

## Micras Firmware

Generated by Doxygen 1.9.1



<b>1 Micras</b>	<b>1</b>
<b>2 Todo List</b>	<b>3</b>
<b>3 Namespace Index</b>	<b>5</b>
3.1 Namespace List . . . . .	5
<b>4 Class Index</b>	<b>7</b>
4.1 Class List . . . . .	7
<b>5 File Index</b>	<b>11</b>
5.1 File List . . . . .	11
<b>6 Namespace Documentation</b>	<b>15</b>
6.1 hal Namespace Reference . . . . .	15
6.2 proxy Namespace Reference . . . . .	15
6.2.1 Variable Documentation . . . . .	16
6.2.1.1 Fundamental . . . . .	16
<b>7 Class Documentation</b>	<b>17</b>
7.1 hal::AdcDma Class Reference . . . . .	17
7.1.1 Detailed Description . . . . .	18
7.1.2 Constructor & Destructor Documentation . . . . .	18
7.1.2.1 AdcDma() . . . . .	18
7.1.3 Member Function Documentation . . . . .	18
7.1.3.1 start_dma() . . . . .	18
7.1.3.2 stop_dma() . . . . .	19
7.1.4 Member Data Documentation . . . . .	19
7.1.4.1 max_reading . . . . .	19
7.1.4.2 reference_voltage . . . . .	19
7.2 proxy::Argb< num_of_leds > Class Template Reference . . . . .	20
7.2.1 Detailed Description . . . . .	21
7.2.2 Constructor & Destructor Documentation . . . . .	21
7.2.2.1 Argb() . . . . .	21
7.2.3 Member Function Documentation . . . . .	21
7.2.3.1 set_color() [1/2] . . . . .	22
7.2.3.2 set_color() [2/2] . . . . .	22
7.2.3.3 turn_off() [1/2] . . . . .	22
7.2.3.4 turn_off() [2/2] . . . . .	22
7.3 proxy::Battery Class Reference . . . . .	23
7.3.1 Detailed Description . . . . .	23
7.3.2 Constructor & Destructor Documentation . . . . .	24
7.3.2.1 Battery() . . . . .	24
7.3.3 Member Function Documentation . . . . .	25

7.3.3.1 get_voltage()	25
7.3.3.2 get_voltage_raw()	25
7.4 proxy::Button Class Reference	26
7.4.1 Detailed Description	26
7.4.2 Member Enumeration Documentation	27
7.4.2.1 PullResistor	27
7.4.2.2 Status	28
7.4.3 Constructor & Destructor Documentation	28
7.4.3.1 Button()	28
7.4.4 Member Function Documentation	28
7.4.4.1 get_status()	28
7.4.4.2 is_pressed()	29
7.5 proxy::Buzzer Class Reference	30
7.5.1 Detailed Description	30
7.5.2 Constructor & Destructor Documentation	30
7.5.2.1 Buzzer()	30
7.5.3 Member Function Documentation	31
7.5.3.1 play()	31
7.5.3.2 stop()	32
7.5.3.3 update()	32
7.6 proxy::Argb< num_of_leds >::Color Struct Reference	33
7.6.1 Detailed Description	33
7.6.2 Member Data Documentation	33
7.6.2.1 blue	34
7.6.2.2 green	34
7.6.2.3 red	34
7.7 hal::AdcDma::Config Struct Reference	34
7.7.1 Detailed Description	35
7.7.2 Member Data Documentation	35
7.7.2.1 handle	35
7.7.2.2 init_function	35
7.7.2.3 max_reading	35
7.7.2.4 reference_voltage	35
7.8 hal::Crc::Config Struct Reference	36
7.8.1 Detailed Description	36
7.8.2 Member Data Documentation	36
7.8.2.1 handle	36
7.9 hal::Encoder::Config Struct Reference	37
7.9.1 Detailed Description	37
7.9.2 Member Data Documentation	37
7.9.2.1 handle	37
7.9.2.2 init_function	38

7.9.2.3 timer_channel . . . . .	38
7.10 hal::Gpio::Config Struct Reference . . . . .	38
7.10.1 Detailed Description . . . . .	39
7.10.2 Member Data Documentation . . . . .	39
7.10.2.1 pin . . . . .	39
7.10.2.2 port . . . . .	39
7.11 hal::Pwm::Config Struct Reference . . . . .	39
7.11.1 Detailed Description . . . . .	40
7.11.2 Member Data Documentation . . . . .	40
7.11.2.1 handle . . . . .	40
7.11.2.2 init_function . . . . .	40
7.11.2.3 timer_channel . . . . .	40
7.12 hal::PwmDma::Config Struct Reference . . . . .	41
7.12.1 Detailed Description . . . . .	41
7.12.2 Member Data Documentation . . . . .	41
7.12.2.1 handle . . . . .	41
7.12.2.2 init_function . . . . .	42
7.12.2.3 timer_channel . . . . .	42
7.13 hal::Spi::Config Struct Reference . . . . .	42
7.13.1 Detailed Description . . . . .	43
7.13.2 Member Data Documentation . . . . .	43
7.13.2.1 gpio . . . . .	43
7.13.2.2 handle . . . . .	43
7.13.2.3 init_function . . . . .	43
7.13.2.4 timeout . . . . .	43
7.14 hal::Timer::Config Struct Reference . . . . .	44
7.14.1 Detailed Description . . . . .	44
7.14.2 Member Data Documentation . . . . .	44
7.14.2.1 handle . . . . .	44
7.14.2.2 init_function . . . . .	45
7.15 proxy::Argb< num_of_leds >::Config Struct Reference . . . . .	45
7.15.1 Detailed Description . . . . .	46
7.15.2 Member Data Documentation . . . . .	46
7.15.2.1 pwm . . . . .	46
7.16 proxy::Battery::Config Struct Reference . . . . .	46
7.16.1 Detailed Description . . . . .	47
7.16.2 Member Data Documentation . . . . .	47
7.16.2.1 adc . . . . .	47
7.16.2.2 voltage_divider . . . . .	47
7.17 proxy::Button::Config Struct Reference . . . . .	47
7.17.1 Detailed Description . . . . .	48
7.17.2 Member Data Documentation . . . . .	48

7.17.2.1 debounce_delay	48
7.17.2.2 extra_long_press_delay	49
7.17.2.3 gpio	49
7.17.2.4 long_press_delay	49
7.17.2.5 pull_resistor	49
7.18 proxy::Buzzer::Config Struct Reference	49
7.18.1 Detailed Description	50
7.18.2 Member Data Documentation	50
7.18.2.1 pwm	50
7.19 proxy::CurrentSensors< num_of_sensors >::Config Struct Reference	51
7.19.1 Detailed Description	51
7.19.2 Member Data Documentation	51
7.19.2.1 adc	51
7.19.2.2 shunt_resistor	52
7.20 proxy::DipSwitch< num_of_switches >::Config Struct Reference	52
7.20.1 Detailed Description	52
7.20.2 Member Data Documentation	53
7.20.2.1 gpio_array	53
7.21 proxy::DistanceSensors< num_of_sensors >::Config Struct Reference	53
7.21.1 Detailed Description	53
7.21.2 Member Data Documentation	54
7.21.2.1 adc	54
7.21.2.2 led_pwm	54
7.21.2.3 max_distance	54
7.22 proxy::Fan::Config Struct Reference	54
7.22.1 Detailed Description	55
7.22.2 Member Data Documentation	55
7.22.2.1 direction_gpio	55
7.22.2.2 enable_gpio	55
7.22.2.3 pwm	55
7.23 proxy::Imu::Config Struct Reference	55
7.23.1 Detailed Description	56
7.23.2 Member Data Documentation	56
7.23.2.1 accelerometer_data_rate	56
7.23.2.2 accelerometer_filter	56
7.23.2.3 accelerometer_scale	56
7.23.2.4 gyroscope_data_rate	56
7.23.2.5 gyroscope_filter	57
7.23.2.6 gyroscope_scale	57
7.23.2.7 orientation_data_rate	57
7.23.2.8 spi	57
7.24 proxy::Led::Config Struct Reference	57

7.24.1 Detailed Description	58
7.24.2 Member Data Documentation	58
7.24.2.1 gpio	58
7.25 proxy::Locomotion::Config Struct Reference	59
7.25.1 Detailed Description	59
7.25.2 Member Data Documentation	59
7.25.2.1 enable_gpio	59
7.25.2.2 pwm_left_bwd	60
7.25.2.3 pwm_left_fwd	60
7.25.2.4 pwm_right_bwd	60
7.25.2.5 pwm_right_fwd	60
7.26 proxy::RotarySensor::Config Struct Reference	60
7.26.1 Detailed Description	61
7.26.2 Member Data Documentation	61
7.26.2.1 crc	61
7.26.2.2 encoder	61
7.26.2.3 registers	61
7.26.2.4 resolution	61
7.26.2.5 spi	61
7.27 proxy::Storage::Config Struct Reference	62
7.27.1 Detailed Description	62
7.27.2 Member Data Documentation	62
7.27.2.1 number_of_pages	62
7.27.2.2 start_page	63
7.28 proxy::TorqueSensors< num_of_sensors >::Config Struct Reference	63
7.28.1 Detailed Description	63
7.28.2 Member Data Documentation	64
7.28.2.1 current_sensors	64
7.28.2.2 max_torque	64
7.29 hal::Crc Class Reference	64
7.29.1 Detailed Description	65
7.29.2 Constructor & Destructor Documentation	65
7.29.2.1 Crc()	65
7.29.3 Member Function Documentation	65
7.29.3.1 calculate()	65
7.30 proxy::CurrentSensors< num_of_sensors > Class Template Reference	66
7.30.1 Detailed Description	67
7.30.2 Constructor & Destructor Documentation	67
7.30.2.1 CurrentSensors()	67
7.30.3 Member Function Documentation	67
7.30.3.1 get_current()	67
7.30.3.2 get_current_raw()	68

7.31 proxy::DipSwitch< num_of_switches > Class Template Reference . . . . .	68
7.31.1 Detailed Description . . . . .	70
7.31.2 Constructor & Destructor Documentation . . . . .	70
7.31.2.1 DipSwitch() . . . . .	70
7.31.3 Member Function Documentation . . . . .	70
7.31.3.1 get_switch_state() . . . . .	70
7.31.3.2 get_switches_value() . . . . .	71
7.32 Registers::Disable Union Reference . . . . .	71
7.32.1 Detailed Description . . . . .	72
7.32.2 Member Data Documentation . . . . .	72
7.32.2.1 fields . . . . .	73
7.32.2.2 raw . . . . .	73
7.33 proxy::DistanceSensors< num_of_sensors > Class Template Reference . . . . .	73
7.33.1 Detailed Description . . . . .	74
7.33.2 Constructor & Destructor Documentation . . . . .	74
7.33.2.1 DistanceSensors() . . . . .	74
7.33.3 Member Function Documentation . . . . .	75
7.33.3.1 get_distance() . . . . .	75
7.33.3.2 get_distance_raw() . . . . .	75
7.33.3.3 set_led_intensity() . . . . .	76
7.34 Registers::Ecc Union Reference . . . . .	76
7.34.1 Member Data Documentation . . . . .	77
7.34.1.1 fields . . . . .	77
7.34.1.2 raw . . . . .	78
7.35 hal::Encoder Class Reference . . . . .	78
7.35.1 Detailed Description . . . . .	78
7.35.2 Constructor & Destructor Documentation . . . . .	78
7.35.2.1 Encoder() . . . . .	78
7.35.3 Member Function Documentation . . . . .	79
7.35.3.1 get_counter() . . . . .	79
7.36 proxy::Fan Class Reference . . . . .	79
7.36.1 Detailed Description . . . . .	80
7.36.2 Member Enumeration Documentation . . . . .	80
7.36.2.1 RotationDirection . . . . .	80
7.36.3 Constructor & Destructor Documentation . . . . .	81
7.36.3.1 Fan() . . . . .	81
7.36.4 Member Function Documentation . . . . .	81
7.36.4.1 disable() . . . . .	81
7.36.4.2 enable() . . . . .	82
7.36.4.3 set_speed() . . . . .	82
7.36.4.4 stop() . . . . .	83
7.37 proxy::RotarySensor::CommandFrame::Fields Struct Reference . . . . .	84



7.37.1 Member Data Documentation	84
7.37.1.1 address	84
7.37.1.2 crc	84
7.37.1.3 do_not_care	85
7.37.1.4 rw	85
7.38 proxy::RotarySensor::DataFrame::Fields Struct Reference	85
7.38.1 Member Data Documentation	86
7.38.1.1 crc	86
7.38.1.2 data	86
7.38.1.3 error	86
7.38.1.4 warning	86
7.39 Registers::Disable::Fields Struct Reference	86
7.39.1 Member Data Documentation	87
7.39.1.1 ABI_off	87
7.39.1.2 FILTER_disable	87
7.39.1.3 na	88
7.39.1.4 UVW_off	88
7.40 Registers::Ecc::Fields Struct Reference	88
7.40.1 Member Data Documentation	88
7.40.1.1 ECC_chsum	89
7.40.1.2 ECC_en	89
7.41 Registers::Settings1::Fields Struct Reference	89
7.41.1 Member Data Documentation	90
7.41.1.1 Dia3_en	90
7.41.1.2 Dia4_en	90
7.41.1.3 K_max	90
7.41.1.4 K_min	90
7.42 Registers::Settings2::Fields Struct Reference	90
7.42.1 Member Data Documentation	91
7.42.1.1 ABI_DEC	91
7.42.1.2 DAECDIS	92
7.42.1.3 Data_select	92
7.42.1.4 DIR	92
7.42.1.5 IWIDTH	92
7.42.1.6 NOISESET	92
7.42.1.7 PWMon	92
7.42.1.8 UVW_ABI	92
7.43 Registers::Settings3::Fields Struct Reference	93
7.43.1 Member Data Documentation	93
7.43.1.1 ABIREs	93
7.43.1.2 HYS	93
7.43.1.3 UVWPP	94

7.44 Registers::Zposl::Fields Struct Reference	94
7.44.1 Member Data Documentation	94
7.44.1.1 Dia1_en	95
7.44.1.2 Dia2_en	95
7.44.1.3 ZPOSL	95
7.45 Registers::Zposm::Fields Struct Reference	95
7.45.1 Member Data Documentation	96
7.45.1.1 ZPOSM	96
7.46 hal::Flash Class Reference	96
7.46.1 Detailed Description	97
7.46.2 Constructor & Destructor Documentation	97
7.46.2.1 Flash()	97
7.46.3 Member Function Documentation	97
7.46.3.1 erase_pages()	97
7.46.3.2 read() [1/2]	98
7.46.3.3 read() [2/2]	98
7.46.3.4 write() [1/2]	99
7.46.3.5 write() [2/2]	99
7.47 hal::Gpio Class Reference	100
7.47.1 Detailed Description	101
7.47.2 Constructor & Destructor Documentation	101
7.47.2.1 Gpio()	101
7.47.3 Member Function Documentation	101
7.47.3.1 read()	101
7.47.3.2 toggle()	102
7.47.3.3 write()	102
7.48 proxy::Imu Class Reference	103
7.48.1 Detailed Description	104
7.48.2 Member Enumeration Documentation	104
7.48.2.1 Axis	104
7.48.3 Constructor & Destructor Documentation	104
7.48.3.1 Imu()	105
7.48.4 Member Function Documentation	105
7.48.4.1 get_angular_velocity()	105
7.48.4.2 get_linear_acceleration()	105
7.48.4.3 get_orientation()	106
7.48.4.4 update_data()	106
7.49 ISerializable Class Reference	107
7.49.1 Detailed Description	107
7.49.2 Constructor & Destructor Documentation	107
7.49.2.1 ~ISerializable()	108
7.49.2.2 ISerializable() [1/2]	108

7.49.2.3 ISerializable() [2/2]	108
7.49.3 Member Function Documentation	108
7.49.3.1 deserialize()	108
7.49.3.2 operator=() [1/2]	109
7.49.3.3 operator=() [2/2]	109
7.49.3.4 serialize()	109
7.50 proxy::Led Class Reference	110
7.50.1 Detailed Description	110
7.50.2 Constructor & Destructor Documentation	110
7.50.2.1 Led()	110
7.50.3 Member Function Documentation	111
7.50.3.1 toggle()	111
7.50.3.2 turn_off()	111
7.50.3.3 turn_on()	112
7.51 proxy::Locomotion Class Reference	113
7.51.1 Detailed Description	114
7.51.2 Constructor & Destructor Documentation	114
7.51.2.1 Locomotion()	114
7.51.3 Member Function Documentation	114
7.51.3.1 disable()	114
7.51.3.2 enable()	115
7.51.3.3 set_speed()	115
7.51.3.4 set_wheel_speed()	116
7.51.3.5 stop()	117
7.51.3.6 stop_left()	118
7.51.3.7 stop_right()	118
7.52 hal::Mcu Class Reference	119
7.52.1 Detailed Description	120
7.52.2 Constructor & Destructor Documentation	120
7.52.2.1 Mcu()	120
7.52.3 Member Function Documentation	120
7.52.3.1 init()	120
7.53 MicrasController Class Reference	121
7.53.1 Detailed Description	121
7.53.2 Constructor & Destructor Documentation	121
7.53.2.1 MicrasController()	121
7.53.3 Member Function Documentation	121
7.53.3.1 run()	122
7.54 hal::Pwm Class Reference	122
7.54.1 Detailed Description	123
7.54.2 Constructor & Destructor Documentation	123
7.54.2.1 Pwm()	123

7.54.3 Member Function Documentation	124
7.54.3.1 set_duty_cycle()	124
7.54.3.2 set_frequency()	124
7.55 hal::PwmDma Class Reference	125
7.55.1 Detailed Description	126
7.55.2 Constructor & Destructor Documentation	126
7.55.2.1 PwmDma()	126
7.55.3 Member Function Documentation	126
7.55.3.1 get_compare()	126
7.55.3.2 start_dma()	127
7.55.3.3 stop_dma()	127
7.56 Registers Struct Reference	127
7.56.1 Detailed Description	128
7.56.2 Member Data Documentation	128
7.56.2.1 disable	128
7.56.2.2 disable_addr	129
7.56.2.3 ecc	129
7.56.2.4 ecc_addr	129
7.56.2.5 settings1	129
7.56.2.6 settings1_addr	129
7.56.2.7 settings2	129
7.56.2.8 settings2_addr	129
7.56.2.9 settings3	130
7.56.2.10 settings3_addr	130
7.56.2.11 zposl	130
7.56.2.12 zposl_addr	130
7.56.2.13 zposm	130
7.56.2.14 zposm_addr	130
7.57 proxy::RotarySensor Class Reference	131
7.57.1 Detailed Description	131
7.57.2 Constructor & Destructor Documentation	131
7.57.2.1 RotarySensor()	131
7.57.3 Member Function Documentation	132
7.57.3.1 get_position()	132
7.58 Registers::Settings1 Union Reference	132
7.58.1 Member Data Documentation	133
7.58.1.1 fields	133
7.58.1.2 raw	134
7.59 Registers::Settings2 Union Reference	134
7.59.1 Member Data Documentation	135
7.59.1.1 fields	135
7.59.1.2 raw	135

7.60 Registers::Settings3 Union Reference . . . . .	135
7.60.1 Member Data Documentation . . . . .	136
7.60.1.1 fields . . . . .	136
7.60.1.2 raw . . . . .	137
7.61 hal::Spi Class Reference . . . . .	137
7.61.1 Detailed Description . . . . .	138
7.61.2 Constructor & Destructor Documentation . . . . .	138
7.61.2.1 Spi() . . . . .	138
7.61.3 Member Function Documentation . . . . .	138
7.61.3.1 receive() . . . . .	138
7.61.3.2 select_device() . . . . .	138
7.61.3.3 transmit() . . . . .	139
7.61.3.4 unselect_device() . . . . .	139
7.62 proxy::Storage Class Reference . . . . .	140
7.62.1 Detailed Description . . . . .	141
7.62.2 Constructor & Destructor Documentation . . . . .	141
7.62.2.1 Storage() . . . . .	141
7.62.3 Member Function Documentation . . . . .	141
7.62.3.1 create() [1/2] . . . . .	141
7.62.3.2 create() [2/2] . . . . .	142
7.62.3.3 save() . . . . .	142
7.62.3.4 sync() [1/2] . . . . .	143
7.62.3.5 sync() [2/2] . . . . .	143
7.63 hal::Timer Class Reference . . . . .	144
7.63.1 Detailed Description . . . . .	145
7.63.2 Constructor & Destructor Documentation . . . . .	145
7.63.2.1 Timer() [1/2] . . . . .	145
7.63.2.2 Timer() [2/2] . . . . .	145
7.63.3 Member Function Documentation . . . . .	145
7.63.3.1 elapsed_time_ms() . . . . .	145
7.63.3.2 elapsed_time_us() . . . . .	146
7.63.3.3 reset_ms() . . . . .	146
7.63.3.4 reset_us() . . . . .	147
7.63.3.5 sleep_ms() . . . . .	147
7.63.3.6 sleep_us() . . . . .	147
7.64 proxy::TorqueSensors< num_of_sensors > Class Template Reference . . . . .	148
7.64.1 Detailed Description . . . . .	149
7.64.2 Constructor & Destructor Documentation . . . . .	149
7.64.2.1 TorqueSensors() . . . . .	149
7.64.3 Member Function Documentation . . . . .	149
7.64.3.1 get_current() . . . . .	149
7.64.3.2 get_torque() . . . . .	150

7.65 Registers::Zposl Union Reference	150
7.65.1 Member Data Documentation	151
7.65.1.1 Fields	151
7.65.1.2 raw	152
7.66 Registers::Zposm Union Reference	152
7.66.1 Member Data Documentation	153
7.66.1.1 fields	153
7.66.1.2 raw	153
<b>8 File Documentation</b>	<b>155</b>
8.1 cfg/target.cpp File Reference	155
8.1.1 Detailed Description	155
8.1.2 Variable Documentation	156
8.1.2.1 argb_config	156
8.1.2.2 battery_config	156
8.1.2.3 button_config	156
8.1.2.4 buzzer_config	156
8.1.2.5 dip_switch_config	157
8.1.2.6 distance_sensors_config	157
8.1.2.7 fan_config	157
8.1.2.8 imu_config	158
8.1.2.9 led_config	158
8.1.2.10 locomotion_config	158
8.1.2.11 rotary_sensor_left_config	159
8.1.2.12 rotary_sensor_reg_config	159
8.1.2.13 rotary_sensor_right_config	159
8.1.2.14 torque_sensors_config	160
8.2 cfg/target.hpp File Reference	160
8.2.1 Detailed Description	161
8.2.2 Variable Documentation	161
8.2.2.1 argb_config	161
8.2.2.2 battery_config	161
8.2.2.3 button_config	161
8.2.2.4 buzzer_config	162
8.2.2.5 dip_switch_config	162
8.2.2.6 distance_sensors_config	162
8.2.2.7 fan_config	162
8.2.2.8 imu_config	162
8.2.2.9 led_config	162
8.2.2.10 locomotion_config	162
8.2.2.11 rotary_sensor_left_config	162
8.2.2.12 rotary_sensor_right_config	163

8.2.2.13 torque_sensors_config . . . . .	163
8.3 inc/controller/micras_controller.hpp File Reference . . . . .	163
8.4 inc/hal/adc_dma.hpp File Reference . . . . .	164
8.4.1 Detailed Description . . . . .	164
8.5 inc/hal/crc.hpp File Reference . . . . .	165
8.5.1 Detailed Description . . . . .	165
8.6 inc/hal/encoder.hpp File Reference . . . . .	166
8.6.1 Detailed Description . . . . .	166
8.7 inc/hal/flash.hpp File Reference . . . . .	167
8.7.1 Detailed Description . . . . .	167
8.8 inc/hal/gpio.hpp File Reference . . . . .	167
8.8.1 Detailed Description . . . . .	168
8.9 inc/hal/mcu.hpp File Reference . . . . .	168
8.9.1 Detailed Description . . . . .	169
8.10 inc/hal/pwm.hpp File Reference . . . . .	169
8.10.1 Detailed Description . . . . .	169
8.11 inc/hal/pwm_dma.hpp File Reference . . . . .	170
8.11.1 Detailed Description . . . . .	170
8.12 inc/hal/spi.hpp File Reference . . . . .	171
8.12.1 Detailed Description . . . . .	172
8.13 inc/hal/timer.hpp File Reference . . . . .	172
8.13.1 Detailed Description . . . . .	172
8.14 inc/proxy/argb.hpp File Reference . . . . .	173
8.14.1 Detailed Description . . . . .	174
8.15 inc/proxy/battery.hpp File Reference . . . . .	174
8.15.1 Detailed Description . . . . .	175
8.16 inc/proxy/button.hpp File Reference . . . . .	175
8.16.1 Detailed Description . . . . .	176
8.17 inc/proxy/buzzer.hpp File Reference . . . . .	176
8.17.1 Detailed Description . . . . .	177
8.18 inc/proxy/current_sensors.hpp File Reference . . . . .	177
8.18.1 Detailed Description . . . . .	178
8.19 inc/proxy/dip_switch.hpp File Reference . . . . .	179
8.19.1 Detailed Description . . . . .	180
8.20 inc/proxy/distance_sensors.hpp File Reference . . . . .	180
8.20.1 Detailed Description . . . . .	181
8.21 inc/proxy/fan.hpp File Reference . . . . .	181
8.21.1 Detailed Description . . . . .	182
8.22 inc/proxy/imu.hpp File Reference . . . . .	182
8.22.1 Detailed Description . . . . .	183
8.23 inc/proxy/led.hpp File Reference . . . . .	183
8.23.1 Detailed Description . . . . .	184

8.24 inc/proxy/locomotion.hpp File Reference . . . . .	185
8.24.1 Detailed Description . . . . .	186
8.25 inc/proxy/rotary_sensor.hpp File Reference . . . . .	186
8.25.1 Detailed Description . . . . .	187
8.26 inc/proxy/rotary_sensor_reg.hpp File Reference . . . . .	187
8.26.1 Detailed Description . . . . .	188
8.27 inc/proxy/serializable_interface.hpp File Reference . . . . .	188
8.27.1 Detailed Description . . . . .	189
8.28 inc/proxy/storage.hpp File Reference . . . . .	189
8.28.1 Detailed Description . . . . .	190
8.29 inc/proxy/torque_sensors.hpp File Reference . . . . .	190
8.29.1 Detailed Description . . . . .	191
8.30 README.md File Reference . . . . .	191
8.31 src/controller/micras_controller.cpp File Reference . . . . .	191
8.31.1 Detailed Description . . . . .	192
8.32 src/hal/adc_dma.cpp File Reference . . . . .	192
8.32.1 Detailed Description . . . . .	192
8.33 src/hal/crc.cpp File Reference . . . . .	193
8.33.1 Detailed Description . . . . .	193
8.34 src/hal/encoder.cpp File Reference . . . . .	193
8.34.1 Detailed Description . . . . .	194
8.35 src/hal/flash.cpp File Reference . . . . .	194
8.35.1 Detailed Description . . . . .	194
8.36 src/hal/gpio.cpp File Reference . . . . .	195
8.36.1 Detailed Description . . . . .	195
8.37 src/hal/mcu.cpp File Reference . . . . .	195
8.37.1 Detailed Description . . . . .	196
8.37.2 Function Documentation . . . . .	196
8.37.2.1 SystemClock_Config() . . . . .	196
8.38 src/hal/pwm.cpp File Reference . . . . .	197
8.38.1 Detailed Description . . . . .	197
8.39 src/hal/pwm_dma.cpp File Reference . . . . .	197
8.39.1 Detailed Description . . . . .	198
8.40 src/hal/spi.cpp File Reference . . . . .	198
8.40.1 Detailed Description . . . . .	198
8.41 src/hal/timer.cpp File Reference . . . . .	199
8.41.1 Detailed Description . . . . .	199
8.42 src/main.cpp File Reference . . . . .	199
8.42.1 Detailed Description . . . . .	200
8.42.2 Function Documentation . . . . .	200
8.42.2.1 main() . . . . .	200
8.43 src/proxy/argb.cpp File Reference . . . . .	200



8.43.1 Detailed Description . . . . .	201
8.43.2 Macro Definition Documentation . . . . .	201
8.43.2.1 MICRAS_PROXY_ARGB_CPP . . . . .	201
8.44 src/proxy/battery.cpp File Reference . . . . .	202
8.44.1 Detailed Description . . . . .	202
8.45 src/proxy/button.cpp File Reference . . . . .	202
8.45.1 Detailed Description . . . . .	203
8.46 src/proxy/buzzer.cpp File Reference . . . . .	203
8.46.1 Detailed Description . . . . .	204
8.47 src/proxy/current_sensors.cpp File Reference . . . . .	204
8.47.1 Detailed Description . . . . .	205
8.47.2 Macro Definition Documentation . . . . .	205
8.47.2.1 MICRAS_PROXY_CURRENT_SENSORS_CPP . . . . .	205
8.48 src/proxy/dip_switch.cpp File Reference . . . . .	206
8.48.1 Detailed Description . . . . .	206
8.48.2 Macro Definition Documentation . . . . .	206
8.48.2.1 MICRAS_PROXY_DIP_SWITCH_CPP . . . . .	206
8.49 src/proxy/distance_sensors.cpp File Reference . . . . .	207
8.49.1 Macro Definition Documentation . . . . .	207
8.49.1.1 MICRAS_PROXY_DISTANCE_SENSORS_CPP . . . . .	207
8.50 src/proxy/fan.cpp File Reference . . . . .	208
8.50.1 Detailed Description . . . . .	208
8.51 src/proxy/imu.cpp File Reference . . . . .	208
8.51.1 Detailed Description . . . . .	209
8.52 src/proxy/led.cpp File Reference . . . . .	209
8.52.1 Detailed Description . . . . .	210
8.53 src/proxy/locomotion.cpp File Reference . . . . .	210
8.53.1 Detailed Description . . . . .	211
8.54 src/proxy/rotary_sensor.cpp File Reference . . . . .	211
8.54.1 Detailed Description . . . . .	212
8.55 src/proxy/storage.cpp File Reference . . . . .	212
8.55.1 Detailed Description . . . . .	213
8.56 src/proxy/torque_sensors.cpp File Reference . . . . .	213
8.56.1 Detailed Description . . . . .	214
8.56.2 Macro Definition Documentation . . . . .	214
8.56.2.1 MICRAS_PROXY_TORQUE_SENSORS_CPP . . . . .	214
<b>Index</b>	<b>215</b>



## Chapter 1

# Micras



## Chapter 2

## Todo List

Member [proxy::Imu::get\\_orientation](#) (Axis axis) const  
implement function using sensor fusion



## Chapter 3

# Namespace Index

### 3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">hal</a> . . . . .	<a href="#">15</a>
<a href="#">proxy</a> . . . . .	<a href="#">15</a>





## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">hal::AdcDma</a>	Class to handle ADC peripheral on STM32 microcontrollers using DMA . . . . .	17
<a href="#">proxy::Argb&lt; num_of_leds &gt;</a>	Class for controlling an addressable RGB LED . . . . .	20
<a href="#">proxy::Battery</a>	Class for getting the battery voltage . . . . .	23
<a href="#">proxy::Button</a>	Class for controlling a button . . . . .	26
<a href="#">proxy::Buzzer</a>	Class for controlling a buzzer . . . . .	30
<a href="#">proxy::Argb&lt; num_of_leds &gt;::Color</a>	Structure for storing color information . . . . .	33
<a href="#">hal::AdcDma::Config</a>	Configuration structure for ADC DMA . . . . .	34
<a href="#">hal::Crc::Config</a>	CRC configuration struct . . . . .	36
<a href="#">hal::Encoder::Config</a>	Encoder configuration struct . . . . .	37
<a href="#">hal::Gpio::Config</a>	Configuration structure for GPIO pin . . . . .	38
<a href="#">hal::Pwm::Config</a>	PWM configuration struct . . . . .	39
<a href="#">hal::PwmDma::Config</a>	PWM configuration struct . . . . .	41
<a href="#">hal::Spi::Config</a>	SPI configuration struct . . . . .	42
<a href="#">hal::Timer::Config</a>	Timer configuration struct . . . . .	44
<a href="#">proxy::Argb&lt; num_of_leds &gt;::Config</a>	Configuration structure for the addressable RGB LED . . . . .	45
<a href="#">proxy::Battery::Config</a>	Configuration structure for the battery . . . . .	46
<a href="#">proxy::Button::Config</a>	Configuration structure for button . . . . .	47
<a href="#">proxy::Buzzer::Config</a>	Configuration structure for the buzzer . . . . .	49

<a href="#">proxy::CurrentSensors&lt; num_of_sensors &gt;::Config</a>	
Configuration structure for current sensors	51
<a href="#">proxy::DipSwitch&lt; num_of_switches &gt;::Config</a>	
Configuration struct for <a href="#">DipSwitch</a>	52
<a href="#">proxy::DistanceSensors&lt; num_of_sensors &gt;::Config</a>	
Configuration structure for distance sensors	53
<a href="#">proxy::Fan::Config</a>	
Configuration structure for the fan	54
<a href="#">proxy::Imu::Config</a>	
IMU configuration struct	55
<a href="#">proxy::Led::Config</a>	
Configuration structure for LED	57
<a href="#">proxy::Locomotion::Config</a>	
Configuration structure for the locomotion	59
<a href="#">proxy::RotarySensor::Config</a>	
Rotary sensor configuration struct	60
<a href="#">proxy::Storage::Config</a>	
Configuration structure for the storage	62
<a href="#">proxy::TorqueSensors&lt; num_of_sensors &gt;::Config</a>	
Configuration structure for torque sensors	63
<a href="#">hal::Crc</a>	
Class to handle the cyclic redundancy check peripheral on STM32 microcontrollers	64
<a href="#">proxy::CurrentSensors&lt; num_of_sensors &gt;</a>	
Class for controlling <a href="#">CurrentSensors</a>	66
<a href="#">proxy::DipSwitch&lt; num_of_switches &gt;</a>	
Class for controlling a dip switch	68
<a href="#">Registers::Disable</a>	
<a href="#">Registers</a> union types definition	71
<a href="#">proxy::DistanceSensors&lt; num_of_sensors &gt;</a>	
Class for controlling <a href="#">DistanceSensors</a>	73
<a href="#">Registers::Ecc</a>	76
<a href="#">hal::Encoder</a>	
Class to handle encoder peripheral on STM32 microcontrollers	78
<a href="#">proxy::Fan</a>	
Class for controlling the fan driver	79
<a href="#">proxy::RotarySensor::CommandFrame::Fields</a>	84
<a href="#">proxy::RotarySensor::DataFrame::Fields</a>	85
<a href="#">Registers::Disable::Fields</a>	86
<a href="#">Registers::Ecc::Fields</a>	88
<a href="#">Registers::Settings1::Fields</a>	89
<a href="#">Registers::Settings2::Fields</a>	90
<a href="#">Registers::Settings3::Fields</a>	93
<a href="#">Registers::Zposl::Fields</a>	94
<a href="#">Registers::Zposm::Fields</a>	95
<a href="#">hal::Flash</a>	
Class to handle flash memory on STM32 microcontrollers	96
<a href="#">hal::Gpio</a>	
Class for controlling GPIO pins on STM32 microcontrollers	100
<a href="#">proxy::Imu</a>	
Class to handle IMU peripheral on STM32 microcontrollers	103
<a href="#">ISerializable</a>	
Interface class for serializable classes	107
<a href="#">proxy::Led</a>	
Class for controlling an LED	110
<a href="#">proxy::Locomotion</a>	
Class for controlling the locomotion driver	113
<a href="#">hal::Mcu</a>	
Microcontroller unit class	119

<a href="#">MicrasController</a>	
Class for controlling the Micras robot . . . . .	121
<a href="#">hal::Pwm</a>	
Class to handle PWM peripheral on STM32 microcontrollers . . . . .	122
<a href="#">hal::PwmDma</a>	
Class to handle PWM peripheral on STM32 microcontrollers using DMA . . . . .	125
<a href="#">Registers</a>	
<a href="#">Registers</a> class for the rotary sensor configuration . . . . .	127
<a href="#">proxy::RotarySensor</a>	
Class to handle rotary sensor peripheral on STM32 microcontrollers . . . . .	131
<a href="#">Registers::Settings1</a> . . . . .	132
<a href="#">Registers::Settings2</a> . . . . .	134
<a href="#">Registers::Settings3</a> . . . . .	135
<a href="#">hal::Spi</a>	
Class to handle SPI peripheral on STM32 microcontrollers . . . . .	137
<a href="#">proxy::Storage</a>	
Class for controlling the storage . . . . .	140
<a href="#">hal::Timer</a>	
Class to handle timer peripheral on STM32 microcontrollers . . . . .	144
<a href="#">proxy::TorqueSensors&lt; num_of_sensors &gt;</a>	
Class for controlling <a href="#">TorqueSensors</a> . . . . .	148
<a href="#">Registers::Zposl</a> . . . . .	150
<a href="#">Registers::Zposm</a> . . . . .	152



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">cfg/target.cpp</a>		
Target specific configuration	155	
<a href="#">cfg/target.hpp</a>		
Target specific configuration	160	
<a href="#">inc/controller/micras_controller.hpp</a>		163
<a href="#">inc/hal/adc_dma.hpp</a>		
ADC DMA HAL header	164	
<a href="#">inc/hal/crc.hpp</a>		
STM32 CRC HAL wrapper	165	
<a href="#">inc/hal/encoder.hpp</a>		
STM32 encoder HAL wrapper	166	
<a href="#">inc/hal/flash.hpp</a>		
STM32 flash HAL wrapper	167	
<a href="#">inc/hal/gpio.hpp</a>		
HAL GPIO class header	167	
<a href="#">inc/hal/mcu.hpp</a>		
MCU related	168	
<a href="#">inc/hal/pwm.hpp</a>		
STM32 PWM HAL wrapper	169	
<a href="#">inc/hal/pwm_dma.hpp</a>		
STM32 PWM DMA HAL wrapper	170	
<a href="#">inc/hal/spi.hpp</a>		
STM32 SPI HAL wrapper	171	
<a href="#">inc/hal/timer.hpp</a>		
STM32 Timer HAL wrapper	172	
<a href="#">inc/proxy/argb.hpp</a>		
Proxy Argb class declaration	173	
<a href="#">inc/proxy/battery.hpp</a>		
Proxy Battery class declaration	174	
<a href="#">inc/proxy/button.hpp</a>		
Proxy Button class header	175	
<a href="#">inc/proxy/buzzer.hpp</a>		
Proxy Buzzer class declaration	176	
<a href="#">inc/proxy/current_sensors.hpp</a>		
Proxy CurrentSensors class header	177	

inc/proxy/dip_switch.hpp	
Proxy Dip Switch class header	179
inc/proxy/distance_sensors.hpp	
Proxy DistanceSensors class header	180
inc/proxy/fan.hpp	
Proxy Fan class declaration	181
inc/proxy/imu.hpp	
STM32 IMU HAL wrapper	182
inc/proxy/led.hpp	
Proxy Led class header	183
inc/proxy/locomotion.hpp	
Proxy Locomotion class declaration	185
inc/proxy/rotary_sensor.hpp	
STM32 rotary sensor HAL wrapper	186
inc/proxy/rotary_sensor_reg.hpp	
AS5047U rotary sensor registers definition	187
inc/proxy/serializable_interface.hpp	
Serializable interface for all classes that need to be serialized	188
inc/proxy/storage.hpp	
Proxy Storage class declaration	189
inc/proxy/torque_sensors.hpp	
Proxy TorqueSensors class header	190
src/main.cpp	
Main function	199
src/controller/micras_controller.cpp	
Micras Controller class implementation	191
src/hal/adc_dma.cpp	
STM32 ADC DMA HAL wrapper	192
src/hal/crc.cpp	
STM32 CRC HAL wrapper	193
src/hal/encoder.cpp	
STM32 encoder HAL wrapper	193
src/hal/flash.cpp	
STM32 flash HAL wrapper	194
src/hal/gpio.cpp	
HAL GPIO class source	195
src/hal/mcu.cpp	
MCU related	195
src/hal/pwm.cpp	
STM32 PWM HAL wrapper	197
src/hal/pwm_dma.cpp	
STM32 PWM DMA HAL wrapper	197
src/hal/spi.cpp	
Proxy SPI Switch class source	198
src/hal/timer.cpp	
STM32 TIM HAL wrapper	199
src/proxy/argb.cpp	
Proxy Argb class implementation	200
src/proxy/battery.cpp	
Proxy Battery class implementation	202
src/proxy/button.cpp	
Proxy Button class source	202
src/proxy/buzzer.cpp	
Proxy Buzzer class implementation	203
src/proxy/current_sensors.cpp	
Proxy CurrentSensors class implementation	204
src/proxy/dip_switch.cpp	
Proxy DIP Switch class source	206

src/proxy/ <a href="#">distance_sensors.cpp</a> . . . . .	207
src/proxy/ <a href="#">fan.cpp</a>	
Proxy Fan class source . . . . .	208
src/proxy/ <a href="#">imu.cpp</a>	
Proxy Imu class source . . . . .	208
src/proxy/ <a href="#">led.cpp</a>	
Proxy Led class source . . . . .	209
src/proxy/ <a href="#">locomotion.cpp</a>	
Proxy Locomotion class source . . . . .	210
src/proxy/ <a href="#">rotary_sensor.cpp</a>	
Proxy RotarySensor class source . . . . .	211
src/proxy/ <a href="#">storage.cpp</a>	
Proxy Storage class source . . . . .	212
src/proxy/ <a href="#">torque_sensors.cpp</a>	
Proxy TorqueSensors class implementation . . . . .	213





## Chapter 6

# Namespace Documentation

### 6.1 hal Namespace Reference

#### Classes

- class [AdcDma](#)  
*Class to handle ADC peripheral on STM32 microcontrollers using DMA.*
- class [Crc](#)  
*Class to handle the cyclic redundancy check peripheral on STM32 microcontrollers.*
- class [Encoder](#)  
*Class to handle encoder peripheral on STM32 microcontrollers.*
- class [Flash](#)  
*Class to handle flash memory on STM32 microcontrollers.*
- class [Gpio](#)  
*Class for controlling GPIO pins on STM32 microcontrollers.*
- class [Mcu](#)  
*Microcontroller unit class.*
- class [Pwm](#)  
*Class to handle PWM peripheral on STM32 microcontrollers.*
- class [PwmDma](#)  
*Class to handle PWM peripheral on STM32 microcontrollers using DMA.*
- class [Spi](#)  
*Class to handle SPI peripheral on STM32 microcontrollers.*
- class [Timer](#)  
*Class to handle timer peripheral on STM32 microcontrollers.*

### 6.2 proxy Namespace Reference

#### Classes

- class [Argb](#)  
*Class for controlling an addressable RGB LED.*
- class [Battery](#)  
*Class for getting the battery voltage.*

- class [Button](#)  
*Class for controlling a button.*
- class [Buzzer](#)  
*Class for controlling a buzzer.*
- class [CurrentSensors](#)  
*Class for controlling [CurrentSensors](#).*
- class [DipSwitch](#)  
*Class for controlling a dip switch.*
- class [DistanceSensors](#)  
*Class for controlling [DistanceSensors](#).*
- class [Fan](#)  
*Class for controlling the fan driver.*
- class [Imu](#)  
*Class to handle IMU peripheral on STM32 microcontrollers.*
- class [Led](#)  
*Class for controlling an LED.*
- class [Locomotion](#)  
*Class for controlling the locomotion driver.*
- class [RotarySensor](#)  
*Class to handle rotary sensor peripheral on STM32 microcontrollers.*
- class [Storage](#)  
*Class for controlling the storage.*
- class [TorqueSensors](#)  
*Class for controlling [TorqueSensors](#).*

## Variables

- `template<typename T >`  
`concept Fundamental = std::is_fundamental<T>::value`

## 6.2.1 Variable Documentation

### 6.2.1.1 Fundamental

```
template<typename T >
concept proxy::Fundamental = std::is_fundamental<T>::value
```

## Chapter 7

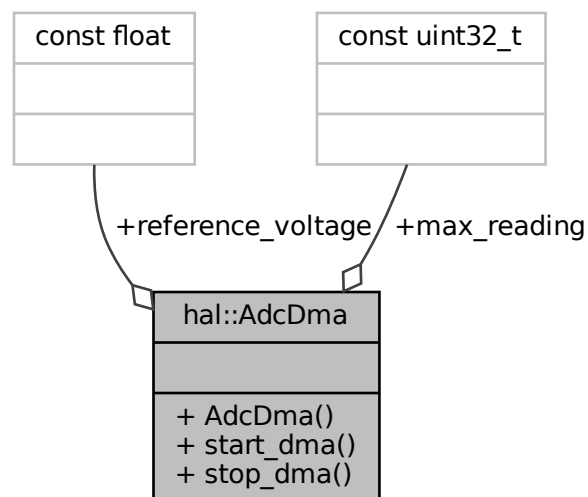
# Class Documentation

### 7.1 hal::AdcDma Class Reference

Class to handle ADC peripheral on STM32 microcontrollers using DMA.

```
#include <adc_dma.hpp>
```

Collaboration diagram for hal::AdcDma:



#### Classes

- struct [Config](#)

*Configuration structure for ADC DMA.*

## Public Member Functions

- [AdcDma](#) (const [Config](#) &config)  
*Construct a new [AdcDma](#) object.*
- void [start\\_dma](#) (uint32\_t buffer[], uint32\_t size)  
*Enable ADC, start conversion of regular group and transfer result through DMA.*
- void [stop\\_dma](#) ()  
*Stop ADC conversion of regular group (and injected group in case of auto\_injection mode)*

## Public Attributes

- const uint32\_t [max\\_reading](#)  
*Maximum ADC reading.*
- const float [reference\\_voltage](#)  
*Reference voltage for the ADC measurement.*

### 7.1.1 Detailed Description

Class to handle ADC peripheral on STM32 microcontrollers using DMA.

### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 AdcDma()

```
hal::AdcDma::AdcDma (
    const Config & config ) [explicit]
```

Construct a new [AdcDma](#) object.

Parameters

<i>config</i>	ADC DMA configuration struct
---------------	------------------------------

### 7.1.3 Member Function Documentation

#### 7.1.3.1 start\_dma()

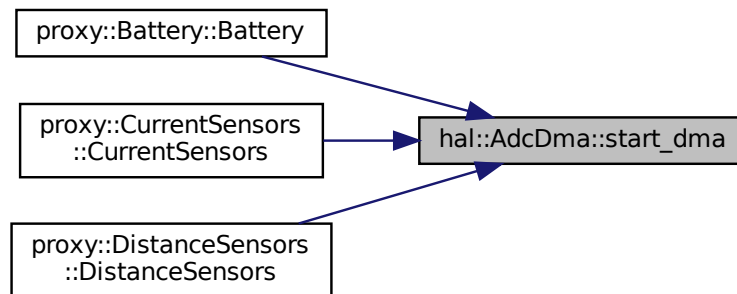
```
void hal::AdcDma::start_dma (
    uint32_t buffer[],
    uint32_t size )
```

Enable ADC, start conversion of regular group and transfer result through DMA.

## Parameters

<i>buffer</i>	Destination Buffer address
<i>size</i>	Number of data to be transferred from ADC DMA peripheral to memory

Here is the caller graph for this function:



### 7.1.3.2 stop\_dma()

```
void hal::AdcDma::stop_dma ( )
```

Stop ADC conversion of regular group (and injected group in case of auto\_injection mode)

## 7.1.4 Member Data Documentation

### 7.1.4.1 max\_reading

```
const uint32_t hal::AdcDma::max_reading
```

Maximum ADC reading.

### 7.1.4.2 reference\_voltage

```
const float hal::AdcDma::reference_voltage
```

Reference voltage for the ADC measurement.

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

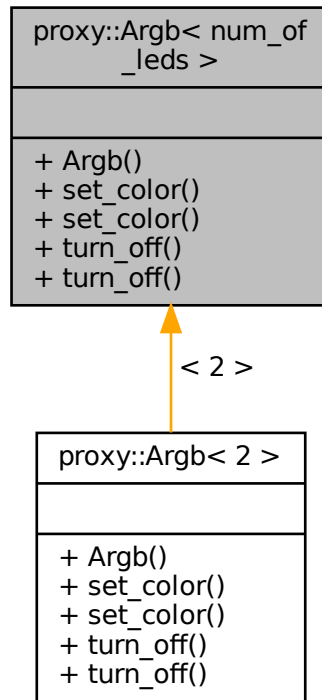
- [inc/hal/adc\\_dma.hpp](#)
- [src/hal/adc\\_dma.cpp](#)

## 7.2 proxy::Argb< num\_of\_leds > Class Template Reference

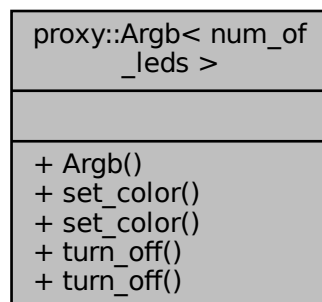
Class for controlling an addressable RGB LED.

```
#include <argb.hpp>
```

Inheritance diagram for proxy::Argb< num\_of\_leds >:



Collaboration diagram for proxy::Argb< num\_of\_leds >:



## Classes

- struct [Color](#)  
*Structure for storing color information.*
- struct [Config](#)  
*Configuration structure for the addressable RGB LED.*

## Public Member Functions

- [Argb](#) (const [Config](#) &config)  
*Constructor for the [Argb](#) class.*
- void [set\\_color](#) (const [Color](#) &color, uint8\_t index)  
*Set the color of the ARGB at the specified index.*
- void [set\\_color](#) (const [Color](#) &color)  
*Set the color of all ARGBs.*
- void [turn\\_off](#) (uint8\_t index)  
*Turn off the ARGB at the specified index.*
- void [turn\\_off](#) ()  
*Turn off all ARGBs.*

### 7.2.1 Detailed Description

```
template<uint8_t num_of_leds>
class proxy::Argb< num_of_leds >
```

Class for controlling an addressable RGB LED.

### 7.2.2 Constructor & Destructor Documentation

#### 7.2.2.1 Argb()

```
template<uint8_t num_of_leds>
proxy::Argb< num_of_leds >::Argb (
    const Config & config ) [explicit]
```

Constructor for the [Argb](#) class.

#### Parameters

<i>config</i>	Configuration for the addressable RGB LED
---------------	---

### 7.2.3 Member Function Documentation

**7.2.3.1 set\_color()** [1/2]

```
template<uint8_t num_of_leds>
void proxy::Argb< num_of_leds >::set_color (
    const Color & color )
```

Set the color of all ARGBs.

**Parameters**

<i>color</i>	The color to set all ARGBs to
--------------	-------------------------------

**7.2.3.2 set\_color()** [2/2]

```
template<uint8_t num_of_leds>
void proxy::Argb< num_of_leds >::set_color (
    const Color & color,
    uint8_t index )
```

Set the color of the ARGB at the specified index.

**Parameters**

<i>index</i>	The index of the ARGB to set the color of
<i>color</i>	The color to set the ARGB to

**7.2.3.3 turn\_off()** [1/2]

```
template<uint8_t num_of_leds>
void proxy::Argb< num_of_leds >::turn_off
```

Turn off all ARGBs.

**7.2.3.4 turn\_off()** [2/2]

```
template<uint8_t num_of_leds>
void proxy::Argb< num_of_leds >::turn_off (
    uint8_t index )
```

Turn off the ARGB at the specified index.

**Parameters**

<i>index</i>	The index of the ARGB to turn off
--------------	-----------------------------------



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

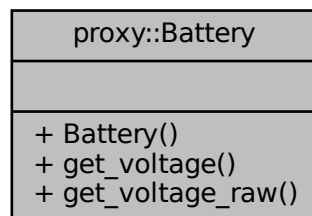
- inc/proxy/[argb.hpp](#)
- src/proxy/[argb.cpp](#)

## 7.3 proxy::Battery Class Reference

Class for getting the battery voltage.

```
#include <battery.hpp>
```

Collaboration diagram for proxy::Battery:



### Classes

- struct [Config](#)  
*Configuration structure for the battery.*

### Public Member Functions

- [Battery](#) (const [Config](#) &config)  
*Constructor for the [Battery](#) class.*
- float [get\\_voltage](#) () const  
*Get the battery voltage.*
- uint32\_t [get\\_voltage\\_raw](#) () const  
*Get the raw reading from the battery.*

#### 7.3.1 Detailed Description

Class for getting the battery voltage.

## 7.3.2 Constructor & Destructor Documentation

### 7.3.2.1 Battery()

```
proxy::Battery::Battery (  
    const Config & config ) [explicit]
```

Constructor for the [Battery](#) class.

## Parameters

<i>config</i>	Configuration for the battery
---------------	-------------------------------

Here is the call graph for this function:



### 7.3.3 Member Function Documentation

#### 7.3.3.1 get\_voltage()

```
float proxy::Battery::get_voltage ( ) const
```

Get the battery voltage.

## Returns

float [Battery](#) voltage in volts

#### 7.3.3.2 get\_voltage\_raw()

```
uint32_t proxy::Battery::get_voltage_raw ( ) const
```

Get the raw reading from the battery.

## Returns

uint32\_t Raw reading from the battery

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

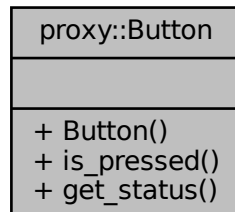
- inc/proxy/[battery.hpp](#)
- src/proxy/[battery.cpp](#)

## 7.4 proxy::Button Class Reference

Class for controlling a button.

```
#include <button.hpp>
```

Collaboration diagram for proxy::Button:



### Classes

- struct [Config](#)  
*Configuration structure for button.*

### Public Types

- enum [Status](#) { [NO\\_PRESS](#) , [SHORT\\_PRESS](#) , [LONG\\_PRESS](#) , [EXTRA\\_LONG\\_PRESS](#) }  
*Enum for button status.*
- enum [PullResistor](#) { [PULL\\_UP](#) , [PULL\\_DOWN](#) }  
*Enum for button pull resistor.*

### Public Member Functions

- [Button](#) (const [Config](#) &config)  
*Constructor for [Button](#) class.*
- bool [is\\_pressed](#) ()  
*Check if button is pressed.*
- [Status](#) [get\\_status](#) ()  
*Get button status.*

#### 7.4.1 Detailed Description

Class for controlling a button.

## 7.4.2 Member Enumeration Documentation

### 7.4.2.1 PullResistor

enum `proxy::Button::PullResistor`

Enum for button pull resistor.

## Enumerator

PULL_UP	
PULL_DOWN	

### 7.4.2.2 Status

```
enum proxy::Button::Status
```

Enum for button status.

## Enumerator

NO_PRESS	
SHORT_PRESS	
LONG_PRESS	
EXTRA_LONG_PRESS	

## 7.4.3 Constructor & Destructor Documentation

### 7.4.3.1 Button()

```
proxy::Button::Button (  
    const Config & config ) [explicit]
```

Constructor for [Button](#) class.

## Parameters

<i>config</i>	<a href="#">Button</a> configuration
---------------	--------------------------------------

## 7.4.4 Member Function Documentation

### 7.4.4.1 get\_status()

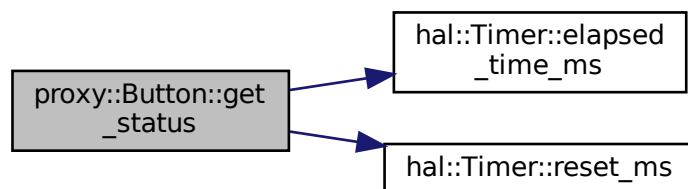
```
Button::Status proxy::Button::get_status ( )
```

Get button status.

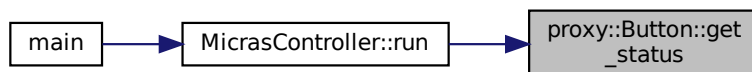
**Returns**

Status [Button](#) status

Here is the call graph for this function:



Here is the caller graph for this function:

**7.4.4.2 is\_pressed()**

```
bool proxy::Button::is_pressed ( )
```

Check if button is pressed.

**Returns**

bool True if button is pressed, false otherwise

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

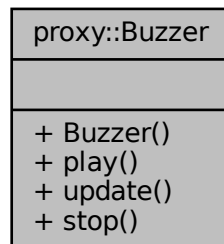
- [inc/proxy/button.hpp](#)
- [src/proxy/button.cpp](#)

## 7.5 proxy::Buzzer Class Reference

Class for controlling a buzzer.

```
#include <buzzer.hpp>
```

Collaboration diagram for proxy::Buzzer:



### Classes

- struct [Config](#)  
*Configuration structure for the buzzer.*

### Public Member Functions

- [Buzzer](#) (const [Config](#) &config)  
*Constructor for the [Buzzer](#) class.*
- void [play](#) (uint32\_t frequency, uint32\_t duration=0)  
*Play a tone for a duration.*
- void [update](#) ()  
*Update the buzzer state.*
- void [stop](#) ()  
*Stop the buzzer sound.*

#### 7.5.1 Detailed Description

Class for controlling a buzzer.

#### 7.5.2 Constructor & Destructor Documentation

##### 7.5.2.1 Buzzer()

```
proxy::Buzzer::Buzzer (
    const Config & config ) [explicit]
```

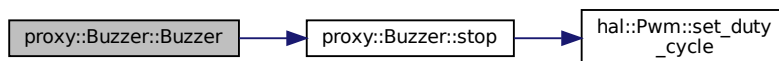
Constructor for the [Buzzer](#) class.



## Parameters

<i>config</i>	Configuration for the buzzer
---------------	------------------------------

Here is the call graph for this function:



### 7.5.3 Member Function Documentation

#### 7.5.3.1 play()

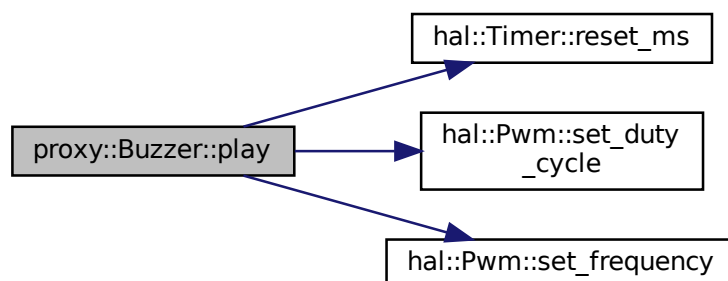
```
void proxy::Buzzer::play (
    uint32_t frequency,
    uint32_t duration = 0 )
```

Play a tone for a duration.

## Parameters

<i>frequency</i>	Buzzer sound frequency in Hz
<i>duration</i>	Duration of the sound in ms

Here is the call graph for this function:

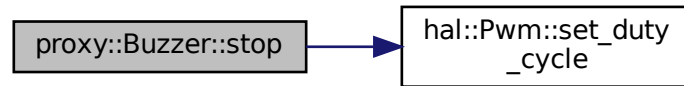


### 7.5.3.2 stop()

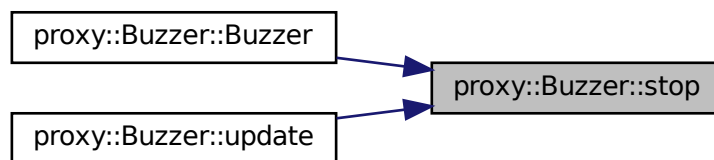
```
void proxy::Buzzer::stop ( )
```

Stop the buzzer sound.

Here is the call graph for this function:



Here is the caller graph for this function:

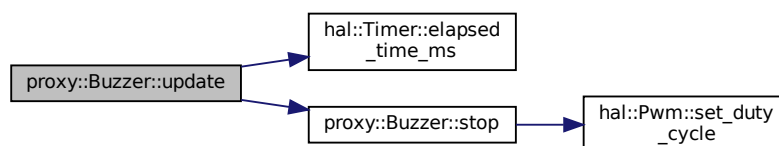


### 7.5.3.3 update()

```
void proxy::Buzzer::update ( )
```

Update the buzzer state.

Here is the call graph for this function:



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

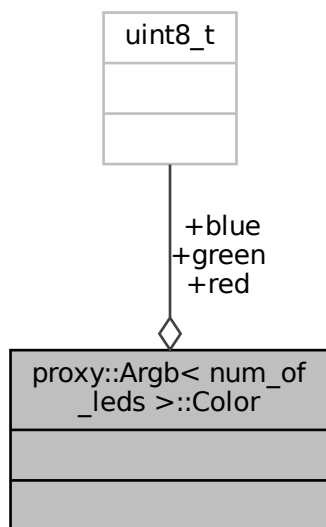
- [inc/proxy/buzzer.hpp](#)
- [src/proxy/buzzer.cpp](#)

## 7.6 proxy::Argb< num\_of\_leds >::Color Struct Reference

Structure for storing color information.

```
#include <argb.hpp>
```

Collaboration diagram for proxy::Argb< num\_of\_leds >::Color:



### Public Attributes

- `uint8_t` [red](#)
- `uint8_t` [green](#)
- `uint8_t` [blue](#)

### 7.6.1 Detailed Description

```
template<uint8_t num_of_leds>
struct proxy::Argb< num_of_leds >::Color
```

Structure for storing color information.

### 7.6.2 Member Data Documentation

### 7.6.2.1 blue

```
template<uint8_t num_of_leds>
uint8_t proxy::Argb< num_of_leds >::Color::blue
```

### 7.6.2.2 green

```
template<uint8_t num_of_leds>
uint8_t proxy::Argb< num_of_leds >::Color::green
```

### 7.6.2.3 red

```
template<uint8_t num_of_leds>
uint8_t proxy::Argb< num_of_leds >::Color::red
```

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

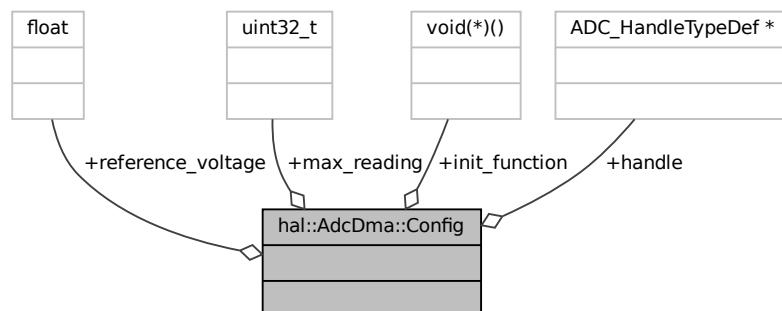
- [inc/proxy/argb.hpp](#)

## 7.7 hal::AdcDma::Config Struct Reference

Configuration structure for ADC DMA.

```
#include <adc_dma.hpp>
```

Collaboration diagram for hal::AdcDma::Config:



## Public Attributes

- ADC\_HandleTypeDef \* [handle](#)
- void(\* [init\\_function](#) )()
- uint32\_t [max\\_reading](#)
- float [reference\\_voltage](#)

### 7.7.1 Detailed Description

Configuration structure for ADC DMA.

### 7.7.2 Member Data Documentation

#### 7.7.2.1 handle

```
ADC_HandleTypeDef* hal::AdcDma::Config::handle
```

#### 7.7.2.2 init\_function

```
void(* hal::AdcDma::Config::init_function) ()
```

#### 7.7.2.3 max\_reading

```
uint32_t hal::AdcDma::Config::max_reading
```

#### 7.7.2.4 reference\_voltage

```
float hal::AdcDma::Config::reference_voltage
```

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

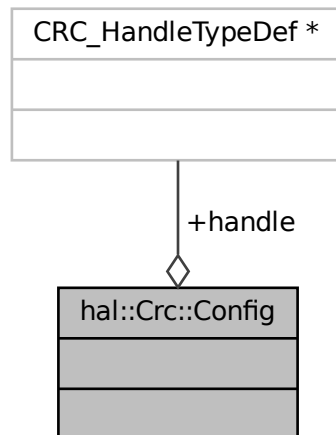
- inc/hal/[adc\\_dma.hpp](#)

## 7.8 hal::Crc::Config Struct Reference

CRC configuration struct.

```
#include <crc.hpp>
```

Collaboration diagram for hal::Crc::Config:



### Public Attributes

- CRC\_HandleTypeDef \* [handle](#)

### 7.8.1 Detailed Description

CRC configuration struct.

### 7.8.2 Member Data Documentation

#### 7.8.2.1 handle

```
CRC_HandleTypeDef* hal::Crc::Config::handle
```

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

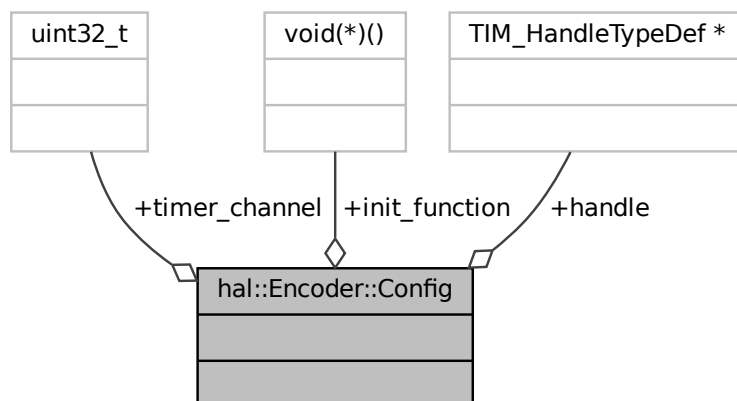
- inc/hal/[crc.hpp](#)

## 7.9 hal::Encoder::Config Struct Reference

[Encoder](#) configuration struct.

```
#include <encoder.hpp>
```

Collaboration diagram for hal::Encoder::Config:



### Public Attributes

- TIM\_HandleTypeDef \* [handle](#)
- void(\* [init\\_function](#) )()
- uint32\_t [timer\\_channel](#)

### 7.9.1 Detailed Description

[Encoder](#) configuration struct.

### 7.9.2 Member Data Documentation

#### 7.9.2.1 handle

```
TIM_HandleTypeDef* hal::Encoder::Config::handle
```

### 7.9.2.2 init\_function

```
void(* hal::Encoder::Config::init_function) ()
```

### 7.9.2.3 timer\_channel

```
uint32_t hal::Encoder::Config::timer_channel
```

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

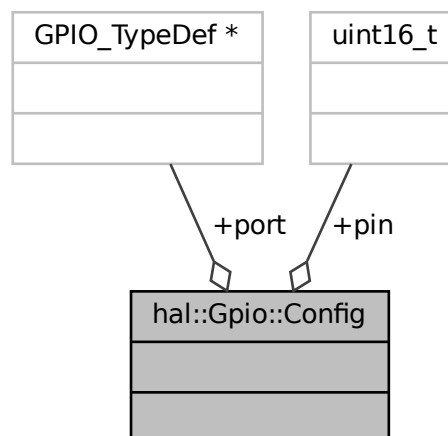
- [inc/hal/encoder.hpp](#)

## 7.10 hal::Gpio::Config Struct Reference

Configuration structure for GPIO pin.

```
#include <gpio.hpp>
```

Collaboration diagram for hal::Gpio::Config:



### Public Attributes

- `GPIO_TypeDef *` [port](#)
- `uint16_t` [pin](#)



### 7.10.1 Detailed Description

Configuration structure for GPIO pin.

### 7.10.2 Member Data Documentation

#### 7.10.2.1 pin

```
uint16_t hal::Gpio::Config::pin
```

#### 7.10.2.2 port

```
GPIO_TypeDef* hal::Gpio::Config::port
```

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

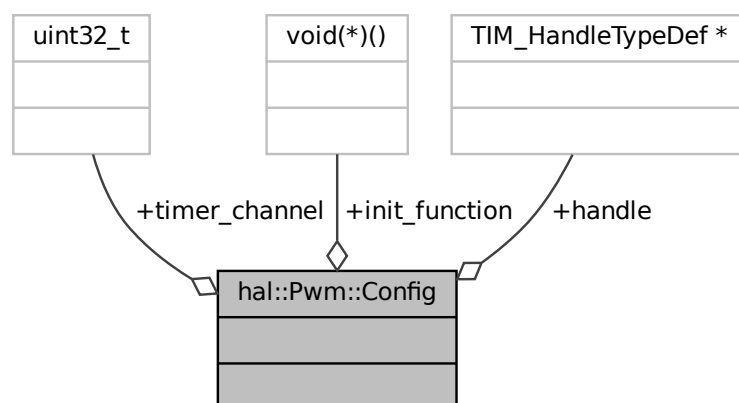
- [inc/hal/gpio.hpp](#)

## 7.11 hal::Pwm::Config Struct Reference

PWM configuration struct.

```
#include <pwm.hpp>
```

Collaboration diagram for hal::Pwm::Config:



## Public Attributes

- TIM\_HandleTypeDef \* [handle](#)
- void(\* [init\\_function](#) )()
- uint32\_t [timer\\_channel](#)

### 7.11.1 Detailed Description

PWM configuration struct.

### 7.11.2 Member Data Documentation

#### 7.11.2.1 [handle](#)

```
TIM_HandleTypeDef* hal::Pwm::Config::handle
```

#### 7.11.2.2 [init\\_function](#)

```
void(* hal::Pwm::Config::init_function) ()
```

#### 7.11.2.3 [timer\\_channel](#)

```
uint32_t hal::Pwm::Config::timer_channel
```

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

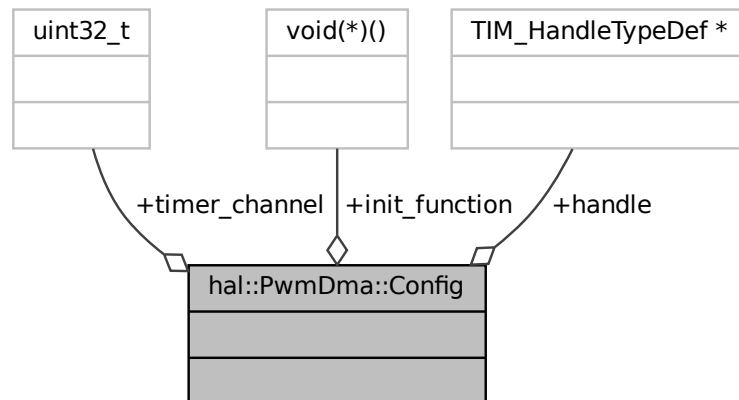
- [inc/hal/pwm.hpp](#)

## 7.12 hal::PwmDma::Config Struct Reference

PWM configuration struct.

```
#include <pwm_dma.hpp>
```

Collaboration diagram for hal::PwmDma::Config:



### Public Attributes

- TIM\_HandleTypeDef \* [handle](#)
- void(\* [init\\_function](#) )()
- uint32\_t [timer\\_channel](#)

### 7.12.1 Detailed Description

PWM configuration struct.

### 7.12.2 Member Data Documentation

#### 7.12.2.1 handle

```
TIM_HandleTypeDef* hal::PwmDma::Config::handle
```

### 7.12.2.2 init\_function

```
void(* hal::PwmDma::Config::init_function) ()
```

### 7.12.2.3 timer\_channel

```
uint32_t hal::PwmDma::Config::timer_channel
```

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

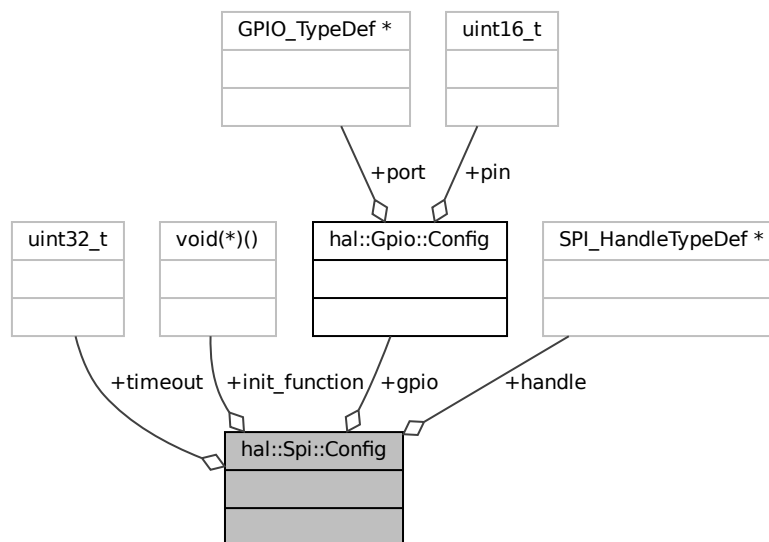
- [inc/hal/pwm\\_dma.hpp](#)

## 7.13 hal::Spi::Config Struct Reference

SPI configuration struct.

```
#include <spi.hpp>
```

Collaboration diagram for hal::Spi::Config:



### Public Attributes

- SPI\_HandleTypeDef \* [handle](#)
- void(\* [init\\_function](#))()
- [hal::Gpio::Config](#) [gpio](#)
- uint32\_t [timeout](#)

### 7.13.1 Detailed Description

SPI configuration struct.

### 7.13.2 Member Data Documentation

#### 7.13.2.1 gpio

`hal::Gpio::Config` `hal::Spi::Config::gpio`

#### 7.13.2.2 handle

`SPI_HandleTypeDef*` `hal::Spi::Config::handle`

#### 7.13.2.3 init\_function

`void(*` `hal::Spi::Config::init_function)` `()`

#### 7.13.2.4 timeout

`uint32_t` `hal::Spi::Config::timeout`

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

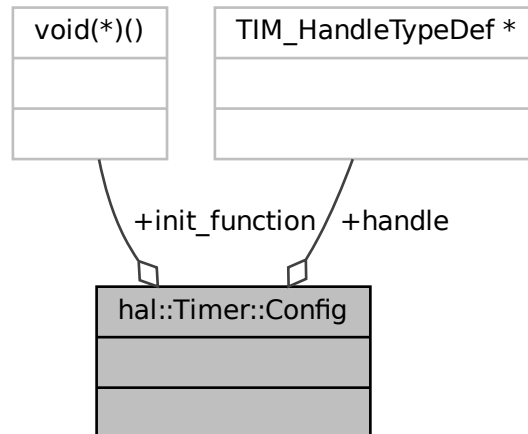
- `inc/hal/spi.hpp`

## 7.14 hal::Timer::Config Struct Reference

Timer configuration struct.

```
#include <timer.hpp>
```

Collaboration diagram for hal::Timer::Config:



### Public Attributes

- TIM\_HandleTypeDef \* [handle](#)
- void(\* [init\\_function](#) )()

#### 7.14.1 Detailed Description

Timer configuration struct.

#### 7.14.2 Member Data Documentation

##### 7.14.2.1 handle

```
TIM_HandleTypeDef* hal::Timer::Config::handle
```

### 7.14.2.2 init\_function

```
void(* hal::Timer::Config::init_function) ()
```

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

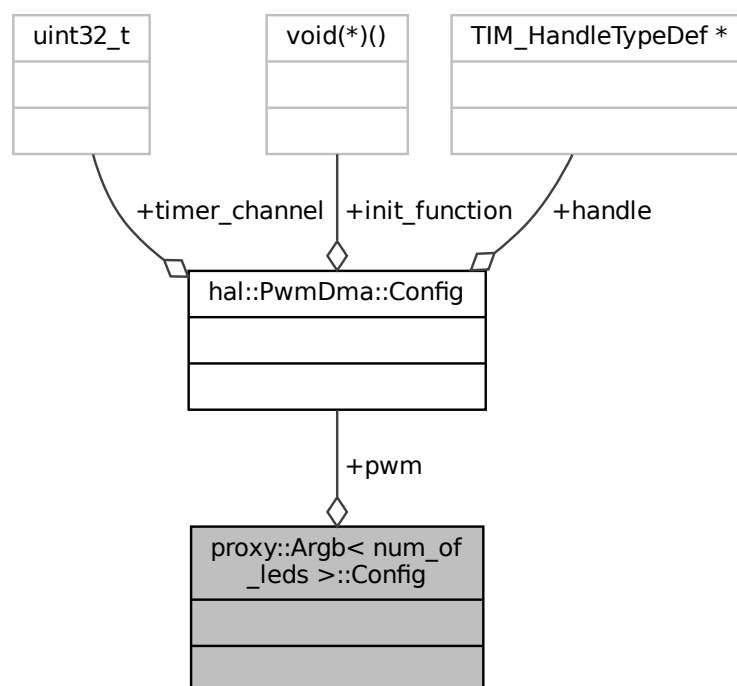
- [inc/hal/timer.hpp](#)

## 7.15 proxy::Argb< num\_of\_leds >::Config Struct Reference

Configuration structure for the addressable RGB LED.

```
#include <argb.hpp>
```

Collaboration diagram for proxy::Argb< num\_of\_leds >::Config:



### Public Attributes

- [hal::PwmDma::Config pwm](#)

### 7.15.1 Detailed Description

```
template<uint8_t num_of_leds>
struct proxy::Argb< num_of_leds >::Config
```

Configuration structure for the addressable RGB LED.

### 7.15.2 Member Data Documentation

#### 7.15.2.1 pwm

```
template<uint8_t num_of_leds>
hal::PwmDma::Config proxy::Argb< num_of_leds >::Config::pwm
```

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

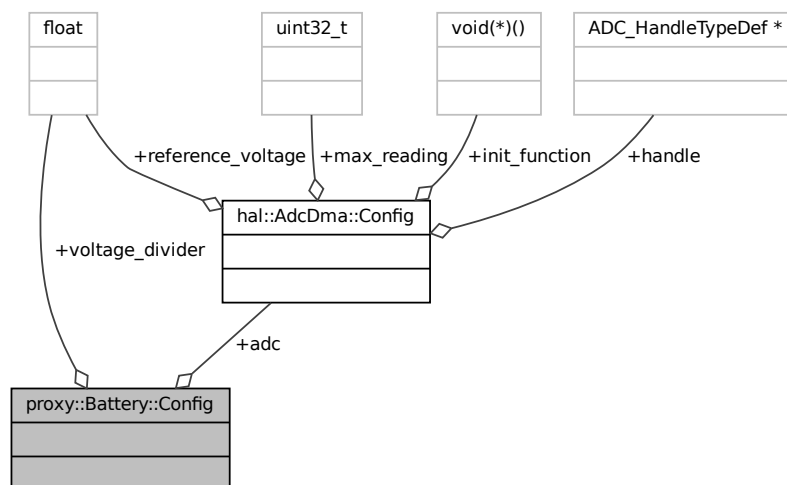
- [inc/proxy/argb.hpp](#)

## 7.16 proxy::Battery::Config Struct Reference

Configuration structure for the battery.

```
#include <battery.hpp>
```

Collaboration diagram for proxy::Battery::Config:





## Public Attributes

- [hal::AdcDma::Config](#) `adc`
- float `voltage_divider`

### 7.16.1 Detailed Description

Configuration structure for the battery.

### 7.16.2 Member Data Documentation

#### 7.16.2.1 `adc`

`hal::AdcDma::Config` `proxy::Battery::Config::adc`

#### 7.16.2.2 `voltage_divider`

float `proxy::Battery::Config::voltage_divider`

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

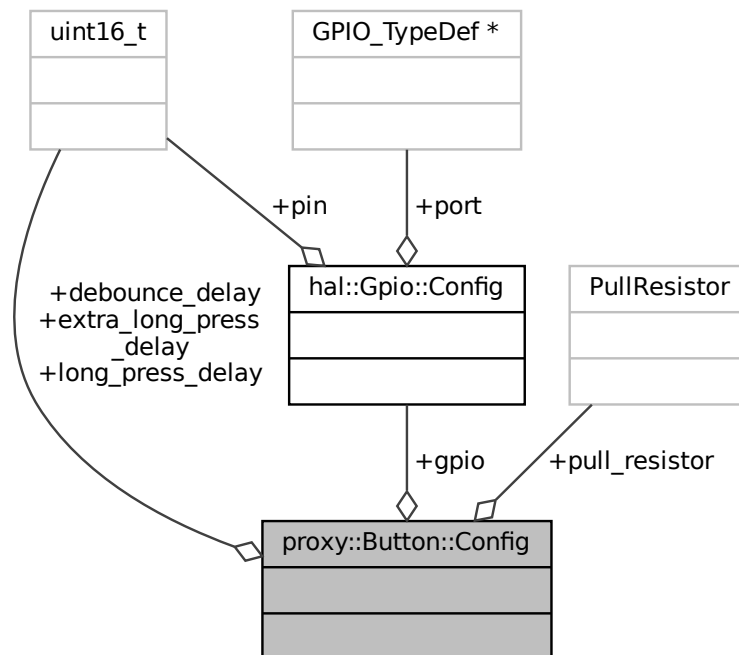
- `inc/proxy/battery.hpp`

## 7.17 proxy::Button::Config Struct Reference

Configuration structure for button.

```
#include <button.hpp>
```

Collaboration diagram for proxy::Button::Config:



## Public Attributes

- `hal::Gpio::Config gpio { }`
- `PullResistor pull_resistor { }`
- `uint16_t debounce_delay = 10`
- `uint16_t long_press_delay = 1000`
- `uint16_t extra_long_press_delay = 5000`

## 7.17.1 Detailed Description

Configuration structure for button.

## 7.17.2 Member Data Documentation

### 7.17.2.1 debounce\_delay

```
uint16_t proxy::Button::Config::debounce_delay = 10
```

### 7.17.2.2 extra\_long\_press\_delay

```
uint16_t proxy::Button::Config::extra_long_press_delay = 5000
```

### 7.17.2.3 gpio

```
hal::Gpio::Config proxy::Button::Config::gpio { }
```

### 7.17.2.4 long\_press\_delay

```
uint16_t proxy::Button::Config::long_press_delay = 1000
```

### 7.17.2.5 pull\_resistor

```
PullResistor proxy::Button::Config::pull_resistor { }
```

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

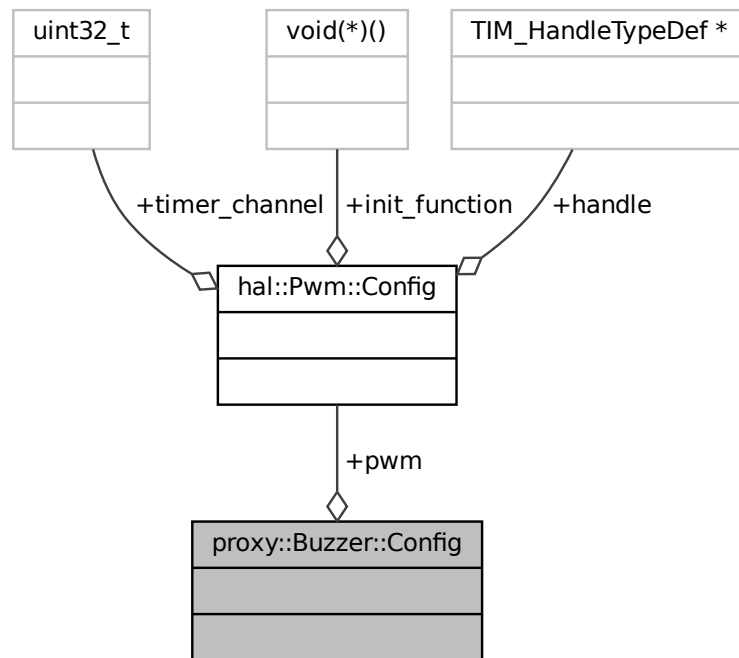
- inc/proxy/[button.hpp](#)

## 7.18 proxy::Buzzer::Config Struct Reference

Configuration structure for the buzzer.

```
#include <buzzer.hpp>
```

Collaboration diagram for proxy::Buzzer::Config:



## Public Attributes

- [hal::Pwm::Config pwm](#)

## 7.18.1 Detailed Description

Configuration structure for the buzzer.

## 7.18.2 Member Data Documentation

### 7.18.2.1 pwm

[hal::Pwm::Config](#) proxy::Buzzer::Config::pwm

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

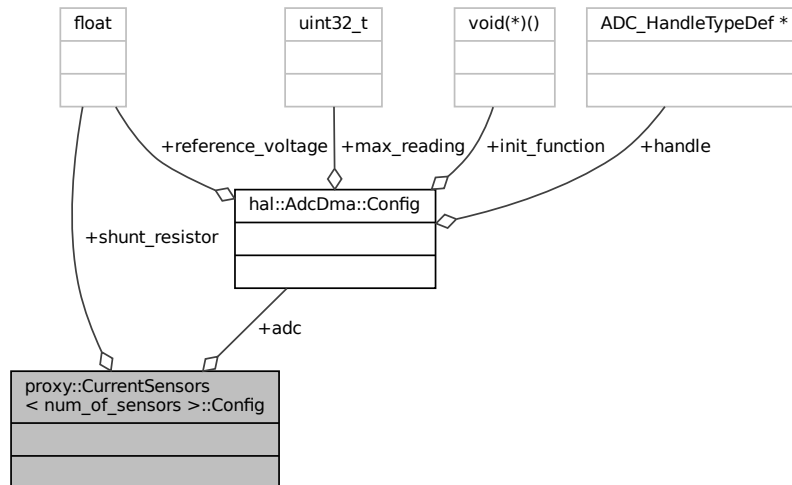
- [inc/proxy/buzzer.hpp](#)

## 7.19 proxy::CurrentSensors< num\_of\_sensors >::Config Struct Reference

Configuration structure for current sensors.

```
#include <current_sensors.hpp>
```

Collaboration diagram for proxy::CurrentSensors< num\_of\_sensors >::Config:



### Public Attributes

- [hal::AdcDma::Config](#) `adc`
- float `shunt_resistor`

### 7.19.1 Detailed Description

```
template<uint8_t num_of_sensors>
struct proxy::CurrentSensors< num_of_sensors >::Config
```

Configuration structure for current sensors.

### 7.19.2 Member Data Documentation

#### 7.19.2.1 adc

```
template<uint8_t num_of_sensors>
hal::AdcDma::Config proxy::CurrentSensors< num_of_sensors >::Config::adc
```

### 7.19.2.2 shunt\_resistor

```
template<uint8_t num_of_sensors>
float proxy::CurrentSensors< num_of_sensors >::Config::shunt_resistor
```

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

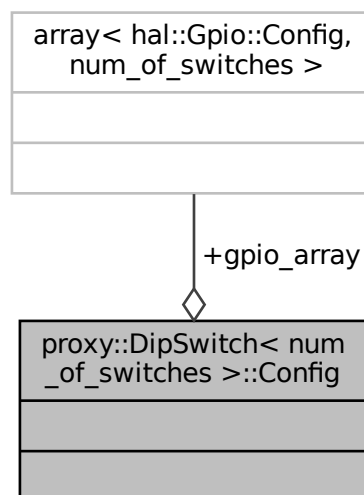
- inc/proxy/current\_sensors.hpp

## 7.20 proxy::DipSwitch< num\_of\_switches >::Config Struct Reference

Configuration struct for [DipSwitch](#).

```
#include <dip_switch.hpp>
```

Collaboration diagram for proxy::DipSwitch< num\_of\_switches >::Config:



### Public Attributes

- std::array< [hal::Gpio::Config](#), num\_of\_switches > [gpio\\_array](#)

### 7.20.1 Detailed Description

```
template<uint8_t num_of_switches>
struct proxy::DipSwitch< num_of_switches >::Config
```

Configuration struct for [DipSwitch](#).

## 7.20.2 Member Data Documentation

### 7.20.2.1 gpio\_array

```
template<uint8_t num_of_switches>
std::array<hal::Gpio::Config, num_of_switches> proxy::DipSwitch< num_of_switches >::Config<
::gpio_array
```

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

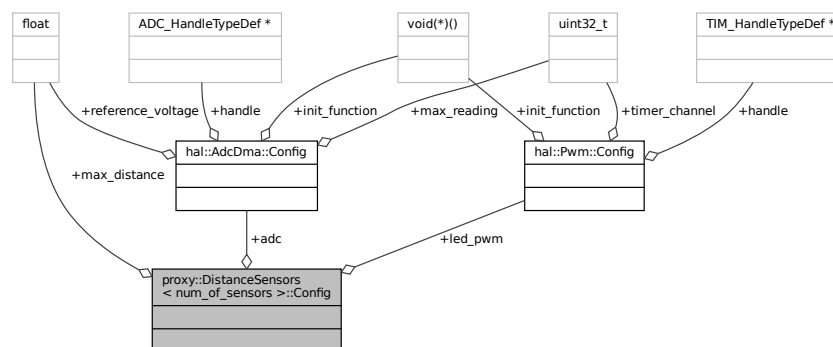
- [inc/proxy/dip\\_switch.hpp](#)

## 7.21 proxy::DistanceSensors< num\_of\_sensors >::Config Struct Reference

Configuration structure for distance sensors.

```
#include <distance_sensors.hpp>
```

Collaboration diagram for proxy::DistanceSensors< num\_of\_sensors >::Config:



### Public Attributes

- [hal::AdcDma::Config](#) `adc`
- [hal::Pwm::Config](#) `led_pwm`
- [float](#) `max_distance`

### 7.21.1 Detailed Description

```
template<uint8_t num_of_sensors>
struct proxy::DistanceSensors< num_of_sensors >::Config
```

Configuration structure for distance sensors.

## 7.21.2 Member Data Documentation

### 7.21.2.1 adc

```
template<uint8_t num_of_sensors>
hal::AdcDma::Config proxy::DistanceSensors< num_of_sensors >::Config::adc
```

### 7.21.2.2 led\_pwm

```
template<uint8_t num_of_sensors>
hal::Pwm::Config proxy::DistanceSensors< num_of_sensors >::Config::led_pwm
```

### 7.21.2.3 max\_distance

```
template<uint8_t num_of_sensors>
float proxy::DistanceSensors< num_of_sensors >::Config::max_distance
```

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

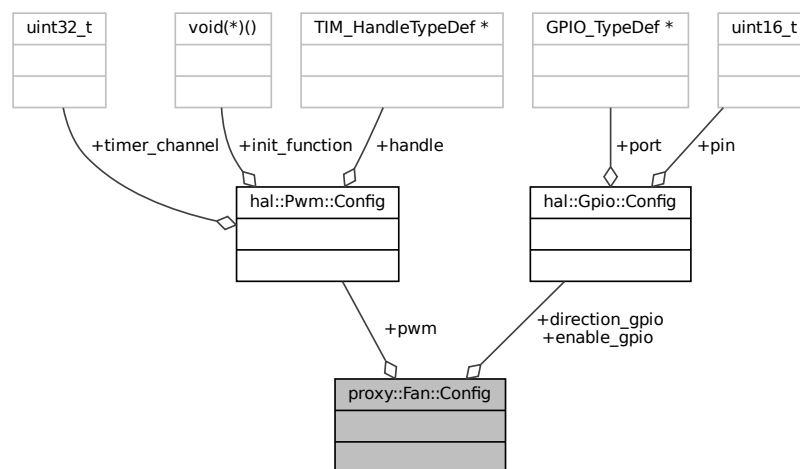
- [inc/proxy/distance\\_sensors.hpp](#)

## 7.22 proxy::Fan::Config Struct Reference

Configuration structure for the fan.

```
#include <fan.hpp>
```

Collaboration diagram for proxy::Fan::Config:





## Public Attributes

- [hal::Pwm::Config](#) pwm
- [hal::Gpio::Config](#) direction\_gpio
- [hal::Gpio::Config](#) enable\_gpio

### 7.22.1 Detailed Description

Configuration structure for the fan.

### 7.22.2 Member Data Documentation

#### 7.22.2.1 direction\_gpio

`hal::Gpio::Config` proxy::Fan::Config::direction\_gpio

#### 7.22.2.2 enable\_gpio

`hal::Gpio::Config` proxy::Fan::Config::enable\_gpio

#### 7.22.2.3 pwm

`hal::Pwm::Config` proxy::Fan::Config::pwm

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

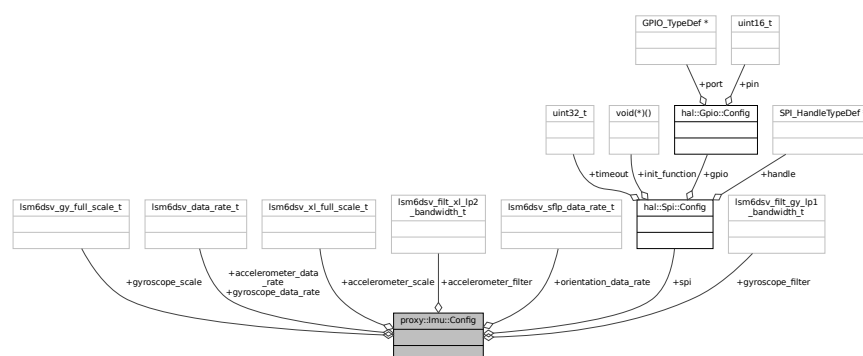
- [inc/proxy/fan.hpp](#)

## 7.23 proxy::Imu::Config Struct Reference

IMU configuration struct.

```
#include <imu.hpp>
```

Collaboration diagram for proxy::Imu::Config:



## Public Attributes

- [hal::Spi::Config spi](#)
- lsm6dsv\_data\_rate\_t [gyroscope\\_data\\_rate](#)
- lsm6dsv\_data\_rate\_t [accelerometer\\_data\\_rate](#)
- lsm6dsv\_sflp\_data\_rate\_t [orientation\\_data\\_rate](#)
- lsm6dsv\_gy\_full\_scale\_t [gyroscope\\_scale](#)
- lsm6dsv\_xl\_full\_scale\_t [accelerometer\\_scale](#)
- lsm6dsv\_filt\_gy\_lp1\_bandwidth\_t [gyroscope\\_filter](#)
- lsm6dsv\_filt\_xl\_lp2\_bandwidth\_t [accelerometer\\_filter](#)

### 7.23.1 Detailed Description

IMU configuration struct.

### 7.23.2 Member Data Documentation

#### 7.23.2.1 accelerometer\_data\_rate

```
lsm6dsv_data_rate_t proxy::Imu::Config::accelerometer_data_rate
```

#### 7.23.2.2 accelerometer\_filter

```
lsm6dsv_filt_xl_lp2_bandwidth_t proxy::Imu::Config::accelerometer_filter
```

#### 7.23.2.3 accelerometer\_scale

```
lsm6dsv_xl_full_scale_t proxy::Imu::Config::accelerometer_scale
```

#### 7.23.2.4 gyroscope\_data\_rate

```
lsm6dsv_data_rate_t proxy::Imu::Config::gyroscope_data_rate
```

### 7.23.2.5 gyroscope\_filter

```
lsm6dsv_filt_gy_lpl_bandwidth_t proxy::Imu::Config::gyroscope_filter
```

### 7.23.2.6 gyroscope\_scale

```
lsm6dsv_gy_full_scale_t proxy::Imu::Config::gyroscope_scale
```

### 7.23.2.7 orientation\_data\_rate

```
lsm6dsv_sflp_data_rate_t proxy::Imu::Config::orientation_data_rate
```

### 7.23.2.8 spi

```
hal::Spi::Config proxy::Imu::Config::spi
```

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

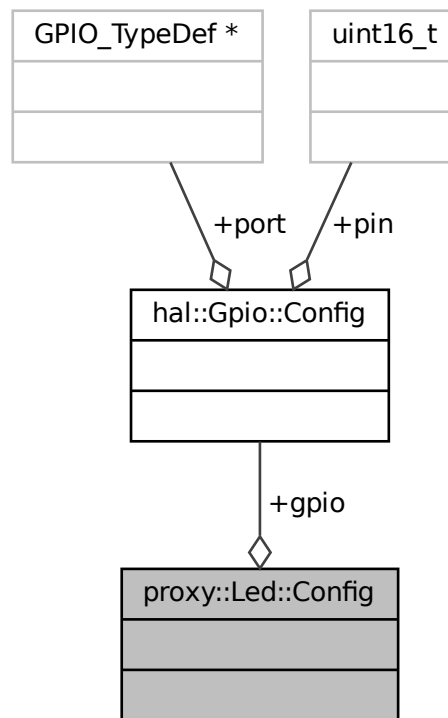
- [inc/proxy/imu.hpp](#)

## 7.24 proxy::Led::Config Struct Reference

Configuration structure for LED.

```
#include <led.hpp>
```

Collaboration diagram for proxy::Led::Config:



## Public Attributes

- [hal::Gpio::Config gpio](#)

## 7.24.1 Detailed Description

Configuration structure for LED.

## 7.24.2 Member Data Documentation

### 7.24.2.1 gpio

[hal::Gpio::Config](#) proxy::Led::Config::gpio

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

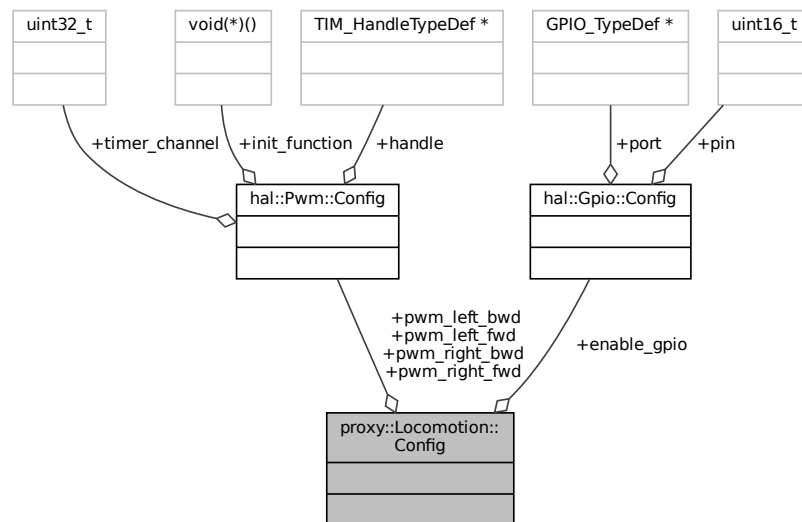
- inc/proxy/[led.hpp](#)

## 7.25 proxy::Locomotion::Config Struct Reference

Configuration structure for the locomotion.

```
#include <locomotion.hpp>
```

Collaboration diagram for proxy::Locomotion::Config:



### Public Attributes

- [hal::Pwm::Config pwm\\_left\\_fwd](#)
- [hal::Pwm::Config pwm\\_left\\_bwd](#)
- [hal::Pwm::Config pwm\\_right\\_fwd](#)
- [hal::Pwm::Config pwm\\_right\\_bwd](#)
- [hal::Gpio::Config enable\\_gpio](#)

### 7.25.1 Detailed Description

Configuration structure for the locomotion.

### 7.25.2 Member Data Documentation

#### 7.25.2.1 enable\_gpio

```
hal::Gpio::Config proxy::Locomotion::Config::enable_gpio
```

### 7.25.2.2 pwm\_left\_bwd

`hal::Pwm::Config proxy::Locomotion::Config::pwm_left_bwd`

### 7.25.2.3 pwm\_left\_fwd

`hal::Pwm::Config proxy::Locomotion::Config::pwm_left_fwd`

### 7.25.2.4 pwm\_right\_bwd

`hal::Pwm::Config proxy::Locomotion::Config::pwm_right_bwd`

### 7.25.2.5 pwm\_right\_fwd

`hal::Pwm::Config proxy::Locomotion::Config::pwm_right_fwd`

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

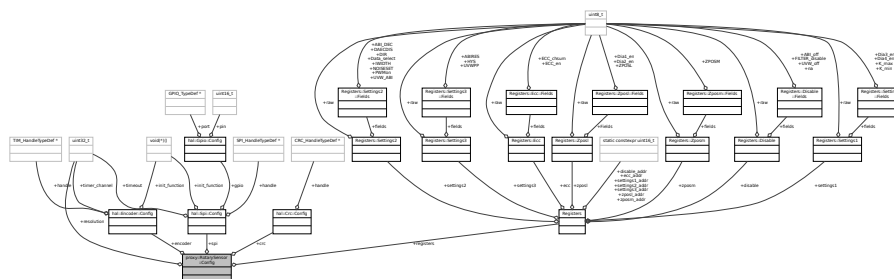
- `inc/proxy/locomotion.hpp`

## 7.26 proxy::RotarySensor::Config Struct Reference

Rotary sensor configuration struct.

```
#include <rotary_sensor.hpp>
```

Collaboration diagram for `proxy::RotarySensor::Config`:



## Public Attributes

- [hal::Spi::Config](#) spi
- [hal::Encoder::Config](#) encoder
- [hal::Crc::Config](#) crc
- [uint32\\_t](#) resolution
- [Registers](#) registers

### 7.26.1 Detailed Description

Rotary sensor configuration struct.

### 7.26.2 Member Data Documentation

#### 7.26.2.1 crc

[hal::Crc::Config](#) proxy::RotarySensor::Config::crc

#### 7.26.2.2 encoder

[hal::Encoder::Config](#) proxy::RotarySensor::Config::encoder

#### 7.26.2.3 registers

[Registers](#) proxy::RotarySensor::Config::registers

#### 7.26.2.4 resolution

[uint32\\_t](#) proxy::RotarySensor::Config::resolution

#### 7.26.2.5 spi

[hal::Spi::Config](#) proxy::RotarySensor::Config::spi

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

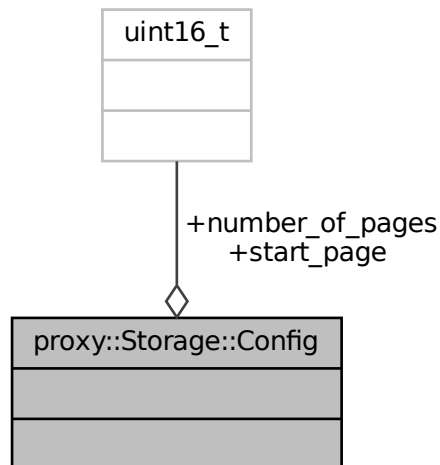
- [inc/proxy/rotary\\_sensor.hpp](#)

## 7.27 proxy::Storage::Config Struct Reference

Configuration structure for the storage.

```
#include <storage.hpp>
```

Collaboration diagram for proxy::Storage::Config:



### Public Attributes

- `uint16_t` [start\\_page](#)
- `uint16_t` [number\\_of\\_pages](#)

### 7.27.1 Detailed Description

Configuration structure for the storage.

### 7.27.2 Member Data Documentation

#### 7.27.2.1 number\_of\_pages

```
uint16_t proxy::Storage::Config::number_of_pages
```



### 7.27.2.2 start\_page

```
uint16_t proxy::Storage::Config::start_page
```

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

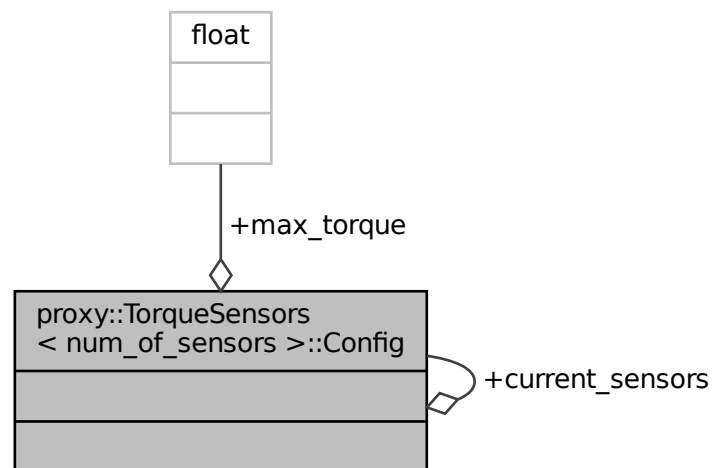
- inc/proxy/storage.hpp

## 7.28 proxy::TorqueSensors< num\_of\_sensors >::Config Struct Reference

Configuration structure for torque sensors.

```
#include <torque_sensors.hpp>
```

Collaboration diagram for proxy::TorqueSensors< num\_of\_sensors >::Config:



### Public Attributes

- [CurrentSensors< num\\_of\\_sensors >::Config current\\_sensors](#)
- float [max\\_torque](#)

### 7.28.1 Detailed Description

```
template<uint8_t num_of_sensors>
struct proxy::TorqueSensors< num_of_sensors >::Config
```

Configuration structure for torque sensors.

## 7.28.2 Member Data Documentation

### 7.28.2.1 current\_sensors

```
template<uint8_t num_of_sensors>
CurrentSensors<num_of_sensors>::Config proxy::TorqueSensors< num_of_sensors >::Config::current←
_sensors
```

### 7.28.2.2 max\_torque

```
template<uint8_t num_of_sensors>
float proxy::TorqueSensors< num_of_sensors >::Config::max_torque
```

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

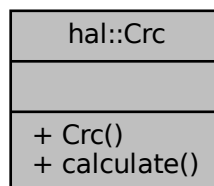
- [inc/proxy/torque\\_sensors.hpp](#)

## 7.29 hal::Crc Class Reference

Class to handle the cyclic redundancy check peripheral on STM32 microcontrollers.

```
#include <crc.hpp>
```

Collaboration diagram for hal::Crc:



## Classes

- struct [Config](#)  
*CRC configuration struct.*

## Public Member Functions

- [Crc](#) (const [Config](#) &config)  
*Construct a new [Crc](#) object.*
- uint32\_t [calculate](#) (uint32\_t data[], uint32\_t size)  
*Calculate the CRC value.*

### 7.29.1 Detailed Description

Class to handle the cyclic redundancy check peripheral on STM32 microcontrollers.

### 7.29.2 Constructor & Destructor Documentation

#### 7.29.2.1 Crc()

```
hal::Crc::Crc (  
    const Config & config ) [explicit]
```

Construct a new [Crc](#) object.

##### Parameters

<i>config</i>	Configuration for the CRC
---------------	---------------------------

### 7.29.3 Member Function Documentation

#### 7.29.3.1 calculate()

```
uint32_t hal::Crc::calculate (  
    uint32_t data[],  
    uint32_t size )
```

Calculate the CRC value.

##### Parameters

<i>data</i>	Data to calculate the CRC
<i>size</i>	Size of the buffer

**Returns**

uint32\_t CRC value

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

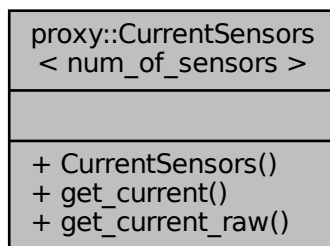
- [inc/hal/crc.hpp](#)
- [src/hal/crc.cpp](#)

## 7.30 proxy::CurrentSensors< num\_of\_sensors > Class Template Reference

Class for controlling [CurrentSensors](#).

```
#include <current_sensors.hpp>
```

Collaboration diagram for proxy::CurrentSensors< num\_of\_sensors >:

**Classes**

- struct [Config](#)  
*Configuration structure for current sensors.*

**Public Member Functions**

- [CurrentSensors](#) (const [Config](#) &config)  
*Constructor for the [CurrentSensors](#) class.*
- float [get\\_current](#) (uint8\_t sensor\_index) const  
*Get the current from the sensor.*
- uint32\_t [get\\_current\\_raw](#) (uint8\_t sensor\_index) const  
*Get the raw reading from the current sensor.*

### 7.30.1 Detailed Description

```
template<uint8_t num_of_sensors>
class proxy::CurrentSensors< num_of_sensors >
```

Class for controlling [CurrentSensors](#).

### 7.30.2 Constructor & Destructor Documentation

#### 7.30.2.1 CurrentSensors()

```
template<uint8_t num_of_sensors>
proxy::CurrentSensors< num_of_sensors >::CurrentSensors (
    const Config & config ) [explicit]
```

Constructor for the [CurrentSensors](#) class.

Parameters

<i>config</i>	Configuration for the current sensors
---------------	---------------------------------------

Here is the call graph for this function:



### 7.30.3 Member Function Documentation

#### 7.30.3.1 get\_current()

```
template<uint8_t num_of_sensors>
float proxy::CurrentSensors< num_of_sensors >::get_current (
    uint8_t sensor_index ) const
```

Get the current from the sensor.

**Parameters**

<i>sensor_index</i>	Index of the sensor
---------------------	---------------------

**Returns**

float Current reading from the sensor in amps

**7.30.3.2 get\_current\_raw()**

```
template<uint8_t num_of_sensors>
uint32_t proxy::CurrentSensors< num_of_sensors >::get_current_raw (
    uint8_t sensor_index ) const
```

Get the raw reading from the current sensor.

**Parameters**

<i>sensor_index</i>	Index of the sensor
---------------------	---------------------

**Returns**

uint16\_t Current reading from the sensor

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

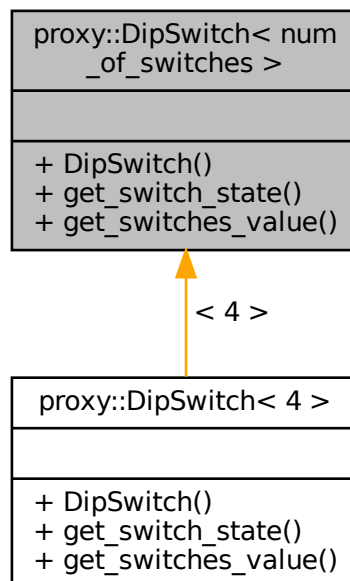
- [inc/proxy/current\\_sensors.hpp](#)
- [src/proxy/current\\_sensors.cpp](#)

**7.31 proxy::DipSwitch< num\_of\_switches > Class Template Reference**

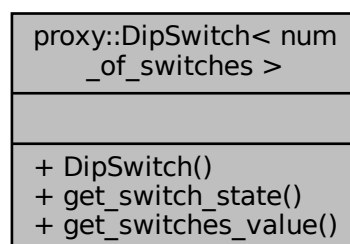
Class for controlling a dip switch.

```
#include <dip_switch.hpp>
```

Inheritance diagram for proxy::DipSwitch< num\_of\_switches >:



Collaboration diagram for proxy::DipSwitch< num\_of\_switches >:



## Classes

- struct [Config](#)  
Configuration struct for [DipSwitch](#).

## Public Member Functions

- [DipSwitch](#) (const [Config](#) &config)

*Construct a new Dip Switch object.*

- bool [get\\_switch\\_state](#) (uint8\_t switch\_index) const

*Get the state of a switch.*

- uint8\_t [get\\_switches\\_value](#) () const

*Get the value of all switches.*

### 7.31.1 Detailed Description

```
template<uint8_t num_of_switches>
class proxy::DipSwitch< num_of_switches >
```

Class for controlling a dip switch.

### 7.31.2 Constructor & Destructor Documentation

#### 7.31.2.1 DipSwitch()

```
template<uint8_t num_of_sensors>
proxy::DipSwitch< num_of_sensors >::DipSwitch (
    const Config & config ) [explicit]
```

Construct a new Dip Switch object.

##### Parameters

<i>config</i>	Configuration struct for <a href="#">DipSwitch</a>
---------------	--

### 7.31.3 Member Function Documentation

#### 7.31.3.1 get\_switch\_state()

```
template<uint8_t num_of_sensors>
bool proxy::DipSwitch< num_of_sensors >::get_switch_state (
    uint8_t switch_index ) const
```

Get the state of a switch.

##### Parameters

<i>switch_index</i>	Index of the switch
---------------------	---------------------



**Returns**

bool True if the switch is on, false otherwise

**7.31.3.2 get\_switches\_value()**

```
template<uint8_t num_of_sensors>
uint8_t proxy::DipSwitch< num_of_sensors >::get_switches_value
```

Get the value of all switches.

**Returns**

uint8\_t Value of all switches

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

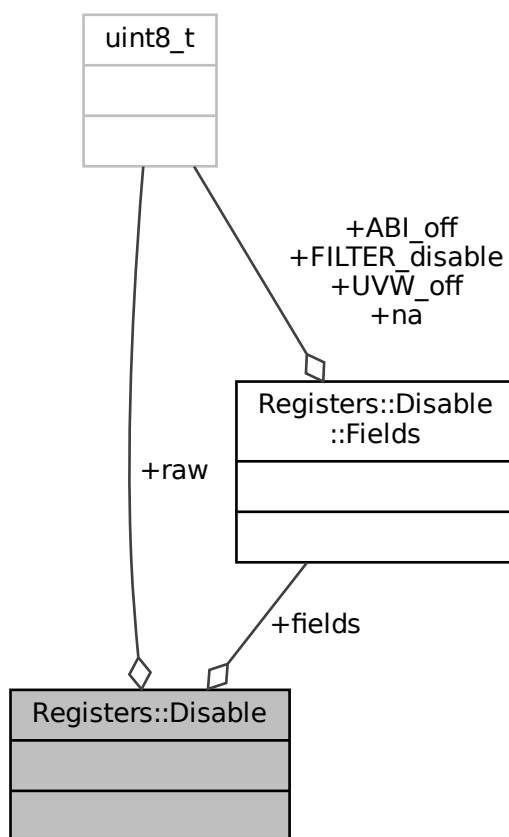
- [inc/proxy/dip\\_switch.hpp](#)
- [src/proxy/dip\\_switch.cpp](#)

## 7.32 Registers::Disable Union Reference

[Registers](#) union types definition.

```
#include <rotary_sensor_reg.hpp>
```

Collaboration diagram for `Registers::Disable`:



## Classes

- struct [Fields](#)

## Public Attributes

- [Fields](#) `fields`
- `uint8_t` `raw`

### 7.32.1 Detailed Description

[Registers](#) union types definition.

### 7.32.2 Member Data Documentation

## 7.32.2.1 fields

`Fields` `Registers::Disable::fields`

## 7.32.2.2 raw

`uint8_t` `Registers::Disable::raw`

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

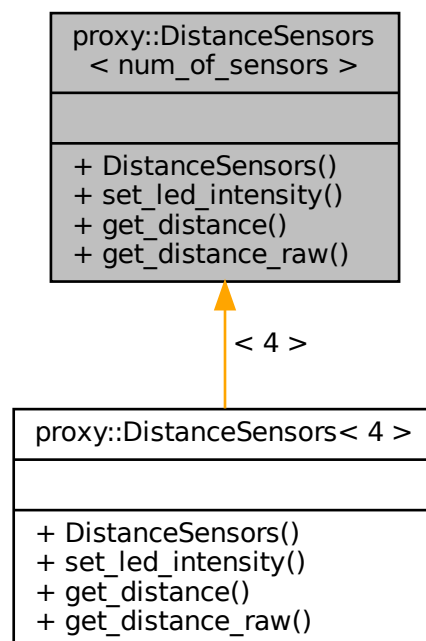
- `inc/proxy/rotary_sensor_reg.hpp`

## 7.33 proxy::DistanceSensors< num\_of\_sensors > Class Template Reference

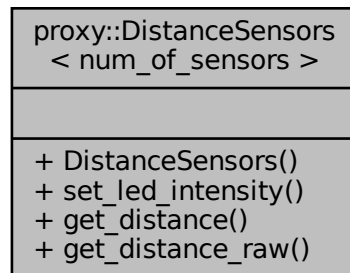
Class for controlling [DistanceSensors](#).

```
#include <distance_sensors.hpp>
```

Inheritance diagram for proxy::DistanceSensors< num\_of\_sensors >:



Collaboration diagram for proxy::DistanceSensors< num\_of\_sensors >:



## Classes

- struct [Config](#)  
*Configuration structure for distance sensors.*

## Public Member Functions

- [DistanceSensors](#) (const [Config](#) &config)  
*Constructor for the [DistanceSensors](#) class.*
- void [set\\_led\\_intensity](#) (float intensity)  
*Set the distance sensors led intensity.*
- float [get\\_distance](#) (uint8\_t sensor\_index) const  
*Get the distance from a sensor.*
- uint32\_t [get\\_distance\\_raw](#) (uint8\_t sensor\_index) const  
*Get the distance from a sensor.*

### 7.33.1 Detailed Description

```
template<uint8_t num_of_sensors>
class proxy::DistanceSensors< num_of_sensors >
```

Class for controlling [DistanceSensors](#).

### 7.33.2 Constructor & Destructor Documentation

#### 7.33.2.1 DistanceSensors()

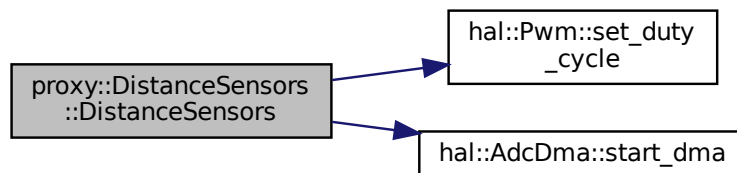
```
template<uint8_t num_of_sensors>
proxy::DistanceSensors< num_of_sensors >::DistanceSensors (
    const Config & config ) [explicit]
```

Constructor for the [DistanceSensors](#) class.

## Parameters

<i>config</i>	Configuration for the distance sensors
---------------	--

Here is the call graph for this function:



### 7.33.3 Member Function Documentation

#### 7.33.3.1 get\_distance()

```
template<uint8_t num_of_sensors>
float proxy::DistanceSensors< num_of_sensors >::get_distance (
    uint8_t sensor_index ) const
```

Get the distance from a sensor.

## Parameters

<i>sensor_index</i>	Index of the sensor
---------------------	---------------------

## Returns

float Distance reading from the sensors

#### 7.33.3.2 get\_distance\_raw()

```
template<uint8_t num_of_sensors>
uint32_t proxy::DistanceSensors< num_of_sensors >::get_distance_raw (
    uint8_t sensor_index ) const
```

Get the distance from a sensor.

**Parameters**

<i>sensor_index</i>	Index of the sensor
---------------------	---------------------

**Returns**

uint32\_t Raw reading from the distance sensor

**7.33.3.3 set\_led\_intensity()**

```
template<uint8_t num_of_sensors>
void proxy::DistanceSensors< num_of_sensors >::set_led_intensity (
    float intensity )
```

Set the distance sensors led intensity.

**Parameters**

<i>intensity</i>	Intensity percentage of the infrared LED
------------------	--

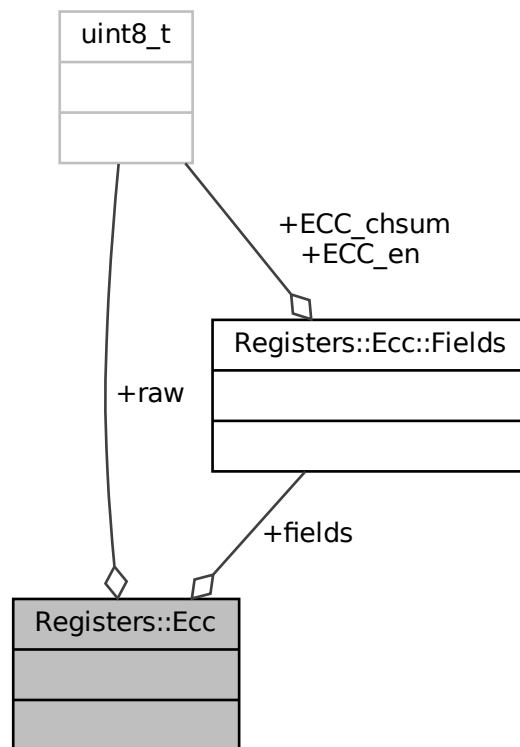
The documentation for this class was generated from the following files:

- inc/proxy/[distance\\_sensors.hpp](#)
- src/proxy/[distance\\_sensors.cpp](#)

**7.34 Registers::Ecc Union Reference**

```
#include <rotary_sensor_reg.hpp>
```

Collaboration diagram for Registers::Ecc:



## Classes

- struct [Fields](#)

## Public Attributes

- [Fields](#) `fields`
- `uint8_t` `raw`

### 7.34.1 Member Data Documentation

#### 7.34.1.1 fields

[Fields](#) `Registers::Ecc::fields`

### 7.34.1.2 raw

```
uint8_t Registers::Ecc::raw
```

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

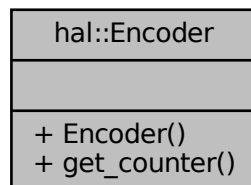
- [inc/proxy/rotary\\_sensor\\_reg.hpp](#)

## 7.35 hal::Encoder Class Reference

Class to handle encoder peripheral on STM32 microcontrollers.

```
#include <encoder.hpp>
```

Collaboration diagram for hal::Encoder:



### Classes

- struct [Config](#)  
*Encoder configuration struct.*

### Public Member Functions

- [Encoder](#) (const [Config](#) &config)  
*Construct a new Encoder object.*
- int32\_t [get\\_counter](#) () const  
*Get the counter value.*

### 7.35.1 Detailed Description

Class to handle encoder peripheral on STM32 microcontrollers.

### 7.35.2 Constructor & Destructor Documentation

#### 7.35.2.1 Encoder()

```
hal::Encoder::Encoder (
    const Config & config ) [explicit]
```

Construct a new [Encoder](#) object.



## Parameters

<i>config</i>	Configuration for the encoder
---------------	-------------------------------

### 7.35.3 Member Function Documentation

#### 7.35.3.1 get\_counter()

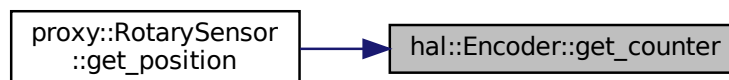
```
int32_t hal::Encoder::get_counter ( ) const
```

Get the counter value.

## Returns

int32\_t Current value of the counter

Here is the caller graph for this function:



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

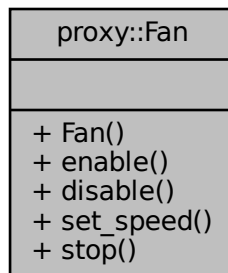
- [inc/hal/encoder.hpp](#)
- [src/hal/encoder.cpp](#)

## 7.36 proxy::Fan Class Reference

Class for controlling the fan driver.

```
#include <fan.hpp>
```

Collaboration diagram for proxy::Fan:



## Classes

- struct [Config](#)  
*Configuration structure for the fan.*

## Public Types

- enum [RotationDirection](#) { [FORWARD](#) , [BACKWARD](#) }  
*Enum for rotation direction.*

## Public Member Functions

- [Fan](#) (const [Config](#) &config)  
*Construct a new fan object.*
- void [enable](#) ()  
*Enable the fan.*
- void [disable](#) ()  
*Disable the fan.*
- void [set\\_speed](#) (float speed)  
*Set the speed of the fans.*
- void [stop](#) ()  
*Stop the fan.*

### 7.36.1 Detailed Description

Class for controlling the fan driver.

### 7.36.2 Member Enumeration Documentation

#### 7.36.2.1 RotationDirection

```
enum proxy::Fan::RotationDirection
```

Enum for rotation direction.

## Enumerator

FORWARD	
BACKWARD	

### 7.36.3 Constructor & Destructor Documentation

#### 7.36.3.1 Fan()

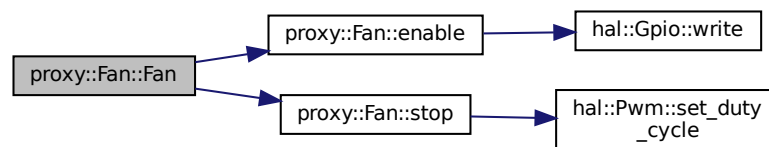
```
proxy::Fan::Fan (  
    const Config & config ) [explicit]
```

Construct a new fan object.

## Parameters

<i>config</i>	Configuration for the fan driver
---------------	----------------------------------

Here is the call graph for this function:



### 7.36.4 Member Function Documentation

#### 7.36.4.1 disable()

```
void proxy::Fan::disable ( )
```

Disable the fan.

Here is the call graph for this function:



#### 7.36.4.2 enable()

```
void proxy::Fan::enable ( )
```

Enable the fan.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 7.36.4.3 set\_speed()

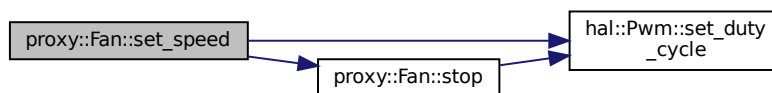
```
void proxy::Fan::set_speed (
    float speed )
```

Set the speed of the fans.

## Parameters

<i>speed</i>	Speed percentage of the fan
--------------	-----------------------------

Here is the call graph for this function:

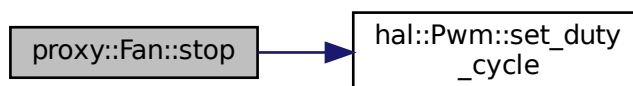


#### 7.36.4.4 stop()

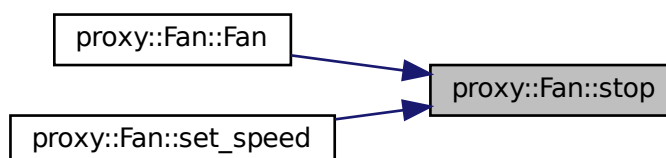
```
void proxy::Fan::stop ( )
```

Stop the fan.

Here is the call graph for this function:



Here is the caller graph for this function:



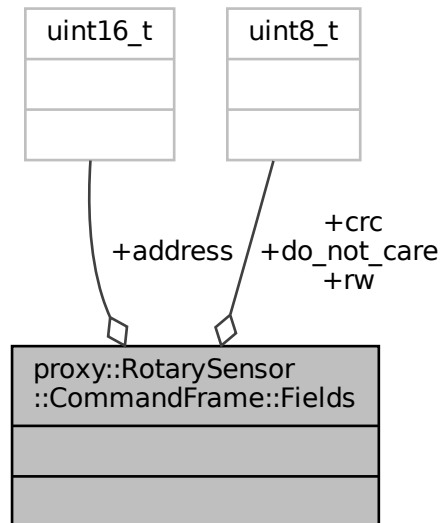
The documentation for this class was generated from the following files:

- [inc/proxy/fan.hpp](#)
- [src/proxy/fan.cpp](#)

## 7.37 proxy::RotarySensor::CommandFrame::Fields Struct Reference

```
#include <rotary_sensor.hpp>
```

Collaboration diagram for proxy::RotarySensor::CommandFrame::Fields:



### Public Attributes

- uint8\_t [do\\_not\\_care](#): 1
- uint8\_t [rw](#): 1
- uint16\_t [address](#): 14
- uint8\_t [crc](#): 8

### 7.37.1 Member Data Documentation

#### 7.37.1.1 address

```
uint16_t proxy::RotarySensor::CommandFrame::Fields::address
```

#### 7.37.1.2 crc

```
uint8_t proxy::RotarySensor::CommandFrame::Fields::crc
```

## 7.37.1.3 do\_not\_care

```
uint8_t proxy::RotarySensor::CommandFrame::Fields::do_not_care
```

## 7.37.1.4 rw

```
uint8_t proxy::RotarySensor::CommandFrame::Fields::rw
```

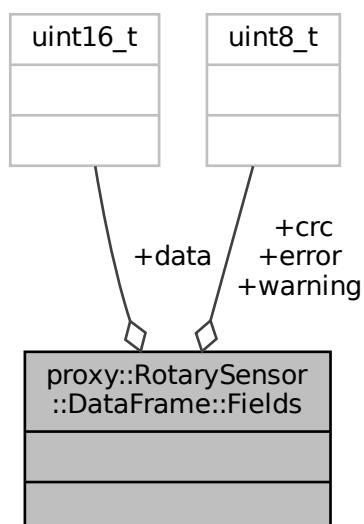
The documentation for this struct was generated from the following file:

- [inc/proxy/rotary\\_sensor.hpp](#)

## 7.38 proxy::RotarySensor::DataFrame::Fields Struct Reference

```
#include <rotary_sensor.hpp>
```

Collaboration diagram for proxy::RotarySensor::DataFrame::Fields:



## Public Attributes

- [uint8\\_t warning](#): 1
- [uint8\\_t error](#): 1
- [uint16\\_t data](#): 14
- [uint8\\_t crc](#): 8

### 7.38.1 Member Data Documentation

#### 7.38.1.1 crc

```
uint8_t proxy::RotarySensor::DataFrame::Fields::crc
```

#### 7.38.1.2 data

```
uint16_t proxy::RotarySensor::DataFrame::Fields::data
```

#### 7.38.1.3 error

```
uint8_t proxy::RotarySensor::DataFrame::Fields::error
```

#### 7.38.1.4 warning

```
uint8_t proxy::RotarySensor::DataFrame::Fields::warning
```

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

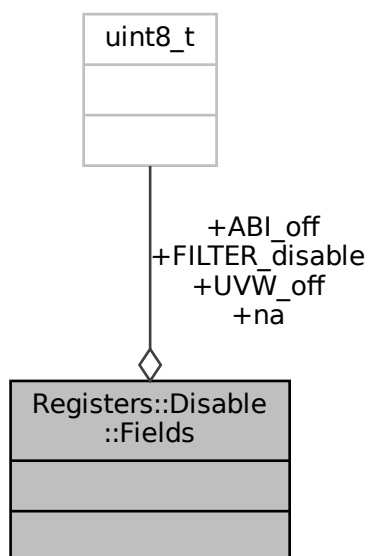
- [inc/proxy/rotary\\_sensor.hpp](#)

## 7.39 Registers::Disable::Fields Struct Reference

```
#include <rotary_sensor_reg.hpp>
```



Collaboration diagram for Registers::Disable::Fields:



## Public Attributes

- uint8\_t [UVW\\_off](#): 1
- uint8\_t [ABI\\_off](#): 1
- uint8\_t [na](#): 4
- uint8\_t [FILTER\\_disable](#): 1

## 7.39.1 Member Data Documentation

### 7.39.1.1 ABI\_off

```
uint8_t Registers::Disable::Fields::ABI_off
```

### 7.39.1.2 FILTER\_disable

```
uint8_t Registers::Disable::Fields::FILTER_disable
```

### 7.39.1.3 na

```
uint8_t Registers::Disable::Fields::na
```

### 7.39.1.4 UVW\_off

```
uint8_t Registers::Disable::Fields::UVW_off
```

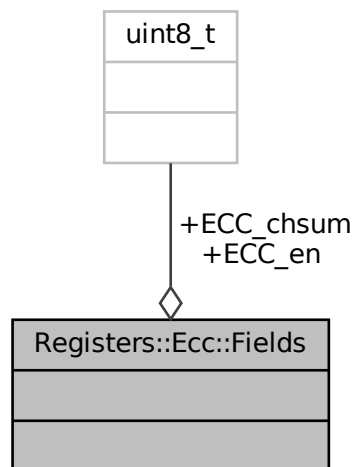
The documentation for this struct was generated from the following file:

- [inc/proxy/rotary\\_sensor\\_reg.hpp](#)

## 7.40 Registers::Ecc::Fields Struct Reference

```
#include <rotary_sensor_reg.hpp>
```

Collaboration diagram for Registers::Ecc::Fields:



### Public Attributes

- `uint8_t` [ECC\\_chsum](#): 7
- `uint8_t` [ECC\\_en](#): 1

### 7.40.1 Member Data Documentation

### 7.40.1.1 ECC\_chsum

```
uint8_t Registers::Ecc::Fields::ECC_chsum
```

### 7.40.1.2 ECC\_en

```
uint8_t Registers::Ecc::Fields::ECC_en
```

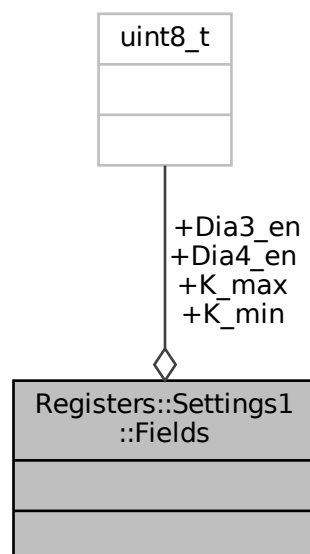
The documentation for this struct was generated from the following file:

- [inc/proxy/rotary\\_sensor\\_reg.hpp](#)

## 7.41 Registers::Settings1::Fields Struct Reference

```
#include <rotary_sensor_reg.hpp>
```

Collaboration diagram for Registers::Settings1::Fields:



### Public Attributes

- `uint8_t K_max`: 3
- `uint8_t K_min`: 3
- `uint8_t Dia3_en`: 1
- `uint8_t Dia4_en`: 1

### 7.41.1 Member Data Documentation

#### 7.41.1.1 Dia3\_en

```
uint8_t Registers::Settings1::Fields::Dia3_en
```

#### 7.41.1.2 Dia4\_en

```
uint8_t Registers::Settings1::Fields::Dia4_en
```

#### 7.41.1.3 K\_max

```
uint8_t Registers::Settings1::Fields::K_max
```

#### 7.41.1.4 K\_min

```
uint8_t Registers::Settings1::Fields::K_min
```

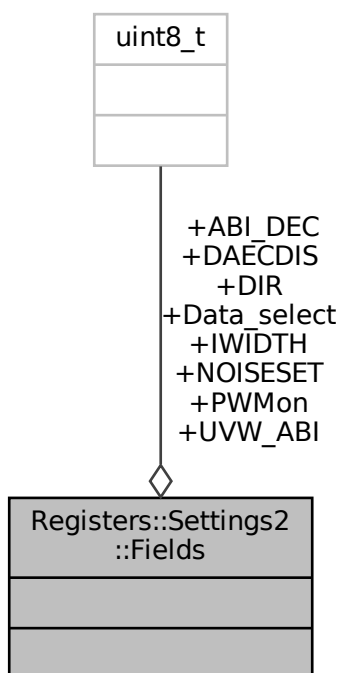
The documentation for this struct was generated from the following file:

- [inc/proxy/rotary\\_sensor\\_reg.hpp](#)

## 7.42 Registers::Settings2::Fields Struct Reference

```
#include <rotary_sensor_reg.hpp>
```

Collaboration diagram for Registers::Settings2::Fields:



## Public Attributes

- uint8\_t [IWIDTH](#): 1
- uint8\_t [NOISESET](#): 1
- uint8\_t [DIR](#): 1
- uint8\_t [UVW\\_ABI](#): 1
- uint8\_t [DAECDIS](#): 1
- uint8\_t [ABI\\_DEC](#): 1
- uint8\_t [Data\\_select](#): 1
- uint8\_t [PWMon](#): 1

## 7.42.1 Member Data Documentation

### 7.42.1.1 ABI\_DEC

uint8\_t Registers::Settings2::Fields::ABI\_DEC

#### 7.42.1.2 DAECDIS

```
uint8_t Registers::Settings2::Fields::DAECDIS
```

#### 7.42.1.3 Data\_select

```
uint8_t Registers::Settings2::Fields::Data_select
```

#### 7.42.1.4 DIR

```
uint8_t Registers::Settings2::Fields::DIR
```

#### 7.42.1.5 IWIDTH

```
uint8_t Registers::Settings2::Fields::IWIDTH
```

#### 7.42.1.6 NOISESET

```
uint8_t Registers::Settings2::Fields::NOISESET
```

#### 7.42.1.7 PWMon

```
uint8_t Registers::Settings2::Fields::PWMon
```

#### 7.42.1.8 UVW\_ABI

```
uint8_t Registers::Settings2::Fields::UVW_ABI
```

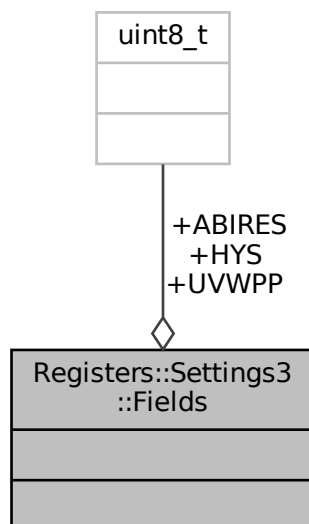
The documentation for this struct was generated from the following file:

- [inc/proxy/rotary\\_sensor\\_reg.hpp](#)

## 7.43 Registers::Settings3::Fields Struct Reference

```
#include <rotary_sensor_reg.hpp>
```

Collaboration diagram for Registers::Settings3::Fields:



### Public Attributes

- `uint8_t UVWPP`: 3
- `uint8_t HYS`: 2
- `uint8_t ABIRES`: 3

### 7.43.1 Member Data Documentation

#### 7.43.1.1 ABIRES

```
uint8_t Registers::Settings3::Fields::ABIRES
```

#### 7.43.1.2 HYS

```
uint8_t Registers::Settings3::Fields::HYS
```

### 7.43.1.3 UVWPP

```
uint8_t Registers::Settings3::Fields::UVWPP
```

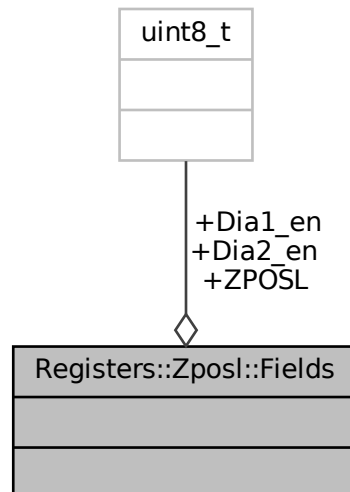
The documentation for this struct was generated from the following file:

- [inc/proxy/rotary\\_sensor\\_reg.hpp](#)

## 7.44 Registers::Zposl::Fields Struct Reference

```
#include <rotary_sensor_reg.hpp>
```

Collaboration diagram for Registers::Zposl::Fields:



### Public Attributes

- [uint8\\_t ZPOS\\_L](#): 6
- [uint8\\_t Dia1\\_en](#): 1
- [uint8\\_t Dia2\\_en](#): 1

### 7.44.1 Member Data Documentation



#### 7.44.1.1 Dia1\_en

```
uint8_t Registers::Zposl::Fields::Dia1_en
```

#### 7.44.1.2 Dia2\_en

```
uint8_t Registers::Zposl::Fields::Dia2_en
```

#### 7.44.1.3 ZPSL

```
uint8_t Registers::Zposl::Fields::ZPSL
```

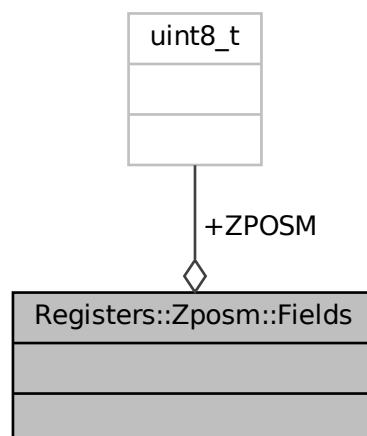
The documentation for this struct was generated from the following file:

- [inc/proxy/rotary\\_sensor\\_reg.hpp](#)

## 7.45 Registers::Zposm::Fields Struct Reference

```
#include <rotary_sensor_reg.hpp>
```

Collaboration diagram for Registers::Zposm::Fields:



### Public Attributes

- `uint8_t` [ZPSM](#): 8

## 7.45.1 Member Data Documentation

### 7.45.1.1 ZPOSM

```
uint8_t Registers::Zposm::Fields::ZPOSM
```

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

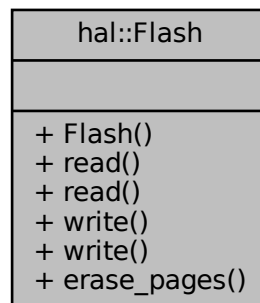
- [inc/proxy/rotary\\_sensor\\_reg.hpp](#)

## 7.46 hal::Flash Class Reference

Class to handle flash memory on STM32 microcontrollers.

```
#include <flash.hpp>
```

Collaboration diagram for hal::Flash:



### Public Member Functions

- [Flash](#) ()=delete  
*Deleted constructor for static class.*

### Static Public Member Functions

- static void [read](#) (uint32\_t address, uint64\_t data[], uint32\_t size=1)  
*Read data from flash memory.*
- static void [read](#) (uint16\_t page, uint16\_t page\_address, uint64\_t data[], uint32\_t size=1)  
*Read data from flash memory.*
- static void [write](#) (uint32\_t address, const uint64\_t data[], uint32\_t size=1)  
*Write data to flash memory.*
- static void [write](#) (uint16\_t page, uint16\_t page\_address, const uint64\_t data[], uint32\_t size=1)  
*Write data to flash memory.*
- static void [erase\\_pages](#) (uint16\_t page, uint16\_t number\_of\_pages=1)  
*Erase flash memory pages.*

### 7.46.1 Detailed Description

Class to handle flash memory on STM32 microcontrollers.

### 7.46.2 Constructor & Destructor Documentation

#### 7.46.2.1 Flash()

```
hal::Flash::Flash ( ) [delete]
```

Deleted constructor for static class.

### 7.46.3 Member Function Documentation

#### 7.46.3.1 erase\_pages()

```
void hal::Flash::erase_pages (
    uint16_t page,
    uint16_t number_of_pages = 1 ) [static]
```

Erase flash memory pages.

##### Parameters

<i>page</i>	first page to erase (counting from the last to the first)
<i>number_of_pages</i>	number of pages to erase

Here is the caller graph for this function:



**7.46.3.2 read()** [1/2]

```
void hal::Flash::read (
    uint16_t page,
    uint16_t page_address,
    uint64_t data[],
    uint32_t size = 1 ) [static]
```

Read data from flash memory.

**Parameters**

<i>page</i>	page to read from (counting from the last to the first)
<i>page_address</i>	address inside the page to read from (indexed by double words)
<i>data</i>	pointer to store the data read
<i>size</i>	size in double words of the data to read

Here is the call graph for this function:

**7.46.3.3 read()** [2/2]

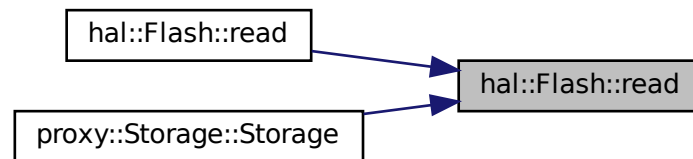
```
void hal::Flash::read (
    uint32_t address,
    uint64_t data[],
    uint32_t size = 1 ) [static]
```

Read data from flash memory.

**Parameters**

<i>address</i>	address to read from (indexed by double words)
<i>data</i>	pointer to store the data read
<i>size</i>	size in double words of the data to read

Here is the caller graph for this function:



#### 7.46.3.4 write() [1/2]

```

void hal::Flash::write (
    uint16_t page,
    uint16_t page_address,
    const uint64_t data[],
    uint32_t size = 1 ) [static]
  
```

Write data to flash memory.

##### Parameters

<i>page</i>	page to write to (counting from the last to the first)
<i>page_address</i>	address inside the page to write to (indexed by double words)
<i>data</i>	pointer to the data to write
<i>size</i>	size in double words of the data to write

Here is the call graph for this function:



#### 7.46.3.5 write() [2/2]

```

void hal::Flash::write (
    uint32_t address,
  
```

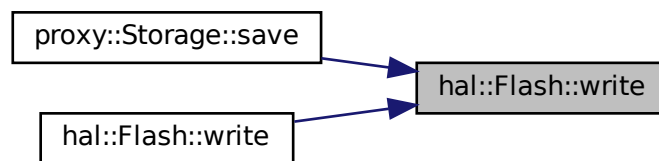
```
const uint64_t data[],
uint32_t size = 1 ) [static]
```

Write data to flash memory.

#### Parameters

<i>address</i>	address to write to (indexed by double words)
<i>data</i>	pointer to the data to write
<i>size</i>	size in double words of the data to write

Here is the caller graph for this function:



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

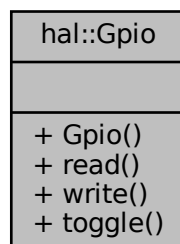
- [inc/hal/flash.hpp](#)
- [src/hal/flash.cpp](#)

## 7.47 hal::Gpio Class Reference

Class for controlling GPIO pins on STM32 microcontrollers.

```
#include <gpio.hpp>
```

Collaboration diagram for `hal::Gpio`:



## Classes

- struct [Config](#)  
*Configuration structure for GPIO pin.*

## Public Member Functions

- [Gpio](#) (const [Config](#) &config)  
*Constructor for the [Gpio](#) class.*
- bool [read](#) () const  
*Read the current state of the GPIO pin.*
- void [write](#) (bool state)  
*Write a new state to the GPIO pin.*
- void [toggle](#) ()  
*Toggle the state of the GPIO pin.*

### 7.47.1 Detailed Description

Class for controlling GPIO pins on STM32 microcontrollers.

### 7.47.2 Constructor & Destructor Documentation

#### 7.47.2.1 Gpio()

```
hal::Gpio::Gpio (  
    const Config & config ) [explicit]
```

Constructor for the [Gpio](#) class.

#### Parameters

<i>config</i>	Configuration for the GPIO pin
---------------	--------------------------------

### 7.47.3 Member Function Documentation

#### 7.47.3.1 read()

```
bool hal::Gpio::read ( ) const
```

Read the current state of the GPIO pin.

**Returns**

bool The current state of the GPIO pin (true for high, false for low)

**7.47.3.2 toggle()**

```
void hal::Gpio::toggle ( )
```

Toggle the state of the GPIO pin.

Here is the caller graph for this function:

**7.47.3.3 write()**

```
void hal::Gpio::write (
    bool state )
```

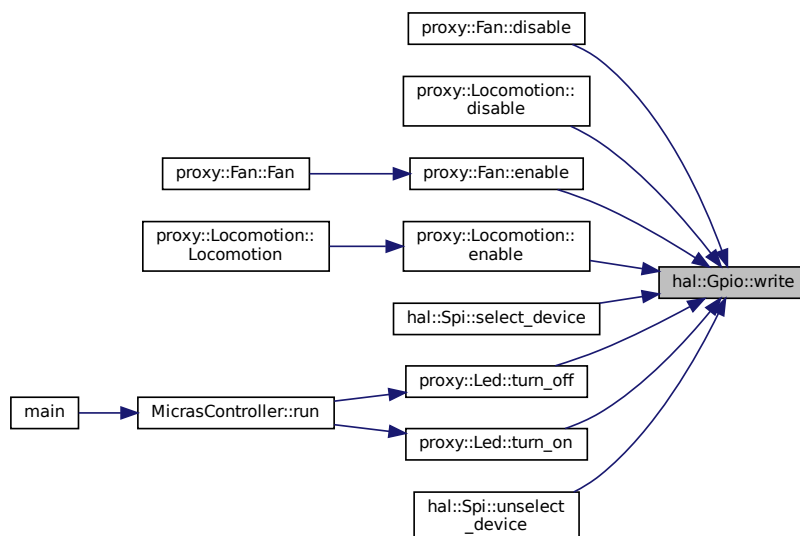
Write a new state to the GPIO pin.

**Parameters**

<i>pin_state</i>	The state to be written (true for high, false for low)
------------------	--



Here is the caller graph for this function:



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

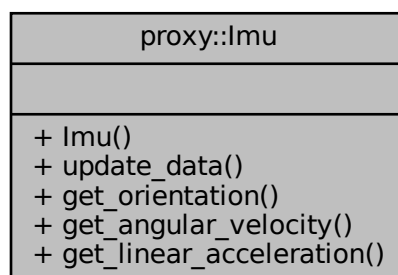
- [inc/hal/gpio.hpp](#)
- [src/hal/gpio.cpp](#)

## 7.48 proxy::Imu Class Reference

Class to handle IMU peripheral on STM32 microcontrollers.

```
#include <imu.hpp>
```

Collaboration diagram for proxy::Imu:



## Classes

- struct [Config](#)  
*IMU configuration struct.*

## Public Types

- enum [Axis](#) { [W](#) , [X](#) , [Y](#) , [Z](#) }

## Public Member Functions

- [Imu](#) (const [Config](#) &config)  
*Construct a new [Imu](#) object.*
- void [update\\_data](#) ()  
*Update the IMU data.*
- float [get\\_orientation](#) ([Axis](#) axis) const  
*Get the IMU orientation over an axis.*
- float [get\\_angular\\_velocity](#) ([Axis](#) axis) const  
*Get the IMU angular velocity over an axis.*
- float [get\\_linear\\_acceleration](#) ([Axis](#) axis) const  
*Get the IMU linear acceleration over an axis.*

### 7.48.1 Detailed Description

Class to handle IMU peripheral on STM32 microcontrollers.

### 7.48.2 Member Enumeration Documentation

#### 7.48.2.1 Axis

```
enum proxy::Imu::Axis
```

##### Enumerator

W	
X	
Y	
Z	

### 7.48.3 Constructor & Destructor Documentation

### 7.48.3.1 Imu()

```
proxy::Imu::Imu (
    const Config & config ) [explicit]
```

Construct a new `Imu` object.

#### Parameters

<code>config</code>	Configuration for the IMU
---------------------	---------------------------

Here is the call graph for this function:



## 7.48.4 Member Function Documentation

### 7.48.4.1 get\_angular\_velocity()

```
float proxy::Imu::get_angular_velocity (
    Axis axis ) const
```

Get the IMU angular velocity over an axis.

#### Parameters

<code>axis</code>	Axis to get the angular velocity from
-------------------	---------------------------------------

#### Returns

float Angular velocity over the desired axis in rad/s

### 7.48.4.2 get\_linear\_acceleration()

```
float proxy::Imu::get_linear_acceleration (
    Axis axis ) const
```

Get the IMU linear acceleration over an axis.

**Parameters**

<i>axis</i>	Axis to get the linear acceleration from
-------------	--

**Returns**

float Linear acceleration over the desired axis in  $\text{m/s}^2$

**7.48.4.3 get\_orientation()**

```
float proxy::Imu::get_orientation (
    Axis axis ) const
```

Get the IMU orientation over an axis.

**Todo** implement function using sensor fusion

**Parameters**

<i>axis</i>	Axis to get the orientation from
-------------	----------------------------------

**Returns**

float Orientation over the desired axis using quaternions

**7.48.4.4 update\_data()**

```
void proxy::Imu::update_data ( )
```

Update the IMU data.

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

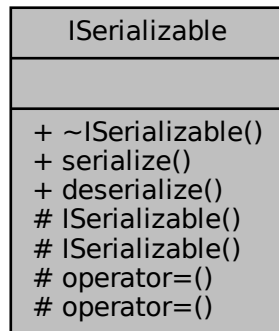
- [inc/proxy/imu.hpp](#)
- [src/proxy/imu.cpp](#)

## 7.49 ISerializable Class Reference

Interface class for serializable classes.

```
#include <serializable_interface.hpp>
```

Collaboration diagram for ISerializable:



### Public Member Functions

- virtual [~ISerializable](#) ()=default  
*Virtual destructor for the [ISerializable](#) class.*
- virtual std::vector< uint8\_t > [serialize](#) () const =0  
*Serialize the class instance.*
- virtual void [deserialize](#) (uint8\_t \*serial\_data, uint16\_t size)=0  
*Deserialize the class instance.*

### Protected Member Functions

- [ISerializable](#) (const [ISerializable](#) &)=default  
*Special member functions declared as default.*
- [ISerializable](#) ([ISerializable](#) &&)=default
- [ISerializable](#) & [operator=](#) (const [ISerializable](#) &)=default
- [ISerializable](#) & [operator=](#) ([ISerializable](#) &&)=default

#### 7.49.1 Detailed Description

Interface class for serializable classes.

#### 7.49.2 Constructor & Destructor Documentation

### 7.49.2.1 ~ISerializable()

```
virtual ISerializable::~~ISerializable ( ) [virtual], [default]
```

Virtual destructor for the [ISerializable](#) class.

### 7.49.2.2 ISerializable() [1/2]

```
ISerializable::ISerializable (
    const ISerializable & ) [protected], [default]
```

Special member functions declared as default.

### 7.49.2.3 ISerializable() [2/2]

```
ISerializable::ISerializable (
    ISerializable && ) [protected], [default]
```

## 7.49.3 Member Function Documentation

### 7.49.3.1 deserialize()

```
virtual void ISerializable::deserialize (
    uint8_t * serial_data,
    uint16_t size ) [pure virtual]
```

Deserialize the class instance.

#### Parameters

<i>serial_data</i>	Serialized data
<i>size</i>	Size of the serialized data

Here is the caller graph for this function:



### 7.49.3.2 operator=() [1/2]

```
ISerializable& ISerializable::operator= (  
    const ISerializable & ) [protected], [default]
```

### 7.49.3.3 operator=() [2/2]

```
ISerializable& ISerializable::operator= (  
    ISerializable && ) [protected], [default]
```

### 7.49.3.4 serialize()

```
virtual std::vector<uint8_t> ISerializable::serialize ( ) const [pure virtual]
```

Serialize the class instance.

#### Returns

`std::vector<uint8_t>` Serialized data

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

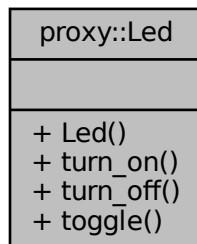
- `inc/proxy/serializable_interface.hpp`

## 7.50 proxy::Led Class Reference

Class for controlling an LED.

```
#include <led.hpp>
```

Collaboration diagram for proxy::Led:



### Classes

- struct [Config](#)  
*Configuration structure for LED.*

### Public Member Functions

- [Led](#) (const [Config](#) &config)  
*Constructor for the [Led](#) class.*
- void [turn\\_on](#) ()  
*Turn the LED on.*
- void [turn\\_off](#) ()  
*Turn the LED off.*
- void [toggle](#) ()  
*Toggle the LED.*

#### 7.50.1 Detailed Description

Class for controlling an LED.

#### 7.50.2 Constructor & Destructor Documentation

##### 7.50.2.1 Led()

```
proxy::Led::Led (  
    const Config & config ) [explicit]
```

Constructor for the [Led](#) class.



## Parameters

<i>config</i>	Configuration for the LED
---------------	---------------------------

### 7.50.3 Member Function Documentation

#### 7.50.3.1 toggle()

```
void proxy::Led::toggle ( )
```

Toggle the LED.

Here is the call graph for this function:



#### 7.50.3.2 turn\_off()

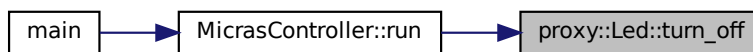
```
void proxy::Led::turn_off ( )
```

Turn the LED off.

Here is the call graph for this function:



Here is the caller graph for this function:



### 7.50.3.3 turn\_on()

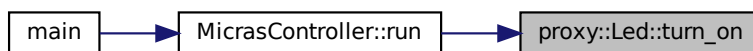
```
void proxy::Led::turn_on ( )
```

Turn the LED on.

Here is the call graph for this function:



Here is the caller graph for this function:



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

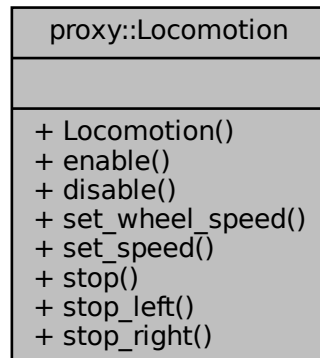
- [inc/proxy/led.hpp](#)
- [src/proxy/led.cpp](#)

## 7.51 proxy::Locomotion Class Reference

Class for controlling the locomotion driver.

```
#include <locomotion.hpp>
```

Collaboration diagram for proxy::Locomotion:



### Classes

- struct [Config](#)  
*Configuration structure for the locomotion.*

### Public Member Functions

- [Locomotion](#) (const [Config](#) &config)  
*Construct a new locomotion object.*
- void [enable](#) ()  
*Enable the locomotion driver.*
- void [disable](#) ()  
*Disable the locomotion driver.*
- void [set\\_wheel\\_speed](#) (float left\_speed, float right\_speed)  
*Set the speed of the wheels.*
- void [set\\_speed](#) (float linear, float angular)  
*Set the linear and angular speeds of the robot.*
- void [stop](#) ()  
*Stop the motors.*
- void [stop\\_left](#) ()  
*Stop the left motor.*
- void [stop\\_right](#) ()  
*Stop the right motor.*

### 7.51.1 Detailed Description

Class for controlling the locomotion driver.

### 7.51.2 Constructor & Destructor Documentation

#### 7.51.2.1 Locomotion()

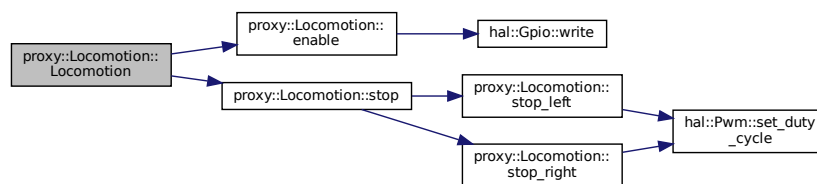
```
proxy::Locomotion::Locomotion (
    const Config & config ) [explicit]
```

Construct a new locomotion object.

##### Parameters

<i>config</i>	Configuration for the locomotion driver
---------------	---

Here is the call graph for this function:



### 7.51.3 Member Function Documentation

#### 7.51.3.1 disable()

```
void proxy::Locomotion::disable ( )
```

Disable the locomotion driver.

Here is the call graph for this function:



### 7.51.3.2 enable()

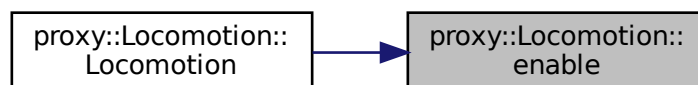
```
void proxy::Locomotion::enable ( )
```

Enable the locomotion driver.

Here is the call graph for this function:



Here is the caller graph for this function:



### 7.51.3.3 set\_speed()

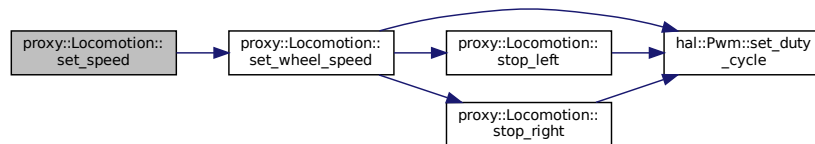
```
void proxy::Locomotion::set_speed (
    float linear,
    float angular )
```

Set the linear and angular speeds of the robot.

## Parameters

<i>linear</i>	Linear speed of the robot
<i>angular</i>	Angular speed of the robot

Here is the call graph for this function:



### 7.51.3.4 set\_wheel\_speed()

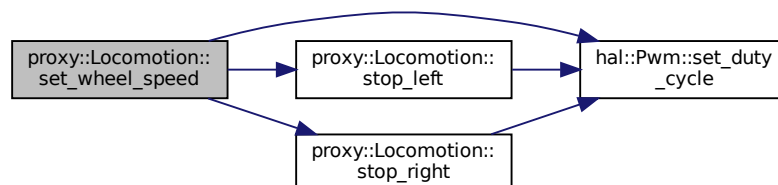
```
void proxy::Locomotion::set_wheel_speed (
    float left_speed,
    float right_speed )
```

Set the speed of the wheels.

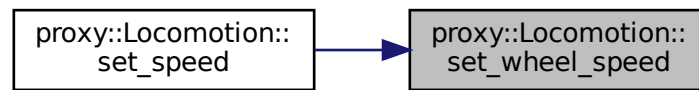
## Parameters

<i>left_speed</i>	Speed of the left wheels
<i>right_speed</i>	Speed of the right wheels

Here is the call graph for this function:



Here is the caller graph for this function:

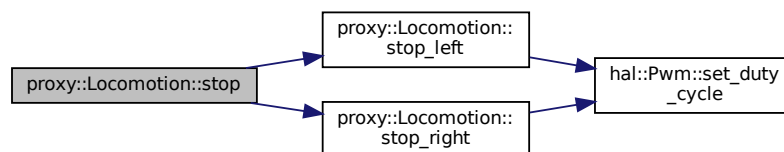


### 7.51.3.5 stop()

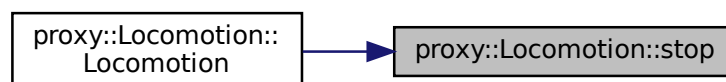
```
void proxy::Locomotion::stop ( )
```

Stop the motors.

Here is the call graph for this function:



Here is the caller graph for this function:

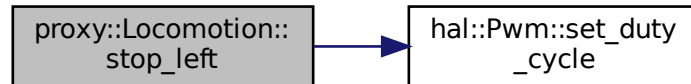


### 7.51.3.6 stop\_left()

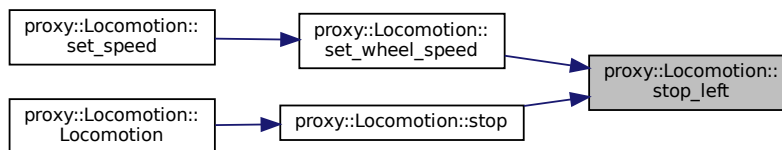
```
void proxy::Locomotion::stop_left ( )
```

Stop the left motor.

Here is the call graph for this function:



Here is the caller graph for this function:

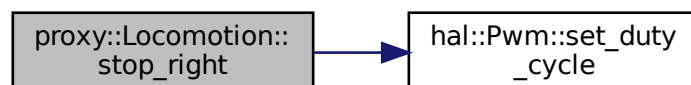


### 7.51.3.7 stop\_right()

```
void proxy::Locomotion::stop_right ( )
```

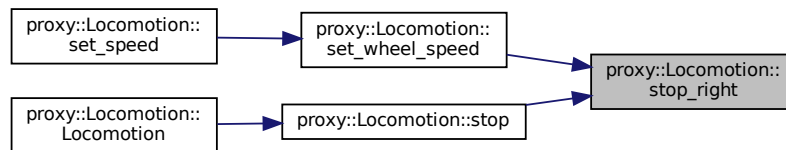
Stop the right motor.

Here is the call graph for this function:





Here is the caller graph for this function:



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

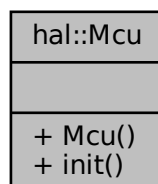
- [inc/proxy/locomotion.hpp](#)
- [src/proxy/locomotion.cpp](#)

## 7.52 hal::Mcu Class Reference

Microcontroller unit class.

```
#include <mcu.hpp>
```

Collaboration diagram for hal::Mcu:



### Public Member Functions

- [Mcu](#) ()=delete  
*Deleted constructor for static class.*

### Static Public Member Functions

- static void [init](#) ()  
*Initializes MCU and some peripherals.*

### 7.52.1 Detailed Description

Microcontroller unit class.

### 7.52.2 Constructor & Destructor Documentation

#### 7.52.2.1 Mcu()

```
hal::Mcu::Mcu ( ) [delete]
```

Deleted constructor for static class.

### 7.52.3 Member Function Documentation

#### 7.52.3.1 init()

```
void hal::Mcu::init ( ) [static]
```

Initializes MCU and some peripherals.

Here is the call graph for this function:



Here is the caller graph for this function:



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

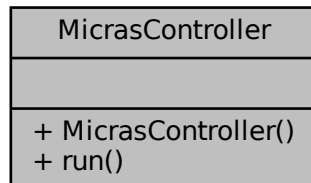
- [inc/hal/mcu.hpp](#)
- [src/hal/mcu.cpp](#)

## 7.53 MicrasController Class Reference

Class for controlling the Micras robot.

```
#include <micras_controller.hpp>
```

Collaboration diagram for MicrasController:



### Public Member Functions

- [MicrasController](#) ()  
*Constructor for the [MicrasController](#) class.*
- void [run](#) ()  
*Runs the controller loop once.*

#### 7.53.1 Detailed Description

Class for controlling the Micras robot.

#### 7.53.2 Constructor & Destructor Documentation

##### 7.53.2.1 MicrasController()

```
MicrasController::MicrasController ( )
```

Constructor for the [MicrasController](#) class.

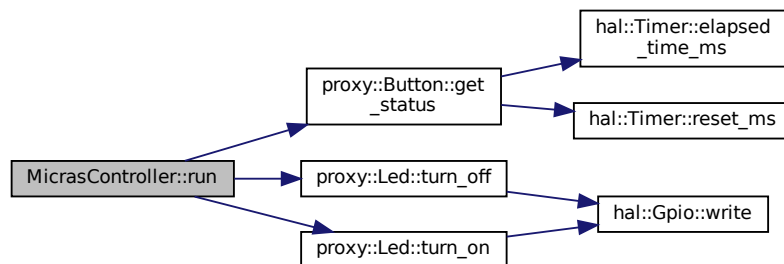
#### 7.53.3 Member Function Documentation

### 7.53.3.1 run()

```
void MicrasController::run ( )
```

Runs the controller loop once.

Here is the call graph for this function:



Here is the caller graph for this function:



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

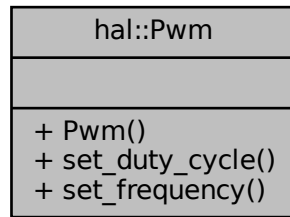
- [inc/controller/micras\\_controller.hpp](#)
- [src/controller/micras\\_controller.cpp](#)

## 7.54 hal::Pwm Class Reference

Class to handle PWM peripheral on STM32 microcontrollers.

```
#include <pwm.hpp>
```

Collaboration diagram for hal::Pwm:



## Classes

- struct [Config](#)  
*PWM configuration struct.*

## Public Member Functions

- [Pwm](#) (const [Config](#) &config)  
*Construct a new [Pwm](#) object.*
- void [set\\_duty\\_cycle](#) (float duty\_cycle)  
*Set the PWM duty cycle.*
- void [set\\_frequency](#) (uint32\_t frequency)  
*Set the PWM frequency.*

### 7.54.1 Detailed Description

Class to handle PWM peripheral on STM32 microcontrollers.

### 7.54.2 Constructor & Destructor Documentation

#### 7.54.2.1 Pwm()

```
hal::Pwm::Pwm (
    const Config & config ) [explicit]
```

Construct a new [Pwm](#) object.

## Parameters

<i>config</i>	Configuration for the PWM
---------------	---------------------------

### 7.54.3 Member Function Documentation

#### 7.54.3.1 set\_duty\_cycle()

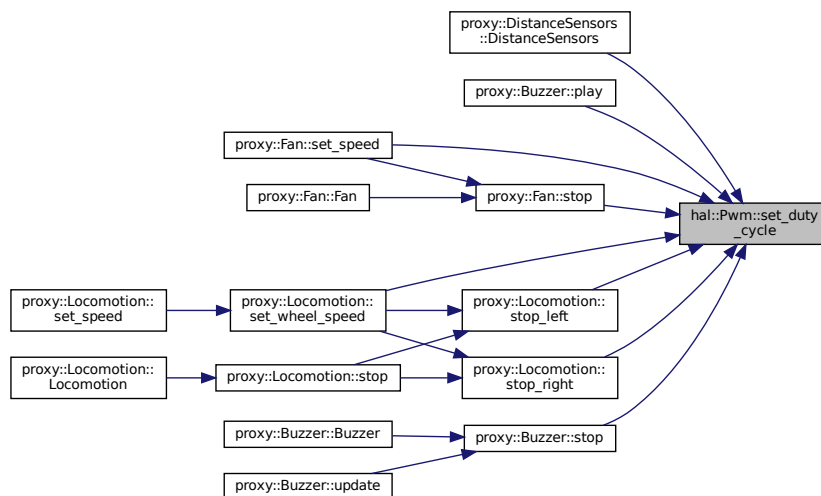
```
void hal::Pwm::set_duty_cycle (
    float duty_cycle )
```

Set the PWM duty cycle.

## Parameters

<i>duty_cycle</i>	Duty cycle value
-------------------	------------------

Here is the caller graph for this function:



#### 7.54.3.2 set\_frequency()

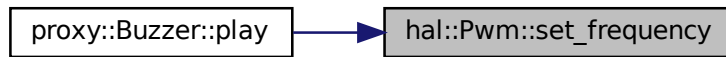
```
void hal::Pwm::set_frequency (
    uint32_t frequency )
```

Set the PWM frequency.

## Parameters

<i>frequency</i>	Frequency value in Hz
------------------	-----------------------

Here is the caller graph for this function:



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

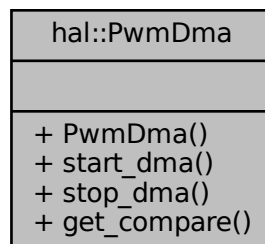
- [inc/hal/pwm.hpp](#)
- [src/hal/pwm.cpp](#)

## 7.55 hal::PwmDma Class Reference

Class to handle PWM peripheral on STM32 microcontrollers using DMA.

```
#include <pwm_dma.hpp>
```

Collaboration diagram for hal::PwmDma:



### Classes

- struct [Config](#)  
*PWM configuration struct.*

## Public Member Functions

- [PwmDma](#) (const [Config](#) &config)  
*Construct a new [PwmDma](#) object.*
- void [start\\_dma](#) (uint32\_t buffer[], uint32\_t size)  
*Start PWM and DMA transfer.*
- void [stop\\_dma](#) ()  
*Stop PWM and DMA transfer.*
- uint32\_t [get\\_compare](#) (float duty\_cycle) const  
*Get the compare value for ad duty cycle.*

### 7.55.1 Detailed Description

Class to handle PWM peripheral on STM32 microcontrollers using DMA.

### 7.55.2 Constructor & Destructor Documentation

#### 7.55.2.1 PwmDma()

```
hal::PwmDma::PwmDma (
    const Config & config ) [explicit]
```

Construct a new [PwmDma](#) object.

##### Parameters

<i>config</i>	Configuration for the PWM
---------------	---------------------------

### 7.55.3 Member Function Documentation

#### 7.55.3.1 get\_compare()

```
uint32_t hal::PwmDma::get_compare (
    float duty_cycle ) const
```

Get the compare value for ad duty cycle.

##### Parameters

<i>duty_cycle</i>	Duty cycle to get the compare value for
-------------------	---



## Returns

uint32\_t Compare value for the duty cycle

## 7.55.3.2 start\_dma()

```
void hal::PwmDma::start_dma (
    uint32_t buffer[],
    uint32_t size )
```

Start PWM and DMA transfer.

## Parameters

<i>buffer</i>	Buffer to transfer
<i>size</i>	Size of the buffer

## 7.55.3.3 stop\_dma()

```
void hal::PwmDma::stop_dma ( )
```

Stop PWM and DMA transfer.

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

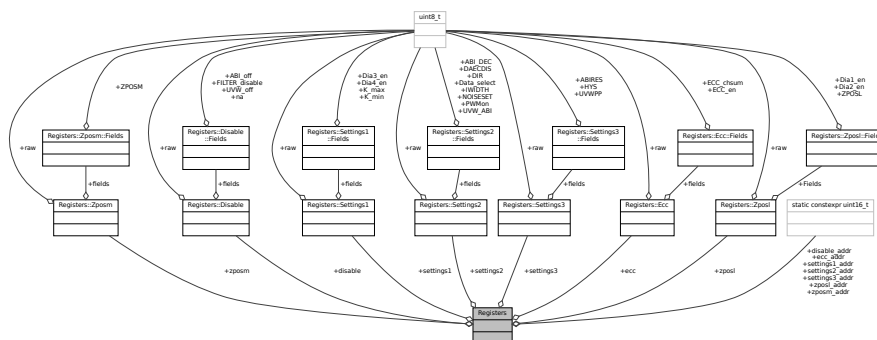
- [inc/hal/pwm\\_dma.hpp](#)
- [src/hal/pwm\\_dma.cpp](#)

## 7.56 Registers Struct Reference

[Registers](#) class for the rotary sensor configuration.

```
#include <rotary_sensor_reg.hpp>
```

Collaboration diagram for Registers:



## Classes

- union [Disable](#)  
*Registers union types definition.*
- union [Ecc](#)
- union [Settings1](#)
- union [Settings2](#)
- union [Settings3](#)
- union [Zposl](#)
- union [Zposm](#)

## Public Attributes

- [Disable](#) `disable`  
*Member variables to be configured in the rotary sensor.*
- [Zposm](#) `zposm`
- [Zposl](#) `zposl`
- [Settings1](#) `settings1`
- [Settings2](#) `settings2`
- [Settings3](#) `settings3`
- [Ecc](#) `ecc`

## Static Public Attributes

- static constexpr uint16\_t [disable\\_addr](#) {0x0015}  
*Register addresses in the rotary sensor memory.*
- static constexpr uint16\_t [zposm\\_addr](#) {0x0016}
- static constexpr uint16\_t [zposl\\_addr](#) {0x0017}
- static constexpr uint16\_t [settings1\\_addr](#) {0x0018}
- static constexpr uint16\_t [settings2\\_addr](#) {0x0019}
- static constexpr uint16\_t [settings3\\_addr](#) {0x001A}
- static constexpr uint16\_t [ecc\\_addr](#) {0x001B}

### 7.56.1 Detailed Description

[Registers](#) class for the rotary sensor configuration.

### 7.56.2 Member Data Documentation

#### 7.56.2.1 `disable`

[Disable](#) `Registers::disable`

Member variables to be configured in the rotary sensor.

### 7.56.2.2 disable\_addr

```
constexpr uint16_t Registers::disable_addr {0x0015} [static], [constexpr]
```

Register addresses in the rotary sensor memory.

### 7.56.2.3 ecc

```
Ecc Registers::ecc
```

### 7.56.2.4 ecc\_addr

```
constexpr uint16_t Registers::ecc_addr {0x001B} [static], [constexpr]
```

### 7.56.2.5 settings1

```
Settings1 Registers::settings1
```

### 7.56.2.6 settings1\_addr

```
constexpr uint16_t Registers::settings1_addr {0x0018} [static], [constexpr]
```

### 7.56.2.7 settings2

```
Settings2 Registers::settings2
```

### 7.56.2.8 settings2\_addr

```
constexpr uint16_t Registers::settings2_addr {0x0019} [static], [constexpr]
```

### 7.56.2.9 settings3

[Settings3](#) Registers::settings3

### 7.56.2.10 settings3\_addr

```
constexpr uint16_t Registers::settings3_addr {0x001A} [static], [constexpr]
```

### 7.56.2.11 zposl

[Zposl](#) Registers::zposl

### 7.56.2.12 zposl\_addr

```
constexpr uint16_t Registers::zposl_addr {0x0017} [static], [constexpr]
```

### 7.56.2.13 zposm

[Zposm](#) Registers::zposm

### 7.56.2.14 zposm\_addr

```
constexpr uint16_t Registers::zposm_addr {0x0016} [static], [constexpr]
```

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

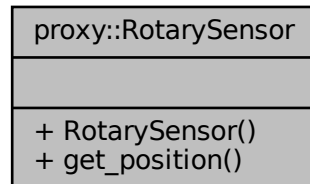
- [inc/proxy/rotary\\_sensor\\_reg.hpp](#)

## 7.57 proxy::RotarySensor Class Reference

Class to handle rotary sensor peripheral on STM32 microcontrollers.

```
#include <rotary_sensor.hpp>
```

Collaboration diagram for proxy::RotarySensor:



### Classes

- struct [Config](#)  
*Rotary sensor configuration struct.*

### Public Member Functions

- [RotarySensor](#) (const [Config](#) &config)  
*Construct a new [RotarySensor](#) object.*
- float [get\\_position](#) () const  
*Get the rotary sensor position over an axis.*

### 7.57.1 Detailed Description

Class to handle rotary sensor peripheral on STM32 microcontrollers.

### 7.57.2 Constructor & Destructor Documentation

#### 7.57.2.1 RotarySensor()

```
proxy::RotarySensor::RotarySensor (  
    const Config & config ) [explicit]
```

Construct a new [RotarySensor](#) object.

## Parameters

<i>config</i>	Configuration for the rotary sensor
---------------	-------------------------------------

### 7.57.3 Member Function Documentation

#### 7.57.3.1 `get_position()`

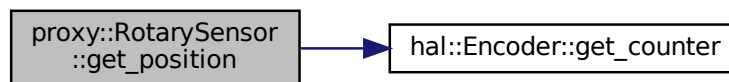
```
float proxy::RotarySensor::get_position ( ) const
```

Get the rotary sensor position over an axis.

## Returns

float Current angular position of the sensor in radians

Here is the call graph for this function:



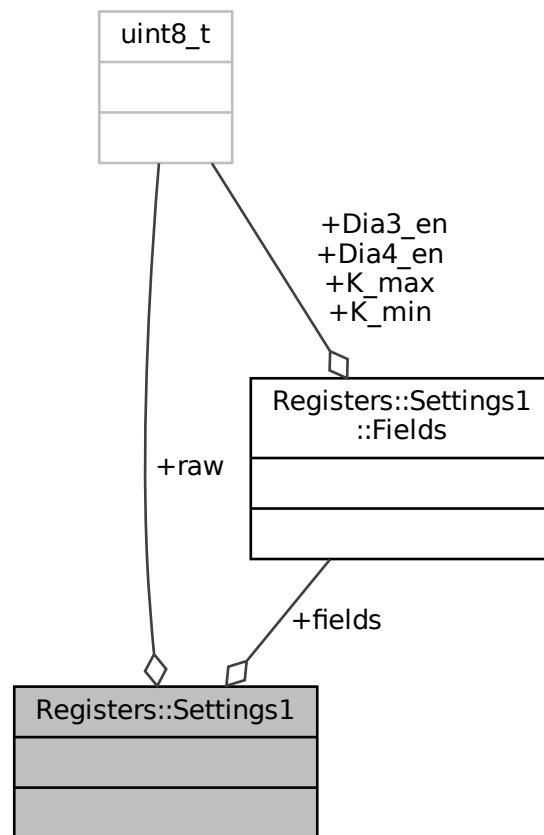
The documentation for this class was generated from the following files:

- [inc/proxy/rotary\\_sensor.hpp](#)
- [src/proxy/rotary\\_sensor.cpp](#)

## 7.58 Registers::Settings1 Union Reference

```
#include <rotary_sensor_reg.hpp>
```

Collaboration diagram for Registers::Settings1:



## Classes

- struct [Fields](#)

## Public Attributes

- [Fields](#) `fields`
- `uint8_t` `raw`

## 7.58.1 Member Data Documentation

### 7.58.1.1 fields

[Fields](#) `Registers::Settings1::fields`

### 7.58.1.2 raw

```
uint8_t Registers::Settings1::raw
```

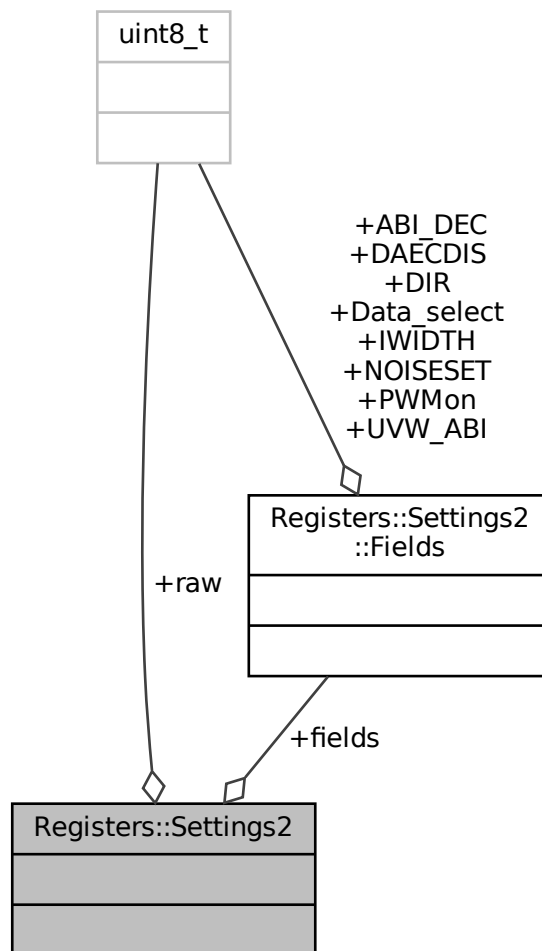
The documentation for this union was generated from the following file:

- [inc/proxy/rotary\\_sensor\\_reg.hpp](#)

## 7.59 Registers::Settings2 Union Reference

```
#include <rotary_sensor_reg.hpp>
```

Collaboration diagram for Registers::Settings2:



### Classes

- struct [Fields](#)



## Public Attributes

- [Fields](#) `fields`
- `uint8_t` `raw`

### 7.59.1 Member Data Documentation

#### 7.59.1.1 `fields`

`Fields` `Registers::Settings2::fields`

#### 7.59.1.2 `raw`

`uint8_t` `Registers::Settings2::raw`

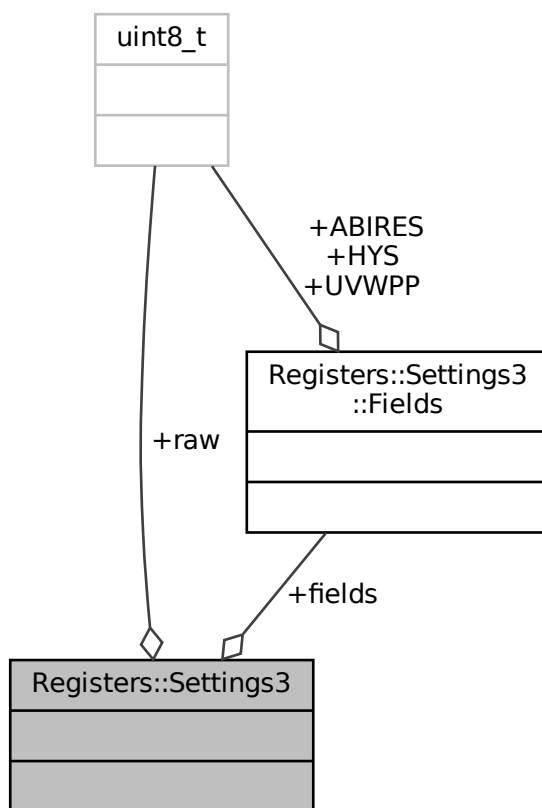
The documentation for this union was generated from the following file:

- `inc/proxy/rotary_sensor_reg.hpp`

## 7.60 Registers::Settings3 Union Reference

```
#include <rotary_sensor_reg.hpp>
```

Collaboration diagram for Registers::Settings3:



## Classes

- struct [Fields](#)

## Public Attributes

- [Fields](#) `fields`
- `uint8_t` `raw`

### 7.60.1 Member Data Documentation

#### 7.60.1.1 fields

[Fields](#) `Registers::Settings3::fields`

### 7.60.1.2 raw

```
uint8_t Registers::Settings3::raw
```

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

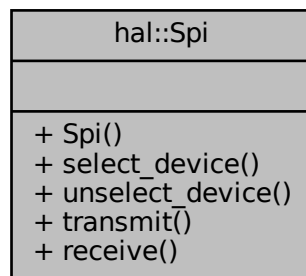
- [inc/proxy/rotary\\_sensor\\_reg.hpp](#)

## 7.61 hal::Spi Class Reference

Class to handle SPI peripheral on STM32 microcontrollers.

```
#include <spi.hpp>
```

Collaboration diagram for hal::Spi:



### Classes

- struct [Config](#)  
*SPI configuration struct.*

### Public Member Functions

- [Spi](#) (const [Config](#) &config)  
*Construct a new Spi object.*
- bool [select\\_device](#) ()  
*Activate the chip select.*
- void [unselect\\_device](#) ()  
*Deactivate the chip select.*
- void [transmit](#) (uint8\_t data[], uint32\_t size)  
*Transmit data over SPI.*
- void [receive](#) (uint8\_t data[], uint32\_t size)  
*Receive data over SPI.*

### 7.61.1 Detailed Description

Class to handle SPI peripheral on STM32 microcontrollers.

### 7.61.2 Constructor & Destructor Documentation

#### 7.61.2.1 Spi()

```
hal::Spi::Spi (
    const Config & config ) [explicit]
```

Construct a new [Spi](#) object.

##### Parameters

<i>config</i>	Configuration for the SPI
---------------	---------------------------

### 7.61.3 Member Function Documentation

#### 7.61.3.1 receive()

```
void hal::Spi::receive (
    uint8_t data[],
    uint32_t size )
```

Receive data over SPI.

##### Parameters

<i>data</i>	Data to receive data
<i>size</i>	Size of the data

#### 7.61.3.2 select\_device()

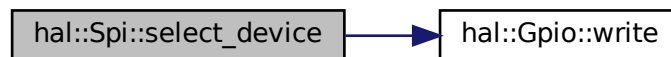
```
bool hal::Spi::select_device ( )
```

Activate the chip select.

**Returns**

bool True if the device was successfully selected, false otherwise

Here is the call graph for this function:

**7.61.3.3 transmit()**

```
void hal::Spi::transmit (
    uint8_t data[],
    uint32_t size )
```

Transmit data over SPI.

**Parameters**

<i>data</i>	Data to transmit
<i>size</i>	Size of the buffer

**7.61.3.4 unselect\_device()**

```
void hal::Spi::unselect_device ( )
```

Deactivate the chip select.

Here is the call graph for this function:



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

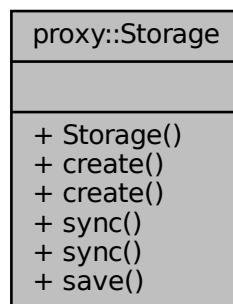
- [inc/hal/spi.hpp](#)
- [src/hal/spi.cpp](#)

## 7.62 proxy::Storage Class Reference

Class for controlling the storage.

```
#include <storage.hpp>
```

Collaboration diagram for proxy::Storage:



### Classes

- struct [Config](#)  
*Configuration structure for the storage.*

### Public Member Functions

- [Storage](#) (const [Config](#) &config)  
*Constructor for the [Storage](#) class.*
- template<Fundamental T>  
void [create](#) (const std::string &name, const T &data)  
*Create a new primitive variable in the storage.*
- void [create](#) (const std::string &name, const [ISerializable](#) &data)  
*Create a new serializable variable in the storage.*
- template<Fundamental T>  
void [sync](#) (const std::string &name, T &data)  
*Sync a primitive variable with the storage.*
- void [sync](#) (const std::string &name, [ISerializable](#) &data)  
*Sync a serializable variable with the storage.*
- void [save](#) ()  
*Save the storage to the flash.*

### 7.62.1 Detailed Description

Class for controlling the storage.

### 7.62.2 Constructor & Destructor Documentation

#### 7.62.2.1 Storage()

```
proxy::Storage::Storage (  
    const Config & config ) [explicit]
```

Constructor for the [Storage](#) class.

##### Parameters

<i>config</i>	Configuration for the storage
---------------	-------------------------------

Here is the call graph for this function:



### 7.62.3 Member Function Documentation

#### 7.62.3.1 create() [1/2]

```
void proxy::Storage::create (  
    const std::string & name,  
    const ISerializable & data )
```

Create a new serializable variable in the storage.

##### Parameters

<i>name</i>	Name of the variable
<i>data</i>	Reference to the variable

### 7.62.3.2 create() [2/2]

```
template<Fundamental T>
void proxy::Storage::create (
    const std::string & name,
    const T & data )
```

Create a new primitive variable in the storage.

#### Template Parameters

<i>T</i>	Type of the variable
----------	----------------------

#### Parameters

<i>name</i>	Name of the variable
<i>data</i>	Reference to the variable

Here is the caller graph for this function:

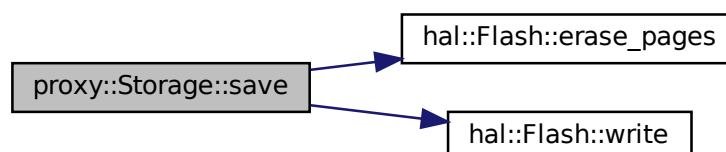


### 7.62.3.3 save()

```
void proxy::Storage::save ( )
```

Save the storage to the flash.

Here is the call graph for this function:





**7.62.3.4 sync()** [1/2]

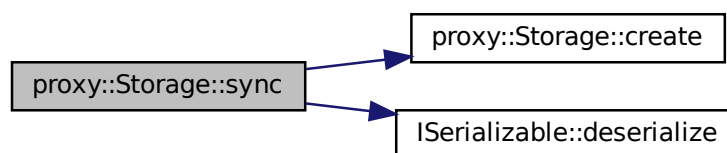
```
void proxy::Storage::sync (
    const std::string & name,
    ISerializable & data )
```

Sync a serializable variable with the storage.

**Parameters**

<i>name</i>	Name of the variable
<i>data</i>	Reference to the variable

Here is the call graph for this function:

**7.62.3.5 sync()** [2/2]

```
template<Fundamental T>
void proxy::Storage::sync (
    const std::string & name,
    T & data )
```

Sync a primitive variable with the storage.

**Template Parameters**

<i>T</i>	Type of the variable
----------	----------------------

**Parameters**

<i>name</i>	Name of the variable
<i>data</i>	Reference to the variable

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

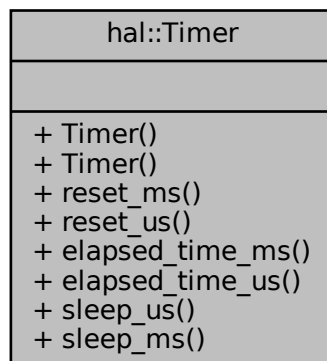
- [inc/proxy/storage.hpp](#)
- [src/proxy/storage.cpp](#)

## 7.63 hal::Timer Class Reference

Class to handle timer peripheral on STM32 microcontrollers.

```
#include <timer.hpp>
```

Collaboration diagram for hal::Timer:



### Classes

- struct [Config](#)  
*Timer configuration struct.*

### Public Member Functions

- [Timer](#) ()=default  
*Construct a new [Timer](#) object.*
- [Timer](#) (const [Config](#) &config)  
*Construct a new [Timer](#) object.*
- void [reset\\_ms](#) ()  
*Reset the timer counter in milliseconds.*
- void [reset\\_us](#) ()  
*Reset the timer counter in microseconds.*
- uint32\_t [elapsed\\_time\\_ms](#) () const  
*Get the time elapsed since the last reset.*
- uint32\_t [elapsed\\_time\\_us](#) () const  
*Get the time elapsed since the last reset.*
- void [sleep\\_us](#) (uint32\_t time) const  
*Sleep for a given amount of time.*

## Static Public Member Functions

- static void `sleep_ms` (uint32\_t time)  
*Sleep for a given amount of time.*

### 7.63.1 Detailed Description

Class to handle timer peripheral on STM32 microcontrollers.

### 7.63.2 Constructor & Destructor Documentation

#### 7.63.2.1 `Timer()` [1/2]

```
hal::Timer::Timer ( ) [default]
```

Construct a new `Timer` object.

#### 7.63.2.2 `Timer()` [2/2]

```
hal::Timer::Timer (
    const Config & config ) [explicit]
```

Construct a new `Timer` object.

##### Parameters

<code>config</code>	Configuration for the timer
---------------------	-----------------------------

### 7.63.3 Member Function Documentation

#### 7.63.3.1 `elapsed_time_ms()`

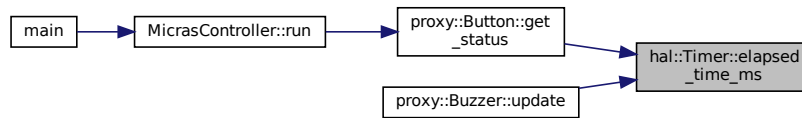
```
uint32_t hal::Timer::elapsed_time_ms ( ) const
```

Get the time elapsed since the last reset.

**Returns**

uint32\_t Time elapsed in milliseconds

Here is the caller graph for this function:

**7.63.3.2 elapsed\_time\_us()**

```
uint32_t hal::Timer::elapsed_time_us ( ) const
```

Get the time elapsed since the last reset.

**Returns**

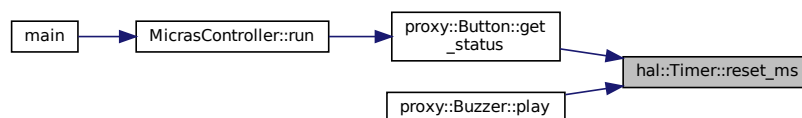
uint32\_t Time elapsed in microseconds

**7.63.3.3 reset\_ms()**

```
void hal::Timer::reset_ms ( )
```

Reset the timer counter in milliseconds.

Here is the caller graph for this function:



### 7.63.3.4 reset\_us()

```
void hal::Timer::reset_us ( )
```

Reset the timer counter in microseconds.

### 7.63.3.5 sleep\_ms()

```
void hal::Timer::sleep_ms (
    uint32_t time ) [static]
```

Sleep for a given amount of time.

#### Parameters

<i>time</i>	Time to sleep in milliseconds
-------------	-------------------------------

Here is the caller graph for this function:



### 7.63.3.6 sleep\_us()

```
void hal::Timer::sleep_us (
    uint32_t time ) const
```

Sleep for a given amount of time.

#### Parameters

<i>time</i>	Time to sleep in microseconds
-------------	-------------------------------

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

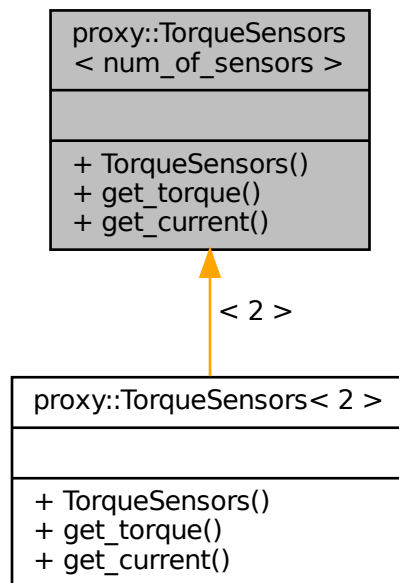
- [inc/hal/timer.hpp](#)
- [src/hal/timer.cpp](#)

## 7.64 proxy::TorqueSensors< num\_of\_sensors > Class Template Reference

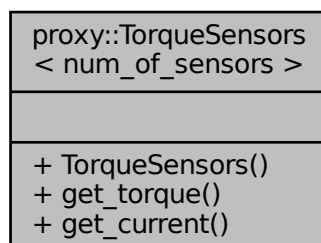
Class for controlling [TorqueSensors](#).

```
#include <torque_sensors.hpp>
```

Inheritance diagram for proxy::TorqueSensors< num\_of\_sensors >:



Collaboration diagram for proxy::TorqueSensors< num\_of\_sensors >:



## Classes

- struct [Config](#)

*Configuration structure for torque sensors.*

## Public Member Functions

- [TorqueSensors](#) (const [Config](#) &config)  
*Constructor for the [TorqueSensors](#) class.*
- float [get\\_torque](#) (uint8\_t sensor\_index) const  
*Get the torque from the sensor.*
- float [get\\_current](#) (uint8\_t sensor\_index) const  
*Get the current from the sensor.*

### 7.64.1 Detailed Description

```
template<uint8_t num_of_sensors>
class proxy::TorqueSensors< num_of_sensors >
```

Class for controlling [TorqueSensors](#).

### 7.64.2 Constructor & Destructor Documentation

#### 7.64.2.1 TorqueSensors()

```
template<uint8_t num_of_sensors>
proxy::TorqueSensors< num_of_sensors >::TorqueSensors (
    const Config & config ) [explicit]
```

Constructor for the [TorqueSensors](#) class.

#### Parameters

<i>config</i>	Configuration for the torque sensors
---------------	--------------------------------------

### 7.64.3 Member Function Documentation

#### 7.64.3.1 get\_current()

```
template<uint8_t num_of_sensors>
float proxy::TorqueSensors< num_of_sensors >::get_current (
    uint8_t sensor_index ) const
```

Get the current from the sensor.

#### Parameters

<i>sensor_index</i>	Index of the sensor
---------------------	---------------------

#### Returns

float Current reading from the sensor in amps

### 7.64.3.2 get\_torque()

```
template<uint8_t num_of_sensors>
float proxy::TorqueSensors< num_of_sensors >::get_torque (
    uint8_t sensor_index ) const
```

Get the torque from the sensor.

#### Parameters

<i>sensor_index</i>	Index of the sensor
---------------------	---------------------

#### Returns

float Torque reading from the sensor in N\*m

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

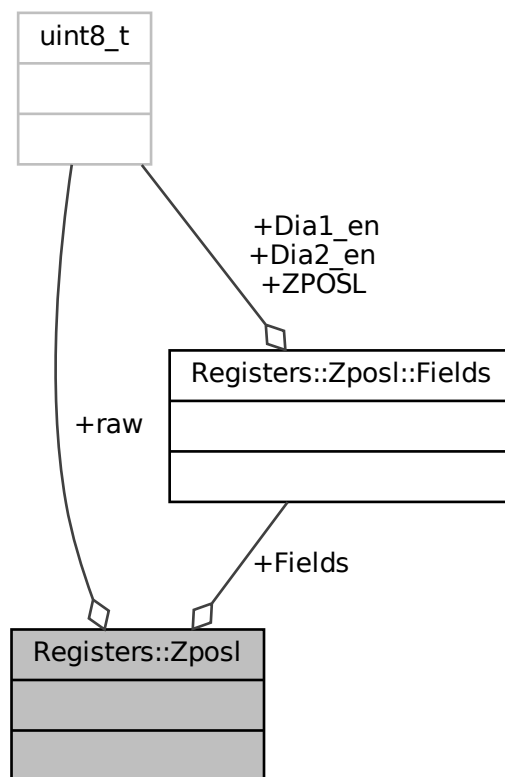
- [inc/proxy/torque\\_sensors.hpp](#)
- [src/proxy/torque\\_sensors.cpp](#)

## 7.65 Registers::Zposl Union Reference

```
#include <rotary_sensor_reg.hpp>
```



Collaboration diagram for Registers::Zposl:



## Classes

- struct [Fields](#)

## Public Attributes

- [Fields](#) `Fields`
- `uint8_t` `raw`

## 7.65.1 Member Data Documentation

### 7.65.1.1 Fields

`Fields` `Registers::Zposl::Fields`

### 7.65.1.2 raw

```
uint8_t Registers::Zposl::raw
```

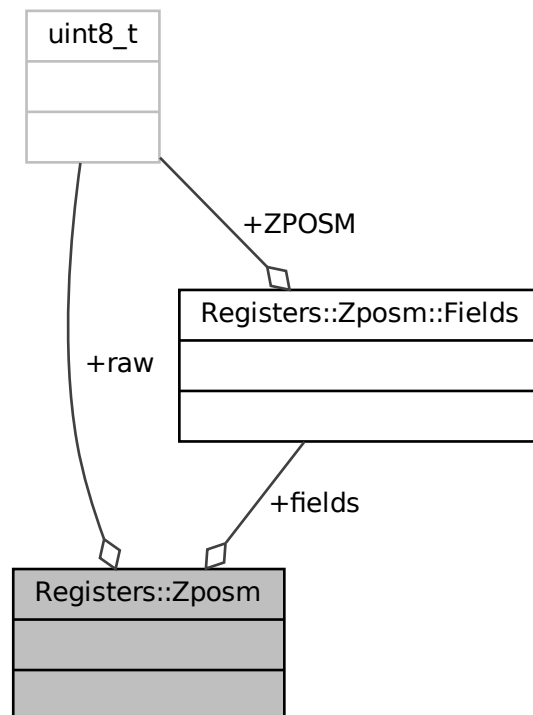
The documentation for this union was generated from the following file:

- inc/proxy/[rotary\\_sensor\\_reg.hpp](#)

## 7.66 Registers::Zposm Union Reference

```
#include <rotary_sensor_reg.hpp>
```

Collaboration diagram for Registers::Zposm:



### Classes

- struct [Fields](#)

### Public Attributes

- [Fields](#) `fields`
- `uint8_t` `raw`

## 7.66.1 Member Data Documentation

### 7.66.1.1 fields

[Fields](#) Registers::Zposm::fields

### 7.66.1.2 raw

uint8\_t Registers::Zposm::raw

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

- [inc/proxy/rotary\\_sensor\\_reg.hpp](#)



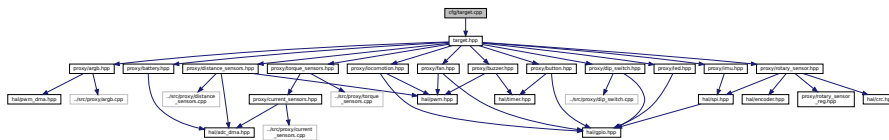
## Chapter 8

# File Documentation

### 8.1 cfg/target.cpp File Reference

Target specific configuration.

```
#include "target.hpp"  
Include dependency graph for target.cpp:
```



### Variables

- const [proxy::Argb< 2 >::Config](#) [argb\\_config](#)
- const [proxy::Battery::Config](#) [battery\\_config](#)
- const [proxy::Button::Config](#) [button\\_config](#)
- const [proxy::Buzzer::Config](#) [buzzer\\_config](#)
- const [proxy::DipSwitch< 4 >::Config](#) [dip\\_switch\\_config](#)
- const [proxy::DistanceSensors< 4 >::Config](#) [distance\\_sensors\\_config](#)
- const [proxy::Fan::Config](#) [fan\\_config](#)
- const [proxy::Imu::Config](#) [imu\\_config](#)
- const [proxy::Led::Config](#) [led\\_config](#)
- const [proxy::Locomotion::Config](#) [locomotion\\_config](#)
- const [proxy::RotarySensor::Registers](#) [rotary\\_sensor\\_reg\\_config](#)
- const [proxy::RotarySensor::Config](#) [rotary\\_sensor\\_left\\_config](#)
- const [proxy::RotarySensor::Config](#) [rotary\\_sensor\\_right\\_config](#)
- const [proxy::TorqueSensors< 2 >::Config](#) [torque\\_sensors\\_config](#)

#### 8.1.1 Detailed Description

Target specific configuration.

Date

03/2024

## 8.1.2 Variable Documentation

### 8.1.2.1 argb\_config

```
const proxy::Argb<2>::Config argb_config
```

**Initial value:**

```
= {  
    .pwm = {  
        .handle = &htim8,  
        .init_function = MX_TIM8_Init,  
        .timer_channel = TIM_CHANNEL_1  
    }  
}
```

### 8.1.2.2 battery\_config

```
const proxy::Battery::Config battery_config
```

**Initial value:**

```
= {  
    .adc = {  
        .handle = &hadc3,  
        .init_function = MX_ADC3_Init,  
        .max_reading = 4095,  
        .reference_voltage = 3.0F  
    },  
    .voltage_divider = 3.0F  
}
```

### 8.1.2.3 button\_config

```
const proxy::Button::Config button_config
```

**Initial value:**

```
= {  
    .gpio = {  
        .port = GPIOA,  
        .pin = GPIO_PIN_12  
    },  
    .pull_resistor = proxy::Button::PullResistor::PULL_UP  
}
```

### 8.1.2.4 buzzer\_config

```
const proxy::Buzzer::Config buzzer_config
```

**Initial value:**

```
= {  
    .pwm = {  
        .handle = &htim4,  
        .init_function = MX_TIM4_Init,  
        .timer_channel = TIM_CHANNEL_1  
    }  
}
```

### 8.1.2.5 dip\_switch\_config

```
const proxy::DipSwitch<4>::Config dip_switch_config
```

**Initial value:**

```
= {  
    .gpio_array = {{  
        {  
            .port = GPIOC,  
            .pin = GPIO_PIN_7  
        },  
        {  
            .port = GPIOB,  
            .pin = GPIO_PIN_15  
        },  
        {  
            .port = GPIOB,  
            .pin = GPIO_PIN_14  
        },  
        {  
            .port = GPIOB,  
            .pin = GPIO_PIN_13  
        }  
    }}  
}
```

### 8.1.2.6 distance\_sensors\_config

```
const proxy::DistanceSensors<4>::Config distance_sensors_config
```

**Initial value:**

```
= {  
    .adc = {  
        .handle = &hadc1,  
        .init_function = MX_ADC1_Init,  
        .max_reading = 4095,  
        .reference_voltage = 3.3F  
    },  
    .led_pwm = {  
        .handle = &htim15,  
        .init_function = MX_TIM15_Init,  
        .timer_channel = TIM_CHANNEL_1  
    },  
    .max_distance = 0.3F  
}
```

### 8.1.2.7 fan\_config

```
const proxy::Fan::Config fan_config
```

**Initial value:**

```
= {  
    .pwm = {  
        .handle = &htim17,  
        .init_function = MX_TIM17_Init,  
        .timer_channel = TIM_CHANNEL_1  
    },  
    .direction_gpio = {  
        .port = GPIOC,  
        .pin = GPIO_PIN_13  
    },  
    .enable_gpio = {  
        .port = GPIOB,  
        .pin = GPIO_PIN_7  
    }  
}
```

### 8.1.2.8 imu\_config

```
const proxy::Imu::Config imu_config
```

#### Initial value:

```
= {  
    .spi = {  
        .handle = &hspi1,  
        .init_function = MX_SPI1_Init,  
        .gpio = {  
            .port = GPIOB,  
            .pin = GPIO_PIN_6  
        },  
        .timeout = 2  
    },  
    .gyroscope_data_rate = LSM6DSV_ODR_AT_480Hz,  
    .accelerometer_data_rate = LSM6DSV_ODR_AT_480Hz,  
    .orientation_data_rate = LSM6DSV_SFLP_480Hz,  
    .gyroscope_scale = LSM6DSV_4000dps,  
    .accelerometer_scale = LSM6DSV_8g,  
    .gyroscope_filter = LSM6DSV_GY_ULTRA_LIGHT,  
    .accelerometer_filter = LSM6DSV_XL_ULTRA_LIGHT  
}
```

### 8.1.2.9 led\_config

```
const proxy::Led::Config led_config
```

#### Initial value:

```
= {  
    .gpio = {  
        .port = GPIOA,  
        .pin = GPIO_PIN_15  
    }  
}
```

### 8.1.2.10 locomotion\_config

```
const proxy::Locomotion::Config locomotion_config
```

#### Initial value:

```
= {  
    .pwm_left_fwd = {  
        .handle = &htim4,  
        .init_function = MX_TIM4_Init,  
        .timer_channel = TIM_CHANNEL_4  
    },  
    .pwm_left_bwd = {  
        .handle = &htim4,  
        .init_function = MX_TIM4_Init,  
        .timer_channel = TIM_CHANNEL_3  
    },  
    .pwm_right_fwd = {  
        .handle = &htim1,  
        .init_function = MX_TIM1_Init,  
        .timer_channel = TIM_CHANNEL_2  
    },  
    .pwm_right_bwd = {  
        .handle = &htim1,  
        .init_function = MX_TIM1_Init,  
        .timer_channel = TIM_CHANNEL_1  
    },  
    .enable_gpio = {  
        .port = GPIOA,  
        .pin = GPIO_PIN_10  
    }  
}
```



### 8.1.2.11 rotary\_sensor\_left\_config

```
const proxy::RotarySensor::Config rotary_sensor_left_config
```

#### Initial value:

```
= {  
    .spi = {  
        .handle = &hspi1,  
        .init_function = MX_SPI1_Init,  
        .gpio = {  
            .port = GPIOA,  
            .pin = GPIO_PIN_6  
        },  
        .timeout = 2  
    },  
    .encoder = {  
        .handle = &htim2,  
        .init_function = MX_TIM2_Init,  
        .timer_channel = TIM_CHANNEL_ALL  
    },  
    .crc = {  
        .handle = &hcrc  
    },  
    .resolution = 4096,  
    .registers = rotary_sensor_reg_config  
}
```

### 8.1.2.12 rotary\_sensor\_reg\_config

```
const proxy::RotarySensor::Registers rotary_sensor_reg_config
```

### 8.1.2.13 rotary\_sensor\_right\_config

```
const proxy::RotarySensor::Config rotary_sensor_right_config
```

#### Initial value:

```
= {  
    .spi = {  
        .handle = &hspi1,  
        .init_function = MX_SPI1_Init,  
        .gpio = {  
            .port = GPIOB,  
            .pin = GPIO_PIN_1  
        },  
        .timeout = 2  
    },  
    .encoder = {  
        .handle = &htim5,  
        .init_function = MX_TIM5_Init,  
        .timer_channel = TIM_CHANNEL_ALL  
    },  
    .crc = {  
        .handle = &hcrc  
    },  
    .resolution = 4096,  
    .registers = rotary_sensor_reg_config  
}
```

### 8.1.2.14 torque\_sensors\_config

```
const proxy::TorqueSensors<2>::Config torque_sensors_config
```

Initial value:

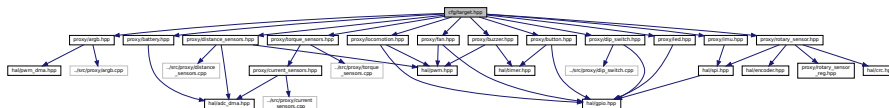
```
= {
    .current_sensors = {
        .adc = {
            .handle = &hadc2,
            .init_function = MX_ADC2_Init,
            .max_reading = 4095,
            .reference_voltage = 3.3F
        },
        .shunt_resistor = 0.04F
    },
    .max_torque = 0.5F
}
```

## 8.2 cfg/target.hpp File Reference

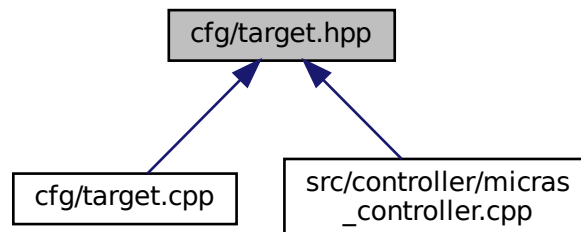
Target specific configuration.

```
#include "proxy/argb.hpp"
#include "proxy/battery.hpp"
#include "proxy/button.hpp"
#include "proxy/buzzer.hpp"
#include "proxy/dip_switch.hpp"
#include "proxy/distance_sensors.hpp"
#include "proxy/fan.hpp"
#include "proxy/imu.hpp"
#include "proxy/led.hpp"
#include "proxy/locomotion.hpp"
#include "proxy/rotary_sensor.hpp"
#include "proxy/torque_sensors.hpp"
```

Include dependency graph for target.hpp:



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



## Variables

- const [proxy::Argb< 2 >::Config](#) [argb\\_config](#)
- const [proxy::Battery::Config](#) [battery\\_config](#)
- const [proxy::Button::Config](#) [button\\_config](#)
- const [proxy::Buzzer::Config](#) [buzzer\\_config](#)
- const [proxy::DipSwitch< 4 >::Config](#) [dip\\_switch\\_config](#)
- const [proxy::DistanceSensors< 4 >::Config](#) [distance\\_sensors\\_config](#)
- const [proxy::Fan::Config](#) [fan\\_config](#)
- const [proxy::Imu::Config](#) [imu\\_config](#)
- const [proxy::Led::Config](#) [led\\_config](#)
- const [proxy::Locomotion::Config](#) [locomotion\\_config](#)
- const [proxy::RotarySensor::Config](#) [rotary\\_sensor\\_left\\_config](#)
- const [proxy::RotarySensor::Config](#) [rotary\\_sensor\\_right\\_config](#)
- const [proxy::TorqueSensors< 2 >::Config](#) [torque\\_sensors\\_config](#)

### 8.2.1 Detailed Description

Target specific configuration.

Date

03/2024

### 8.2.2 Variable Documentation

#### 8.2.2.1 [argb\\_config](#)

```
const proxy::Argb<2>::Config argb\_config [extern]
```

#### 8.2.2.2 [battery\\_config](#)

```
const proxy::Battery::Config battery\_config [extern]
```

#### 8.2.2.3 [button\\_config](#)

```
const proxy::Button::Config button\_config [extern]
```

#### 8.2.2.4 buzzer\_config

```
const proxy::Buzzer::Config buzzer_config [extern]
```

#### 8.2.2.5 dip\_switch\_config

```
const proxy::DipSwitch<4>::Config dip_switch_config [extern]
```

#### 8.2.2.6 distance\_sensors\_config

```
const proxy::DistanceSensors<4>::Config distance_sensors_config [extern]
```

#### 8.2.2.7 fan\_config

```
const proxy::Fan::Config fan_config [extern]
```

#### 8.2.2.8 imu\_config

```
const proxy::Imu::Config imu_config [extern]
```

#### 8.2.2.9 led\_config

```
const proxy::Led::Config led_config [extern]
```

#### 8.2.2.10 locomotion\_config

```
const proxy::Locomotion::Config locomotion_config [extern]
```

#### 8.2.2.11 rotary\_sensor\_left\_config

```
const proxy::RotarySensor::Config rotary_sensor_left_config [extern]
```

## 8.2.2.12 rotary\_sensor\_right\_config

```
const proxy::RotarySensor::Config rotary_sensor_right_config [extern]
```

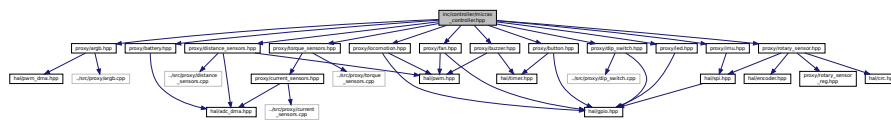
## 8.2.2.13 torque\_sensors\_config

```
const proxy::TorqueSensors<2>::Config torque_sensors_config [extern]
```

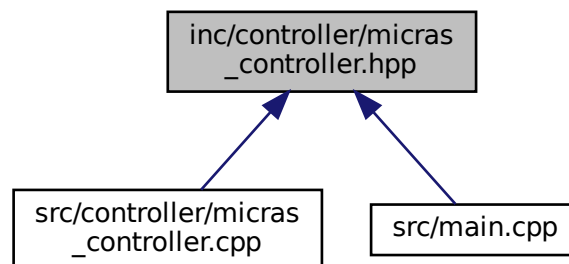
## 8.3 inc/controller/micras\_controller.hpp File Reference

```
#include "proxy/argb.hpp"
#include "proxy/battery.hpp"
#include "proxy/button.hpp"
#include "proxy/buzzer.hpp"
#include "proxy/dip_switch.hpp"
#include "proxy/distance_sensors.hpp"
#include "proxy/fan.hpp"
#include "proxy/imu.hpp"
#include "proxy/led.hpp"
#include "proxy/locomotion.hpp"
#include "proxy/rotary_sensor.hpp"
#include "proxy/torque_sensors.hpp"
```

Include dependency graph for micras\_controller.hpp:



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



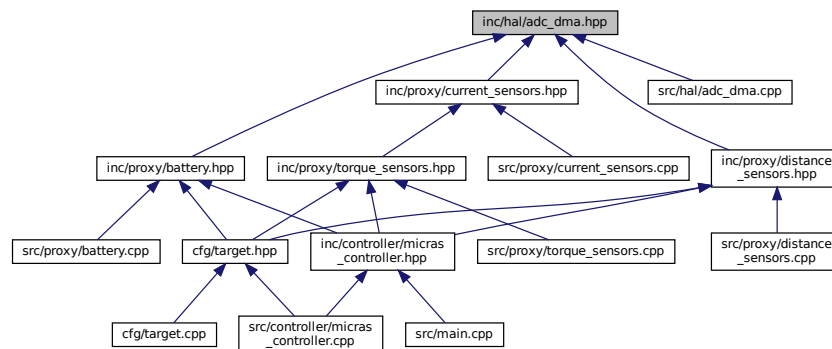
## Classes

- class [MicrasController](#)  
*Class for controlling the Micras robot.*

## 8.4 inc/hal/adc\_dma.hpp File Reference

ADC DMA HAL header.

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



## Classes

- class [hal::AdcDma](#)  
*Class to handle ADC peripheral on STM32 microcontrollers using DMA.*
- struct [hal::AdcDma::Config](#)  
*Configuration structure for ADC DMA.*

## Namespaces

- [hal](#)

### 8.4.1 Detailed Description

ADC DMA HAL header.

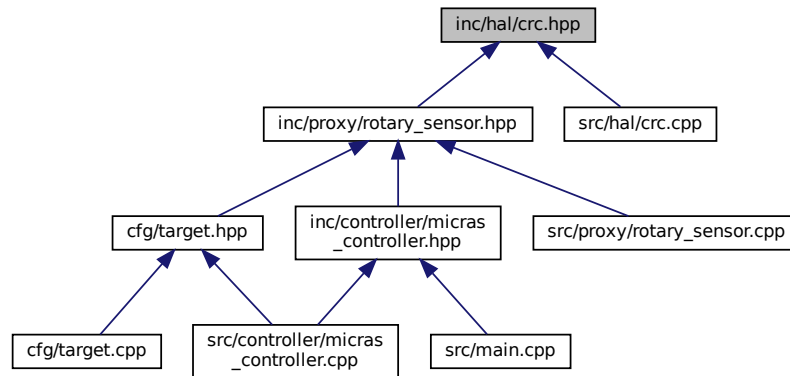
Date

03/2024

## 8.5 inc/hal/crc.hpp File Reference

STM32 CRC HAL wrapper.

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



### Classes

- class [hal::Crc](#)  
*Class to handle the cyclic redundancy check peripheral on STM32 microcontrollers.*
- struct [hal::Crc::Config](#)  
*CRC configuration struct.*

### Namespaces

- [hal](#)

#### 8.5.1 Detailed Description

STM32 CRC HAL wrapper.

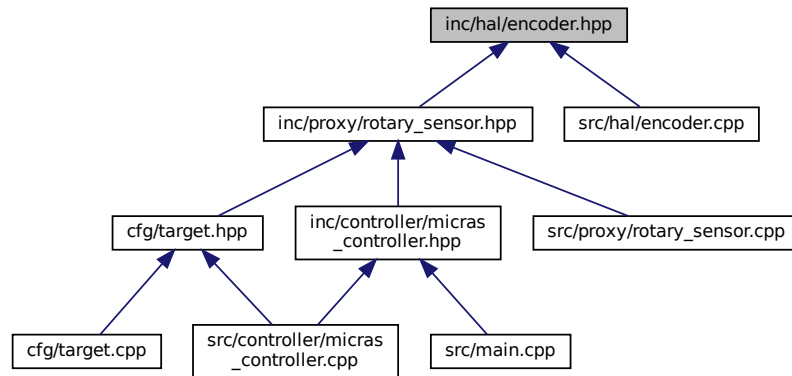
Date

03/2024

## 8.6 inc/hal/encoder.hpp File Reference

STM32 encoder HAL wrapper.

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



### Classes

- class [hal::Encoder](#)  
*Class to handle encoder peripheral on STM32 microcontrollers.*
- struct [hal::Encoder::Config](#)  
*Encoder configuration struct.*

### Namespaces

- [hal](#)

#### 8.6.1 Detailed Description

STM32 encoder HAL wrapper.

Date

03/2024





## Classes

- class [hal::Gpio](#)  
*Class for controlling GPIO pins on STM32 microcontrollers.*
- struct [hal::Gpio::Config](#)  
*Configuration structure for GPIO pin.*

## Namespaces

- [hal](#)

### 8.8.1 Detailed Description

HAL GPIO class header.

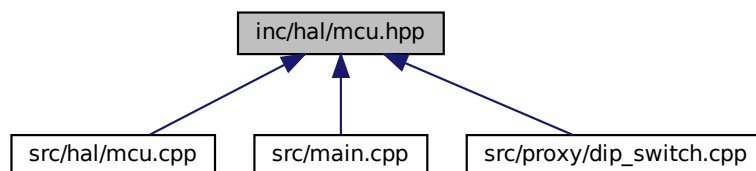
Date

03/2024

### 8.9 inc/hal/mcu.hpp File Reference

MCU related.

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



## Classes

- class [hal::Mcu](#)  
*Microcontroller unit class.*

## Namespaces

- [hal](#)

### 8.9.1 Detailed Description

MCU related.

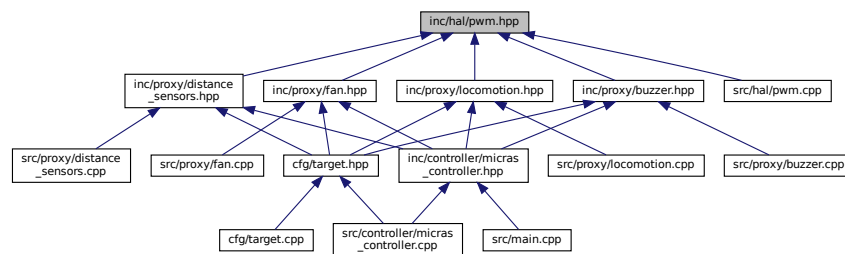
Date

03/2024

## 8.10 inc/hal/pwm.hpp File Reference

STM32 PWM HAL wrapper.

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



### Classes

- class [hal::Pwm](#)  
Class to handle PWM peripheral on STM32 microcontrollers.
- struct [hal::Pwm::Config](#)  
PWM configuration struct.

### Namespaces

- [hal](#)

### 8.10.1 Detailed Description

STM32 PWM HAL wrapper.

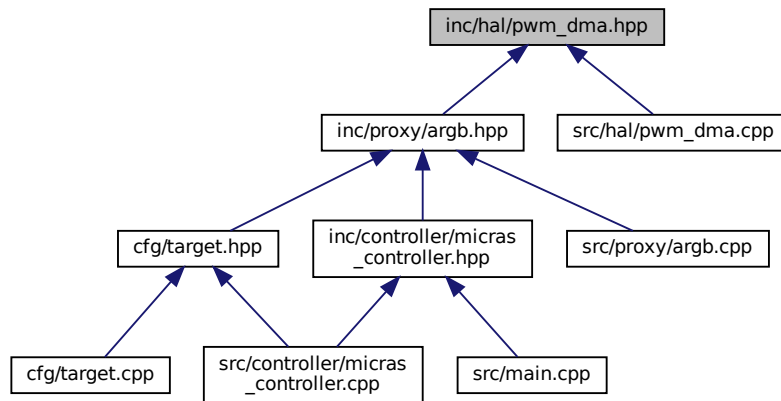
Date

03/2024

## 8.11 inc/hal/pwm\_dma.hpp File Reference

STM32 PWM DMA HAL wrapper.

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



### Classes

- class [hal::PwmDma](#)  
*Class to handle PWM peripheral on STM32 microcontrollers using DMA.*
- struct [hal::PwmDma::Config](#)  
*PWM configuration struct.*

### Namespaces

- [hal](#)

#### 8.11.1 Detailed Description

STM32 PWM DMA HAL wrapper.

Date

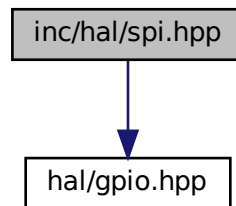
03/2024

## 8.12 inc/hal/spi.hpp File Reference

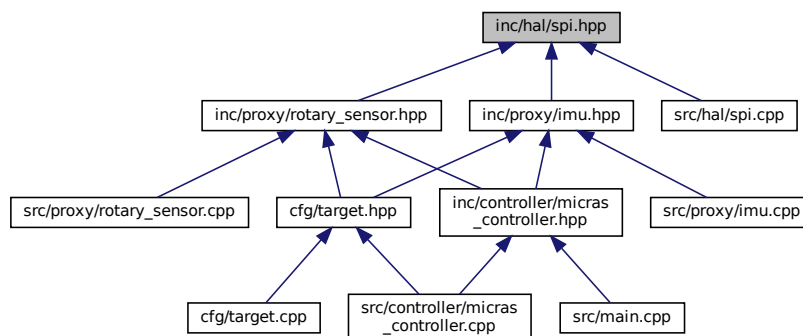
STM32 SPI HAL wrapper.

```
#include "hal/gpio.hpp"
```

Include dependency graph for spi.hpp:



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



### Classes

- class [hal::Spi](#)  
*Class to handle SPI peripheral on STM32 microcontrollers.*
- struct [hal::Spi::Config](#)  
*SPI configuration struct.*

### Namespaces

- [hal](#)

### 8.12.1 Detailed Description

STM32 SPI HAL wrapper.

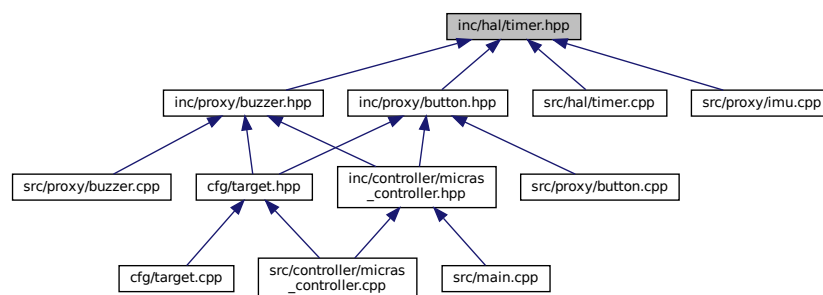
Date

03/2024

## 8.13 inc/hal/timer.hpp File Reference

STM32 Timer HAL wrapper.

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



### Classes

- class [hal::Timer](#)  
*Class to handle timer peripheral on STM32 microcontrollers.*
- struct [hal::Timer::Config](#)  
*Timer configuration struct.*

### Namespaces

- [hal](#)

### 8.13.1 Detailed Description

STM32 Timer HAL wrapper.

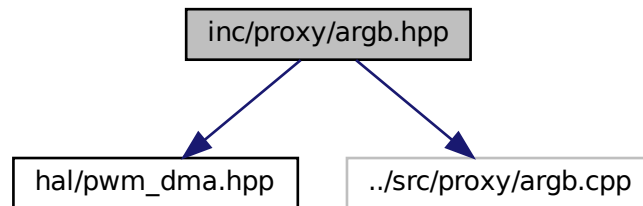
Date

03/2024

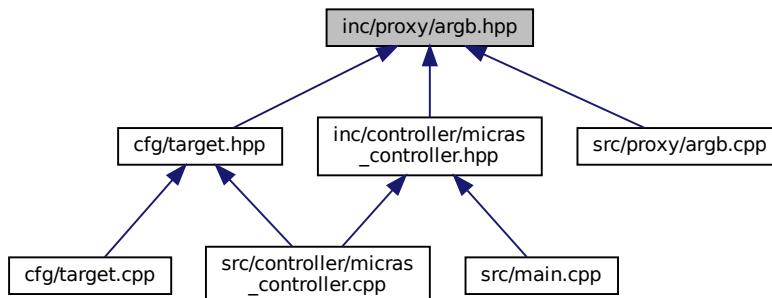
## 8.14 inc/proxy/argb.hpp File Reference

Proxy Argb class declaration.

```
#include "hal/pwm_dma.hpp"
#include "../src/proxy/argb.cpp"
Include dependency graph for argb.hpp:
```



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



### Classes

- class `proxy::Argb< num_of_leds >`  
*Class for controlling an addressable RGB LED.*
- struct `proxy::Argb< num_of_leds >::Config`  
*Configuration structure for the addressable RGB LED.*
- struct `proxy::Argb< num_of_leds >::Color`  
*Structure for storing color information.*

### Namespaces

- `proxy`

### 8.14.1 Detailed Description

Proxy Argb class declaration.

Date

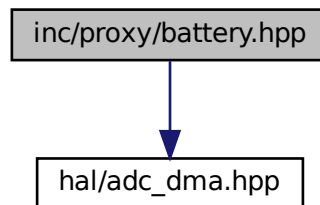
03/2024

## 8.15 inc/proxy/battery.hpp File Reference

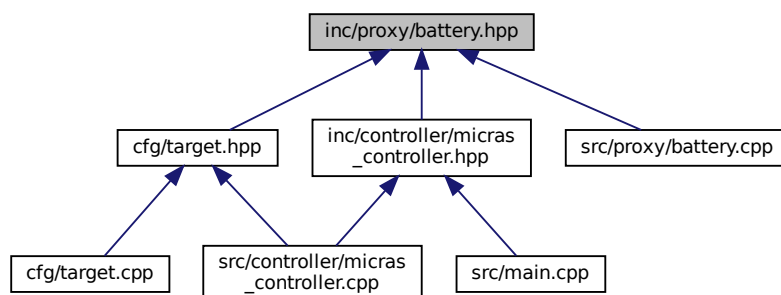
Proxy Battery class declaration.

```
#include "hal/adc_dma.hpp"
```

Include dependency graph for battery.hpp:



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



## Classes

- class `proxy::Battery`  
*Class for getting the battery voltage.*
- struct `proxy::Battery::Config`  
*Configuration structure for the battery.*



## Namespaces

- [proxy](#)

### 8.15.1 Detailed Description

Proxy Battery class declaration.

Date

03/2024

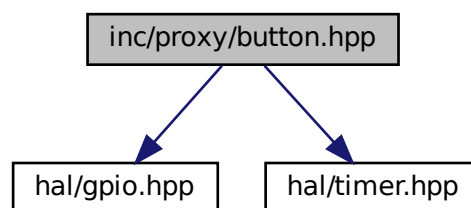
## 8.16 inc/proxy/button.hpp File Reference

Proxy Button class header.

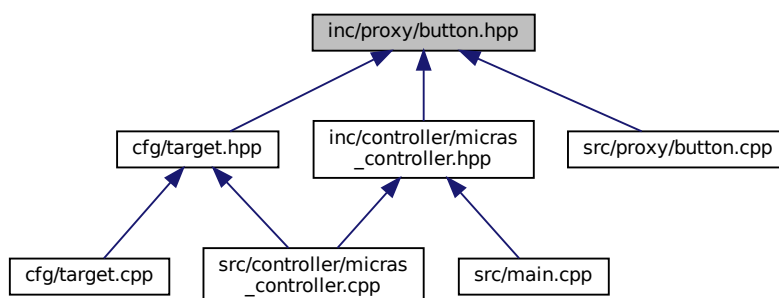
```
#include "hal/gpio.hpp"
```

```
#include "hal/timer.hpp"
```

Include dependency graph for button.hpp:



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



## Classes

- class `proxy::Button`  
*Class for controlling a button.*
- struct `proxy::Button::Config`  
*Configuration structure for button.*

## Namespaces

- `proxy`

### 8.16.1 Detailed Description

Proxy Button class header.

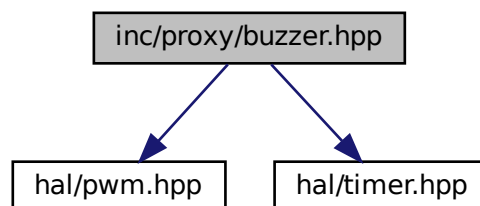
Date

03/2024

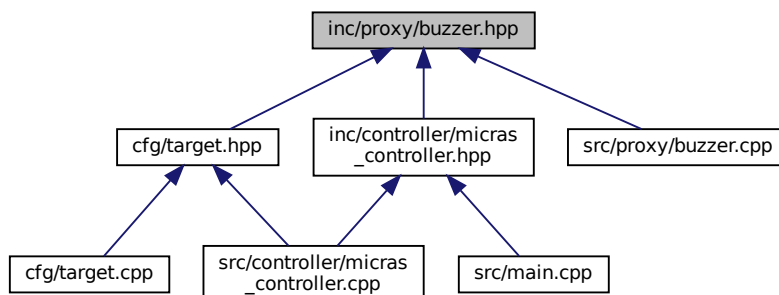
### 8.17 inc/proxy/buzzer.hpp File Reference

Proxy Buzzer class declaration.

```
#include "hal/pwm.hpp"  
#include "hal/timer.hpp"  
Include dependency graph for buzzer.hpp:
```



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



## Classes

- class `proxy::Buzzer`  
*Class for controlling a buzzer.*
- struct `proxy::Buzzer::Config`  
*Configuration structure for the buzzer.*

## Namespaces

- `proxy`

### 8.17.1 Detailed Description

Proxy Buzzer class declaration.

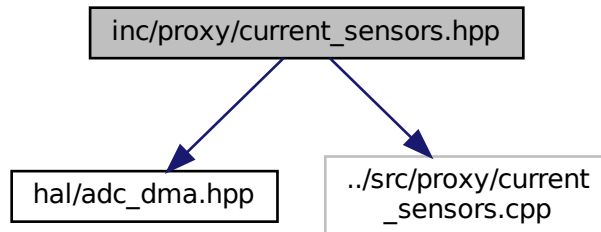
Date

03/2024

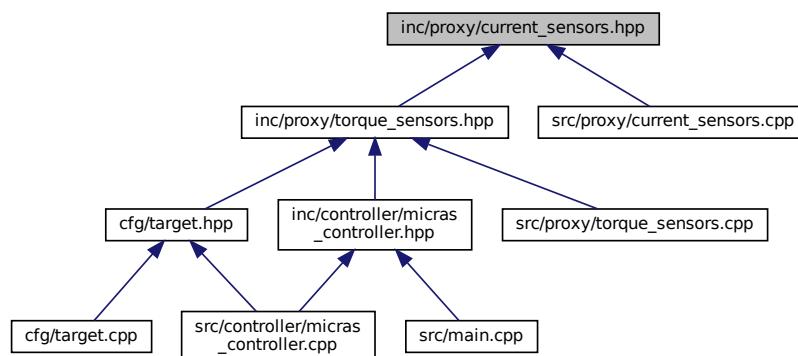
## 8.18 inc/proxy/current\_sensors.hpp File Reference

Proxy CurrentSensors class header.

```
#include "hal/adc_dma.hpp"
#include "../src/proxy/current_sensors.cpp"
Include dependency graph for current_sensors.hpp:
```



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



## Classes

- class `proxy::CurrentSensors< num_of_sensors >`  
*Class for controlling `CurrentSensors`.*
- struct `proxy::CurrentSensors< num_of_sensors >::Config`  
*Configuration structure for current sensors.*

## Namespaces

- `proxy`

### 8.18.1 Detailed Description

Proxy `CurrentSensors` class header.

Date

03/2024



### 8.19.1 Detailed Description

Proxy Dip Switch class header.

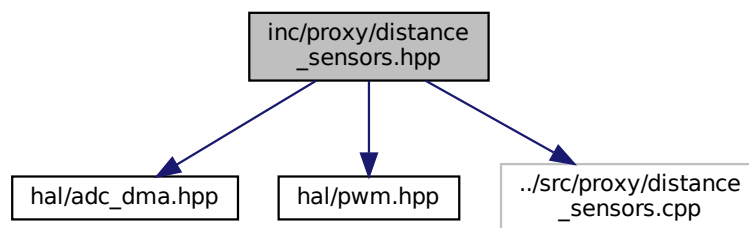
Date

03/2024

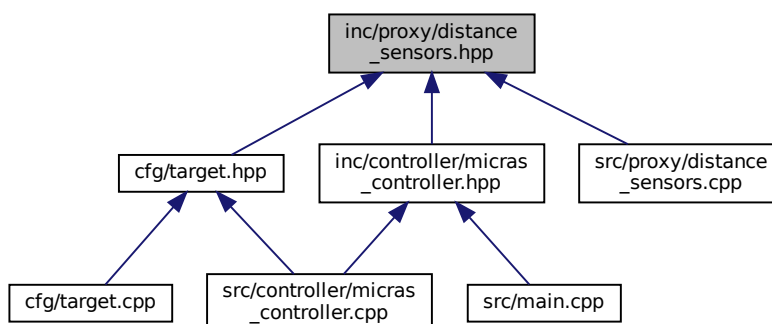
## 8.20 inc/proxy/distance\_sensors.hpp File Reference

Proxy DistanceSensors class header.

```
#include "hal/adc_dma.hpp"
#include "hal/pwm.hpp"
#include "../src/proxy/distance_sensors.cpp"
Include dependency graph for distance_sensors.hpp:
```



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



### Classes

- class `proxy::DistanceSensors< num_of_sensors >`  
Class for controlling *DistanceSensors*.
- struct `proxy::DistanceSensors< num_of_sensors >::Config`  
Configuration structure for distance sensors.

## Namespaces

- [proxy](#)

### 8.20.1 Detailed Description

Proxy DistanceSensors class header.

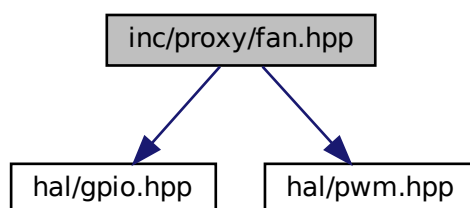
Date

03/2024

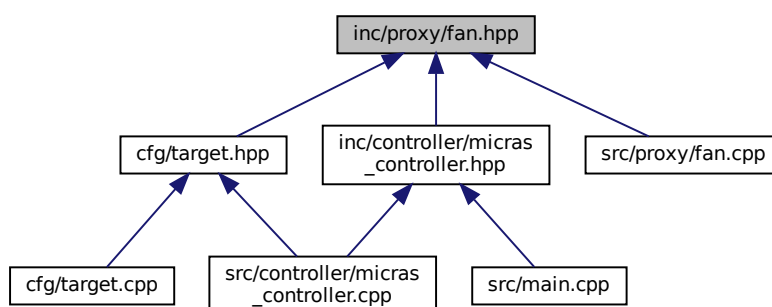
## 8.21 inc/proxy/fan.hpp File Reference

Proxy Fan class declaration.

```
#include "hal/gpio.hpp"  
#include "hal/pwm.hpp"  
Include dependency graph for fan.hpp:
```



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



## Classes

- class [proxy::Fan](#)  
*Class for controlling the fan driver.*
- struct [proxy::Fan::Config](#)  
*Configuration structure for the fan.*

## Namespaces

- [proxy](#)

### 8.21.1 Detailed Description

Proxy Fan class declaration.

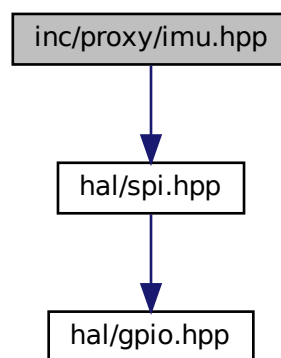
Date

03/2024

## 8.22 inc/proxy/imu.hpp File Reference

STM32 IMU HAL wrapper.

```
#include "hal/spi.hpp"  
Include dependency graph for imu.hpp:
```

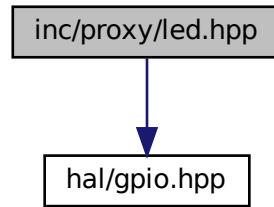




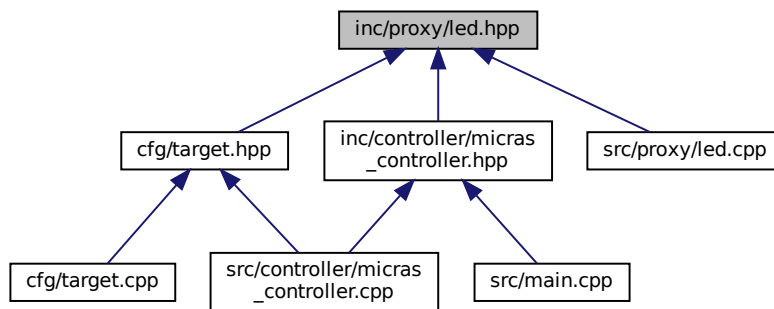


```
#include "hal/gpio.hpp"
```

Include dependency graph for led.hpp:



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



## Classes

- class [proxy::Led](#)  
*Class for controlling an LED.*
- struct [proxy::Led::Config](#)  
*Configuration structure for LED.*

## Namespaces

- [proxy](#)

### 8.23.1 Detailed Description

Proxy Led class header.

Date

03/2024

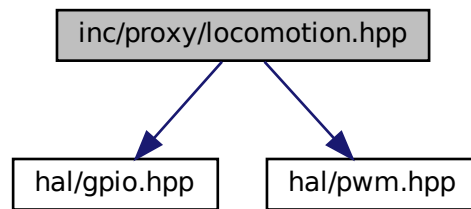
## 8.24 inc/proxy/locomotion.hpp File Reference

Proxy Locomotion class declaration.

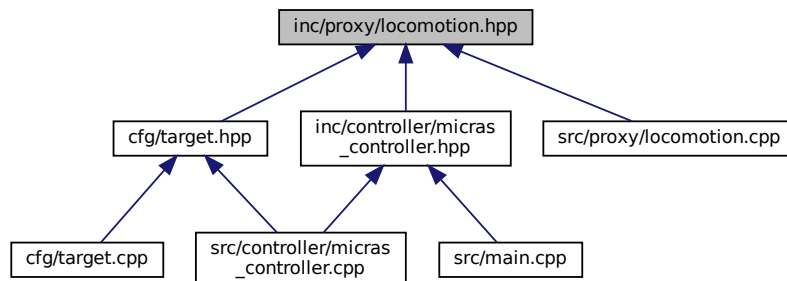
```
#include "hal/gpio.hpp"
```

```
#include "hal/pwm.hpp"
```

Include dependency graph for locomotion.hpp:



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



### Classes

- class [proxy::Locomotion](#)  
*Class for controlling the locomotion driver.*
- struct [proxy::Locomotion::Config](#)  
*Configuration structure for the locomotion.*

### Namespaces

- [proxy](#)

### 8.24.1 Detailed Description

Proxy Locomotion class declaration.

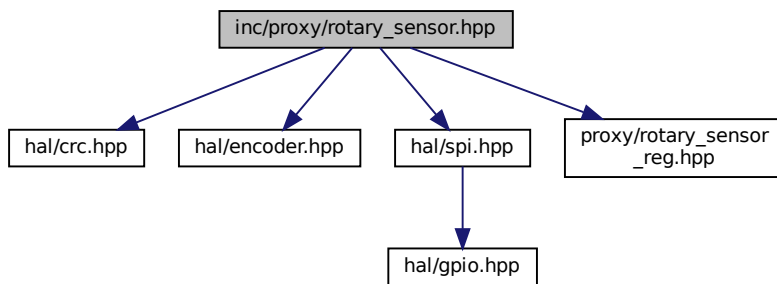
Date

03/2024

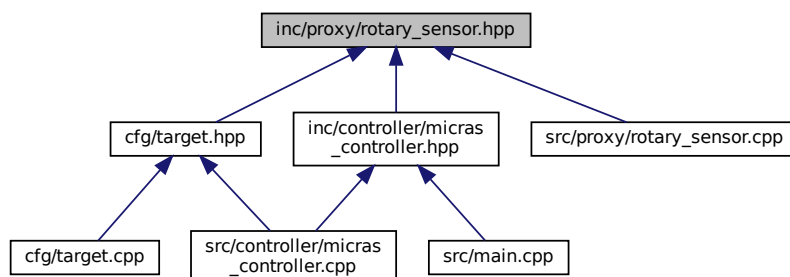
## 8.25 inc/proxy/rotary\_sensor.hpp File Reference

STM32 rotary sensor HAL wrapper.

```
#include "hal/crc.hpp"
#include "hal/encoder.hpp"
#include "hal/spi.hpp"
#include "proxy/rotary_sensor_reg.hpp"
Include dependency graph for rotary_sensor.hpp:
```



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



## Classes

- class [proxy::RotarySensor](#)  
*Class to handle rotary sensor peripheral on STM32 microcontrollers.*
- struct [proxy::RotarySensor::Config](#)  
*Rotary sensor configuration struct.*
- struct [proxy::RotarySensor::CommandFrame::Fields](#)
- struct [proxy::RotarySensor::DataFrame::Fields](#)

## Namespaces

- [proxy](#)

### 8.25.1 Detailed Description

STM32 rotary sensor HAL wrapper.

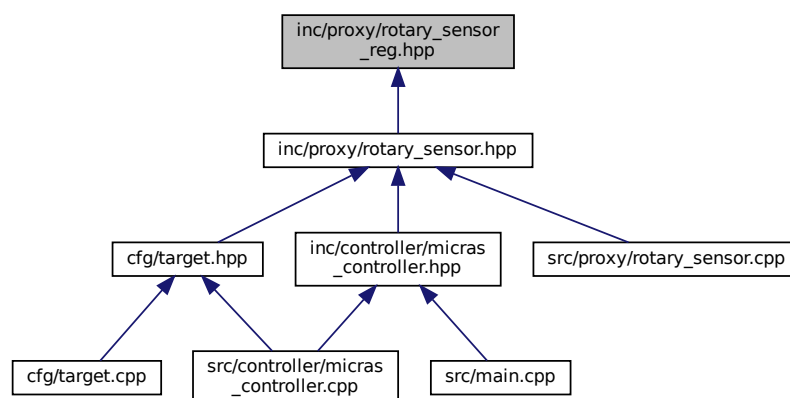
Date

03/2024

## 8.26 inc/proxy/rotary\_sensor\_reg.hpp File Reference

AS5047U rotary sensor registers definition.

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



## Classes

- struct [Registers](#)  
*[Registers](#) class for the rotary sensor configuration.*
- union [Registers::Disable](#)  
*[Registers](#) union types definition.*
- struct [Registers::Disable::Fields](#)
- union [Registers::Zposm](#)
- struct [Registers::Zposm::Fields](#)
- union [Registers::Zposl](#)
- struct [Registers::Zposl::Fields](#)
- union [Registers::Settings1](#)
- struct [Registers::Settings1::Fields](#)
- union [Registers::Settings2](#)
- struct [Registers::Settings2::Fields](#)
- union [Registers::Settings3](#)
- struct [Registers::Settings3::Fields](#)
- union [Registers::Ecc](#)
- struct [Registers::Ecc::Fields](#)

### 8.26.1 Detailed Description

AS5047U rotary sensor registers definition.

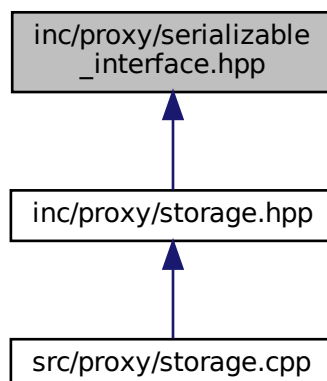
Date

03/2024

## 8.27 inc/proxy/serializable\_interface.hpp File Reference

Serializable interface for all classes that need to be serialized.

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



## Classes

- class [ISerializable](#)  
*Interface class for serializable classes.*

### 8.27.1 Detailed Description

Serializable interface for all classes that need to be serialized.

Date

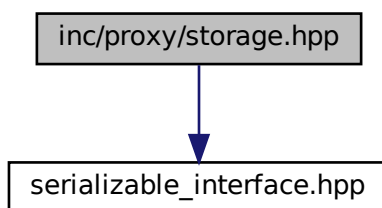
03/2024

## 8.28 inc/proxy/storage.hpp File Reference

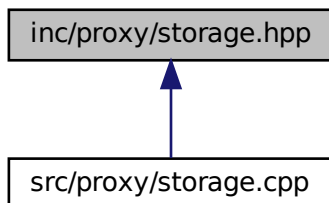
Proxy Storage class declaration.

```
#include "serializable_interface.hpp"
```

Include dependency graph for storage.hpp:



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



## Classes

- class `proxy::Storage`  
*Class for controlling the storage.*
- struct `proxy::Storage::Config`  
*Configuration structure for the storage.*

## Namespaces

- `proxy`

## Variables

- `template<typename T>`  
`concept proxy::Fundamental = std::is_fundamental<T>::value`

### 8.28.1 Detailed Description

Proxy Storage class declaration.

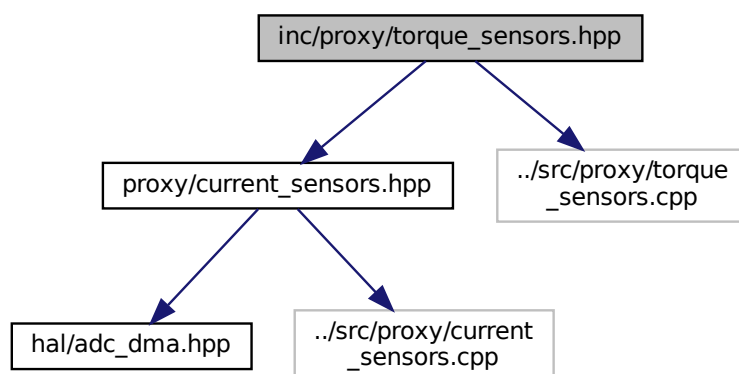
Date

03/2024

## 8.29 inc/proxy/torque\_sensors.hpp File Reference

Proxy TorqueSensors class header.

```
#include "proxy/current_sensors.hpp"  
#include "../src/proxy/torque_sensors.cpp"  
Include dependency graph for torque_sensors.hpp:
```







### 8.31.1 Detailed Description

Micras Controller class implementation.

Date

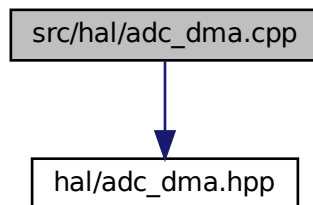
03/2024

## 8.32 src/hal/adc\_dma.cpp File Reference

STM32 ADC DMA HAL wrapper.

```
#include "hal/adc_dma.hpp"
```

Include dependency graph for adc\_dma.cpp:



### Namespaces

- [hal](#)

### 8.32.1 Detailed Description

STM32 ADC DMA HAL wrapper.

Date

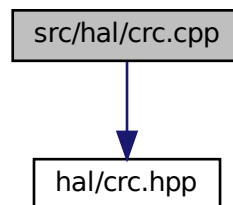
03/2024

## 8.33 src/hal/crc.cpp File Reference

STM32 CRC HAL wrapper.

```
#include "hal/crc.hpp"
```

Include dependency graph for crc.cpp:



### Namespaces

- [hal](#)

#### 8.33.1 Detailed Description

STM32 CRC HAL wrapper.

Date

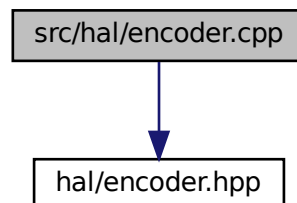
03/2024

## 8.34 src/hal/encoder.cpp File Reference

STM32 encoder HAL wrapper.

```
#include "hal/encoder.hpp"
```

Include dependency graph for encoder.cpp:



## Namespaces

- [hal](#)

### 8.34.1 Detailed Description

STM32 encoder HAL wrapper.

Date

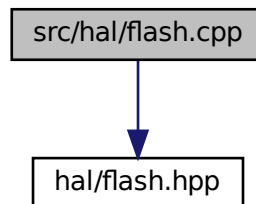
03/2024

## 8.35 src/hal/flash.cpp File Reference

STM32 flash HAL wrapper.

```
#include "hal/flash.hpp"
```

Include dependency graph for flash.cpp:



## Namespaces

- [hal](#)

### 8.35.1 Detailed Description

STM32 flash HAL wrapper.

Date

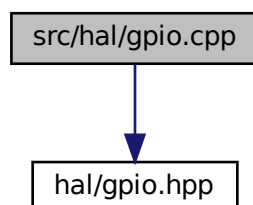
03/2024

## 8.36 src/hal/gpio.cpp File Reference

HAL GPIO class source.

```
#include "hal/gpio.hpp"
```

Include dependency graph for gpio.cpp:



### Namespaces

- [hal](#)

#### 8.36.1 Detailed Description

HAL GPIO class source.

Date

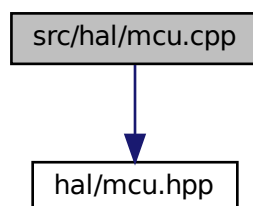
03/2024

## 8.37 src/hal/mcu.cpp File Reference

MCU related.

```
#include "hal/mcu.hpp"
```

Include dependency graph for mcu.cpp:



## Namespaces

- [hal](#)

## Functions

- void [SystemClock\\_Config](#) ()  
*Initializes System Clock.*

### 8.37.1 Detailed Description

MCU related.

### 8.37.2 Function Documentation

#### 8.37.2.1 SystemClock\_Config()

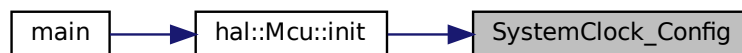
```
void SystemClock_Config ( )
```

Initializes System Clock.

#### Note

Defined by cube

Here is the caller graph for this function:

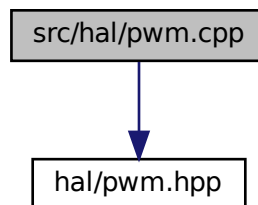


## 8.38 src/hal/pwm.cpp File Reference

STM32 PWM HAL wrapper.

```
#include "hal/pwm.hpp"
```

Include dependency graph for pwm.cpp:



### Namespaces

- [hal](#)

#### 8.38.1 Detailed Description

STM32 PWM HAL wrapper.

Date

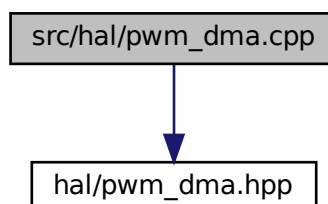
03/2024

## 8.39 src/hal/pwm\_dma.cpp File Reference

STM32 PWM DMA HAL wrapper.

```
#include "hal/pwm_dma.hpp"
```

Include dependency graph for pwm\_dma.cpp:



## Namespaces

- [hal](#)

### 8.39.1 Detailed Description

STM32 PWM DMA HAL wrapper.

Date

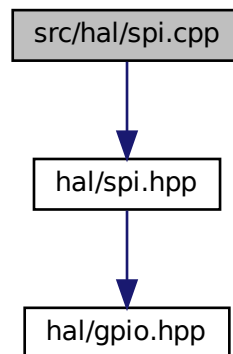
03/2024

## 8.40 src/hal/spi.cpp File Reference

Proxy SPI Switch class source.

```
#include "hal/spi.hpp"
```

Include dependency graph for spi.cpp:



## Namespaces

- [hal](#)

### 8.40.1 Detailed Description

Proxy SPI Switch class source.

Date

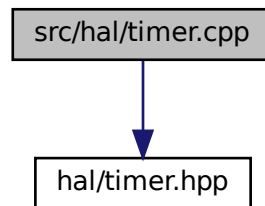
03/2024



# STM32 TIM HAL wrapper.

```
#include "hal/timer.hpp"
```

Include dependency graph for timer.cpp:



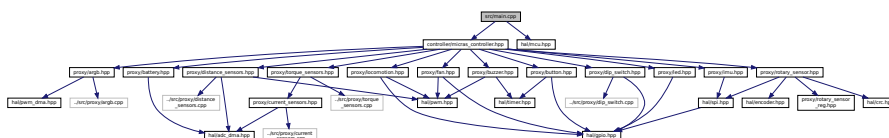
- hal

# STM32 TIM HAL wrapper.

03/2024

Main function.

```
#include "controller/micras_controller.hpp"
#include "hal/mcu.hpp"
Include dependency graph for main.cpp:
```



## Functions

- int `main` ()

### 8.42.1 Detailed Description

Main function.

Date

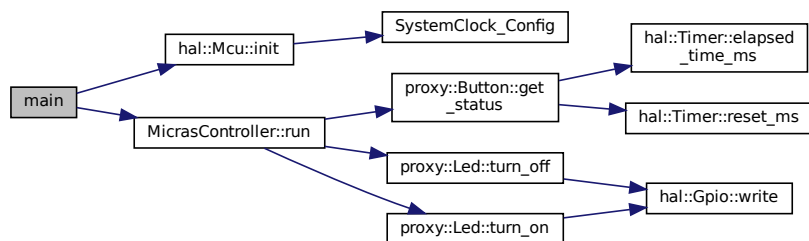
03/2024

### 8.42.2 Function Documentation

#### 8.42.2.1 `main()`

```
int main ( )
```

Here is the call graph for this function:

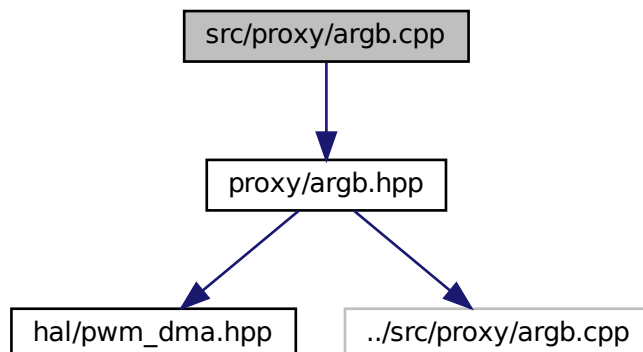


## 8.43 `src/proxy/argb.cpp` File Reference

Proxy Argb class implementation.

```
#include "proxy/argb.hpp"
```

Include dependency graph for argb.cpp:



## Namespaces

- [proxy](#)

## Macros

- `#define` [MICRAS\\_PROXY\\_ARGB\\_CPP](#)

### 8.43.1 Detailed Description

Proxy Argb class implementation.

Date

03/2024

### 8.43.2 Macro Definition Documentation

#### 8.43.2.1 MICRAS\_PROXY\_ARGB\_CPP

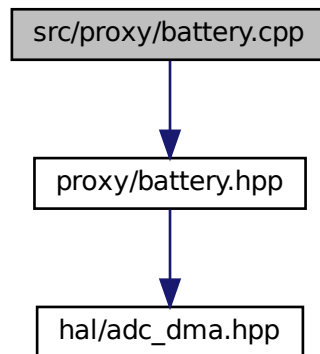
```
#define MICRAS_PROXY_ARGB_CPP
```

## 8.44 src/proxy/battery.cpp File Reference

Proxy Battery class implementation.

```
#include "proxy/battery.hpp"
```

Include dependency graph for battery.cpp:



### Namespaces

- [proxy](#)

### 8.44.1 Detailed Description

Proxy Battery class implementation.

Date

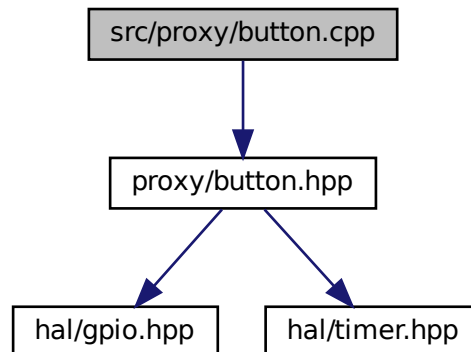
03/2024

## 8.45 src/proxy/button.cpp File Reference

Proxy Button class source.

```
#include "proxy/button.hpp"
```

Include dependency graph for button.cpp:



## Namespaces

- [proxy](#)

### 8.45.1 Detailed Description

Proxy Button class source.

Date

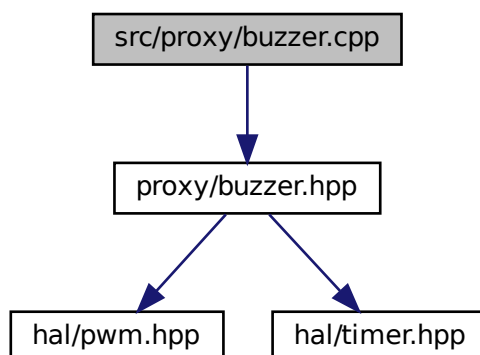
03/2024

## 8.46 src/proxy/buzzer.cpp File Reference

Proxy Buzzer class implementation.

```
#include "proxy/buzzer.hpp"
```

Include dependency graph for buzzer.cpp:



## Namespaces

- [proxy](#)

### 8.46.1 Detailed Description

Proxy Buzzer class implementation.

Date

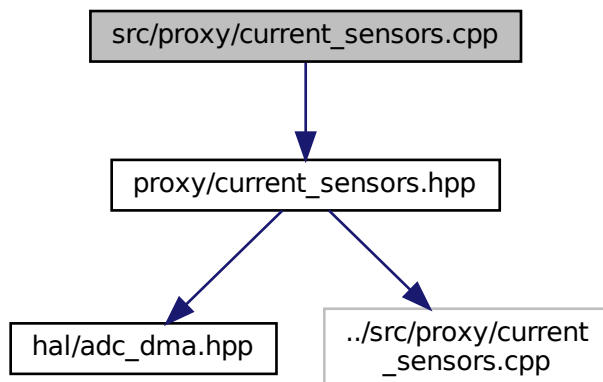
03/2024

## 8.47 `src/proxy/current_sensors.cpp` File Reference

Proxy CurrentSensors class implementation.

```
#include "proxy/current_sensors.hpp"
```

Include dependency graph for current\_sensors.cpp:



## Namespaces

- [proxy](#)

## Macros

- `#define MICRAS_PROXY_CURRENT_SENSORS_CPP`

### 8.47.1 Detailed Description

Proxy CurrentSensors class implementation.

Date

03/2024

### 8.47.2 Macro Definition Documentation

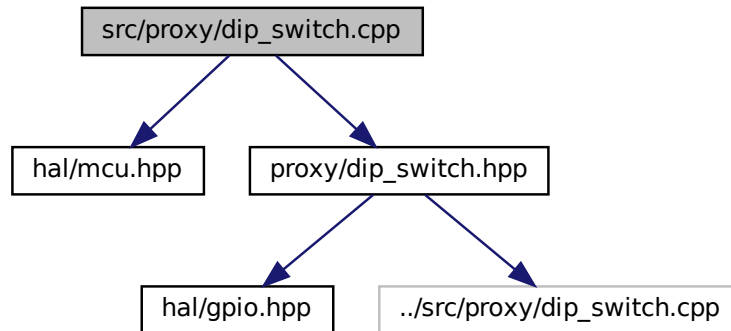
#### 8.47.2.1 MICRAS\_PROXY\_CURRENT\_SENSORS\_CPP

```
#define MICRAS_PROXY_CURRENT_SENSORS_CPP
```

## 8.48 src/proxy/dip\_switch.cpp File Reference

Proxy DIP Switch class source.

```
#include "hal/mcu.hpp"  
#include "proxy/dip_switch.hpp"  
Include dependency graph for dip_switch.cpp:
```



### Namespaces

- [proxy](#)

### Macros

- `#define MICRAS_PROXY_DIP_SWITCH_CPP`

#### 8.48.1 Detailed Description

Proxy DIP Switch class source.

Date

03/2024

#### 8.48.2 Macro Definition Documentation

##### 8.48.2.1 MICRAS\_PROXY\_DIP\_SWITCH\_CPP

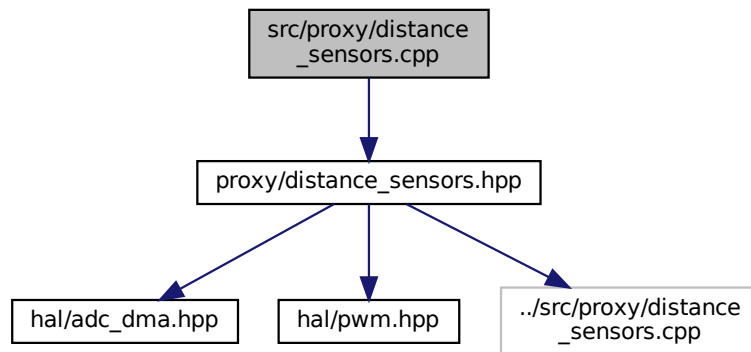
```
#define MICRAS_PROXY_DIP_SWITCH_CPP
```



## 8.49 src/proxy/distance\_sensors.cpp File Reference

```
#include "proxy/distance_sensors.hpp"
```

Include dependency graph for distance\_sensors.cpp:



### Namespaces

- [proxy](#)

### Macros

- `#define MICRAS_PROXY_DISTANCE_SENSORS_CPP`

#### 8.49.1 Macro Definition Documentation

##### 8.49.1.1 MICRAS\_PROXY\_DISTANCE\_SENSORS\_CPP

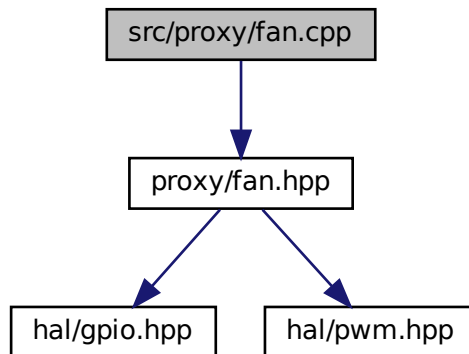
```
#define MICRAS_PROXY_DISTANCE_SENSORS_CPP
```

## 8.50 src/proxy/fan.cpp File Reference

Proxy Fan class source.

```
#include "proxy/fan.hpp"
```

Include dependency graph for fan.cpp:



### Namespaces

- [proxy](#)

### 8.50.1 Detailed Description

Proxy Fan class source.

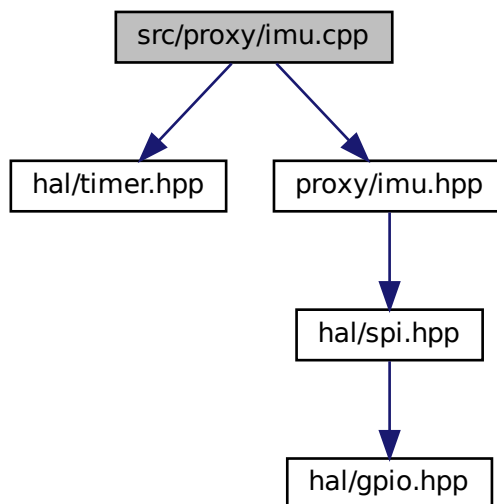
Date

03/2024

## 8.51 src/proxy/imu.cpp File Reference

Proxy Imu class source.

```
#include "hal/timer.hpp"
#include "proxy/imu.hpp"
Include dependency graph for imu.cpp:
```



## Namespaces

- [proxy](#)

### 8.51.1 Detailed Description

Proxy Imu class source.

Date

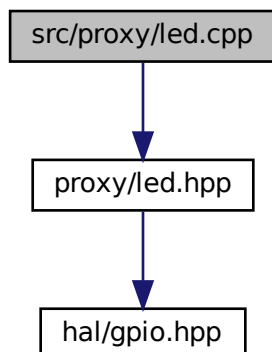
03/2024

## 8.52 src/proxy/led.cpp File Reference

Proxy Led class source.

```
#include "proxy/led.hpp"
```

Include dependency graph for led.cpp:



## Namespaces

- [proxy](#)

### 8.52.1 Detailed Description

Proxy Led class source.

Date

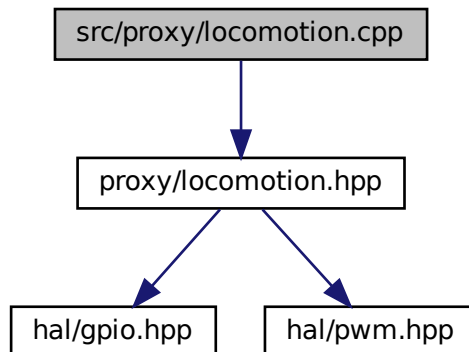
03/2024

## 8.53 src/proxy/locomotion.cpp File Reference

Proxy Locomotion class source.

```
#include "proxy/locomotion.hpp"
```

Include dependency graph for locomotion.cpp:



## Namespaces

- [proxy](#)

### 8.53.1 Detailed Description

Proxy Locomotion class source.

Date

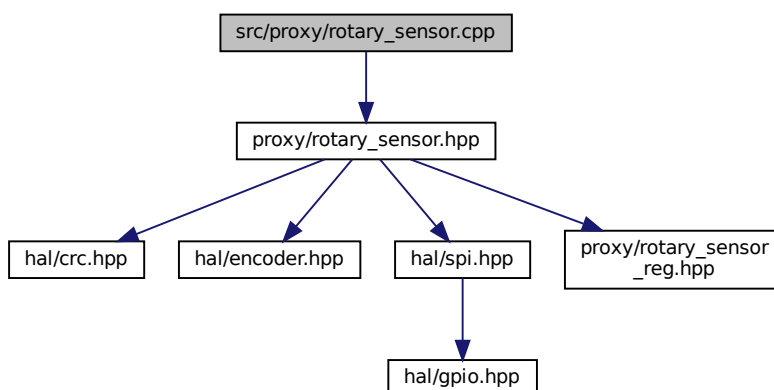
03/2024

## 8.54 src/proxy/rotary\_sensor.cpp File Reference

Proxy RotarySensor class source.

```
#include "proxy/rotary_sensor.hpp"
```

Include dependency graph for rotary\_sensor.cpp:



## Namespaces

- [proxy](#)

### 8.54.1 Detailed Description

Proxy RotarySensor class source.

Date

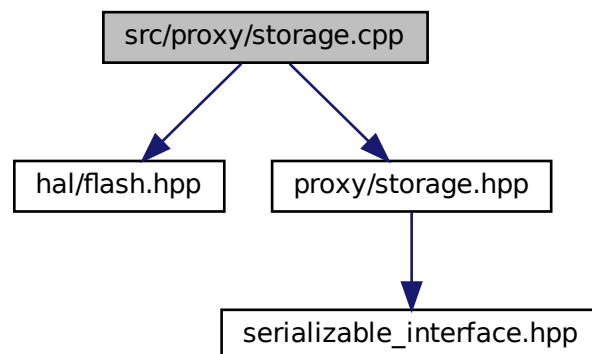
03/2024

## 8.55 src/proxy/storage.cpp File Reference

Proxy Storage class source.

```
#include "hal/flash.hpp"
#include "proxy/storage.hpp"
```

Include dependency graph for storage.cpp:



## Namespaces

- [proxy](#)

### 8.55.1 Detailed Description

Proxy Storage class source.

Date

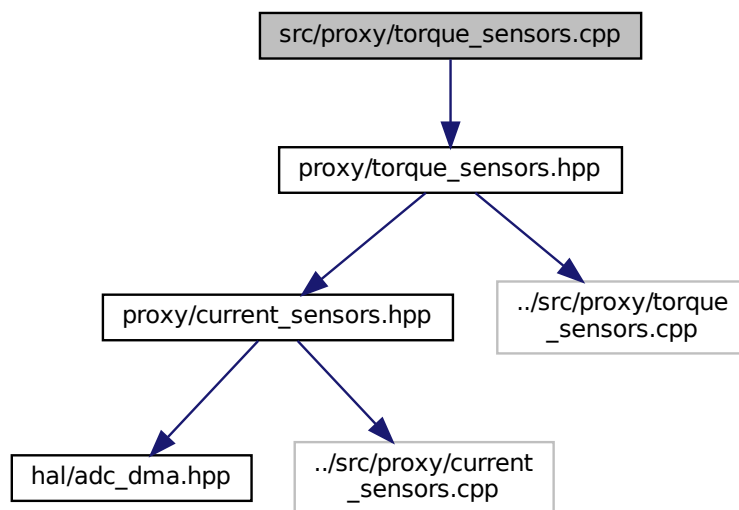
03/2024

## 8.56 src/proxy/torque\_sensors.cpp File Reference

Proxy TorqueSensors class implementation.

```
#include "proxy/torque_sensors.hpp"
```

Include dependency graph for torque\_sensors.cpp:



## Namespaces

- [proxy](#)

## Macros

- `#define` [MICRAS\\_PROXY\\_TORQUE\\_SENSORS\\_CPP](#)

### 8.56.1 Detailed Description

Proxy TorqueSensors class implementation.

Date

03/2024

### 8.56.2 Macro Definition Documentation

#### 8.56.2.1 MICRAS\_PROXY\_TORQUE\_SENSORS\_CPP

```
#define MICRAS_PROXY_TORQUE_SENSORS_CPP
```



# Index

- ~ISerializable
  - ISerializable, [107](#)
- ABI\_DEC
  - Registers::Settings2::Fields, [91](#)
- ABI\_off
  - Registers::Disable::Fields, [87](#)
- ABIRES
  - Registers::Settings3::Fields, [93](#)
- accelerometer\_data\_rate
  - proxy::Imu::Config, [56](#)
- accelerometer\_filter
  - proxy::Imu::Config, [56](#)
- accelerometer\_scale
  - proxy::Imu::Config, [56](#)
- adc
  - proxy::Battery::Config, [47](#)
  - proxy::CurrentSensors< num\_of\_sensors >::Config, [51](#)
  - proxy::DistanceSensors< num\_of\_sensors >::Config, [54](#)
- AdcDma
  - hal::AdcDma, [18](#)
- address
  - proxy::RotarySensor::CommandFrame::Fields, [84](#)
- Argb
  - proxy::Argb< num\_of\_leds >, [21](#)
- argb.cpp
  - MICRAS\_PROXY\_ARGB\_CPP, [201](#)
- argb\_config
  - target.cpp, [156](#)
  - target.hpp, [161](#)
- Axis
  - proxy::Imu, [104](#)
- BACKWARD
  - proxy::Fan, [81](#)
- Battery
  - proxy::Battery, [24](#)
- battery\_config
  - target.cpp, [156](#)
  - target.hpp, [161](#)
- blue
  - proxy::Argb< num\_of\_leds >::Color, [33](#)
- Button
  - proxy::Button, [28](#)
- button\_config
  - target.cpp, [156](#)
  - target.hpp, [161](#)
- Buzzer
  - proxy::Buzzer, [30](#)
- buzzer\_config
  - target.cpp, [156](#)
  - target.hpp, [161](#)
- calculate
  - hal::Crc, [65](#)
- cfg/target.cpp, [155](#)
- cfg/target.hpp, [160](#)
- Crc
  - hal::Crc, [65](#)
- crc
  - proxy::RotarySensor::CommandFrame::Fields, [84](#)
  - proxy::RotarySensor::Config, [61](#)
  - proxy::RotarySensor::DataFrame::Fields, [86](#)
- create
  - proxy::Storage, [141](#), [142](#)
- current\_sensors
  - proxy::TorqueSensors< num\_of\_sensors >::Config, [64](#)
- current\_sensors.cpp
  - MICRAS\_PROXY\_CURRENT\_SENSORS\_CPP, [205](#)
- CurrentSensors
  - proxy::CurrentSensors< num\_of\_sensors >, [67](#)
- DAECDIS
  - Registers::Settings2::Fields, [91](#)
- data
  - proxy::RotarySensor::DataFrame::Fields, [86](#)
- Data\_select
  - Registers::Settings2::Fields, [92](#)
- debounce\_delay
  - proxy::Button::Config, [48](#)
- deserialize
  - ISerializable, [108](#)
- Dia1\_en
  - Registers::Zposl::Fields, [94](#)
- Dia2\_en
  - Registers::Zposl::Fields, [95](#)
- Dia3\_en
  - Registers::Settings1::Fields, [90](#)
- Dia4\_en
  - Registers::Settings1::Fields, [90](#)
- dip\_switch.cpp
  - MICRAS\_PROXY\_DIP\_SWITCH\_CPP, [206](#)
- dip\_switch\_config
  - target.cpp, [156](#)
  - target.hpp, [162](#)
- DipSwitch

- proxy::DipSwitch< num\_of\_switches >, 70
- DIR
  - Registers::Settings2::Fields, 92
- direction\_gpio
  - proxy::Fan::Config, 55
- disable
  - proxy::Fan, 81
  - proxy::Locomotion, 114
  - Registers, 128
- disable\_addr
  - Registers, 128
- distance\_sensors.cpp
  - MICRAS\_PROXY\_DISTANCE\_SENSORS\_CPP, 207
- distance\_sensors\_config
  - target.cpp, 157
  - target.hpp, 162
- DistanceSensors
  - proxy::DistanceSensors< num\_of\_sensors >, 74
- do\_not\_care
  - proxy::RotarySensor::CommandFrame::Fields, 84
- ecc
  - Registers, 129
- ecc\_addr
  - Registers, 129
- ECC\_chsum
  - Registers::Ecc::Fields, 88
- ECC\_en
  - Registers::Ecc::Fields, 89
- elapsed\_time\_ms
  - hal::Timer, 145
- elapsed\_time\_us
  - hal::Timer, 146
- enable
  - proxy::Fan, 82
  - proxy::Locomotion, 115
- enable\_gpio
  - proxy::Fan::Config, 55
  - proxy::Locomotion::Config, 59
- Encoder
  - hal::Encoder, 78
- encoder
  - proxy::RotarySensor::Config, 61
- erase\_pages
  - hal::Flash, 97
- error
  - proxy::RotarySensor::DataFrame::Fields, 86
- EXTRA\_LONG\_PRESS
  - proxy::Button, 28
- extra\_long\_press\_delay
  - proxy::Button::Config, 48
- Fan
  - proxy::Fan, 81
- fan\_config
  - target.cpp, 157
  - target.hpp, 162
- Fields
  - Registers::Zposl, 151
- fields
  - Registers::Disable, 72
  - Registers::Ecc, 77
  - Registers::Settings1, 133
  - Registers::Settings2, 135
  - Registers::Settings3, 136
  - Registers::Zposm, 153
- FILTER\_disable
  - Registers::Disable::Fields, 87
- Flash
  - hal::Flash, 97
- FORWARD
  - proxy::Fan, 81
- Fundamental
  - proxy, 16
- get\_angular\_velocity
  - proxy::Imu, 105
- get\_compare
  - hal::PwmDma, 126
- get\_counter
  - hal::Encoder, 79
- get\_current
  - proxy::CurrentSensors< num\_of\_sensors >, 67
  - proxy::TorqueSensors< num\_of\_sensors >, 149
- get\_current\_raw
  - proxy::CurrentSensors< num\_of\_sensors >, 68
- get\_distance
  - proxy::DistanceSensors< num\_of\_sensors >, 75
- get\_distance\_raw
  - proxy::DistanceSensors< num\_of\_sensors >, 75
- get\_linear\_acceleration
  - proxy::Imu, 105
- get\_orientation
  - proxy::Imu, 106
- get\_position
  - proxy::RotarySensor, 132
- get\_status
  - proxy::Button, 28
- get\_switch\_state
  - proxy::DipSwitch< num\_of\_switches >, 70
- get\_switches\_value
  - proxy::DipSwitch< num\_of\_switches >, 71
- get\_torque
  - proxy::TorqueSensors< num\_of\_sensors >, 150
- get\_voltage
  - proxy::Battery, 25
- get\_voltage\_raw
  - proxy::Battery, 25
- Gpio
  - hal::Gpio, 101
- gpio
  - hal::Spi::Config, 43
  - proxy::Button::Config, 49
  - proxy::Led::Config, 58
- gpio\_array
  - proxy::DipSwitch< num\_of\_switches >::Config, 53
- green

- proxy::Argb< num\_of\_leds >::Color, 34
- gyroscope\_data\_rate
  - proxy::Imu::Config, 56
- gyroscope\_filter
  - proxy::Imu::Config, 56
- gyroscope\_scale
  - proxy::Imu::Config, 57
- hal, 15
- hal::AdcDma, 17
  - AdcDma, 18
  - max\_reading, 19
  - reference\_voltage, 19
  - start\_dma, 18
  - stop\_dma, 19
- hal::AdcDma::Config, 34
  - handle, 35
  - init\_function, 35
  - max\_reading, 35
  - reference\_voltage, 35
- hal::Crc, 64
  - calculate, 65
  - Crc, 65
- hal::Crc::Config, 36
  - handle, 36
- hal::Encoder, 78
  - Encoder, 78
  - get\_counter, 79
- hal::Encoder::Config, 37
  - handle, 37
  - init\_function, 37
  - timer\_channel, 38
- hal::Flash, 96
  - erase\_pages, 97
  - Flash, 97
  - read, 97, 98
  - write, 99
- hal::Gpio, 100
  - Gpio, 101
  - read, 101
  - toggle, 102
  - write, 102
- hal::Gpio::Config, 38
  - pin, 39
  - port, 39
- hal::Mcu, 119
  - init, 120
  - Mcu, 120
- hal::Pwm, 122
  - Pwm, 123
  - set\_duty\_cycle, 124
  - set\_frequency, 124
- hal::Pwm::Config, 39
  - handle, 40
  - init\_function, 40
  - timer\_channel, 40
- hal::PwmDma, 125
  - get\_compare, 126
  - PwmDma, 126
  - start\_dma, 127
  - stop\_dma, 127
- hal::PwmDma::Config, 41
  - handle, 41
  - init\_function, 41
  - timer\_channel, 42
- hal::Spi, 137
  - receive, 138
  - select\_device, 138
  - Spi, 138
  - transmit, 139
  - unselect\_device, 139
- hal::Spi::Config, 42
  - gpio, 43
  - handle, 43
  - init\_function, 43
  - timeout, 43
- hal::Timer, 144
  - elapsed\_time\_ms, 145
  - elapsed\_time\_us, 146
  - reset\_ms, 146
  - reset\_us, 146
  - sleep\_ms, 147
  - sleep\_us, 147
  - Timer, 145
- hal::Timer::Config, 44
  - handle, 44
  - init\_function, 44
- handle
  - hal::AdcDma::Config, 35
  - hal::Crc::Config, 36
  - hal::Encoder::Config, 37
  - hal::Pwm::Config, 40
  - hal::PwmDma::Config, 41
  - hal::Spi::Config, 43
  - hal::Timer::Config, 44
- HYS
  - Registers::Settings3::Fields, 93
- Imu
  - proxy::Imu, 104
- imu\_config
  - target.cpp, 157
  - target.hpp, 162
- inc/controller/micras\_controller.hpp, 163
- inc/hal/adc\_dma.hpp, 164
- inc/hal/crc.hpp, 165
- inc/hal/encoder.hpp, 166
- inc/hal/flash.hpp, 167
- inc/hal/gpio.hpp, 167
- inc/hal/mcu.hpp, 168
- inc/hal/pwm.hpp, 169
- inc/hal/pwm\_dma.hpp, 170
- inc/hal/spi.hpp, 171
- inc/hal/timer.hpp, 172
- inc/proxy/argb.hpp, 173
- inc/proxy/battery.hpp, 174
- inc/proxy/button.hpp, 175
- inc/proxy/buzzer.hpp, 176

- inc/proxy/current\_sensors.hpp, 177
- inc/proxy/dip\_switch.hpp, 179
- inc/proxy/distance\_sensors.hpp, 180
- inc/proxy/fan.hpp, 181
- inc/proxy/imu.hpp, 182
- inc/proxy/led.hpp, 183
- inc/proxy/locomotion.hpp, 185
- inc/proxy/rotary\_sensor.hpp, 186
- inc/proxy/rotary\_sensor\_reg.hpp, 187
- inc/proxy/serializable\_interface.hpp, 188
- inc/proxy/storage.hpp, 189
- inc/proxy/torque\_sensors.hpp, 190
- init
  - hal::Mcu, 120
- init\_function
  - hal::AdcDma::Config, 35
  - hal::Encoder::Config, 37
  - hal::Pwm::Config, 40
  - hal::PwmDma::Config, 41
  - hal::Spi::Config, 43
  - hal::Timer::Config, 44
- is\_pressed
  - proxy::Button, 29
- ISerializable, 107
  - ~ISerializable, 107
  - deserialize, 108
  - ISerializable, 108
  - operator=, 109
  - serialize, 109
- IWIDTH
  - Registers::Settings2::Fields, 92
- K\_max
  - Registers::Settings1::Fields, 90
- K\_min
  - Registers::Settings1::Fields, 90
- Led
  - proxy::Led, 110
- led\_config
  - target.cpp, 158
  - target.hpp, 162
- led\_pwm
  - proxy::DistanceSensors< num\_of\_sensors >::Config, 54
- Locomotion
  - proxy::Locomotion, 114
- locomotion\_config
  - target.cpp, 158
  - target.hpp, 162
- LONG\_PRESS
  - proxy::Button, 28
- long\_press\_delay
  - proxy::Button::Config, 49
- main
  - main.cpp, 200
- main.cpp
  - main, 200
- max\_distance
  - proxy::DistanceSensors< num\_of\_sensors >::Config, 54
- max\_reading
  - hal::AdcDma, 19
  - hal::AdcDma::Config, 35
- max\_torque
  - proxy::TorqueSensors< num\_of\_sensors >::Config, 64
- Mcu
  - hal::Mcu, 120
- mcu.cpp
  - SystemClock\_Config, 196
- MICRAS\_PROXY\_ARGB\_CPP
  - argb.cpp, 201
- MICRAS\_PROXY\_CURRENT\_SENSORS\_CPP
  - current\_sensors.cpp, 205
- MICRAS\_PROXY\_DIP\_SWITCH\_CPP
  - dip\_switch.cpp, 206
- MICRAS\_PROXY\_DISTANCE\_SENSORS\_CPP
  - distance\_sensors.cpp, 207
- MICRAS\_PROXY\_TORQUE\_SENSORS\_CPP
  - torque\_sensors.cpp, 214
- MicrasController, 121
  - MicrasController, 121
  - run, 121
- na
  - Registers::Disable::Fields, 87
- NO\_PRESS
  - proxy::Button, 28
- NOISESET
  - Registers::Settings2::Fields, 92
- number\_of\_pages
  - proxy::Storage::Config, 62
- operator=
  - ISerializable, 109
- orientation\_data\_rate
  - proxy::Imu::Config, 57
- pin
  - hal::Gpio::Config, 39
- play
  - proxy::Buzzer, 31
- port
  - hal::Gpio::Config, 39
- proxy, 15
  - Fundamental, 16
- proxy::Argb< num\_of\_leds >, 20
  - Argb, 21
  - set\_color, 21, 22
  - turn\_off, 22
- proxy::Argb< num\_of\_leds >::Color, 33
  - blue, 33
  - green, 34
  - red, 34
- proxy::Argb< num\_of\_leds >::Config, 45
  - pwm, 46

- proxy::Battery, 23
  - Battery, 24
  - get\_voltage, 25
  - get\_voltage\_raw, 25
- proxy::Battery::Config, 46
  - adc, 47
  - voltage\_divider, 47
- proxy::Button, 26
  - Button, 28
  - EXTRA\_LONG\_PRESS, 28
  - get\_status, 28
  - is\_pressed, 29
  - LONG\_PRESS, 28
  - NO\_PRESS, 28
  - PULL\_DOWN, 28
  - PULL\_UP, 28
  - PullResistor, 27
  - SHORT\_PRESS, 28
  - Status, 28
- proxy::Button::Config, 47
  - debounce\_delay, 48
  - extra\_long\_press\_delay, 48
  - gpio, 49
  - long\_press\_delay, 49
  - pull\_resistor, 49
- proxy::Buzzer, 30
  - Buzzer, 30
  - play, 31
  - stop, 31
  - update, 32
- proxy::Buzzer::Config, 49
  - pwm, 50
- proxy::CurrentSensors< num\_of\_sensors >, 66
  - CurrentSensors, 67
  - get\_current, 67
  - get\_current\_raw, 68
- proxy::CurrentSensors< num\_of\_sensors >::Config, 51
  - adc, 51
  - shunt\_resistor, 51
- proxy::DipSwitch< num\_of\_switches >, 68
  - DipSwitch, 70
  - get\_switch\_state, 70
  - get\_switches\_value, 71
- proxy::DipSwitch< num\_of\_switches >::Config, 52
  - gpio\_array, 53
- proxy::DistanceSensors< num\_of\_sensors >, 73
  - DistanceSensors, 74
  - get\_distance, 75
  - get\_distance\_raw, 75
  - set\_led\_intensity, 76
- proxy::DistanceSensors< num\_of\_sensors >::Config, 53
  - adc, 54
  - led\_pwm, 54
  - max\_distance, 54
- proxy::Fan, 79
  - BACKWARD, 81
  - disable, 81
  - enable, 82
  - Fan, 81
  - FORWARD, 81
  - RotationDirection, 80
  - set\_speed, 82
  - stop, 83
- proxy::Fan::Config, 54
  - direction\_gpio, 55
  - enable\_gpio, 55
  - pwm, 55
- proxy::Imu, 103
  - Axis, 104
  - get\_angular\_velocity, 105
  - get\_linear\_acceleration, 105
  - get\_orientation, 106
  - Imu, 104
  - update\_data, 106
  - W, 104
  - X, 104
  - Y, 104
  - Z, 104
- proxy::Imu::Config, 55
  - accelerometer\_data\_rate, 56
  - accelerometer\_filter, 56
  - accelerometer\_scale, 56
  - gyroscope\_data\_rate, 56
  - gyroscope\_filter, 56
  - gyroscope\_scale, 57
  - orientation\_data\_rate, 57
  - spi, 57
- proxy::Led, 110
  - Led, 110
  - toggle, 111
  - turn\_off, 111
  - turn\_on, 112
- proxy::Led::Config, 57
  - gpio, 58
- proxy::Locomotion, 113
  - disable, 114
  - enable, 115
  - Locomotion, 114
  - set\_speed, 115
  - set\_wheel\_speed, 116
  - stop, 117
  - stop\_left, 117
  - stop\_right, 118
- proxy::Locomotion::Config, 59
  - enable\_gpio, 59
  - pwm\_left\_bwd, 59
  - pwm\_left\_fwd, 60
  - pwm\_right\_bwd, 60
  - pwm\_right\_fwd, 60
- proxy::RotarySensor, 131
  - get\_position, 132
  - RotarySensor, 131
- proxy::RotarySensor::CommandFrame::Fields, 84
  - address, 84
  - crc, 84

- do\_not\_care, [84](#)
- rw, [85](#)
- proxy::RotarySensor::Config, [60](#)
  - crc, [61](#)
  - encoder, [61](#)
  - registers, [61](#)
  - resolution, [61](#)
  - spi, [61](#)
- proxy::RotarySensor::DataFrame::Fields, [85](#)
  - crc, [86](#)
  - data, [86](#)
  - error, [86](#)
  - warning, [86](#)
- proxy::Storage, [140](#)
  - create, [141](#), [142](#)
  - save, [142](#)
  - Storage, [141](#)
  - sync, [143](#)
- proxy::Storage::Config, [62](#)
  - number\_of\_pages, [62](#)
  - start\_page, [62](#)
- proxy::TorqueSensors< num\_of\_sensors >, [148](#)
  - get\_current, [149](#)
  - get\_torque, [150](#)
  - TorqueSensors, [149](#)
- proxy::TorqueSensors< num\_of\_sensors >::Config, [63](#)
  - current\_sensors, [64](#)
  - max\_torque, [64](#)
- PULL\_DOWN
  - proxy::Button, [28](#)
- pull\_resistor
  - proxy::Button::Config, [49](#)
- PULL\_UP
  - proxy::Button, [28](#)
- PullResistor
  - proxy::Button, [27](#)
- Pwm
  - hal::Pwm, [123](#)
- pwm
  - proxy::Argb< num\_of\_leds >::Config, [46](#)
  - proxy::Buzzer::Config, [50](#)
  - proxy::Fan::Config, [55](#)
- pwm\_left\_bwd
  - proxy::Locomotion::Config, [59](#)
- pwm\_left\_fwd
  - proxy::Locomotion::Config, [60](#)
- pwm\_right\_bwd
  - proxy::Locomotion::Config, [60](#)
- pwm\_right\_fwd
  - proxy::Locomotion::Config, [60](#)
- PwmDma
  - hal::PwmDma, [126](#)
- PWMon
  - Registers::Settings2::Fields, [92](#)
- raw
  - Registers::Disable, [73](#)
  - Registers::Ecc, [77](#)
  - Registers::Settings1, [133](#)
  - Registers::Settings2, [135](#)
  - Registers::Settings3, [136](#)
  - Registers::Zposl, [151](#)
  - Registers::Zposm, [153](#)
- read
  - hal::Flash, [97](#), [98](#)
  - hal::Gpio, [101](#)
- README.md, [191](#)
- receive
  - hal::Spi, [138](#)
- red
  - proxy::Argb< num\_of\_leds >::Color, [34](#)
- reference\_voltage
  - hal::AdcDma, [19](#)
  - hal::AdcDma::Config, [35](#)
- Registers, [127](#)
  - disable, [128](#)
  - disable\_addr, [128](#)
  - ecc, [129](#)
  - ecc\_addr, [129](#)
  - settings1, [129](#)
  - settings1\_addr, [129](#)
  - settings2, [129](#)
  - settings2\_addr, [129](#)
  - settings3, [129](#)
  - settings3\_addr, [130](#)
  - zposl, [130](#)
  - zposl\_addr, [130](#)
  - zposm, [130](#)
  - zposm\_addr, [130](#)
- registers
  - proxy::RotarySensor::Config, [61](#)
- Registers::Disable, [71](#)
  - fields, [72](#)
  - raw, [73](#)
- Registers::Disable::Fields, [86](#)
  - ABI\_off, [87](#)
  - FILTER\_disable, [87](#)
  - na, [87](#)
  - UVW\_off, [88](#)
- Registers::Ecc, [76](#)
  - fields, [77](#)
  - raw, [77](#)
- Registers::Ecc::Fields, [88](#)
  - ECC\_chsum, [88](#)
  - ECC\_en, [89](#)
- Registers::Settings1, [132](#)
  - fields, [133](#)
  - raw, [133](#)
- Registers::Settings1::Fields, [89](#)
  - Dia3\_en, [90](#)
  - Dia4\_en, [90](#)
  - K\_max, [90](#)
  - K\_min, [90](#)
- Registers::Settings2, [134](#)
  - fields, [135](#)
  - raw, [135](#)
- Registers::Settings2::Fields, [90](#)

- ABI\_DEC, [91](#)
- DAECDIS, [91](#)
- Data\_select, [92](#)
- DIR, [92](#)
- IWIDTH, [92](#)
- NOISESET, [92](#)
- PWMon, [92](#)
- UVW\_ABI, [92](#)
- Registers::Settings3, [135](#)
  - fields, [136](#)
  - raw, [136](#)
- Registers::Settings3::Fields, [93](#)
  - ABIRES, [93](#)
  - HYS, [93](#)
  - UVWPP, [93](#)
- Registers::Zposl, [150](#)
  - Fields, [151](#)
  - raw, [151](#)
- Registers::Zposl::Fields, [94](#)
  - Dia1\_en, [94](#)
  - Dia2\_en, [95](#)
  - ZPOSL, [95](#)
- Registers::Zposm, [152](#)
  - fields, [153](#)
  - raw, [153](#)
- Registers::Zposm::Fields, [95](#)
  - ZPOSM, [96](#)
- reset\_ms
  - hal::Timer, [146](#)
- reset\_us
  - hal::Timer, [146](#)
- resolution
  - proxy::RotarySensor::Config, [61](#)
- rotary\_sensor\_left\_config
  - target.cpp, [158](#)
  - target.hpp, [162](#)
- rotary\_sensor\_reg\_config
  - target.cpp, [159](#)
- rotary\_sensor\_right\_config
  - target.cpp, [159](#)
  - target.hpp, [162](#)
- RotarySensor
  - proxy::RotarySensor, [131](#)
- RotationDirection
  - proxy::Fan, [80](#)
- run
  - MicrasController, [121](#)
- rw
  - proxy::RotarySensor::CommandFrame::Fields, [85](#)
- save
  - proxy::Storage, [142](#)
- select\_device
  - hal::Spi, [138](#)
- serialize
  - ISerializable, [109](#)
- set\_color
  - proxy::Argb< num\_of\_leds >, [21](#), [22](#)
- set\_duty\_cycle
  - hal::Pwm, [124](#)
- set\_frequency
  - hal::Pwm, [124](#)
- set\_led\_intensity
  - proxy::DistanceSensors< num\_of\_sensors >, [76](#)
- set\_speed
  - proxy::Fan, [82](#)
  - proxy::Locomotion, [115](#)
- set\_wheel\_speed
  - proxy::Locomotion, [116](#)
- settings1
  - Registers, [129](#)
- settings1\_addr
  - Registers, [129](#)
- settings2
  - Registers, [129](#)
- settings2\_addr
  - Registers, [129](#)
- settings3
  - Registers, [129](#)
- settings3\_addr
  - Registers, [130](#)
- SHORT\_PRESS
  - proxy::Button, [28](#)
- shunt\_resistor
  - proxy::CurrentSensors< num\_of\_sensors >::Config, [51](#)
- sleep\_ms
  - hal::Timer, [147](#)
- sleep\_us
  - hal::Timer, [147](#)
- Spi
  - hal::Spi, [138](#)
- spi
  - proxy::Imu::Config, [57](#)
  - proxy::RotarySensor::Config, [61](#)
- src/controller/micras\_controller.cpp, [191](#)
- src/hal/adc\_dma.cpp, [192](#)
- src/hal/crc.cpp, [193](#)
- src/hal/encoder.cpp, [193](#)
- src/hal/flash.cpp, [194](#)
- src/hal/gpio.cpp, [195](#)
- src/hal/mcu.cpp, [195](#)
- src/hal/pwm.cpp, [197](#)
- src/hal/pwm\_dma.cpp, [197](#)
- src/hal/spi.cpp, [198](#)
- src/hal/timer.cpp, [199](#)
- src/main.cpp, [199](#)
- src/proxy/argb.cpp, [200](#)
- src/proxy/battery.cpp, [202](#)
- src/proxy/button.cpp, [202](#)
- src/proxy/buzzer.cpp, [203](#)
- src/proxy/current\_sensors.cpp, [204](#)
- src/proxy/dip\_switch.cpp, [206](#)
- src/proxy/distance\_sensors.cpp, [207](#)
- src/proxy/fan.cpp, [208](#)
- src/proxy/imu.cpp, [208](#)
- src/proxy/led.cpp, [209](#)

- src/proxy/locomotion.cpp, 210
- src/proxy/rotary\_sensor.cpp, 211
- src/proxy/storage.cpp, 212
- src/proxy/torque\_sensors.cpp, 213
- start\_dma
  - hal::AdcDma, 18
  - hal::PwmDma, 127
- start\_page
  - proxy::Storage::Config, 62
- Status
  - proxy::Button, 28
- stop
  - proxy::Buzzer, 31
  - proxy::Fan, 83
  - proxy::Locomotion, 117
- stop\_dma
  - hal::AdcDma, 19
  - hal::PwmDma, 127
- stop\_left
  - proxy::Locomotion, 117
- stop\_right
  - proxy::Locomotion, 118
- Storage
  - proxy::Storage, 141
- sync
  - proxy::Storage, 143
- SystemClock\_Config
  - mcu.cpp, 196
- target.cpp
  - argb\_config, 156
  - battery\_config, 156
  - button\_config, 156
  - buzzer\_config, 156
  - dip\_switch\_config, 156
  - distance\_sensors\_config, 157
  - fan\_config, 157
  - imu\_config, 157
  - led\_config, 158
  - locomotion\_config, 158
  - rotary\_sensor\_left\_config, 158
  - rotary\_sensor\_reg\_config, 159
  - rotary\_sensor\_right\_config, 159
  - torque\_sensors\_config, 159
- target.hpp
  - argb\_config, 161
  - battery\_config, 161
  - button\_config, 161
  - buzzer\_config, 161
  - dip\_switch\_config, 162
  - distance\_sensors\_config, 162
  - fan\_config, 162
  - imu\_config, 162
  - led\_config, 162
  - locomotion\_config, 162
  - rotary\_sensor\_left\_config, 162
  - rotary\_sensor\_right\_config, 162
  - torque\_sensors\_config, 163
- timeout
  - hal::Spi::Config, 43
- Timer
  - hal::Timer, 145
- timer\_channel
  - hal::Encoder::Config, 38
  - hal::Pwm::Config, 40
  - hal::PwmDma::Config, 42
- toggle
  - hal::Gpio, 102
  - proxy::Led, 111
- torque\_sensors.cpp
  - MICRAS\_PROXY\_TORQUE\_SENSORS\_CPP, 214
- torque\_sensors\_config
  - target.cpp, 159
  - target.hpp, 163
- TorqueSensors
  - proxy::TorqueSensors< num\_of\_sensors >, 149
- transmit
  - hal::Spi, 139
- turn\_off
  - proxy::Argb< num\_of\_leds >, 22
  - proxy::Led, 111
- turn\_on
  - proxy::Led, 112
- unselect\_device
  - hal::Spi, 139
- update
  - proxy::Buzzer, 32
- update\_data
  - proxy::Imu, 106
- UVW\_ABI
  - Registers::Settings2::Fields, 92
- UVW\_off
  - Registers::Disable::Fields, 88
- UVWPP
  - Registers::Settings3::Fields, 93
- voltage\_divider
  - proxy::Battery::Config, 47
- W
  - proxy::Imu, 104
- warning
  - proxy::RotarySensor::DataFrame::Fields, 86
- write
  - hal::Flash, 99
  - hal::Gpio, 102
- X
  - proxy::Imu, 104
- Y
  - proxy::Imu, 104
- Z
  - proxy::Imu, 104
- ZPOS
  - Registers::Zposl::Fields, 95



zposl  
    Registers, [130](#)  
zposl\_addr  
    Registers, [130](#)  
ZPOSM  
    Registers::Zposm::Fields, [96](#)  
zposm  
    Registers, [130](#)  
zposm\_addr  
    Registers, [130](#)