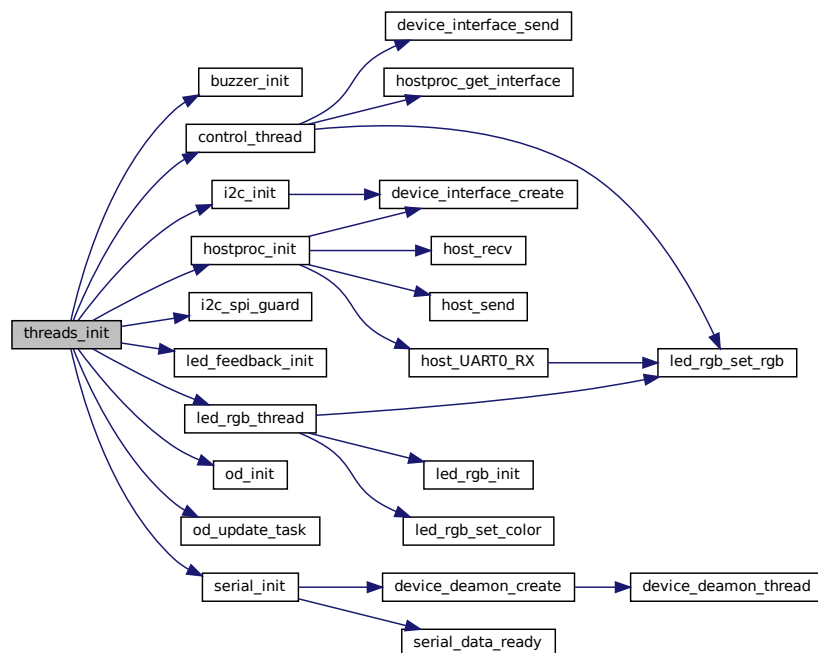
Initialize all the threads of Wildhorn AV.

This is the only function that needs to be called from the ST Auto-generated files. This is clever in case the autogeneration fails. This will minimize the code to be rewritten.

Definition at line 99 of file threads.c.

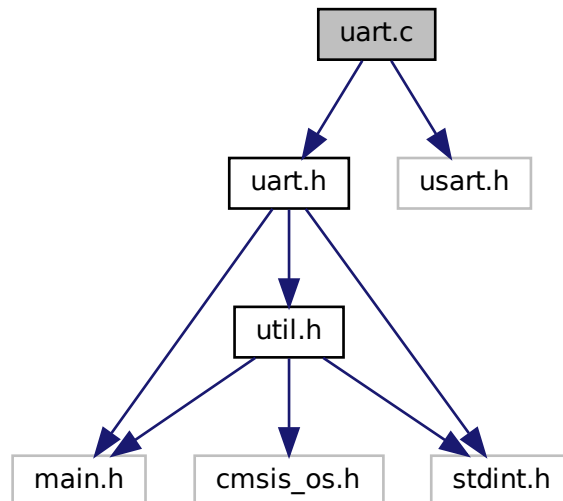
References [buzzer_init\(\)](#), [control_handle](#), [CONTROL_PRIO](#), [CONTROL_SZ](#), [control_thread\(\)](#), [CREATE_THREAD](#), [ER_SUCCESS](#), [hostproc_init\(\)](#), [i2c_init\(\)](#), [i2c_spi_guard\(\)](#), [led_feedback_init\(\)](#), [led_rgb_handle](#), [LED_RGB_PRIO](#), [LED_RGB_SZ](#), [led_rgb_thread\(\)](#), [od_handle](#), [od_init\(\)](#), [OD_PRIO](#), [OD_SZ](#), [od_update_task\(\)](#), and [serial_init\(\)](#).

Here is the call graph for this function:



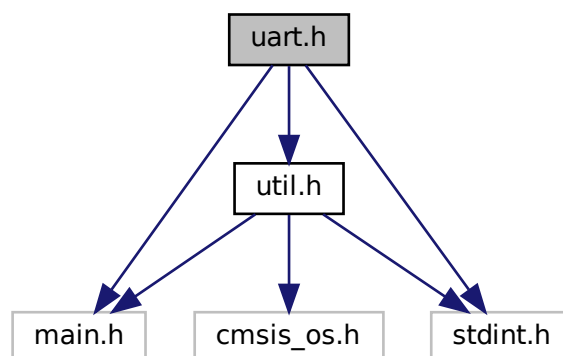
4.43 uart.c File Reference

```
#include "uart.h"  
#include <usart.h>  
Include dependency graph for uart.c:
```

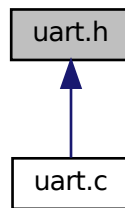


4.44 uart.h File Reference

```
#include <main.h>  
#include <stdint.h>  
#include <util.h>  
Include dependency graph for uart.h:
```



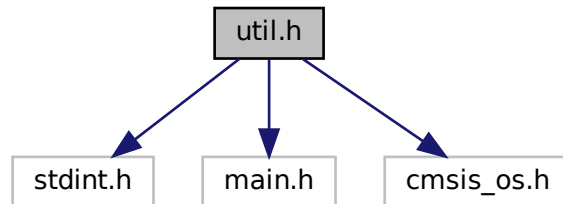
This graph shows which files directly or indirectly include this file:



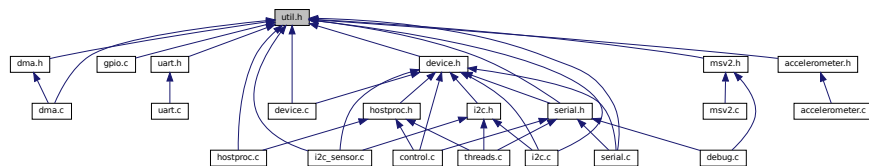
4.45 util.h File Reference

```
#include <stdint.h>
#include <main.h>
#include <cmsis_os.h>
```

Include dependency graph for util.h:



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



Data Structures

- struct [util_buffer_u8](#)
- struct [util_buffer_u16](#)
- struct [util_buffer_i16](#)

Macros

- #define `WRITE_IN_REG`(reg, mask, data) (reg) &= ~(mask); (reg) |= (data)
Macro to write masked data into a register.
- #define `ENTER_CRITICAL` taskENTER_CRITICAL
Macro to enter a critical section.
- #define `EXIT_CRITICAL` taskEXIT_CRITICAL
Macro to exit a critical section.
- #define `UTIL_GENERATE_BUFFER`(type, name)
- #define `util_abs`(a) ((a)<0?-(a):(a))

Typedefs

- typedef enum `util_error` `util_error_t`
Unified error codes for the whole WildhornAV project.
- typedef struct `util_buffer_u8` `util_buffer_u8_t`
- typedef struct `util_buffer_u16` `util_buffer_u16_t`
- typedef struct `util_buffer_i16` `util_buffer_i16_t`

Enumerations

- enum `util_error` {
`ER_SUCCESS` = 0 , `ER_DATA_NOT_RDY` = 0<<1 , `ER_FAILURE` = 1<<1 , `ER_OUT_OF_RANGE` = 1<<2
, `ER_TIMEOUT` = 1<<3 , `ER_RESSOURCE_ERROR` = 1<<4 }
Unified error codes for the whole WildhornAV project.

Functions

- static void `util_encode_u8` (uint8_t *data, uint8_t value)
- static void `util_encode_u16` (uint8_t *data, uint16_t value)
- static void `util_encode_u32` (uint8_t *data, uint32_t value)
- static void `util_encode_i8` (uint8_t *data, int8_t value)
- static void `util_encode_i16` (uint8_t *data, int16_t value)
- static void `util_encode_i32` (uint8_t *data, int32_t value)
- static uint8_t `util_decode_u8` (uint8_t *data)
- static uint16_t `util_decode_u16` (uint8_t *data)
- static uint32_t `util_decode_u32` (uint8_t *data)
- static int8_t `util_decode_i8` (uint8_t *data)
- static int16_t `util_decode_i16` (uint8_t *data)
- static int32_t `util_decode_i32` (uint8_t *data)
- static void `util_buffer_u8_init` (`util_buffer_u8_t` *bfr, uint8_t *buffer, uint16_t bfr_len)
- static void `util_buffer_u8_add` (`util_buffer_u8_t` *bfr, uint8_t d)
- static uint8_t `util_buffer_u8_get` (`util_buffer_u8_t` *bfr)
- static uint8_t `util_buffer_u8_access` (`util_buffer_u8_t` *bfr, int16_t ix)
- static uint8_t `util_buffer_u8_isempty` (`util_buffer_u8_t` *bfr)
- static void `util_buffer_u16_init` (`util_buffer_u16_t` *bfr, uint16_t *buffer, uint16_t bfr_len)
- static void `util_buffer_u16_add` (`util_buffer_u16_t` *bfr, uint16_t d)
- static uint16_t `util_buffer_u16_get` (`util_buffer_u16_t` *bfr)
- static uint8_t `util_buffer_u16_isempty` (`util_buffer_u16_t` *bfr)
- static void `util_buffer_i16_init` (`util_buffer_i16_t` *bfr, int16_t *buffer, uint16_t bfr_len)
- static void `util_buffer_i16_add` (`util_buffer_i16_t` *bfr, int16_t d)
- static int16_t `util_buffer_i16_get` (`util_buffer_i16_t` *bfr)
- static uint8_t `util_buffer_i16_isempty` (`util_buffer_i16_t` *bfr)

4.45.1 Macro Definition Documentation

4.45.1.1 ENTER_CRITICAL

```
#define ENTER_CRITICAL taskENTER_CRITICAL
```

Macro to enter a critical section.

Definition at line 51 of file util.h.

4.45.1.2 EXIT_CRITICAL

```
#define EXIT_CRITICAL taskEXIT_CRITICAL
```

Macro to exit a critical section.

Definition at line 56 of file util.h.

4.45.1.3 util_abs

```
#define util_abs(  
    a ) ((a)<0?- (a) : (a))
```

Definition at line 327 of file util.h.

4.45.1.4 UTIL_GENERATE_BUFFER

```
#define UTIL_GENERATE_BUFFER(  
    type,  
    name )
```

Value:

```
typedef struct UTIL_BUFFER_##name{  
    uint16_t c_ix;  
    uint16_t l_ix;  
    uint16_t bfr_len;  
    type * buffer;  
}UTIL_BUFFER_##name##_t;  
static inline void util_buffer_##name##_init(UTIL_BUFFER_##name##_t * bfr, type * buffer, uint16_t bfr_len)  
{  
    bfr->c_ix = 0;  
}
```

```

    bfr->l_ix = 0;
    bfr->bfr_len = bfr_len;
    bfr->buffer = buffer;
}

static inline void util_buffer_##name##_add(UTIL_BUFFER_##name##_t * bfr, type d) {
    bfr->buffer[bfr->c_ix++] = d;
    if(bfr->c_ix == bfr->bfr_len) bfr->c_ix = 0;
}

static inline type util_buffer_##name##_get(UTIL_BUFFER_##name##_t * bfr) {
    type tmp = bfr->buffer[bfr->l_ix++];
    if(bfr->l_ix == bfr->bfr_len) bfr->l_ix=0;
    return tmp;
}

static inline type util_buffer_##name##_access(UTIL_BUFFER_##name##_t * bfr, uint16_t ix) {
    int16_t i = bfr->c_ix - ix - 1;
    while(i < 0) i += bfr->bfr_len;
    return bfr->buffer[i];
}

static inline uint8_t util_buffer_##name##_isempty(UTIL_BUFFER_##name##_t * bfr) {
    return bfr->l_ix == bfr->c_ix;
}

```

Definition at line 295 of file util.h.

4.45.1.5 WRITE_IN_REG

```

#define WRITE_IN_REG(
    reg,
    mask,
    data ) (reg) &= ~(mask); (reg) |= (data)

```

Macro to write masked data into a register.

Definition at line 36 of file util.h.

4.45.2 Typedef Documentation

4.45.2.1 util_buffer_i16_t

```
typedef struct util_buffer_i16 util_buffer_i16_t
```

4.45.2.2 util_buffer_u16_t

```
typedef struct util_buffer_u16 util_buffer_u16_t
```

4.45.2.3 util_buffer_u8_t

```
typedef struct util_buffer_u8 util_buffer_u8_t
```

4.45.2.4 util_error_t

```
typedef enum util_error util_error_t
```

Unified error codes for the whole WildhornAV project.

Note

The error codes can be ORed together to create more complex errors.

4.45.3 Enumeration Type Documentation

4.45.3.1 util_error

```
enum util_error
```

Unified error codes for the whole WildhornAV project.

Note

The error codes can be ORed together to create more complex errors.

Enumerator

ER_SUCCESS	Operation completed successfully
ER_DATA_NOT_RDY	Error due to lack of readiness
ER_FAILURE	Error due to a generic failure
ER_OUT_OF_RANGE	Error due to a range issue
ER_TIMEOUT	Error due to a timeout
ER_RESSOURCE_ERROR	Error due to a ressource issue

Definition at line 66 of file util.h.

4.45.4 Function Documentation

4.45.4.1 util_buffer_i16_add()

```
static void util_buffer_i16_add (  
    util_buffer_i16_t * bfr,  
    int16_t d ) [inline], [static]
```

Definition at line 278 of file util.h.

References util_buffer_i16::bfr_len, util_buffer_i16::buffer, and util_buffer_i16::c_ix.

4.45.4.2 util_buffer_i16_get()

```
static int16_t util_buffer_i16_get (  
    util_buffer_i16_t * bfr ) [inline], [static]
```

Definition at line 283 of file util.h.

References util_buffer_i16::bfr_len, util_buffer_i16::buffer, and util_buffer_i16::l_ix.

4.45.4.3 util_buffer_i16_init()

```
static void util_buffer_i16_init (  
    util_buffer_i16_t * bfr,  
    int16_t * buffer,  
    uint16_t bfr_len ) [inline], [static]
```

Definition at line 271 of file util.h.

References util_buffer_i16::bfr_len, util_buffer_i16::buffer, util_buffer_i16::c_ix, and util_buffer_i16::l_ix.

4.45.4.4 util_buffer_i16_isempty()

```
static uint8_t util_buffer_i16_isempty (  
    util_buffer_i16_t * bfr ) [inline], [static]
```

Definition at line 289 of file util.h.

References util_buffer_i16::c_ix, and util_buffer_i16::l_ix.

4.45.4.5 util_buffer_u16_add()

```
static void util_buffer_u16_add (
    util_buffer_u16_t * bfr,
    uint16_t d ) [inline], [static]
```

Definition at line 255 of file util.h.

References util_buffer_u16::bfr_len, util_buffer_u16::buffer, and util_buffer_u16::c_ix.

4.45.4.6 util_buffer_u16_get()

```
static uint16_t util_buffer_u16_get (
    util_buffer_u16_t * bfr ) [inline], [static]
```

Definition at line 260 of file util.h.

References util_buffer_u16::bfr_len, util_buffer_u16::buffer, and util_buffer_u16::l_ix.

4.45.4.7 util_buffer_u16_init()

```
static void util_buffer_u16_init (
    util_buffer_u16_t * bfr,
    uint16_t * buffer,
    uint16_t bfr_len ) [inline], [static]
```

Definition at line 248 of file util.h.

References util_buffer_u16::bfr_len, util_buffer_u16::buffer, util_buffer_u16::c_ix, and util_buffer_u16::l_ix.

4.45.4.8 util_buffer_u16_isempty()

```
static uint8_t util_buffer_u16_isempty (
    util_buffer_u16_t * bfr ) [inline], [static]
```

Definition at line 266 of file util.h.

References util_buffer_u16::c_ix, and util_buffer_u16::l_ix.

4.45.4.9 util_buffer_u8_access()

```
static uint8_t util_buffer_u8_access (  
    util_buffer_u8_t * bfr,  
    int16_t ix ) [inline], [static]
```

Definition at line 237 of file util.h.

References util_buffer_u8::bfr_len, util_buffer_u8::buffer, and util_buffer_u8::c_ix.

4.45.4.10 util_buffer_u8_add()

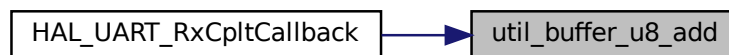
```
static void util_buffer_u8_add (  
    util_buffer_u8_t * bfr,  
    uint8_t d ) [inline], [static]
```

Definition at line 226 of file util.h.

References util_buffer_u8::bfr_len, util_buffer_u8::buffer, and util_buffer_u8::c_ix.

Referenced by HAL_UART_RxCpltCallback().

Here is the caller graph for this function:



4.45.4.11 util_buffer_u8_get()

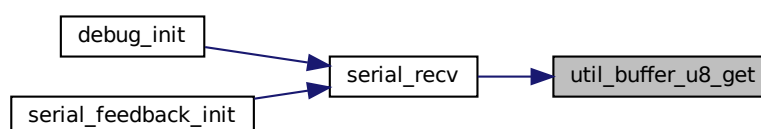
```
static uint8_t util_buffer_u8_get (  
    util_buffer_u8_t * bfr ) [inline], [static]
```

Definition at line 231 of file util.h.

References util_buffer_u8::bfr_len, util_buffer_u8::buffer, and util_buffer_u8::l_ix.

Referenced by serial_rcv().

Here is the caller graph for this function:



4.45.4.12 util_buffer_u8_init()

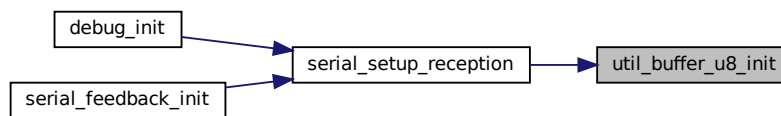
```
static void util_buffer_u8_init (
    util_buffer_u8_t * bfr,
    uint8_t * buffer,
    uint16_t bfr_len ) [inline], [static]
```

Definition at line 219 of file util.h.

References util_buffer_u8::bfr_len, util_buffer_u8::buffer, util_buffer_u8::c_ix, and util_buffer_u8::l_ix.

Referenced by serial_setup_reception().

Here is the caller graph for this function:



4.45.4.13 util_buffer_u8_isempty()

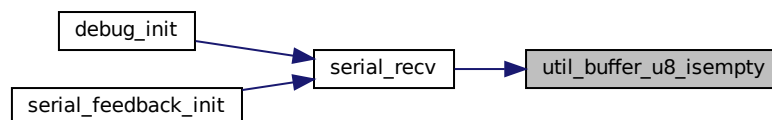
```
static uint8_t util_buffer_u8_isempty (
    util_buffer_u8_t * bfr ) [inline], [static]
```

Definition at line 243 of file util.h.

References util_buffer_u8::c_ix, and util_buffer_u8::l_ix.

Referenced by serial_rcv().

Here is the caller graph for this function:



4.45.4.14 util_decode_i16()

```
static int16_t util_decode_i16 (  
    uint8_t * data )    [inline], [static]
```

Definition at line 209 of file util.h.

Referenced by device_read_i16().

Here is the caller graph for this function:



4.45.4.15 util_decode_i32()

```
static int32_t util_decode_i32 (  
    uint8_t * data )    [inline], [static]
```

Definition at line 213 of file util.h.

Referenced by device_read_i32().

Here is the caller graph for this function:



4.45.4.16 util_decode_i8()

```
static int8_t util_decode_i8 (  
    uint8_t * data ) [inline], [static]
```

Definition at line 205 of file util.h.

Referenced by device_read_i8().

Here is the caller graph for this function:



4.45.4.17 util_decode_u16()

```
static uint16_t util_decode_u16 (  
    uint8_t * data ) [inline], [static]
```

Definition at line 197 of file util.h.

Referenced by device_read_u16().

Here is the caller graph for this function:



4.45.4.18 util_decode_u32()

```
static uint32_t util_decode_u32 (  
    uint8_t * data )    [inline], [static]
```

Definition at line 201 of file util.h.

Referenced by device_read_u32().

Here is the caller graph for this function:



4.45.4.19 util_decode_u8()

```
static uint8_t util_decode_u8 (  
    uint8_t * data )    [inline], [static]
```

Definition at line 193 of file util.h.

Referenced by device_read_u8().

Here is the caller graph for this function:



4.45.4.20 util_encode_i16()

```
static void util_encode_i16 (  
    uint8_t * data,  
    int16_t value ) [inline], [static]
```

Definition at line 181 of file util.h.

Referenced by device_write_i16().

Here is the caller graph for this function:



4.45.4.21 util_encode_i32()

```
static void util_encode_i32 (  
    uint8_t * data,  
    int32_t value ) [inline], [static]
```

Definition at line 186 of file util.h.

Referenced by device_write_i32().

Here is the caller graph for this function:



4.45.4.22 util_encode_i8()

```
static void util_encode_i8 (  
    uint8_t * data,  
    int8_t value ) [inline], [static]
```

Definition at line 177 of file util.h.

Referenced by device_write_i8().

Here is the caller graph for this function:



4.45.4.23 util_encode_u16()

```
static void util_encode_u16 (  
    uint8_t * data,  
    uint16_t value ) [inline], [static]
```

Definition at line 165 of file util.h.

Referenced by device_write_u16().

Here is the caller graph for this function:



4.45.4.24 util_encode_u32()

```
static void util_encode_u32 (  
    uint8_t * data,  
    uint32_t value ) [inline], [static]
```

Definition at line 170 of file util.h.

Referenced by device_write_u32().

Here is the caller graph for this function:



4.45.4.25 util_encode_u8()

```
static void util_encode_u8 (  
    uint8_t * data,  
    uint8_t value ) [inline], [static]
```

Definition at line 161 of file util.h.

Referenced by device_write_u8().

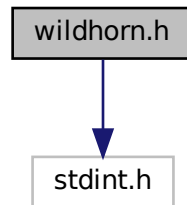
Here is the caller graph for this function:



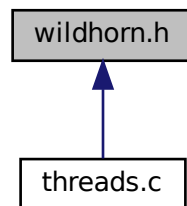
4.46 wildhorn.h File Reference

```
#include <stdint.h>
```

Include dependency graph for wildhorn.h:



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



Macros

- #define `WH_TRUE` 1
- #define `WH_FALSE` 0
- #define `WH_HAS_SENSORS` `WH_TRUE`
- #define `WH_HAS_FEEDBACK` `WH_TRUE`
- #define `WH_HAS_RADIO` `WH_FALSE`
- #define `WH_HAS_GNSS` `WH_FALSE`
- #define `WH_HAS_KRTEK` `WH_FALSE`
- #define `WH_USE_BUZZER` `WH_FALSE`

4.46.1 Macro Definition Documentation

4.46.1.1 WH_FALSE

```
#define WH_FALSE 0
```

Definition at line 28 of file wildhorn.h.

4.46.1.2 WH_HAS_FEEDBACK

```
#define WH_HAS_FEEDBACK WH_TRUE
```

Definition at line 32 of file wildhorn.h.

4.46.1.3 WH_HAS_GNSS

```
#define WH_HAS_GNSS WH_FALSE
```

Definition at line 34 of file wildhorn.h.

4.46.1.4 WH_HAS_KRTEK

```
#define WH_HAS_KRTEK WH_FALSE
```

Definition at line 35 of file wildhorn.h.

4.46.1.5 WH_HAS_RADIO

```
#define WH_HAS_RADIO WH_FALSE
```

Definition at line 33 of file wildhorn.h.

4.46.1.6 WH_HAS_SENSORS

```
#define WH_HAS_SENSORS WH_TRUE
```

Definition at line 31 of file wildhorn.h.

4.46.1.7 WH_TRUE

```
#define WH_TRUE 1
```

Definition at line 27 of file wildhorn.h.

4.46.1.8 WH_USE_BUZZER

```
#define WH_USE_BUZZER WH_FALSE
```

Definition at line 38 of file wildhorn.h.

Index

- A0
 - [note.h](#), [181](#)
- A0H
 - [note.h](#), [181](#)
- A1
 - [note.h](#), [181](#)
- A1H
 - [note.h](#), [181](#)
- A2
 - [note.h](#), [182](#)
- A2H
 - [note.h](#), [182](#)
- A3
 - [note.h](#), [182](#)
- A3H
 - [note.h](#), [182](#)
- A4
 - [note.h](#), [182](#)
- A4H
 - [note.h](#), [182](#)
- A5
 - [note.h](#), [183](#)
- A5H
 - [note.h](#), [183](#)
- A6
 - [note.h](#), [183](#)
- A6H
 - [note.h](#), [183](#)
- A7
 - [note.h](#), [183](#)
- A7H
 - [note.h](#), [183](#)
- A8
 - [note.h](#), [184](#)
- A8H
 - [note.h](#), [184](#)
- accelerometer.c, [43](#)
 - [accelerometer_init](#), [43](#)
- accelerometer.h, [44](#)
 - [accelerometer_init](#), [45](#)
- accelerometer_init
 - [accelerometer.c](#), [43](#)
 - [accelerometer.h](#), [45](#)
- ALLOCATE_OD_ENTRY
 - [od.c](#), [201](#)
- alternate
 - [gpio_config](#), [22](#)
- b
 - [led_color](#), [25](#)
- B0
 - [note.h](#), [184](#)
- B1
 - [note.h](#), [184](#)
- B2
 - [note.h](#), [184](#)
- B3
 - [note.h](#), [184](#)
- B4
 - [note.h](#), [185](#)
- B5
 - [note.h](#), [185](#)
- B6
 - [note.h](#), [185](#)
- B7
 - [note.h](#), [185](#)
- B8
 - [note.h](#), [185](#)
- barometer.c, [45](#)
- barometer.h, [45](#)
- bfr_len
 - [util_buffer_i16](#), [38](#)
 - [util_buffer_u16](#), [39](#)
 - [util_buffer_u8](#), [41](#)
- bias
 - [gpio_config](#), [22](#)
- blink_sequence
 - [led.c](#), [161](#)
- blink_sequence_len
 - [led.c](#), [162](#)
- buffer
 - [device_deamon](#), [10](#)
 - [util_buffer_i16](#), [38](#)
 - [util_buffer_u16](#), [39](#)
 - [util_buffer_u8](#), [41](#)
- buzzer.c, [46](#)
 - [buzzer_disable](#), [49](#)
 - [buzzer_enable](#), [49](#)
 - [buzzer_init](#), [49](#)
 - [buzzer_note_interrupt](#), [50](#)
 - [BUZZER_PIN](#), [47](#)
 - [BUZZER_PORT](#), [47](#)
 - [buzzer_rytm_interrupt](#), [50](#)
 - [COMPUTE_NOTE](#), [47](#)
 - [COMPUTE_RYTM](#), [47](#)
 - [melody_active](#), [51](#)
 - [melody_state](#), [51](#)
 - [NOTE_PRESC](#), [47](#)
 - [NOTE_TIMER](#), [48](#)

- NOTE_TIMER_DEV, [48](#)
- RYTM_MS, [48](#)
- RYTM_PRESC, [48](#)
- RYTM_TIMER, [48](#)
- RYTM_TIMER_DEV, [48](#)
- state, [51](#)
- TIMER_FREQ, [49](#)
- TIMER_TRIM, [49](#)
- buzzer.h, [52](#)
 - buzzer_disable, [53](#)
 - buzzer_enable, [53](#)
 - buzzer_init, [53](#)
 - buzzer_note_interrupt, [53](#)
 - buzzer_rytm_interrupt, [54](#)
- buzzer_disable
 - buzzer.c, [49](#)
 - buzzer.h, [53](#)
- buzzer_enable
 - buzzer.c, [49](#)
 - buzzer.h, [53](#)
- buzzer_init
 - buzzer.c, [49](#)
 - buzzer.h, [53](#)
- buzzer_note_interrupt
 - buzzer.c, [50](#)
 - buzzer.h, [53](#)
- BUZZER_PIN
 - buzzer.c, [47](#)
- BUZZER_PORT
 - buzzer.c, [47](#)
- buzzer_rytm_interrupt
 - buzzer.c, [50](#)
 - buzzer.h, [54](#)
- C0
 - note.h, [185](#)
- C0H
 - note.h, [186](#)
- C1
 - note.h, [186](#)
- C1H
 - note.h, [186](#)
- C2
 - note.h, [186](#)
- C2H
 - note.h, [186](#)
- C3
 - note.h, [186](#)
- C3H
 - note.h, [187](#)
- C4
 - note.h, [187](#)
- C4H
 - note.h, [187](#)
- C5
 - note.h, [187](#)
- C5H
 - note.h, [187](#)
- C6
 - note.h, [187](#)
- C6H
 - note.h, [188](#)
- C7
 - note.h, [188](#)
- C7H
 - note.h, [188](#)
- C8
 - note.h, [188](#)
- C8H
 - note.h, [188](#)
- c_ix
 - util_buffer_i16, [38](#)
 - util_buffer_u16, [40](#)
 - util_buffer_u8, [41](#)
- calc_field_CRC
 - msv2.c, [171](#)
- color_sequence
 - led.c, [162](#)
- color_sequence_len
 - led.c, [162](#)
- COMPUTE_NOTE
 - buzzer.c, [47](#)
- COMPUTE_RYTM
 - buzzer.c, [47](#)
- context
 - device, [7](#)
 - device_deamon, [10](#)
 - device_interface, [12](#)
- control, [5](#)
 - control.c, [66](#)
 - state, [5](#)
- control.c, [55](#)
 - control, [66](#)
 - CONTROL_ABORT, [58](#)
 - control_abort_run, [58](#)
 - control_abort_start, [58](#)
 - CONTROL_APOGEE, [58](#)
 - control_apogee_run, [58](#)
 - control_apogee_start, [59](#)
 - CONTROL_ARMED, [58](#)
 - control_armed_run, [59](#)
 - control_armed_start, [59](#)
 - CONTROL_BALLISTIC, [58](#)
 - control_ballistic_run, [59](#)
 - control_ballistic_start, [60](#)
 - CONTROL_CALIBRATION, [58](#)
 - control_calibration_run, [60](#)
 - control_calibration_start, [60](#)
 - CONTROL_COAST, [58](#)
 - control_coast_run, [60](#)
 - control_coast_start, [61](#)
 - CONTROL_DROGUE, [58](#)
 - control_drogue_run, [61](#)
 - control_drogue_start, [61](#)
 - CONTROL_ERROR, [58](#)
 - control_error_run, [61](#)
 - control_error_start, [62](#)

CONTROL_EVENT, 58
control_event_run, 62
control_event_start, 62
CONTROL_HEART_BEAT, 57
CONTROL_IDLE, 58
control_idle_run, 62
control_idle_start, 63
CONTROL_MAIN, 58
control_main_run, 63
control_main_start, 63
CONTROL_POWERED, 58
control_powered_run, 63
control_powered_start, 64
control_state, 57
control_state_t, 57
CONTROL_SUPERSONIC, 58
control_supersonic_run, 64
control_supersonic_start, 64
control_t, 57
control_thread, 64
CONTROL_TOUCHDOWN, 58
control_touchdown_run, 65
control_touchdown_start, 65
control.h, 66
 control_thread, 67
CONTROL_ABORT
 control.c, 58
control_abort_run
 control.c, 58
control_abort_start
 control.c, 58
CONTROL_APOGEE
 control.c, 58
control_apogee_run
 control.c, 58
control_apogee_start
 control.c, 59
CONTROL_ARMED
 control.c, 58
control_armed_run
 control.c, 59
control_armed_start
 control.c, 59
CONTROL_BALLISTIC
 control.c, 58
control_ballistic_run
 control.c, 59
control_ballistic_start
 control.c, 60
CONTROL_CALIBRATION
 control.c, 58
control_calibration_run
 control.c, 60
control_calibration_start
 control.c, 60
CONTROL_COAST
 control.c, 58
control_coast_run
 control.c, 60
control_coast_start
 control.c, 61
CONTROL_DROGUE
 control.c, 58
control_drogue_run
 control.c, 61
control_drogue_start
 control.c, 61
CONTROL_ERROR
 control.c, 58
control_error_run
 control.c, 61
control_error_start
 control.c, 62
CONTROL_EVENT
 control.c, 58
control_event_run
 control.c, 62
control_event_start
 control.c, 62
control_handle
 threads.c, 230
CONTROL_HEART_BEAT
 control.c, 57
CONTROL_IDLE
 control.c, 58
control_idle_run
 control.c, 62
control_idle_start
 control.c, 63
CONTROL_MAIN
 control.c, 58
control_main_run
 control.c, 63
control_main_start
 control.c, 63
CONTROL_POWERED
 control.c, 58
control_powered_run
 control.c, 63
control_powered_start
 control.c, 64
CONTROL_PRIO
 threads.c, 227
control_state
 control.c, 57
control_state_t
 control.c, 57
CONTROL_SUPERSONIC
 control.c, 58
control_supersonic_run
 control.c, 64
control_supersonic_start
 control.c, 64
CONTROL_SZ
 threads.c, 227
control_t

- control.c, [57](#)
- control_thread
 - control.c, [64](#)
 - control.h, [67](#)
- CONTROL_TOUCHDOWN
 - control.c, [58](#)
- control_touchdown_run
 - control.c, [65](#)
- control_touchdown_start
 - control.c, [65](#)
- counter
 - MSV2_RX_DATA, [28](#)
- crc
 - MSV2_RX_DATA, [28](#)
 - MSV2_TX_DATA, [30](#)
- crc_data
 - MSV2_RX_DATA, [28](#)
 - MSV2_TX_DATA, [30](#)
- CREATE_THREAD
 - threads.c, [228](#)
- D0
 - note.h, [188](#)
- D0H
 - note.h, [189](#)
- D1
 - note.h, [189](#)
- D1H
 - note.h, [189](#)
- D2
 - note.h, [189](#)
- D2H
 - note.h, [189](#)
- D3
 - note.h, [189](#)
- D3H
 - note.h, [190](#)
- D4
 - note.h, [190](#)
- D4H
 - note.h, [190](#)
- D5
 - note.h, [190](#)
- D5H
 - note.h, [190](#)
- D6
 - note.h, [190](#)
- D6H
 - note.h, [191](#)
- D7
 - note.h, [191](#)
- D7H
 - note.h, [191](#)
- D8
 - note.h, [191](#)
- D8H
 - note.h, [191](#)
- data
 - MSV2_RX_DATA, [28](#)
 - MSV2_TX_DATA, [30](#)
 - od_entry_t, [32](#)
 - od_frame_t, [33](#)
- data_id
 - od_entry_t, [32](#)
 - od_frame_t, [33](#)
- data_len
 - MSV2_RX_DATA, [29](#)
 - MSV2_TX_DATA, [31](#)
- data_rdy
 - device_deamon, [10](#)
- DEAMON_STACK_SIZE
 - device.h, [84](#)
- debug.c, [67](#)
 - debug_context, [69](#)
 - debug_context_t, [68](#)
 - debug_init, [68](#)
 - debug_interface_context, [69](#)
 - debug_interface_context_t, [68](#)
 - feedback_interface_context, [69](#)
- debug.h, [70](#)
- debug_context, [6](#)
 - debug.c, [69](#)
- debug_context_t
 - debug.c, [68](#)
- debug_init
 - debug.c, [68](#)
- debug_interface_context, [6](#)
 - debug.c, [69](#)
- msv2, [6](#)
- debug_interface_context_t
 - debug.c, [68](#)
- DEBUG_NO_CAN
 - od.c, [201](#)
- DECLARE_OD_ENTRY
 - od.h, [205](#)
- DEFAULT_SZ
 - threads.c, [228](#)
- device, [7](#)
 - context, [7](#)
 - id, [8](#)
 - interface, [8](#)
 - read_reg, [8](#)
 - write_reg, [8](#)
- device.c, [70](#)
 - device_create, [72](#)
 - device_deamon_create, [73](#)
 - device_deamon_thread, [74](#)
 - device_interface_create, [74](#)
 - device_interface_recv, [75](#)
 - device_interface_send, [75](#)
 - device_read_i16, [76](#)
 - device_read_i32, [76](#)
 - device_read_i8, [77](#)
 - device_read_u16, [77](#)
 - device_read_u32, [78](#)
 - device_read_u8, [78](#)
 - device_write_i16, [79](#)

- device_write_i32, [79](#)
- device_write_i8, [80](#)
- device_write_u16, [80](#)
- device_write_u32, [81](#)
- device_write_u8, [81](#)
- LEN_16, [72](#)
- LEN_32, [72](#)
- LEN_8, [72](#)
- device.h, [82](#)
 - DEAMON_STACK_SIZE, [84](#)
 - device_create, [85](#)
 - device_daemon_create, [86](#)
 - device_daemon_t, [84](#)
 - device_interface_create, [86](#)
 - device_interface_recv, [87](#)
 - device_interface_send, [87](#)
 - device_interface_t, [84](#)
 - DEVICE_MAX_INTERFACES_PER_DEAMON, [84](#)
 - DEVICE_NAME_LEN, [84](#)
 - device_read_i16, [88](#)
 - device_read_i32, [88](#)
 - device_read_i8, [89](#)
 - device_read_u16, [89](#)
 - device_read_u32, [90](#)
 - device_read_u8, [90](#)
 - device_t, [84](#)
 - device_write_i16, [91](#)
 - device_write_i32, [91](#)
 - device_write_i8, [92](#)
 - device_write_u16, [92](#)
 - device_write_u32, [93](#)
 - device_write_u8, [93](#)
- device_address
 - i2c_sensor_context, [25](#)
- device_create
 - device.c, [72](#)
 - device.h, [85](#)
- device_daemon, [9](#)
 - buffer, [10](#)
 - context, [10](#)
 - data_rdy, [10](#)
 - handle, [10](#)
 - id, [10](#)
 - interfaces, [10](#)
 - interfaces_count, [11](#)
 - stack, [11](#)
- device_daemon_create
 - device.c, [73](#)
 - device.h, [86](#)
- device_daemon_t
 - device.h, [84](#)
- device_daemon_thread
 - device.c, [74](#)
- device_interface, [11](#)
 - context, [12](#)
 - handle_data, [12](#)
 - id, [12](#)
 - recv, [12](#)
 - send, [12](#)
- device_interface_create
 - device.c, [74](#)
 - device.h, [86](#)
- device_interface_recv
 - device.c, [75](#)
 - device.h, [87](#)
- device_interface_send
 - device.c, [75](#)
 - device.h, [87](#)
- device_interface_t
 - device.h, [84](#)
- DEVICE_MAX_INTERFACES_PER_DEAMON
 - device.h, [84](#)
- DEVICE_NAME_LEN
 - device.h, [84](#)
- device_read_i16
 - device.c, [76](#)
 - device.h, [88](#)
- device_read_i32
 - device.c, [76](#)
 - device.h, [88](#)
- device_read_i8
 - device.c, [77](#)
 - device.h, [89](#)
- device_read_u16
 - device.c, [77](#)
 - device.h, [89](#)
- device_read_u32
 - device.c, [78](#)
 - device.h, [90](#)
- device_read_u8
 - device.c, [78](#)
 - device.h, [90](#)
- device_t
 - device.h, [84](#)
- device_write_i16
 - device.c, [79](#)
 - device.h, [91](#)
- device_write_i32
 - device.c, [79](#)
 - device.h, [91](#)
- device_write_i8
 - device.c, [80](#)
 - device.h, [92](#)
- device_write_u16
 - device.c, [80](#)
 - device.h, [92](#)
- device_write_u32
 - device.c, [81](#)
 - device.h, [93](#)
- device_write_u8
 - device.c, [81](#)
 - device.h, [93](#)
- direction
 - dma_stream_config, [16](#)
- DLE
 - msv2.c, [170](#)

- dma
 - dma_stream_dev, [19](#)
- dma.c, [94](#)
 - dma2_get_scheduler, [95](#)
 - dma2_get_streams, [95](#)
 - dma2_init_scheduler, [95](#)
 - dma2_scheduler, [97](#)
 - dma2_streams, [97](#)
 - dma_handle_interrupt, [96](#)
 - dma_scheduler_init, [96](#)
 - dma_scheduler_release_stream, [96](#)
 - dma_scheduler_request_stream, [97](#)
 - dma_start_stream, [97](#)
- dma.h, [98](#)
 - dma2_get_scheduler, [125](#)
 - dma2_get_streams, [125](#)
 - dma2_init_scheduler, [125](#)
 - dma_copy, [126](#)
 - dma_request_t, [124](#)
 - dma_scheduler_dev_t, [124](#)
 - dma_scheduler_init, [126](#)
 - dma_scheduler_release_stream, [126](#)
 - dma_scheduler_request_stream, [127](#)
 - dma_start_stream, [127](#)
 - DMA_STATUS_TC, [102](#)
 - DMA_STATUS_TE, [102](#)
 - DMA_STATUS_TH, [102](#)
 - dma_stop_stream, [127](#)
 - DMA_STREAM_BUSY, [125](#)
 - dma_stream_config_t, [124](#)
 - dma_stream_dev_t, [124](#)
 - dma_stream_dir, [124](#)
 - dma_stream_dir_t, [124](#)
 - DMA_STREAM_FREE, [125](#)
 - DMA_STREAM_M2M, [125](#)
 - DMA_STREAM_M2P, [125](#)
 - DMA_STREAM_P2M, [125](#)
 - dma_stream_state, [125](#)
 - dma_stream_state_t, [124](#)
 - DMA_STREAMS_MAX_LEN, [102](#)
 - STM32_DMAMUX1_ADC1, [102](#)
 - STM32_DMAMUX1_ADC2, [103](#)
 - STM32_DMAMUX1_CRY2_IN, [103](#)
 - STM32_DMAMUX1_CRY2_OUT, [103](#)
 - STM32_DMAMUX1_DAC1_CH1, [103](#)
 - STM32_DMAMUX1_DAC1_CH2, [103](#)
 - STM32_DMAMUX1_DCMI, [103](#)
 - STM32_DMAMUX1_DFSDM1_FLT0, [104](#)
 - STM32_DMAMUX1_DFSDM1_FLT1, [104](#)
 - STM32_DMAMUX1_DFSDM1_FLT2, [104](#)
 - STM32_DMAMUX1_DFSDM1_FLT3, [104](#)
 - STM32_DMAMUX1_DFSDM1_FLT4, [104](#)
 - STM32_DMAMUX1_DFSDM1_FLT5, [104](#)
 - STM32_DMAMUX1_HASH2_IN, [105](#)
 - STM32_DMAMUX1_I2C1_RX, [105](#)
 - STM32_DMAMUX1_I2C1_TX, [105](#)
 - STM32_DMAMUX1_I2C2_RX, [105](#)
 - STM32_DMAMUX1_I2C2_TX, [105](#)
 - STM32_DMAMUX1_I2C3_RX, [105](#)
 - STM32_DMAMUX1_I2C3_TX, [106](#)
 - STM32_DMAMUX1_I2C5_RX, [106](#)
 - STM32_DMAMUX1_I2C5_TX, [106](#)
 - STM32_DMAMUX1_REQ_GEN0, [106](#)
 - STM32_DMAMUX1_REQ_GEN1, [106](#)
 - STM32_DMAMUX1_REQ_GEN2, [106](#)
 - STM32_DMAMUX1_REQ_GEN3, [107](#)
 - STM32_DMAMUX1_REQ_GEN4, [107](#)
 - STM32_DMAMUX1_REQ_GEN5, [107](#)
 - STM32_DMAMUX1_REQ_GEN6, [107](#)
 - STM32_DMAMUX1_REQ_GEN7, [107](#)
 - STM32_DMAMUX1_RSVD117, [107](#)
 - STM32_DMAMUX1_RSVD118, [108](#)
 - STM32_DMAMUX1_RSVD119, [108](#)
 - STM32_DMAMUX1_RSVD120, [108](#)
 - STM32_DMAMUX1_RSVD121, [108](#)
 - STM32_DMAMUX1_RSVD122, [108](#)
 - STM32_DMAMUX1_RSVD123, [108](#)
 - STM32_DMAMUX1_RSVD124, [109](#)
 - STM32_DMAMUX1_RSVD125, [109](#)
 - STM32_DMAMUX1_RSVD126, [109](#)
 - STM32_DMAMUX1_RSVD127, [109](#)
 - STM32_DMAMUX1_RSVD41, [109](#)
 - STM32_DMAMUX1_RSVD42, [109](#)
 - STM32_DMAMUX1_RSVD54, [110](#)
 - STM32_DMAMUX1_RSVD95, [110](#)
 - STM32_DMAMUX1_RSVD96, [110](#)
 - STM32_DMAMUX1_RSVD97, [110](#)
 - STM32_DMAMUX1_RSVD98, [110](#)
 - STM32_DMAMUX1_SAI1_A, [110](#)
 - STM32_DMAMUX1_SAI1_B, [111](#)
 - STM32_DMAMUX1_SAI2_A, [111](#)
 - STM32_DMAMUX1_SAI2_B, [111](#)
 - STM32_DMAMUX1_SAI3_A, [111](#)
 - STM32_DMAMUX1_SAI3_B, [111](#)
 - STM32_DMAMUX1_SAI4_A, [111](#)
 - STM32_DMAMUX1_SAI4_B, [112](#)
 - STM32_DMAMUX1_SPDIFRX_CS, [112](#)
 - STM32_DMAMUX1_SPDIFRX_DT, [112](#)
 - STM32_DMAMUX1_SPI1_RX, [112](#)
 - STM32_DMAMUX1_SPI1_TX, [112](#)
 - STM32_DMAMUX1_SPI2_RX, [112](#)
 - STM32_DMAMUX1_SPI2_TX, [113](#)
 - STM32_DMAMUX1_SPI3_RX, [113](#)
 - STM32_DMAMUX1_SPI3_TX, [113](#)
 - STM32_DMAMUX1_SPI4_RX, [113](#)
 - STM32_DMAMUX1_SPI4_TX, [113](#)
 - STM32_DMAMUX1_SPI5_RX, [113](#)
 - STM32_DMAMUX1_SPI5_TX, [114](#)
 - STM32_DMAMUX1_TIM15_CH1, [114](#)
 - STM32_DMAMUX1_TIM15_COM, [114](#)
 - STM32_DMAMUX1_TIM15_TRIG, [114](#)
 - STM32_DMAMUX1_TIM15_UP, [114](#)
 - STM32_DMAMUX1_TIM16_CH1, [114](#)
 - STM32_DMAMUX1_TIM16_UP, [115](#)
 - STM32_DMAMUX1_TIM17_CH1, [115](#)
 - STM32_DMAMUX1_TIM17_UP, [115](#)

- STM32_DMAMUX1_TIM1_CH1, [115](#)
- STM32_DMAMUX1_TIM1_CH2, [115](#)
- STM32_DMAMUX1_TIM1_CH3, [115](#)
- STM32_DMAMUX1_TIM1_CH4, [116](#)
- STM32_DMAMUX1_TIM1_COM, [116](#)
- STM32_DMAMUX1_TIM1_TRIG, [116](#)
- STM32_DMAMUX1_TIM1_UP, [116](#)
- STM32_DMAMUX1_TIM2_CH1, [116](#)
- STM32_DMAMUX1_TIM2_CH2, [116](#)
- STM32_DMAMUX1_TIM2_CH3, [117](#)
- STM32_DMAMUX1_TIM2_CH4, [117](#)
- STM32_DMAMUX1_TIM2_UP, [117](#)
- STM32_DMAMUX1_TIM3_CH1, [117](#)
- STM32_DMAMUX1_TIM3_CH2, [117](#)
- STM32_DMAMUX1_TIM3_CH3, [117](#)
- STM32_DMAMUX1_TIM3_CH4, [118](#)
- STM32_DMAMUX1_TIM3_TRIG, [118](#)
- STM32_DMAMUX1_TIM3_UP, [118](#)
- STM32_DMAMUX1_TIM4_CH1, [118](#)
- STM32_DMAMUX1_TIM4_CH2, [118](#)
- STM32_DMAMUX1_TIM4_CH3, [118](#)
- STM32_DMAMUX1_TIM4_UP, [119](#)
- STM32_DMAMUX1_TIM5_CH1, [119](#)
- STM32_DMAMUX1_TIM5_CH2, [119](#)
- STM32_DMAMUX1_TIM5_CH3, [119](#)
- STM32_DMAMUX1_TIM5_CH4, [119](#)
- STM32_DMAMUX1_TIM5_TRIG, [119](#)
- STM32_DMAMUX1_TIM5_UP, [120](#)
- STM32_DMAMUX1_TIM6_UP, [120](#)
- STM32_DMAMUX1_TIM7_UP, [120](#)
- STM32_DMAMUX1_TIM8_CH1, [120](#)
- STM32_DMAMUX1_TIM8_CH2, [120](#)
- STM32_DMAMUX1_TIM8_CH3, [120](#)
- STM32_DMAMUX1_TIM8_CH4, [121](#)
- STM32_DMAMUX1_TIM8_COM, [121](#)
- STM32_DMAMUX1_TIM8_TRIG, [121](#)
- STM32_DMAMUX1_TIM8_UP, [121](#)
- STM32_DMAMUX1_UART4_RX, [121](#)
- STM32_DMAMUX1_UART4_TX, [121](#)
- STM32_DMAMUX1_UART5_RX, [122](#)
- STM32_DMAMUX1_UART5_TX, [122](#)
- STM32_DMAMUX1_UART7_RX, [122](#)
- STM32_DMAMUX1_UART7_TX, [122](#)
- STM32_DMAMUX1_UART8_RX, [122](#)
- STM32_DMAMUX1_UART8_TX, [122](#)
- STM32_DMAMUX1_USART2_RX, [123](#)
- STM32_DMAMUX1_USART2_TX, [123](#)
- STM32_DMAMUX1_USART3_RX, [123](#)
- STM32_DMAMUX1_USART3_TX, [123](#)
- STM32_DMAMUX1_USART6_RX, [123](#)
- STM32_DMAMUX1_USART6_TX, [123](#)
- dma2_get_scheduler
 - [dma.c, 95](#)
 - [dma.h, 125](#)
- dma2_get_streams
 - [dma.c, 95](#)
 - [dma.h, 125](#)
- dma2_init_scheduler
 - [dma.c, 95](#)
 - [dma.h, 125](#)
- dma2_scheduler
 - [dma.c, 97](#)
- dma2_streams
 - [dma.c, 97](#)
- dma_copy
 - [dma.h, 126](#)
- dma_handle_interrupt
 - [dma.c, 96](#)
- dma_request, [13](#)
 - [dst, 13](#)
 - [dst_inc, 13](#)
 - [src, 13](#)
 - [src_inc, 14](#)
 - [transfer_len, 14](#)
- dma_request_t
 - [dma.h, 124](#)
- dma_scheduler_dev, [14](#)
 - [free_stream_count, 15](#)
 - [stream_count, 15](#)
 - [streams, 15](#)
- dma_scheduler_dev_t
 - [dma.h, 124](#)
- dma_scheduler_init
 - [dma.c, 96](#)
 - [dma.h, 126](#)
- dma_scheduler_release_stream
 - [dma.c, 96](#)
 - [dma.h, 126](#)
- dma_scheduler_request_stream
 - [dma.c, 97](#)
 - [dma.h, 127](#)
- dma_start_stream
 - [dma.c, 97](#)
 - [dma.h, 127](#)
- DMA_STATUS_TC
 - [dma.h, 102](#)
- DMA_STATUS_TE
 - [dma.h, 102](#)
- DMA_STATUS_TH
 - [dma.h, 102](#)
- dma_stop_stream
 - [dma.h, 127](#)
- dma_stream
 - [dma_stream_dev, 19](#)
- DMA_STREAM_BUSY
 - [dma.h, 125](#)
- dma_stream_config, [16](#)
 - [direction, 16](#)
 - [dmamux_request_number, 16](#)
 - [m0_addr, 16](#)
 - [m1_addr, 17](#)
 - [p_addr, 17](#)
 - [peripheral_flow_control, 17](#)
 - [priority, 17](#)
 - [stream_number, 17](#)
 - [transfer_cplt, 18](#)

- transfer_error, [18](#)
- transfer_half, [18](#)
- transfer_size, [18](#)
- user_context, [18](#)
- dma_stream_config_t
 - dma.h, [124](#)
- dma_stream_dev, [19](#)
 - dma, [19](#)
 - dma_stream, [19](#)
 - dmamux_channel, [20](#)
 - dmamux_channel_status, [20](#)
 - number, [20](#)
 - state, [20](#)
 - transfer_cplt, [20](#)
 - transfer_error, [21](#)
 - transfer_half, [21](#)
 - user_context, [21](#)
- dma_stream_dev_t
 - dma.h, [124](#)
- dma_stream_dir
 - dma.h, [124](#)
- dma_stream_dir_t
 - dma.h, [124](#)
- DMA_STREAM_FREE
 - dma.h, [125](#)
- DMA_STREAM_M2M
 - dma.h, [125](#)
- DMA_STREAM_M2P
 - dma.h, [125](#)
- DMA_STREAM_P2M
 - dma.h, [125](#)
- dma_stream_state
 - dma.h, [125](#)
- dma_stream_state_t
 - dma.h, [124](#)
- DMA_STREAMS_MAX_LEN
 - dma.h, [102](#)
- dmamux_channel
 - dma_stream_dev, [20](#)
- dmamux_channel_status
 - dma_stream_dev, [20](#)
- dmamux_request_number
 - dma_stream_config, [16](#)
- drive
 - gpio_config, [22](#)
- dst
 - dma_request, [13](#)
- dst_inc
 - dma_request, [13](#)
- E0
 - note.h, [191](#)
- E1
 - note.h, [192](#)
- E2
 - note.h, [192](#)
- E3
 - note.h, [192](#)
- E4
 - note.h, [192](#)
- E5
 - note.h, [192](#)
- E6
 - note.h, [192](#)
- E7
 - note.h, [193](#)
- E8
 - note.h, [193](#)
- ENTER_CRITICAL
 - util.h, [236](#)
- ER_DATA_NOT_RDY
 - util.h, [238](#)
- ER_FAILURE
 - util.h, [238](#)
- ER_OUT_OF_RANGE
 - util.h, [238](#)
- ER_RESSOURCE_ERROR
 - util.h, [238](#)
- ER_SUCCESS
 - util.h, [238](#)
- ER_TIMEOUT
 - util.h, [238](#)
- escape
 - MSV2_RX_DATA, [29](#)
- EXIT_CRITICAL
 - util.h, [236](#)
- F0
 - note.h, [193](#)
- F0H
 - note.h, [193](#)
- F1
 - note.h, [193](#)
- F1H
 - note.h, [193](#)
- F2
 - note.h, [194](#)
- F2H
 - note.h, [194](#)
- F3
 - note.h, [194](#)
- F3H
 - note.h, [194](#)
- F4
 - note.h, [194](#)
- F4H
 - note.h, [194](#)
- F5
 - note.h, [195](#)
- F5H
 - note.h, [195](#)
- F6
 - note.h, [195](#)
- F6H
 - note.h, [195](#)
- F7
 - note.h, [195](#)
- F7H

- note.h, [195](#)
- F8
 - note.h, [196](#)
- F8H
 - note.h, [196](#)
- feedback_interface
 - serial.c, [215](#)
- feedback_interface_context
 - debug.c, [69](#)
 - serial.c, [215](#)
- free_stream_count
 - dma_scheduler_dev, [15](#)
- freq
 - note, [32](#)
- g
 - led_color, [25](#)
- G0
 - note.h, [196](#)
- G0H
 - note.h, [196](#)
- G1
 - note.h, [196](#)
- G1H
 - note.h, [196](#)
- G2
 - note.h, [197](#)
- G2H
 - note.h, [197](#)
- G3
 - note.h, [197](#)
- G3H
 - note.h, [197](#)
- G4
 - note.h, [197](#)
- G4H
 - note.h, [197](#)
- G5
 - note.h, [198](#)
- G5H
 - note.h, [198](#)
- G6
 - note.h, [198](#)
- G6H
 - note.h, [198](#)
- G7
 - note.h, [198](#)
- G7H
 - note.h, [198](#)
- G8
 - note.h, [199](#)
- G8H
 - note.h, [199](#)
- gnss.c, [128](#)
- gnss.h, [128](#)
- gpio.c, [129](#)
 - gpio_cfg, [129](#)
 - gpio_clr, [130](#)
 - gpio_get, [130](#)
 - gpio_set, [130](#)
- gpio.h, [131](#)
 - gpio_bias, [133](#)
 - GPIO_BIAS_HIGH, [133](#)
 - GPIO_BIAS_LOW, [133](#)
 - GPIO_BIAS_NONE, [133](#)
 - gpio_bias_t, [132](#)
 - gpio_cfg, [134](#)
 - gpio_clr, [134](#)
 - gpio_config_t, [132](#)
 - gpio_drive, [133](#)
 - GPIO_DRIVE_OD, [133](#)
 - GPIO_DRIVE_PP, [133](#)
 - gpio_drive_t, [133](#)
 - gpio_get, [134](#)
 - gpio_mode, [133](#)
 - GPIO_MODE_ALT, [134](#)
 - GPIO_MODE_ANA, [134](#)
 - GPIO_MODE_IN, [134](#)
 - GPIO_MODE_OUT, [134](#)
 - gpio_mode_t, [133](#)
 - gpio_set, [135](#)
- gpio_bias
 - gpio.h, [133](#)
- GPIO_BIAS_HIGH
 - gpio.h, [133](#)
- GPIO_BIAS_LOW
 - gpio.h, [133](#)
- GPIO_BIAS_NONE
 - gpio.h, [133](#)
- gpio_bias_t
 - gpio.h, [132](#)
- gpio_cfg
 - gpio.c, [129](#)
 - gpio.h, [134](#)
- gpio_clr
 - gpio.c, [130](#)
 - gpio.h, [134](#)
- gpio_config, [21](#)
 - alternate, [22](#)
 - bias, [22](#)
 - drive, [22](#)
 - mode, [22](#)
 - speed, [23](#)
- gpio_config_t
 - gpio.h, [132](#)
- gpio_drive
 - gpio.h, [133](#)
- GPIO_DRIVE_OD
 - gpio.h, [133](#)
- GPIO_DRIVE_PP
 - gpio.h, [133](#)
- gpio_drive_t
 - gpio.h, [133](#)
- gpio_get
 - gpio.c, [130](#)
 - gpio.h, [134](#)
- gpio_mode

- gpio.h, [133](#)
- GPIO_MODE_ALT
 - gpio.h, [134](#)
- GPIO_MODE_ANA
 - gpio.h, [134](#)
- GPIO_MODE_IN
 - gpio.h, [134](#)
- GPIO_MODE_OUT
 - gpio.h, [134](#)
- gpio_mode_t
 - gpio.h, [133](#)
- gpio_set
 - gpio.c, [130](#)
 - gpio.h, [135](#)
- gyroscope.c, [135](#)
- gyroscope.h, [136](#)
- HAL_UART_RxCpltCallback
 - serial.c, [210](#)
- handle
 - device_daemon, [10](#)
- handle_data
 - device_interface, [12](#)
- host_recv
 - hostproc.c, [138](#)
- host_send
 - hostproc.c, [138](#)
- host_UART0
 - hostproc.c, [141](#)
- host_UART0_RX
 - hostproc.c, [138](#)
- hostproc.c, [136](#)
 - host_recv, [138](#)
 - host_send, [138](#)
 - host_UART0, [141](#)
 - host_UART0_RX, [138](#)
 - hostproc_device, [141](#)
 - hostproc_get_device, [139](#)
 - hostproc_get_interface, [139](#)
 - hostproc_init, [140](#)
 - hostproc_interface, [141](#)
 - hostproc_interface_context, [141](#)
 - hostproc_interface_context_t, [137](#)
- hostproc.h, [142](#)
 - hostproc_get_device, [143](#)
 - hostproc_get_interface, [143](#)
 - hostproc_init, [143](#)
- hostproc_device
 - hostproc.c, [141](#)
- hostproc_get_device
 - hostproc.c, [139](#)
 - hostproc.h, [143](#)
- hostproc_get_interface
 - hostproc.c, [139](#)
 - hostproc.h, [143](#)
- hostproc_init
 - hostproc.c, [140](#)
 - hostproc.h, [143](#)
- hostproc_interface
 - hostproc.c, [141](#)
- hostproc_interface_context, [23](#)
 - hostproc.c, [141](#)
 - rx_once, [23](#)
 - uart, [23](#)
- hostproc_interface_context_t
 - hostproc.c, [137](#)
- i2c
 - i2c_interface_context, [24](#)
- i2c.c, [144](#)
 - i2c_get_sensor_interface, [146](#)
 - i2c_init, [146](#)
 - i2c_spi_guard, [147](#)
 - S1_I2C, [145](#)
 - S2_I2C, [146](#)
 - S3_I2C, [146](#)
 - sensor_interface, [148](#)
 - sensor_interface_context, [148](#)
- i2c.h, [148](#)
 - i2c_get_sensor_interface, [149](#)
 - i2c_init, [150](#)
 - i2c_interface_context_t, [149](#)
 - i2c_spi_guard, [150](#)
- i2c_accelerometer_device
 - i2c_sensor.c, [154](#)
- i2c_accelerometer_device_context
 - i2c_sensor.c, [154](#)
- i2c_barometer_device
 - i2c_sensor.c, [154](#)
- i2c_barometer_device_context
 - i2c_sensor.c, [155](#)
- i2c_get_accelerometer
 - i2c_sensor.c, [152](#)
- i2c_get_sensor_interface
 - i2c.c, [146](#)
 - i2c.h, [149](#)
- i2c_gyroscope_device
 - i2c_sensor.c, [155](#)
- i2c_gyroscope_device_context
 - i2c_sensor.c, [155](#)
- i2c_init
 - i2c.c, [146](#)
 - i2c.h, [150](#)
- i2c_interface_context, [24](#)
 - i2c, [24](#)
- i2c_interface_context_t
 - i2c.h, [149](#)
- i2c_sensor.c, [151](#)
 - i2c_accelerometer_device, [154](#)
 - i2c_accelerometer_device_context, [154](#)
 - i2c_barometer_device, [154](#)
 - i2c_barometer_device_context, [155](#)
 - i2c_get_accelerometer, [152](#)
 - i2c_gyroscope_device, [155](#)
 - i2c_gyroscope_device_context, [155](#)
 - i2c_sensor_context_t, [152](#)
 - i2c_sensor_init, [152](#)
 - read_reg, [153](#)

- write_reg, 153
- i2c_sensor.h, 156
 - i2c_sensor_init, 156
- i2c_sensor_context, 24
 - device_address, 25
- i2c_sensor_context_t
 - i2c_sensor.c, 152
- i2c_sensor_init
 - i2c_sensor.c, 152
 - i2c_sensor.h, 156
- i2c_spi_guard
 - i2c.c, 147
 - i2c.h, 150
- id
 - device, 8
 - device_daemon, 10
 - device_interface, 12
 - MSV2_INST, 27
- in_q
 - od.c, 204
- interface
 - device, 8
- interfaces
 - device_daemon, 10
- interfaces_count
 - device_daemon, 11
- l_ix
 - util_buffer_i16, 39
 - util_buffer_u16, 40
 - util_buffer_u8, 41
- led.c, 157
 - blink_sequence, 161
 - blink_sequence_len, 162
 - color_sequence, 162
 - color_sequence_len, 162
 - led_blick_state, 158
 - led_blink_state_t, 158
 - LED_FAINT, 159
 - led_feedback_init, 159
 - LED_MAX, 158
 - LED_OFF, 159
 - LED_ON, 159
 - led_rgb_init, 159
 - led_rgb_set_color, 159
 - led_rgb_set_rgb, 160
 - led_rgb_thread, 160
 - LED_TIM, 158
- led.h, 163
 - LED_BLACK, 164
 - led_black, 168
 - LED_BLUE, 164
 - led_blue, 168
 - led_color_t, 166
 - led_feedback_init, 166
 - LED_GREEN, 164
 - led_green, 169
 - LED_LILA, 164
 - LED_ORANGE, 165
 - LED_PINK, 165
 - LED_RED, 165
 - led_red, 169
 - led_rgb_init, 166
 - led_rgb_set_color, 167
 - led_rgb_set_rgb, 167
 - led_rgb_thread, 168
 - LED_TEAL, 165
 - LED_WHITE, 165
 - LED_YELLOW, 165
- LED_BLACK
 - led.h, 164
- led_black
 - led.h, 168
- led_blick_state
 - led.c, 158
- led_blink_state_t
 - led.c, 158
- LED_BLUE
 - led.h, 164
- led_blue
 - led.h, 168
- led_color, 25
 - b, 25
 - g, 25
 - r, 26
- led_color_t
 - led.h, 166
- LED_FAINT
 - led.c, 159
- led_feedback_init
 - led.c, 159
 - led.h, 166
- LED_GREEN
 - led.h, 164
- led_green
 - led.h, 169
- LED_LILA
 - led.h, 164
- LED_MAX
 - led.c, 158
- LED_OFF
 - led.c, 159
- LED_ON
 - led.c, 159
- LED_ORANGE
 - led.h, 165
- LED_PINK
 - led.h, 165
- LED_RED
 - led.h, 165
- led_red
 - led.h, 169
- led_rgb_handle
 - threads.c, 230
- led_rgb_init
 - led.c, 159
 - led.h, 166

- LED_RGB_PRIO
 - threads.c, 228
- led_rgb_set_color
 - led.c, 159
 - led.h, 167
- led_rgb_set_rgb
 - led.c, 160
 - led.h, 167
- LED_RGB_SZ
 - threads.c, 229
- led_rgb_thread
 - led.c, 160
 - led.h, 168
- LED_TEAL
 - led.h, 165
- LED_TIM
 - led.c, 158
- LED_WHITE
 - led.h, 165
- LED_YELLOW
 - led.h, 165
- len
 - packet_def, 34
- LEN_16
 - device.c, 72
- LEN_32
 - device.c, 72
- LEN_8
 - device.c, 72
- length
 - MSV2_RX_DATA, 29
- LINK_OD_ENTRY
 - od.c, 202
- m0_addr
 - dma_stream_config, 16
- m1_addr
 - dma_stream_config, 17
- melody_active
 - buzzer.c, 51
- melody_state
 - buzzer.c, 51
- mode
 - gpio_config, 22
- msv2
 - debug_interface_context, 6
- msv2.c, 170
 - calc_field_CRC, 171
 - DLE, 170
 - msv2_create_frame, 171
 - msv2_decode_fragment, 172
 - msv2_init, 172
 - msv2_rx_data, 173
 - msv2_tx_data, 173
 - STX, 170
- msv2.h, 173
 - msv2_create_frame, 176
 - msv2_decode_fragment, 177
 - MSV2_DECODE_STATE, 175
 - MSV2_DECODE_STATE_t, 175
 - MSV2_ERROR, 176
 - MSV2_ERROR_t, 175
 - msv2_init, 177
 - MSV2_INST, 175
 - MSV2_MAX_DATA_LEN, 175
 - MSV2_MAX_FRAME_LEN, 175
 - MSV2_PROGRESS, 176
 - MSV2_RX_DATA, 175
 - MSV2_TX_DATA, 175
 - MSV2_WRONG_CRC, 176
 - WAITING_CRC1, 176
 - WAITING_CRC2, 176
 - WAITING_DATA, 176
 - WAITING_DLE, 176
 - WAITING_LEN, 176
 - WAITING_OPCODE, 176
 - WAITING_STX, 176
- msv2_create_frame
 - msv2.c, 171
 - msv2.h, 176
- msv2_decode_fragment
 - msv2.c, 172
 - msv2.h, 177
- MSV2_DECODE_STATE
 - msv2.h, 175
- MSV2_DECODE_STATE_t
 - msv2.h, 175
- MSV2_ERROR
 - msv2.h, 176
- MSV2_ERROR_t
 - msv2.h, 175
- msv2_init
 - msv2.c, 172
 - msv2.h, 177
- MSV2_INST, 26
 - id, 27
 - rx, 27
 - tx, 27
- MSV2_INST_t
 - msv2.h, 175
- MSV2_MAX_DATA_LEN
 - msv2.h, 175
- MSV2_MAX_FRAME_LEN
 - msv2.h, 175
- MSV2_PROGRESS
 - msv2.h, 176
- MSV2_RX_DATA, 27
 - counter, 28
 - crc, 28
 - crc_data, 28
 - data, 28
 - data_len, 29
 - escape, 29
 - length, 29

- opcode, [29](#)
- state, [29](#)
- msv2_rx_data
 - msv2.c, [173](#)
 - msv2.h, [178](#)
- MSV2_RX_DATA_t
 - msv2.h, [175](#)
- MSV2_SUCCESS
 - msv2.h, [176](#)
- MSV2_TX_DATA, [30](#)
 - crc, [30](#)
 - crc_data, [30](#)
 - data, [30](#)
 - data_len, [31](#)
 - opcode, [31](#)
- msv2_tx_data
 - msv2.c, [173](#)
 - msv2.h, [178](#)
- MSV2_TX_DATA_t
 - msv2.h, [175](#)
- MSV2_WRONG_CRC
 - msv2.h, [176](#)
- note, [31](#)
 - freq, [32](#)
 - time, [32](#)
- note.h, [178](#)
 - A0, [181](#)
 - A0H, [181](#)
 - A1, [181](#)
 - A1H, [181](#)
 - A2, [182](#)
 - A2H, [182](#)
 - A3, [182](#)
 - A3H, [182](#)
 - A4, [182](#)
 - A4H, [182](#)
 - A5, [183](#)
 - A5H, [183](#)
 - A6, [183](#)
 - A6H, [183](#)
 - A7, [183](#)
 - A7H, [183](#)
 - A8, [184](#)
 - A8H, [184](#)
 - B0, [184](#)
 - B1, [184](#)
 - B2, [184](#)
 - B3, [184](#)
 - B4, [185](#)
 - B5, [185](#)
 - B6, [185](#)
 - B7, [185](#)
 - B8, [185](#)
 - C0, [185](#)
 - C0H, [186](#)
 - C1, [186](#)
 - C1H, [186](#)
 - C2, [186](#)
 - C2H, [186](#)
 - C3, [186](#)
 - C3H, [187](#)
 - C4, [187](#)
 - C4H, [187](#)
 - C5, [187](#)
 - C5H, [187](#)
 - C6, [187](#)
 - C6H, [188](#)
 - C7, [188](#)
 - C7H, [188](#)
 - C8, [188](#)
 - C8H, [188](#)
 - D0, [188](#)
 - D0H, [189](#)
 - D1, [189](#)
 - D1H, [189](#)
 - D2, [189](#)
 - D2H, [189](#)
 - D3, [189](#)
 - D3H, [190](#)
 - D4, [190](#)
 - D4H, [190](#)
 - D5, [190](#)
 - D5H, [190](#)
 - D6, [190](#)
 - D6H, [191](#)
 - D7, [191](#)
 - D7H, [191](#)
 - D8, [191](#)
 - D8H, [191](#)
 - E0, [191](#)
 - E1, [192](#)
 - E2, [192](#)
 - E3, [192](#)
 - E4, [192](#)
 - E5, [192](#)
 - E6, [192](#)
 - E7, [193](#)
 - E8, [193](#)
 - F0, [193](#)
 - F0H, [193](#)
 - F1, [193](#)
 - F1H, [193](#)
 - F2, [194](#)
 - F2H, [194](#)
 - F3, [194](#)
 - F3H, [194](#)
 - F4, [194](#)
 - F4H, [194](#)
 - F5, [195](#)
 - F5H, [195](#)
 - F6, [195](#)
 - F6H, [195](#)
 - F7, [195](#)
 - F7H, [195](#)
 - F8, [196](#)
 - F8H, [196](#)

- G0, [196](#)
- G0H, [196](#)
- G1, [196](#)
- G1H, [196](#)
- G2, [197](#)
- G2H, [197](#)
- G3, [197](#)
- G3H, [197](#)
- G4, [197](#)
- G4H, [197](#)
- G5, [198](#)
- G5H, [198](#)
- G6, [198](#)
- G6H, [198](#)
- G7, [198](#)
- G7H, [198](#)
- G8, [199](#)
- G8H, [199](#)
- note_t, [200](#)
- T1, [199](#)
- T1_1_2, [199](#)
- T1_2, [199](#)
- T1_4, [199](#)
- T2, [200](#)
- T4, [200](#)
- NOTE_PRESC
 - buzzer.c, [47](#)
- note_t
 - note.h, [200](#)
- NOTE_TIMER
 - buzzer.c, [48](#)
- NOTE_TIMER_DEV
 - buzzer.c, [48](#)
- number
 - dma_stream_dev, [20](#)
- od.c, [200](#)
 - ALLOCATE_OD_ENTRY, [201](#)
 - DEBUG_NO_CAN, [201](#)
 - in_q, [204](#)
 - LINK_OD_ENTRY, [202](#)
 - od_entries, [204](#)
 - od_init, [202](#)
 - OD_MSGQ_SIZE, [202](#)
 - od_unsafe_read, [202](#)
 - od_unsafe_write, [203](#)
 - od_update_task, [203](#)
 - out_q, [204](#)
- od.h, [205](#)
 - DECLARE_OD_ENTRY, [205](#)
 - OD_FRAME_MAX_SIZE, [206](#)
 - od_init, [206](#)
 - OD_MAX_DATAID, [206](#)
 - od_update_task, [206](#)
- od_entries
 - od.c, [204](#)
- od_entry_t, [32](#)
 - data, [32](#)
 - data_id, [32](#)
 - size, [33](#)
- OD_FRAME_MAX_SIZE
 - od.h, [206](#)
- od_frame_t, [33](#)
 - data, [33](#)
 - data_id, [33](#)
 - size, [34](#)
- od_handle
 - threads.c, [231](#)
- od_init
 - od.c, [202](#)
 - od.h, [206](#)
- OD_MAX_DATAID
 - od.h, [206](#)
- OD_MSGQ_SIZE
 - od.c, [202](#)
- OD_PRIO
 - threads.c, [229](#)
- OD_SZ
 - threads.c, [229](#)
- od_unsafe_read
 - od.c, [202](#)
- od_unsafe_write
 - od.c, [203](#)
- od_update_task
 - od.c, [203](#)
 - od.h, [206](#)
- opcode
 - MSV2_RX_DATA, [29](#)
 - MSV2_TX_DATA, [31](#)
 - packet_def, [34](#)
- out_q
 - od.c, [204](#)
- p_addr
 - dma_stream_config, [17](#)
- packet.h, [207](#)
 - packet_def_t, [207](#)
 - ping, [208](#)
- packet_def, [34](#)
 - len, [34](#)
 - opcode, [34](#)
- packet_def_t
 - packet.h, [207](#)
- peripheral_flow_control
 - dma_stream_config, [17](#)
- ping
 - packet.h, [208](#)
- priority
 - dma_stream_config, [17](#)
- protocol
 - serial_interface_context, [36](#)
- r
 - led_color, [26](#)
- read_reg
 - device, [8](#)
 - i2c_sensor.c, [153](#)
- recv

- device_interface, 12
- rx
 - MSV2_INST, 27
- rx_buffer
 - serial_interface_context, 36
- rx_data
 - serial_interface_context, 37
- rx_data_len
 - serial_interface_context, 37
- rx_fragment
 - serial_interface_context, 37
- rx_once
 - hostproc_interface_context, 23
- rx_sem
 - serial_daemon_context, 35
- rx_sem_buffer
 - serial_daemon_context, 35
- RYTM_MS
 - buzzer.c, 48
- RYTM_PRESC
 - buzzer.c, 48
- RYTM_TIMER
 - buzzer.c, 48
- RYTM_TIMER_DEV
 - buzzer.c, 48
- S1_I2C
 - i2c.c, 145
- S1_UART
 - serial.c, 209
- S2_I2C
 - i2c.c, 146
- S2_UART
 - serial.c, 209
- S3_I2C
 - i2c.c, 146
- S3_UART
 - serial.c, 209
- send
 - device_interface, 12
- sensor_interface
 - i2c.c, 148
- sensor_interface_context
 - i2c.c, 148
- serial.c, 208
 - feedback_interface, 215
 - feedback_interface_context, 215
 - HAL_UART_RxCpltCallback, 210
 - S1_UART, 209
 - S2_UART, 209
 - S3_UART, 209
 - serial_data_ready, 210
 - serial_daemon, 216
 - serial_daemon_context, 216
 - SERIAL_DMA_LEN, 209
 - serial_feedback_init, 210
 - serial_get_daemon, 211
 - serial_get_feedback_interface, 211
 - serial_handle_data, 212
 - serial_init, 212
 - serial_recv, 213
 - serial_send, 214
 - serial_setup_reception, 214
- serial.h, 216
 - SERIAL_BUFFER_LEN, 218
 - serial_daemon_context_t, 218
 - serial_feedback_init, 219
 - serial_get_daemon, 220
 - serial_get_feedback_interface, 220
 - serial_init, 221
 - serial_interface_context_t, 218
 - serial_interrupt_source, 219
 - serial_interrupt_source_t, 218
 - serial_recv, 221
 - serial_send, 222
 - SERIAL_SOURCE_DMA_FIRST_HALF, 219
 - SERIAL_SOURCE_DMA_SECOND_HALF, 219
 - SERIAL_SOURCE_IDLE, 219
 - SERIAL_TRANSFER_DMA, 219
 - SERIAL_TRANSFER_IT, 219
 - serial_transfer_mode, 219
 - serial_transfer_mode_t, 219
- SERIAL_BUFFER_LEN
 - serial.h, 218
- serial_data_ready
 - serial.c, 210
- serial_daemon
 - serial.c, 216
- serial_daemon_context, 35
 - rx_sem, 35
 - rx_sem_buffer, 35
 - serial.c, 216
- serial_daemon_context_t
 - serial.h, 218
- SERIAL_DMA_LEN
 - serial.c, 209
- serial_feedback_init
 - serial.c, 210
 - serial.h, 219
- serial_get_daemon
 - serial.c, 211
 - serial.h, 220
- serial_get_feedback_interface
 - serial.c, 211
 - serial.h, 220
- serial_handle_data
 - serial.c, 212
- serial_init
 - serial.c, 212
 - serial.h, 221
- serial_interface_context, 36
 - protocol, 36
 - rx_buffer, 36
 - rx_data, 37
 - rx_data_len, 37
 - rx_fragment, 37
 - tx_data, 37

- uart, [37](#)
- serial_interface_context_t
 - serial.h, [218](#)
- serial_interrupt_source
 - serial.h, [219](#)
- serial_interrupt_source_t
 - serial.h, [218](#)
- serial_recv
 - serial.c, [213](#)
 - serial.h, [221](#)
- serial_send
 - serial.c, [214](#)
 - serial.h, [222](#)
- serial_setup_reception
 - serial.c, [214](#)
- SERIAL_SOURCE_DMA_FIRST_HALF
 - serial.h, [219](#)
- SERIAL_SOURCE_DMA_SECOND_HALF
 - serial.h, [219](#)
- SERIAL_SOURCE_IDLE
 - serial.h, [219](#)
- SERIAL_TRANSFER_DMA
 - serial.h, [219](#)
- SERIAL_TRANSFER_IT
 - serial.h, [219](#)
- serial_transfer_mode
 - serial.h, [219](#)
- serial_transfer_mode_t
 - serial.h, [219](#)
- size
 - od_entry_t, [33](#)
 - od_frame_t, [34](#)
- speed
 - gpio_config, [23](#)
- src
 - dma_request, [13](#)
- src_inc
 - dma_request, [14](#)
- stack
 - device_daemon, [11](#)
- state
 - buzzer.c, [51](#)
 - control, [5](#)
 - dma_stream_dev, [20](#)
 - MSV2_RX_DATA, [29](#)
- still_alive
 - still_alive.h, [224](#)
 - still_alive_bak.h, [225](#)
- still_alive.h, [223](#)
 - still_alive, [224](#)
 - still_alive_len, [224](#)
- still_alive_bak.h, [224](#)
 - still_alive, [225](#)
 - still_alive_len, [225](#)
- still_alive_len
 - still_alive.h, [224](#)
 - still_alive_bak.h, [225](#)
- STM32_DMAMUX1_ADC1
 - dma.h, [102](#)
- STM32_DMAMUX1_ADC2
 - dma.h, [103](#)
- STM32_DMAMUX1_Cryp2_IN
 - dma.h, [103](#)
- STM32_DMAMUX1_Cryp2_OUT
 - dma.h, [103](#)
- STM32_DMAMUX1_DAC1_CH1
 - dma.h, [103](#)
- STM32_DMAMUX1_DAC1_CH2
 - dma.h, [103](#)
- STM32_DMAMUX1_DCMI
 - dma.h, [103](#)
- STM32_DMAMUX1_DFSDM1_FLT0
 - dma.h, [104](#)
- STM32_DMAMUX1_DFSDM1_FLT1
 - dma.h, [104](#)
- STM32_DMAMUX1_DFSDM1_FLT2
 - dma.h, [104](#)
- STM32_DMAMUX1_DFSDM1_FLT3
 - dma.h, [104](#)
- STM32_DMAMUX1_DFSDM1_FLT4
 - dma.h, [104](#)
- STM32_DMAMUX1_DFSDM1_FLT5
 - dma.h, [104](#)
- STM32_DMAMUX1_HASH2_IN
 - dma.h, [105](#)
- STM32_DMAMUX1_I2C1_RX
 - dma.h, [105](#)
- STM32_DMAMUX1_I2C1_TX
 - dma.h, [105](#)
- STM32_DMAMUX1_I2C2_RX
 - dma.h, [105](#)
- STM32_DMAMUX1_I2C2_TX
 - dma.h, [105](#)
- STM32_DMAMUX1_I2C3_RX
 - dma.h, [105](#)
- STM32_DMAMUX1_I2C3_TX
 - dma.h, [106](#)
- STM32_DMAMUX1_I2C5_RX
 - dma.h, [106](#)
- STM32_DMAMUX1_I2C5_TX
 - dma.h, [106](#)
- STM32_DMAMUX1_REQ_GEN0
 - dma.h, [106](#)
- STM32_DMAMUX1_REQ_GEN1
 - dma.h, [106](#)
- STM32_DMAMUX1_REQ_GEN2
 - dma.h, [106](#)
- STM32_DMAMUX1_REQ_GEN3
 - dma.h, [107](#)
- STM32_DMAMUX1_REQ_GEN4
 - dma.h, [107](#)
- STM32_DMAMUX1_REQ_GEN5
 - dma.h, [107](#)
- STM32_DMAMUX1_REQ_GEN6
 - dma.h, [107](#)
- STM32_DMAMUX1_REQ_GEN7

dma.h, [107](#)
STM32_DMAMUX1_RSVD117
dma.h, [107](#)
STM32_DMAMUX1_RSVD118
dma.h, [108](#)
STM32_DMAMUX1_RSVD119
dma.h, [108](#)
STM32_DMAMUX1_RSVD120
dma.h, [108](#)
STM32_DMAMUX1_RSVD121
dma.h, [108](#)
STM32_DMAMUX1_RSVD122
dma.h, [108](#)
STM32_DMAMUX1_RSVD123
dma.h, [108](#)
STM32_DMAMUX1_RSVD124
dma.h, [109](#)
STM32_DMAMUX1_RSVD125
dma.h, [109](#)
STM32_DMAMUX1_RSVD126
dma.h, [109](#)
STM32_DMAMUX1_RSVD127
dma.h, [109](#)
STM32_DMAMUX1_RSVD41
dma.h, [109](#)
STM32_DMAMUX1_RSVD42
dma.h, [109](#)
STM32_DMAMUX1_RSVD54
dma.h, [110](#)
STM32_DMAMUX1_RSVD95
dma.h, [110](#)
STM32_DMAMUX1_RSVD96
dma.h, [110](#)
STM32_DMAMUX1_RSVD97
dma.h, [110](#)
STM32_DMAMUX1_RSVD98
dma.h, [110](#)
STM32_DMAMUX1_SAI1_A
dma.h, [110](#)
STM32_DMAMUX1_SAI1_B
dma.h, [111](#)
STM32_DMAMUX1_SAI2_A
dma.h, [111](#)
STM32_DMAMUX1_SAI2_B
dma.h, [111](#)
STM32_DMAMUX1_SAI3_A
dma.h, [111](#)
STM32_DMAMUX1_SAI3_B
dma.h, [111](#)
STM32_DMAMUX1_SAI4_A
dma.h, [111](#)
STM32_DMAMUX1_SAI4_B
dma.h, [112](#)
STM32_DMAMUX1_SPDIFRX_CS
dma.h, [112](#)
STM32_DMAMUX1_SPDIFRX_DT
dma.h, [112](#)
STM32_DMAMUX1_SPI1_RX
dma.h, [112](#)
STM32_DMAMUX1_SPI1_TX
dma.h, [112](#)
STM32_DMAMUX1_SPI2_RX
dma.h, [112](#)
STM32_DMAMUX1_SPI2_TX
dma.h, [113](#)
STM32_DMAMUX1_SPI3_RX
dma.h, [113](#)
STM32_DMAMUX1_SPI3_TX
dma.h, [113](#)
STM32_DMAMUX1_SPI4_RX
dma.h, [113](#)
STM32_DMAMUX1_SPI4_TX
dma.h, [113](#)
STM32_DMAMUX1_SPI5_RX
dma.h, [113](#)
STM32_DMAMUX1_SPI5_TX
dma.h, [114](#)
STM32_DMAMUX1_TIM15_CH1
dma.h, [114](#)
STM32_DMAMUX1_TIM15_COM
dma.h, [114](#)
STM32_DMAMUX1_TIM15_TRIG
dma.h, [114](#)
STM32_DMAMUX1_TIM15_UP
dma.h, [114](#)
STM32_DMAMUX1_TIM16_CH1
dma.h, [114](#)
STM32_DMAMUX1_TIM16_UP
dma.h, [115](#)
STM32_DMAMUX1_TIM17_CH1
dma.h, [115](#)
STM32_DMAMUX1_TIM17_UP
dma.h, [115](#)
STM32_DMAMUX1_TIM1_CH1
dma.h, [115](#)
STM32_DMAMUX1_TIM1_CH2
dma.h, [115](#)
STM32_DMAMUX1_TIM1_CH3
dma.h, [115](#)
STM32_DMAMUX1_TIM1_CH4
dma.h, [116](#)
STM32_DMAMUX1_TIM1_COM
dma.h, [116](#)
STM32_DMAMUX1_TIM1_TRIG
dma.h, [116](#)
STM32_DMAMUX1_TIM1_UP
dma.h, [116](#)
STM32_DMAMUX1_TIM2_CH1
dma.h, [116](#)
STM32_DMAMUX1_TIM2_CH2
dma.h, [116](#)
STM32_DMAMUX1_TIM2_CH3
dma.h, [117](#)
STM32_DMAMUX1_TIM2_CH4
dma.h, [117](#)
STM32_DMAMUX1_TIM2_UP

[dma.h](#), [117](#)
 STM32_DMAMUX1_TIM3_CH1
 [dma.h](#), [117](#)
 STM32_DMAMUX1_TIM3_CH2
 [dma.h](#), [117](#)
 STM32_DMAMUX1_TIM3_CH3
 [dma.h](#), [117](#)
 STM32_DMAMUX1_TIM3_CH4
 [dma.h](#), [118](#)
 STM32_DMAMUX1_TIM3_TRIG
 [dma.h](#), [118](#)
 STM32_DMAMUX1_TIM3_UP
 [dma.h](#), [118](#)
 STM32_DMAMUX1_TIM4_CH1
 [dma.h](#), [118](#)
 STM32_DMAMUX1_TIM4_CH2
 [dma.h](#), [118](#)
 STM32_DMAMUX1_TIM4_CH3
 [dma.h](#), [118](#)
 STM32_DMAMUX1_TIM4_UP
 [dma.h](#), [119](#)
 STM32_DMAMUX1_TIM5_CH1
 [dma.h](#), [119](#)
 STM32_DMAMUX1_TIM5_CH2
 [dma.h](#), [119](#)
 STM32_DMAMUX1_TIM5_CH3
 [dma.h](#), [119](#)
 STM32_DMAMUX1_TIM5_CH4
 [dma.h](#), [119](#)
 STM32_DMAMUX1_TIM5_TRIG
 [dma.h](#), [119](#)
 STM32_DMAMUX1_TIM5_UP
 [dma.h](#), [120](#)
 STM32_DMAMUX1_TIM6_UP
 [dma.h](#), [120](#)
 STM32_DMAMUX1_TIM7_UP
 [dma.h](#), [120](#)
 STM32_DMAMUX1_TIM8_CH1
 [dma.h](#), [120](#)
 STM32_DMAMUX1_TIM8_CH2
 [dma.h](#), [120](#)
 STM32_DMAMUX1_TIM8_CH3
 [dma.h](#), [120](#)
 STM32_DMAMUX1_TIM8_CH4
 [dma.h](#), [121](#)
 STM32_DMAMUX1_TIM8_COM
 [dma.h](#), [121](#)
 STM32_DMAMUX1_TIM8_TRIG
 [dma.h](#), [121](#)
 STM32_DMAMUX1_TIM8_UP
 [dma.h](#), [121](#)
 STM32_DMAMUX1_UART4_RX
 [dma.h](#), [121](#)
 STM32_DMAMUX1_UART4_TX
 [dma.h](#), [121](#)
 STM32_DMAMUX1_UART5_RX
 [dma.h](#), [122](#)
 STM32_DMAMUX1_UART5_TX
 [dma.h](#), [122](#)
 STM32_DMAMUX1_UART7_RX
 [dma.h](#), [122](#)
 STM32_DMAMUX1_UART7_TX
 [dma.h](#), [122](#)
 STM32_DMAMUX1_UART8_RX
 [dma.h](#), [122](#)
 STM32_DMAMUX1_UART8_TX
 [dma.h](#), [122](#)
 STM32_DMAMUX1_USART2_RX
 [dma.h](#), [123](#)
 STM32_DMAMUX1_USART2_TX
 [dma.h](#), [123](#)
 STM32_DMAMUX1_USART3_RX
 [dma.h](#), [123](#)
 STM32_DMAMUX1_USART3_TX
 [dma.h](#), [123](#)
 STM32_DMAMUX1_USART6_RX
 [dma.h](#), [123](#)
 STM32_DMAMUX1_USART6_TX
 [dma.h](#), [123](#)
 stream_count
 [dma_scheduler_dev](#), [15](#)
 stream_number
 [dma_stream_config](#), [17](#)
 streams
 [dma_scheduler_dev](#), [15](#)
 STX
 [msv2.c](#), [170](#)
 T1
 [note.h](#), [199](#)
 T1_1_2
 [note.h](#), [199](#)
 T1_2
 [note.h](#), [199](#)
 T1_4
 [note.h](#), [199](#)
 T2
 [note.h](#), [200](#)
 T4
 [note.h](#), [200](#)
 template.c, [225](#)
 template.h, [226](#)
 threads.c, [226](#)
 [control_handle](#), [230](#)
 [CONTROL_PRIO](#), [227](#)
 [CONTROL_SZ](#), [227](#)
 [CREATE_THREAD](#), [228](#)
 [DEFAULT_SZ](#), [228](#)
 [led_rgb_handle](#), [230](#)
 [LED_RGB_PRIO](#), [228](#)
 [LED_RGB_SZ](#), [229](#)
 [od_handle](#), [231](#)
 [OD_PRIO](#), [229](#)
 [OD_SZ](#), [229](#)
 [threads_init](#), [229](#)
 threads.h, [231](#)
 [threads_init](#), [232](#)

- threads_init
 - threads.c, [229](#)
 - threads.h, [232](#)
- time
 - note, [32](#)
- TIMER_FREQ
 - buzzer.c, [49](#)
- TIMER_TRIM
 - buzzer.c, [49](#)
- transfer_len
 - dma_request, [14](#)
- transfer_cplt
 - dma_stream_config, [18](#)
 - dma_stream_dev, [20](#)
- transfer_error
 - dma_stream_config, [18](#)
 - dma_stream_dev, [21](#)
- transfer_half
 - dma_stream_config, [18](#)
 - dma_stream_dev, [21](#)
- transfer_size
 - dma_stream_config, [18](#)
- tx
 - MSV2_INST, [27](#)
- tx_data
 - serial_interface_context, [37](#)
- uart
 - hostproc_interface_context, [23](#)
 - serial_interface_context, [37](#)
- uart.c, [233](#)
- uart.h, [233](#)
- user_context
 - dma_stream_config, [18](#)
 - dma_stream_dev, [21](#)
- util.h, [234](#)
 - ENTER_CRITICAL, [236](#)
 - ER_DATA_NOT_RDY, [238](#)
 - ER_FAILURE, [238](#)
 - ER_OUT_OF_RANGE, [238](#)
 - ER_RESSOURCE_ERROR, [238](#)
 - ER_SUCCESS, [238](#)
 - ER_TIMEOUT, [238](#)
 - EXIT_CRITICAL, [236](#)
 - util_abs, [236](#)
 - util_buffer_i16_add, [239](#)
 - util_buffer_i16_get, [239](#)
 - util_buffer_i16_init, [239](#)
 - util_buffer_i16_isempty, [239](#)
 - util_buffer_i16_t, [237](#)
 - util_buffer_u16_add, [239](#)
 - util_buffer_u16_get, [240](#)
 - util_buffer_u16_init, [240](#)
 - util_buffer_u16_isempty, [240](#)
 - util_buffer_u16_t, [237](#)
 - util_buffer_u8_access, [240](#)
 - util_buffer_u8_add, [241](#)
 - util_buffer_u8_get, [241](#)
 - util_buffer_u8_init, [241](#)
 - util_buffer_u8_isempty, [242](#)
 - util_buffer_u8_t, [238](#)
 - util_decode_i16, [242](#)
 - util_decode_i32, [243](#)
 - util_decode_i8, [243](#)
 - util_decode_u16, [244](#)
 - util_decode_u32, [244](#)
 - util_decode_u8, [245](#)
 - util_encode_i16, [245](#)
 - util_encode_i32, [246](#)
 - util_encode_i8, [246](#)
 - util_encode_u16, [247](#)
 - util_encode_u32, [247](#)
 - util_encode_u8, [248](#)
 - util_error, [238](#)
 - util_error_t, [238](#)
 - UTIL_GENERATE_BUFFER, [236](#)
 - WRITE_IN_REG, [237](#)
- util_abs
 - util.h, [236](#)
- util_buffer_i16, [38](#)
 - bfr_len, [38](#)
 - buffer, [38](#)
 - c_ix, [38](#)
 - l_ix, [39](#)
- util_buffer_i16_add
 - util.h, [239](#)
- util_buffer_i16_get
 - util.h, [239](#)
- util_buffer_i16_init
 - util.h, [239](#)
- util_buffer_i16_isempty
 - util.h, [239](#)
- util_buffer_i16_t
 - util.h, [237](#)
- util_buffer_u16, [39](#)
 - bfr_len, [39](#)
 - buffer, [39](#)
 - c_ix, [40](#)
 - l_ix, [40](#)
- util_buffer_u16_add
 - util.h, [239](#)
- util_buffer_u16_get
 - util.h, [240](#)
- util_buffer_u16_init
 - util.h, [240](#)
- util_buffer_u16_isempty
 - util.h, [240](#)
- util_buffer_u16_t
 - util.h, [237](#)
- util_buffer_u8, [40](#)
 - bfr_len, [41](#)
 - buffer, [41](#)
 - c_ix, [41](#)
 - l_ix, [41](#)
- util_buffer_u8_access
 - util.h, [240](#)
- util_buffer_u8_add

- util.h, [241](#)
- util_buffer_u8_get
 - util.h, [241](#)
- util_buffer_u8_init
 - util.h, [241](#)
- util_buffer_u8_isempty
 - util.h, [242](#)
- util_buffer_u8_t
 - util.h, [238](#)
- util_decode_i16
 - util.h, [242](#)
- util_decode_i32
 - util.h, [243](#)
- util_decode_i8
 - util.h, [243](#)
- util_decode_u16
 - util.h, [244](#)
- util_decode_u32
 - util.h, [244](#)
- util_decode_u8
 - util.h, [245](#)
- util_encode_i16
 - util.h, [245](#)
- util_encode_i32
 - util.h, [246](#)
- util_encode_i8
 - util.h, [246](#)
- util_encode_u16
 - util.h, [247](#)
- util_encode_u32
 - util.h, [247](#)
- util_encode_u8
 - util.h, [248](#)
- util_error
 - util.h, [238](#)
- util_error_t
 - util.h, [238](#)
- UTIL_GENERATE_BUFFER
 - util.h, [236](#)
- WAITING_CRC1
 - msv2.h, [176](#)
- WAITING_CRC2
 - msv2.h, [176](#)
- WAITING_DATA
 - msv2.h, [176](#)
- WAITING_DLE
 - msv2.h, [176](#)
- WAITING_LEN
 - msv2.h, [176](#)
- WAITING_OPCODE
 - msv2.h, [176](#)
- WAITING_STX
 - msv2.h, [176](#)
- WH_FALSE
 - wildhorn.h, [249](#)
- WH_HAS_FEEDBACK
 - wildhorn.h, [250](#)
- WH_HAS_GNSS
 - wildhorn.h, [250](#)
- util.h, [241](#)
- util_buffer_u8_get
 - util.h, [241](#)
- util_buffer_u8_init
 - util.h, [241](#)
- util_buffer_u8_isempty
 - util.h, [242](#)
- util_buffer_u8_t
 - util.h, [238](#)
- util_decode_i16
 - util.h, [242](#)
- util_decode_i32
 - util.h, [243](#)
- util_decode_i8
 - util.h, [243](#)
- util_decode_u16
 - util.h, [244](#)
- util_decode_u32
 - util.h, [244](#)
- util_decode_u8
 - util.h, [245](#)
- util_encode_i16
 - util.h, [245](#)
- util_encode_i32
 - util.h, [246](#)
- util_encode_i8
 - util.h, [246](#)
- util_encode_u16
 - util.h, [247](#)
- util_encode_u32
 - util.h, [247](#)
- util_encode_u8
 - util.h, [248](#)
- util_error
 - util.h, [238](#)
- util_error_t
 - util.h, [238](#)
- UTIL_GENERATE_BUFFER
 - util.h, [236](#)
- WAITING_CRC1
 - msv2.h, [176](#)
- WAITING_CRC2
 - msv2.h, [176](#)
- WAITING_DATA
 - msv2.h, [176](#)
- WAITING_DLE
 - msv2.h, [176](#)
- WAITING_LEN
 - msv2.h, [176](#)
- WAITING_OPCODE
 - msv2.h, [176](#)
- WAITING_STX
 - msv2.h, [176](#)
- WH_FALSE
 - wildhorn.h, [249](#)
- WH_HAS_FEEDBACK
 - wildhorn.h, [250](#)
- WH_HAS_GNSS
 - wildhorn.h, [250](#)
- wildhorn.h, [250](#)
- WH_HAS_KRTEK
 - wildhorn.h, [250](#)
- WH_HAS_RADIO
 - wildhorn.h, [250](#)
- WH_HAS_SENSORS
 - wildhorn.h, [250](#)
- WH_TRUE
 - wildhorn.h, [250](#)
- WH_USE_BUZZER
 - wildhorn.h, [250](#)
- wildhorn.h, [249](#)
- WH_FALSE, [249](#)
- WH_HAS_FEEDBACK, [250](#)
- WH_HAS_GNSS, [250](#)
- WH_HAS_KRTEK, [250](#)
- WH_HAS_RADIO, [250](#)
- WH_HAS_SENSORS, [250](#)
- WH_TRUE, [250](#)
- WH_USE_BUZZER, [250](#)
- WRITE_IN_REG
 - util.h, [237](#)
- write_reg
 - device, [8](#)
 - i2c_sensor.c, [153](#)