

# Firmware documentation

v. 6.1.5

Generated by Doxygen 1.8.13



# Contents

|          |                                     |          |
|----------|-------------------------------------|----------|
| <b>1</b> | <b>Firmware</b>                     | <b>1</b> |
| <b>2</b> | <b>Data Structure Index</b>         | <b>3</b> |
| 2.1      | Data Structures . . . . .           | 3        |
| <b>3</b> | <b>File Index</b>                   | <b>5</b> |
| 3.1      | File List . . . . .                 | 5        |
| <b>4</b> | <b>Data Structure Documentation</b> | <b>7</b> |
| 4.1      | st_calib Struct Reference . . . . . | 7        |
| 4.1.1    | Detailed Description . . . . .      | 7        |
| 4.1.2    | Field Documentation . . . . .       | 7        |
| 4.1.2.1  | direction . . . . .                 | 7        |
| 4.1.2.2  | enabled . . . . .                   | 7        |
| 4.1.2.3  | repetitions . . . . .               | 8        |
| 4.1.2.4  | speed . . . . .                     | 8        |
| 4.2      | st_data Struct Reference . . . . .  | 8        |
| 4.2.1    | Detailed Description . . . . .      | 8        |
| 4.2.2    | Field Documentation . . . . .       | 8        |
| 4.2.2.1  | buffer . . . . .                    | 8        |
| 4.2.2.2  | ind . . . . .                       | 9        |
| 4.2.2.3  | length . . . . .                    | 9        |
| 4.2.2.4  | ready . . . . .                     | 9        |
| 4.3      | st_dev Struct Reference . . . . .   | 9        |
| 4.3.1    | Detailed Description . . . . .      | 9        |

|          |                                    |    |
|----------|------------------------------------|----|
| 4.3.2    | Field Documentation . . . . .      | 9  |
| 4.3.2.1  | pwm_limit . . . . .                | 10 |
| 4.3.2.2  | tension . . . . .                  | 10 |
| 4.3.2.3  | tension_conv_factor . . . . .      | 10 |
| 4.3.2.4  | tension_valid . . . . .            | 10 |
| 4.4      | st_meas Struct Reference . . . . . | 10 |
| 4.4.1    | Detailed Description . . . . .     | 10 |
| 4.4.2    | Field Documentation . . . . .      | 11 |
| 4.4.2.1  | acc . . . . .                      | 11 |
| 4.4.2.2  | curr . . . . .                     | 11 |
| 4.4.2.3  | emg . . . . .                      | 11 |
| 4.4.2.4  | pos . . . . .                      | 11 |
| 4.4.2.5  | rot . . . . .                      | 11 |
| 4.4.2.6  | vel . . . . .                      | 11 |
| 4.5      | st_mem Struct Reference . . . . .  | 12 |
| 4.5.1    | Detailed Description . . . . .     | 13 |
| 4.5.2    | Field Documentation . . . . .      | 13 |
| 4.5.2.1  | activ . . . . .                    | 13 |
| 4.5.2.2  | activate_pwm_rescaling . . . . .   | 13 |
| 4.5.2.3  | baud_rate . . . . .                | 13 |
| 4.5.2.4  | control_mode . . . . .             | 13 |
| 4.5.2.5  | curr_lookup . . . . .              | 13 |
| 4.5.2.6  | current_hist . . . . .             | 14 |
| 4.5.2.7  | current_limit . . . . .            | 14 |
| 4.5.2.8  | double_encoder_on_off . . . . .    | 14 |
| 4.5.2.9  | emg_calibration_flag . . . . .     | 14 |
| 4.5.2.10 | emg_counter . . . . .              | 14 |
| 4.5.2.11 | emg_max_value . . . . .            | 14 |
| 4.5.2.12 | emg_speed . . . . .                | 14 |
| 4.5.2.13 | emg_threshold . . . . .            | 14 |

|          |                              |    |
|----------|------------------------------|----|
| 4.5.2.14 | flag . . . . .               | 15 |
| 4.5.2.15 | id . . . . .                 | 15 |
| 4.5.2.16 | input_mode . . . . .         | 15 |
| 4.5.2.17 | k_d . . . . .                | 15 |
| 4.5.2.18 | k_d_c . . . . .              | 15 |
| 4.5.2.19 | k_d_c_dl . . . . .           | 15 |
| 4.5.2.20 | k_d_dl . . . . .             | 15 |
| 4.5.2.21 | k_i . . . . .                | 15 |
| 4.5.2.22 | k_i_c . . . . .              | 16 |
| 4.5.2.23 | k_i_c_dl . . . . .           | 16 |
| 4.5.2.24 | k_i_dl . . . . .             | 16 |
| 4.5.2.25 | k_p . . . . .                | 16 |
| 4.5.2.26 | k_p_c . . . . .              | 16 |
| 4.5.2.27 | k_p_c_dl . . . . .           | 16 |
| 4.5.2.28 | k_p_dl . . . . .             | 16 |
| 4.5.2.29 | m_mult . . . . .             | 16 |
| 4.5.2.30 | m_off . . . . .              | 17 |
| 4.5.2.31 | maint_day . . . . .          | 17 |
| 4.5.2.32 | maint_month . . . . .        | 17 |
| 4.5.2.33 | maint_year . . . . .         | 17 |
| 4.5.2.34 | max_step_neg . . . . .       | 17 |
| 4.5.2.35 | max_step_pos . . . . .       | 17 |
| 4.5.2.36 | motor_handle_ratio . . . . . | 17 |
| 4.5.2.37 | pos_lim_flag . . . . .       | 17 |
| 4.5.2.38 | pos_lim_inf . . . . .        | 18 |
| 4.5.2.39 | pos_lim_sup . . . . .        | 18 |
| 4.5.2.40 | position_hist . . . . .      | 18 |
| 4.5.2.41 | res . . . . .                | 18 |
| 4.5.2.42 | rest_counter . . . . .       | 18 |
| 4.5.2.43 | rest_delay . . . . .         | 18 |

|          |   |           |
|----------|---|-----------|
| 4.5.2.44 | rest_pos . . . . .                            | 18        |
| 4.5.2.45 | rest_position_flag . . . . .                  | 18        |
| 4.5.2.46 | rest_ratio . . . . .                          | 19        |
| 4.5.2.47 | rest_vel . . . . .                            | 19        |
| 4.5.2.48 | switch_emg . . . . .                          | 19        |
| 4.5.2.49 | total_time_on . . . . .                       | 19        |
| 4.5.2.50 | total_time_rest . . . . .                     | 19        |
| 4.5.2.51 | watchdog_period . . . . .                     | 19        |
| 4.5.2.52 | wire_disp . . . . .                           | 19        |
| 4.6      | st_ref Struct Reference . . . . .             | 20        |
| 4.6.1    | Detailed Description . . . . .                | 20        |
| 4.6.2    | Field Documentation . . . . .                 | 20        |
| 4.6.2.1  | curr . . . . .                                | 20        |
| 4.6.2.2  | onoff . . . . .                               | 20        |
| 4.6.2.3  | pos . . . . .                                 | 20        |
| 4.6.2.4  | pwm . . . . .                                 | 20        |
| <b>5</b> | <b>File Documentation</b>                     | <b>21</b> |
| 5.1      | command_processing.c File Reference . . . . . | 21        |
| 5.1.1    | Detailed Description . . . . .                | 22        |
| 5.1.2    | Function Documentation . . . . .              | 23        |
| 5.1.2.1  | cmd_activate() . . . . .                      | 23        |
| 5.1.2.2  | cmd_get_accelerations() . . . . .             | 23        |
| 5.1.2.3  | cmd_get_activate() . . . . .                  | 23        |
| 5.1.2.4  | cmd_get_curr_and_meas() . . . . .             | 23        |
| 5.1.2.5  | cmd_get_currents() . . . . .                  | 23        |
| 5.1.2.6  | cmd_get_currents_for_cuff() . . . . .         | 23        |
| 5.1.2.7  | cmd_get_emg() . . . . .                       | 23        |
| 5.1.2.8  | cmd_get_inputs() . . . . .                    | 24        |
| 5.1.2.9  | cmd_get_measurements() . . . . .              | 24        |
| 5.1.2.10 | cmd_get_velocities() . . . . .                | 24        |

|          |                                     |    |
|----------|-------------------------------------|----|
| 5.1.2.11 | cmd_ping()                          | 24 |
| 5.1.2.12 | cmd_set_baudrate()                  | 24 |
| 5.1.2.13 | cmd_set_inputs()                    | 24 |
| 5.1.2.14 | cmd_set_watchdog()                  | 24 |
| 5.1.2.15 | cmd_store_params()                  | 25 |
| 5.1.2.16 | commProcess()                       | 25 |
| 5.1.2.17 | commWrite()                         | 25 |
| 5.1.2.18 | commWrite_old_id()                  | 25 |
| 5.1.2.19 | commWrite_to_cuff()                 | 25 |
| 5.1.2.20 | get_param_list()                    | 26 |
| 5.1.2.21 | infoGet()                           | 26 |
| 5.1.2.22 | infoSend()                          | 26 |
| 5.1.2.23 | LCRChecksum()                       | 27 |
| 5.1.2.24 | memInit()                           | 27 |
| 5.1.2.25 | memRecall()                         | 27 |
| 5.1.2.26 | memRestore()                        | 27 |
| 5.1.2.27 | memStore()                          | 27 |
| 5.1.2.28 | prepare_counter_info()              | 28 |
| 5.1.2.29 | prepare_generic_info()              | 28 |
| 5.1.2.30 | sendAcknowledgment()                | 28 |
| 5.1.2.31 | setZeros()                          | 29 |
| 5.2      | command_processing.h File Reference | 29 |
| 5.2.1    | Detailed Description                | 30 |
| 5.2.2    | Function Documentation              | 31 |
| 5.2.2.1  | cmd_activate()                      | 31 |
| 5.2.2.2  | cmd_get_accelerations()             | 31 |
| 5.2.2.3  | cmd_get_activate()                  | 31 |
| 5.2.2.4  | cmd_get_curr_and_meas()             | 31 |
| 5.2.2.5  | cmd_get_currents()                  | 31 |
| 5.2.2.6  | cmd_get_currents_for_cuff()         | 31 |

|          |                                |    |
|----------|--------------------------------|----|
| 5.2.2.7  | cmd_get_emg()                  | 31 |
| 5.2.2.8  | cmd_get_inputs()               | 32 |
| 5.2.2.9  | cmd_get_measurements()         | 32 |
| 5.2.2.10 | cmd_get_velocities()           | 32 |
| 5.2.2.11 | cmd_ping()                     | 32 |
| 5.2.2.12 | cmd_set_baudrate()             | 32 |
| 5.2.2.13 | cmd_set_inputs()               | 32 |
| 5.2.2.14 | cmd_set_watchdog()             | 32 |
| 5.2.2.15 | cmd_store_params()             | 33 |
| 5.2.2.16 | commProcess()                  | 33 |
| 5.2.2.17 | commWrite()                    | 33 |
| 5.2.2.18 | commWrite_old_id()             | 33 |
| 5.2.2.19 | commWrite_to_cuff()            | 33 |
| 5.2.2.20 | get_param_list()               | 34 |
| 5.2.2.21 | infoGet()                      | 34 |
| 5.2.2.22 | infoSend()                     | 34 |
| 5.2.2.23 | LCRChecksum()                  | 35 |
| 5.2.2.24 | memInit()                      | 35 |
| 5.2.2.25 | memRecall()                    | 35 |
| 5.2.2.26 | memRestore()                   | 35 |
| 5.2.2.27 | memStore()                     | 35 |
| 5.2.2.28 | prepare_counter_info()         | 36 |
| 5.2.2.29 | prepare_generic_info()         | 36 |
| 5.2.2.30 | sendAcknowledgment()           | 36 |
| 5.2.2.31 | setZeros()                     | 37 |
| 5.3      | commands.h File Reference      | 37 |
| 5.3.1    | Detailed Description           | 39 |
| 5.3.2    | Enumeration Type Documentation | 39 |
| 5.3.2.1  | qbmove_command                 | 39 |
| 5.3.2.2  | qbmove_control_mode            | 40 |



|          |                                    |    |
|----------|------------------------------------|----|
| 5.3.2.3  | qbmove_input_mode . . . . .        | 40 |
| 5.3.2.4  | qbmove_parameter . . . . .         | 41 |
| 5.4      | globals.c File Reference . . . . . | 42 |
| 5.4.1    | Detailed Description . . . . .     | 43 |
| 5.4.2    | Variable Documentation . . . . .   | 43 |
| 5.4.2.1  | c_mem . . . . .                    | 43 |
| 5.4.2.2  | calib . . . . .                    | 43 |
| 5.4.2.3  | can_write . . . . .                | 43 |
| 5.4.2.4  | cycle_time . . . . .               | 44 |
| 5.4.2.5  | cycles_interrupt_flag . . . . .    | 44 |
| 5.4.2.6  | cycles_status . . . . .            | 44 |
| 5.4.2.7  | dev_pwm_limit . . . . .            | 44 |
| 5.4.2.8  | dev_pwm_sat . . . . .              | 44 |
| 5.4.2.9  | dev_tension . . . . .              | 44 |
| 5.4.2.10 | dev_tension_f . . . . .            | 44 |
| 5.4.2.11 | emg_1_status . . . . .             | 44 |
| 5.4.2.12 | emg_2_status . . . . .             | 45 |
| 5.4.2.13 | forced_open . . . . .              | 45 |
| 5.4.2.14 | g_measOld . . . . .                | 45 |
| 5.4.2.15 | g_refOld . . . . .                 | 45 |
| 5.4.2.16 | g_rx . . . . .                     | 45 |
| 5.4.2.17 | interrupt_flag . . . . .           | 45 |
| 5.4.2.18 | pow_tension . . . . .              | 45 |
| 5.4.2.19 | pwm_sign . . . . .                 | 45 |
| 5.4.2.20 | reset_last_value_flag . . . . .    | 46 |
| 5.4.2.21 | rest_enabled . . . . .             | 46 |
| 5.4.2.22 | rest_pos_curr_ref . . . . .        | 46 |
| 5.4.2.23 | tau_feedback . . . . .             | 46 |
| 5.4.2.24 | tension_valid . . . . .            | 46 |
| 5.4.2.25 | timer_value . . . . .              | 46 |

|          |  |    |
|----------|--|----|
| 5.4.2.26 | timer_value0 . . . . .                   | 46 |
| 5.4.2.27 | watchdog_flag . . . . .                  | 46 |
| 5.5      | globals.h File Reference . . . . .       | 47 |
| 5.5.1    | Detailed Description . . . . .           | 49 |
| 5.5.2    | Macro Definition Documentation . . . . . | 50 |
| 5.5.2.1  | ANTI_WINDUP . . . . .                    | 50 |
| 5.5.2.2  | CALIBRATION_DIV . . . . .                | 50 |
| 5.5.2.3  | COUNTER_INC . . . . .                    | 50 |
| 5.5.2.4  | CURR_INTEGRAL_SAT_LIMIT . . . . .        | 50 |
| 5.5.2.5  | CURRENT_HYSTERESIS . . . . .             | 50 |
| 5.5.2.6  | DEFAULT_CURRENT_LIMIT . . . . .          | 50 |
| 5.5.2.7  | DEFAULT_EEPROM_DISPLACEMENT . . . . .    | 50 |
| 5.5.2.8  | DIV_INIT_VALUE . . . . .                 | 51 |
| 5.5.2.9  | EEPROM_BYTES_ROW . . . . .               | 51 |
| 5.5.2.10 | EEPROM_COUNTERS_ROWS . . . . .           | 51 |
| 5.5.2.11 | EMG_SAMPLE_TO_DISCARD . . . . .          | 51 |
| 5.5.2.12 | LOOKUP_DIM . . . . .                     | 51 |
| 5.5.2.13 | MAX_WATCHDOG_TIMER . . . . .             | 51 |
| 5.5.2.14 | NUM_OF_ANALOG_INPUTS . . . . .           | 51 |
| 5.5.2.15 | NUM_OF_EMGS . . . . .                    | 51 |
| 5.5.2.16 | NUM_OF_MOTORS . . . . .                  | 52 |
| 5.5.2.17 | NUM_OF_PARAMS . . . . .                  | 52 |
| 5.5.2.18 | NUM_OF_SENSORS . . . . .                 | 52 |
| 5.5.2.19 | POS_INTEGRAL_SAT_LIMIT . . . . .         | 52 |
| 5.5.2.20 | PWM_MAX_VALUE . . . . .                  | 52 |
| 5.5.2.21 | RECEIVE . . . . .                        | 52 |
| 5.5.2.22 | SAMPLES_FOR_EMG_MEAN . . . . .           | 52 |
| 5.5.2.23 | SAMPLES_FOR_MEAN . . . . .               | 52 |
| 5.5.2.24 | STATE_ACTIVE . . . . .                   | 53 |
| 5.5.2.25 | STATE_INACTIVE . . . . .                 | 53 |

|          |  |    |
|----------|--|----|
| 5.5.2.26 | UNLOAD . . . . .                         | 53 |
| 5.5.2.27 | WAIT_ID . . . . .                        | 53 |
| 5.5.2.28 | WAIT_LENGTH . . . . .                    | 53 |
| 5.5.2.29 | WAIT_START . . . . .                     | 53 |
| 5.5.3    | Enumeration Type Documentation . . . . . | 53 |
| 5.5.3.1  | counter_status . . . . .                 | 53 |
| 5.5.3.2  | emg_status . . . . .                     | 54 |
| 5.5.4    | Variable Documentation . . . . .         | 54 |
| 5.5.4.1  | c_mem . . . . .                          | 54 |
| 5.5.4.2  | calib . . . . .                          | 54 |
| 5.5.4.3  | can_write . . . . .                      | 54 |
| 5.5.4.4  | cycle_time . . . . .                     | 55 |
| 5.5.4.5  | cycles_interrupt_flag . . . . .          | 55 |
| 5.5.4.6  | cycles_status . . . . .                  | 55 |
| 5.5.4.7  | dev_pwm_limit . . . . .                  | 55 |
| 5.5.4.8  | dev_pwm_sat . . . . .                    | 55 |
| 5.5.4.9  | dev_tension . . . . .                    | 55 |
| 5.5.4.10 | dev_tension_f . . . . .                  | 55 |
| 5.5.4.11 | emg_1_status . . . . .                   | 56 |
| 5.5.4.12 | emg_2_status . . . . .                   | 56 |
| 5.5.4.13 | forced_open . . . . .                    | 56 |
| 5.5.4.14 | g_measOld . . . . .                      | 56 |
| 5.5.4.15 | g_refOld . . . . .                       | 56 |
| 5.5.4.16 | g_rx . . . . .                           | 56 |
| 5.5.4.17 | interrupt_flag . . . . .                 | 56 |
| 5.5.4.18 | pow_tension . . . . .                    | 56 |
| 5.5.4.19 | pwm_sign . . . . .                       | 57 |
| 5.5.4.20 | reset_last_value_flag . . . . .          | 57 |
| 5.5.4.21 | rest_enabled . . . . .                   | 57 |
| 5.5.4.22 | rest_pos_curr_ref . . . . .              | 57 |

|          |  |    |
|----------|--|----|
| 5.5.4.23 | tau_feedback . . . . .                   | 57 |
| 5.5.4.24 | tension_valid . . . . .                  | 57 |
| 5.5.4.25 | timer_value . . . . .                    | 57 |
| 5.5.4.26 | timer_value0 . . . . .                   | 57 |
| 5.5.4.27 | watchdog_flag . . . . .                  | 58 |
| 5.6      | interruptions.c File Reference . . . . . | 58 |
| 5.6.1    | Detailed Description . . . . .           | 59 |
| 5.6.2    | Function Documentation . . . . .         | 59 |
| 5.6.2.1  | analog_read_end() . . . . .              | 59 |
| 5.6.2.2  | cycles_counter_update() . . . . .        | 59 |
| 5.6.2.3  | encoder_reading() . . . . .              | 59 |
| 5.6.2.4  | function_scheduler() . . . . .           | 60 |
| 5.6.2.5  | interrupt_manager() . . . . .            | 60 |
| 5.6.2.6  | motor_control() . . . . .                | 60 |
| 5.6.2.7  | overcurrent_control() . . . . .          | 60 |
| 5.6.2.8  | pwm_limit_search() . . . . .             | 60 |
| 5.6.2.9  | save_cycles_eeprom() . . . . .           | 60 |
| 5.6.3    | Variable Documentation . . . . .         | 61 |
| 5.6.3.1  | pwm_preload_values . . . . .             | 61 |
| 5.7      | interruptions.h File Reference . . . . . | 61 |
| 5.7.1    | Detailed Description . . . . .           | 63 |
| 5.7.2    | Function Documentation . . . . .         | 63 |
| 5.7.2.1  | analog_read_end() . . . . .              | 63 |
| 5.7.2.2  | CY_ISR_PROTO() [1/3] . . . . .           | 63 |
| 5.7.2.3  | CY_ISR_PROTO() [2/3] . . . . .           | 63 |
| 5.7.2.4  | CY_ISR_PROTO() [3/3] . . . . .           | 64 |
| 5.7.2.5  | cycles_counter_update() . . . . .        | 64 |
| 5.7.2.6  | encoder_reading() . . . . .              | 64 |
| 5.7.2.7  | function_scheduler() . . . . .           | 64 |
| 5.7.2.8  | interrupt_manager() . . . . .            | 64 |

|          |                        |    |
|----------|------------------------|----|
| 5.7.2.9  | motor_control()        | 64 |
| 5.7.2.10 | overcurrent_control()  | 65 |
| 5.7.2.11 | pwm_limit_search()     | 65 |
| 5.7.2.12 | save_cycles_eeprom()   | 65 |
| 5.8      | main.c File Reference  | 65 |
| 5.8.1    | Detailed Description   | 66 |
| 5.9      | utils.c File Reference | 66 |
| 5.9.1    | Detailed Description   | 67 |
| 5.9.2    | Function Documentation | 68 |
| 5.9.2.1  | calc_turns_fcn()       | 68 |
| 5.9.2.2  | calibration()          | 68 |
| 5.9.2.3  | check_enc_data()       | 68 |
| 5.9.2.4  | check_rest_position()  | 69 |
| 5.9.2.5  | curr_estim()           | 69 |
| 5.9.2.6  | filter_acc_1()         | 69 |
| 5.9.2.7  | filter_acc_2()         | 69 |
| 5.9.2.8  | filter_acc_3()         | 70 |
| 5.9.2.9  | filter_ch1()           | 70 |
| 5.9.2.10 | filter_ch2()           | 71 |
| 5.9.2.11 | filter_curr_diff()     | 71 |
| 5.9.2.12 | filter_i1()            | 71 |
| 5.9.2.13 | filter_v()             | 72 |
| 5.9.2.14 | filter_vel_1()         | 72 |
| 5.9.2.15 | filter_vel_2()         | 72 |
| 5.9.2.16 | filter_vel_3()         | 73 |
| 5.9.2.17 | filter_voltage()       | 73 |
| 5.9.2.18 | my_mod()               | 73 |
| 5.9.2.19 | my_round()             | 74 |
| 5.9.2.20 | reset_counters()       | 74 |
| 5.10     | utils.h File Reference | 74 |

|   |    |
|---|----|
| 5.10.1 Detailed Description . . . . .           | 76 |
| 5.10.2 Macro Definition Documentation . . . . . | 77 |
| 5.10.2.1 REFSPEED . . . . .                     | 77 |
| 5.10.2.2 SIGN . . . . .                         | 77 |
| 5.10.2.3 ZERO_TOL . . . . .                     | 77 |
| 5.10.2.4 ZMAX . . . . .                         | 77 |
| 5.10.3 Function Documentation . . . . .         | 77 |
| 5.10.3.1 calc_turns_fcn() . . . . .             | 77 |
| 5.10.3.2 calibration() . . . . .                | 78 |
| 5.10.3.3 check_enc_data() . . . . .             | 78 |
| 5.10.3.4 check_rest_position() . . . . .        | 78 |
| 5.10.3.5 curr_estim() . . . . .                 | 78 |
| 5.10.3.6 filter_acc_1() . . . . .               | 79 |
| 5.10.3.7 filter_acc_2() . . . . .               | 79 |
| 5.10.3.8 filter_acc_3() . . . . .               | 79 |
| 5.10.3.9 filter_ch1() . . . . .                 | 80 |
| 5.10.3.10 filter_ch2() . . . . .                | 80 |
| 5.10.3.11 filter_curr_diff() . . . . .          | 81 |
| 5.10.3.12 filter_i1() . . . . .                 | 81 |
| 5.10.3.13 filter_v() . . . . .                  | 81 |
| 5.10.3.14 filter_vel_1() . . . . .              | 82 |
| 5.10.3.15 filter_vel_2() . . . . .              | 82 |
| 5.10.3.16 filter_vel_3() . . . . .              | 82 |
| 5.10.3.17 filter_voltage() . . . . .            | 83 |
| 5.10.3.18 my_mod() . . . . .                    | 83 |
| 5.10.3.19 my_round() . . . . .                  | 83 |
| 5.10.3.20 reset_counters() . . . . .            | 85 |

# Chapter 1

## Firmware

This is the firmware of the SoftHand board.

**Version**

6.1.5

This is the firmware of the SoftHand board. It can control a motor and read its encoder. Also can read and convert analog measurements connected to the PSoC microcontroller.





## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

|                 |  |    |
|-----------------|--|----|
| <b>st_calib</b> | Hand calibration structure . . . . .   | 7  |
| <b>st_data</b>  | Data sent/received structure . . . . . | 8  |
| <b>st_dev</b>   | Device related structure . . . . .     | 9  |
| <b>st_meas</b>  | Measurements structure . . . . .       | 10 |
| <b>st_mem</b>   | EEPROM stored structure . . . . .      | 12 |
| <b>st_ref</b>   | Motor Reference structure . . . . .    | 20 |



## Chapter 3

# File Index

### 3.1 File List

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

|  |    |
|--|----|
| <b>command_processing.c</b>  |    |
| Command processing functions . . . . .                               | 21 |
| <b>command_processing.h</b>  |    |
| Received commands processing functions . . . . .                     | 29 |
| <b>commands.h</b>  |    |
| Definitions for SoftHand commands, parameters and packages . . . . . | 37 |
| <b>device.h</b> . . . . .  | ?? |
| <b>globals.c</b>   |    |
| Global variables . . . . .   | 42 |
| <b>globals.h</b>   |    |
| Global definitions and macros are set in this file . . . . .         | 47 |
| <b>interruptions.c</b>   |    |
| Interruption handling and firmware core functions . . . . .          | 58 |
| <b>interruptions.h</b>   |    |
| Interruptions header file . . . . .                                  | 61 |
| <b>main.c</b>  |    |
| Firmware main file . . . . .   | 65 |
| <b>utils.c</b>   |    |
| Definition of utility functions . . . . .                            | 66 |
| <b>utils.h</b>   |    |
| Utility functions declaration . . . . .                              | 74 |



## Chapter 4

# Data Structure Documentation

### 4.1 st\_calib Struct Reference

Hand calibration structure.

```
#include <globals.h>
```

#### Data Fields

- uint8 **enabled**
- uint8 **direction**
- int16 **speed**
- int16 **repetitions**

#### 4.1.1 Detailed Description

Hand calibration structure.

#### 4.1.2 Field Documentation

##### 4.1.2.1 direction

```
uint8 direction
```

Direction of motor winding.

##### 4.1.2.2 enabled

```
uint8 enabled
```

Calibration enabling flag.

#### 4.1.2.3 repetitions

`int16 repetitions`

Number of cycles of hand closing/opening.

#### 4.1.2.4 speed

`int16 speed`

Speed of hand opening/closing.

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

- **globals.h**

## 4.2 st\_data Struct Reference

Data sent/received structure.

```
#include <globals.h>
```

### Data Fields

- `uint8` **buffer** [128]
- `int16` **length**
- `int16` **ind**
- `uint8` **ready**

### 4.2.1 Detailed Description

Data sent/received structure.

### 4.2.2 Field Documentation

#### 4.2.2.1 buffer

`uint8 buffer[128]`

Data buffer [CMD | DATA | CHECKSUM].

#### 4.2.2.2 ind

```
int16 ind
```

Data buffer index.

#### 4.2.2.3 length

```
int16 length
```

Data buffer length.

#### 4.2.2.4 ready

```
uint8 ready
```

Data buffer flag to see if the data is ready.

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

- **globals.h**

## 4.3 st\_dev Struct Reference

Device related structure.

```
#include <globals.h>
```

### Data Fields

- int32 **tension**
- float **tension\_conv\_factor**
- int8 **pwm\_limit**
- uint8 **tension\_valid**

### 4.3.1 Detailed Description

Device related structure.

### 4.3.2 Field Documentation

#### 4.3.2.1 pwm\_limit

```
int8 pwm_limit
```

Limit on pwm value driven to the motor.

#### 4.3.2.2 tension

```
int32 tension
```

Power supply tension.

#### 4.3.2.3 tension\_conv\_factor

```
float tension_conv_factor
```

Used to calculate input tension.

#### 4.3.2.4 tension\_valid

```
uint8 tension_valid
```

Flag checked if the power supply has a valid value.

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

- **globals.h**

## 4.4 st\_meas Struct Reference

Measurements structure.

```
#include <globals.h>
```

### Data Fields

- int32 **pos** [ **NUM\_OF\_SENSORS**]
- int32 **curr** [ **NUM\_OF\_MOTORS**]
- int8 **rot** [ **NUM\_OF\_SENSORS**]
- int32 **emg** [ **NUM\_OF\_EMGS**]
- int32 **vel** [ **NUM\_OF\_SENSORS**]
- int32 **acc** [ **NUM\_OF\_SENSORS**]

#### 4.4.1 Detailed Description

Measurements structure.



## 4.4.2 Field Documentation

### 4.4.2.1 acc

```
int32 acc[ NUM_OF_SENSORS]
```

Encoder rotational acceleration.

### 4.4.2.2 curr

```
int32 curr[ NUM_OF_MOTORS]
```

Motor current and current estimation.

### 4.4.2.3 emg

```
int32 emg[ NUM_OF_EMGS]
```

EMG sensors values.

### 4.4.2.4 pos

```
int32 pos[ NUM_OF_SENSORS]
```

Encoder sensor position.

### 4.4.2.5 rot

```
int8 rot[ NUM_OF_SENSORS]
```

Encoder sensor rotations.

### 4.4.2.6 vel

```
int32 vel[ NUM_OF_SENSORS]
```

Encoder rotational velocity.

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

- **globals.h**

## 4.5 st\_mem Struct Reference

EEPROM stored structure.

```
#include <globals.h>
```

### Data Fields

- uint8 **flag**
- uint8 **id**
- int32 **k\_p**
- int32 **k\_i**
- int32 **k\_d**
- int32 **k\_p\_c**
- int32 **k\_i\_c**
- int32 **k\_d\_c**
- int32 **k\_p\_dl**
- int32 **k\_i\_dl**
- int32 **k\_d\_dl**
- int32 **k\_p\_c\_dl**
- int32 **k\_i\_c\_dl**
- int32 **k\_d\_c\_dl**
- uint8 **activ**
- uint8 **input\_mode**
- uint8 **control\_mode**
- uint8 **res** [ NUM\_OF\_SENSORS]
- int32 **m\_off** [ NUM\_OF\_SENSORS]
- float **m\_mult** [ NUM\_OF\_SENSORS]
- uint8 **pos\_lim\_flag**
- int32 **pos\_lim\_inf** [ NUM\_OF\_MOTORS]
- int32 **pos\_lim\_sup** [ NUM\_OF\_MOTORS]
- int32 **max\_step\_pos**
- int32 **max\_step\_neg**
- int16 **current\_limit**
- uint16 **emg\_threshold** [ NUM\_OF\_EMGS]
- uint8 **emg\_calibration\_flag**
- uint32 **emg\_max\_value** [ NUM\_OF\_EMGS]
- uint8 **emg\_speed**
- uint8 **double\_encoder\_on\_off**
- int8 **motor\_handle\_ratio**
- uint8 **activate\_pwm\_rescaling**
- float **curr\_lookup** [ LOOKUP\_DIM]
- uint8 **baud\_rate**
- uint8 **watchdog\_period**
- int32 **rest\_pos**
- float **rest\_delay**
- float **rest\_vel**
- float **rest\_ratio**
- uint8 **maint\_day**
- uint8 **maint\_month**
- uint8 **maint\_year**
- uint8 **switch\_emg**
- uint8 **rest\_position\_flag**

- uint8 **unused\_bytes** [5]
- uint32 **emg\_counter** [2]
- uint32 **position\_hist** [10]
- uint32 **current\_hist** [4]
- uint32 **rest\_counter**
- uint32 **wire\_disp**
- uint32 **total\_time\_on**
- uint32 **total\_time\_rest**

### 4.5.1 Detailed Description

EEPROM stored structure.

### 4.5.2 Field Documentation

#### 4.5.2.1 **activ**

uint8 **activ**

Startup activation.

#### 4.5.2.2 **activate\_pwm\_rescaling**

uint8 **activate\_pwm\_rescaling**

Activation of PWM rescaling for 12V motors.

#### 4.5.2.3 **baud\_rate**

uint8 **baud\_rate**

Baud Rate setted.

#### 4.5.2.4 **control\_mode**

uint8 **control\_mode**

Motor Control mode.

#### 4.5.2.5 **curr\_lookup**

float **curr\_lookup**[ **LOOKUP\_DIM**]

Table of values to get estimated curr.

#### 4.5.2.6 current\_hist

```
uint32 current_hist[4]
```

Current histogram - 4 zones.

#### 4.5.2.7 current\_limit

```
int16 current_limit
```

Limit for absorbed current.

#### 4.5.2.8 double\_encoder\_on\_off

```
uint8 double_encoder_on_off
```

Double encoder ON/OFF.

#### 4.5.2.9 emg\_calibration\_flag

```
uint8 emg_calibration_flag
```

Enable emg calibration on startup.

#### 4.5.2.10 emg\_counter

```
uint32 emg_counter[2]
```

Counter for EMG activation - both channels.

#### 4.5.2.11 emg\_max\_value

```
uint32 emg_max_value[ NUM_OF_EMGS]
```

Maximum value for EMG.

#### 4.5.2.12 emg\_speed

```
uint8 emg_speed
```

Maximum closure speed when using EMG.

#### 4.5.2.13 emg\_threshold

```
uint16 emg_threshold[ NUM_OF_EMGS]
```

Minimum value for activation of EMG control.

#### 4.5.2.14 flag

uint8 flag

If checked the device has been configured.

#### 4.5.2.15 id

uint8 id

Device id.

#### 4.5.2.16 input\_mode

uint8 input\_mode

Motor Input mode.

#### 4.5.2.17 k\_d

int32 k\_d

Position controller derivative constant.

#### 4.5.2.18 k\_d\_c

int32 k\_d\_c

Current controller derivative constant.

#### 4.5.2.19 k\_d\_c\_dl

int32 k\_d\_c\_dl

Double loop current controller deriv. constant.

#### 4.5.2.20 k\_d\_dl

int32 k\_d\_dl

Double loop position controller deriv. constant.

#### 4.5.2.21 k\_i

int32 k\_i

Position controller integrative constant.

**4.5.2.22 k\_i\_c**

```
int32 k_i_c
```

Current controller integrative constant.

**4.5.2.23 k\_i\_c\_dl**

```
int32 k_i_c_dl
```

Double loop current controller integr. constant.

**4.5.2.24 k\_i\_dl**

```
int32 k_i_dl
```

Double loop position controller integr. constant.

**4.5.2.25 k\_p**

```
int32 k_p
```

Position controller proportional constant.

**4.5.2.26 k\_p\_c**

```
int32 k_p_c
```

Current controller proportional constant.

**4.5.2.27 k\_p\_c\_dl**

```
int32 k_p_c_dl
```

Double loop current controller prop. constant.

**4.5.2.28 k\_p\_dl**

```
int32 k_p_dl
```

Double loop position controller prop. constant.

**4.5.2.29 m\_mult**

```
float m_mult[ NUM_OF_SENSORS]
```

Measurement multiplier.

#### 4.5.2.30 m\_off

```
int32 m_off[ NUM_OF_SENSORS]
```

Measurement offset.

#### 4.5.2.31 maint\_day

```
uint8 maint_day
```

Day of maintenance.

#### 4.5.2.32 maint\_month

```
uint8 maint_month
```

Month of maintenance.

#### 4.5.2.33 maint\_year

```
uint8 maint_year
```

Year of maintenance.

#### 4.5.2.34 max\_step\_neg

```
int32 max_step_neg
```

Maximum number of steps per cycle for negative positions.

#### 4.5.2.35 max\_step\_pos

```
int32 max_step_pos
```

Maximum number of steps per cycle for positive positions.

#### 4.5.2.36 motor\_handle\_ratio

```
int8 motor_handle_ratio
```

Discrete multiplier for handle device.

#### 4.5.2.37 pos\_lim\_flag

```
uint8 pos_lim_flag
```

Position limit active/inactive.

**4.5.2.38 pos\_lim\_inf**

```
int32 pos_lim_inf[ NUM_OF_MOTORS]
```

Inferior position limit for motors.

**4.5.2.39 pos\_lim\_sup**

```
int32 pos_lim_sup[ NUM_OF_MOTORS]
```

Superior position limit for motors.

**4.5.2.40 position\_hist**

```
uint32 position_hist[10]
```

Positions histogram - 10 zones.

**4.5.2.41 res**

```
uint8 res[ NUM_OF_SENSORS]
```

Angle resolution.

**4.5.2.42 rest\_counter**

```
uint32 rest_counter
```

Counter for rest position occurrences.

**4.5.2.43 rest\_delay**

```
float rest_delay
```

Hand rest position delay while in EMG mode.

**4.5.2.44 rest\_pos**

```
int32 rest_pos
```

Hand rest position while in EMG mode.

**4.5.2.45 rest\_position\_flag**

```
uint8 rest_position_flag
```

Enable rest position feature.



**4.5.2.46 rest\_ratio**

```
float rest_ratio
```

Hand rest ratio between velocity closure and rest position error

**4.5.2.47 rest\_vel**

```
float rest_vel
```

Hand velocity closure for rest position reaching

**4.5.2.48 switch\_emg**

```
uint8 switch_emg
```

EMG opening/closure switch.

**4.5.2.49 total\_time\_on**

```
uint32 total_time_on
```

Total time of system power (in seconds).

**4.5.2.50 total\_time\_rest**

```
uint32 total_time_rest
```

Total time of system while rest position is maintained.

**4.5.2.51 watchdog\_period**

```
uint8 watchdog_period
```

Watchdog period setted, 255 = disable.

**4.5.2.52 wire\_disp**

```
uint32 wire_disp
```

Counter for total wire displacement measurement.

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

- **globals.h**

## 4.6 st\_ref Struct Reference

Motor Reference structure.

```
#include <globals.h>
```

### Data Fields

- int32 **pos** [ **NUM\_OF\_MOTORS**]
- int32 **curr** [ **NUM\_OF\_MOTORS**]
- int32 **pwm** [ **NUM\_OF\_MOTORS**]
- uint8 **onoff**

### 4.6.1 Detailed Description

Motor Reference structure.

### 4.6.2 Field Documentation

#### 4.6.2.1 curr

```
int32 curr[ NUM_OF_MOTORS]
```

Motor current reference.

#### 4.6.2.2 onoff

```
uint8 onoff
```

Motor drivers enable.

#### 4.6.2.3 pos

```
int32 pos[ NUM_OF_MOTORS]
```

Motor position reference.

#### 4.6.2.4 pwm

```
int32 pwm[ NUM_OF_MOTORS]
```

Motor direct pwm control.

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

- **globals.h**

## Chapter 5

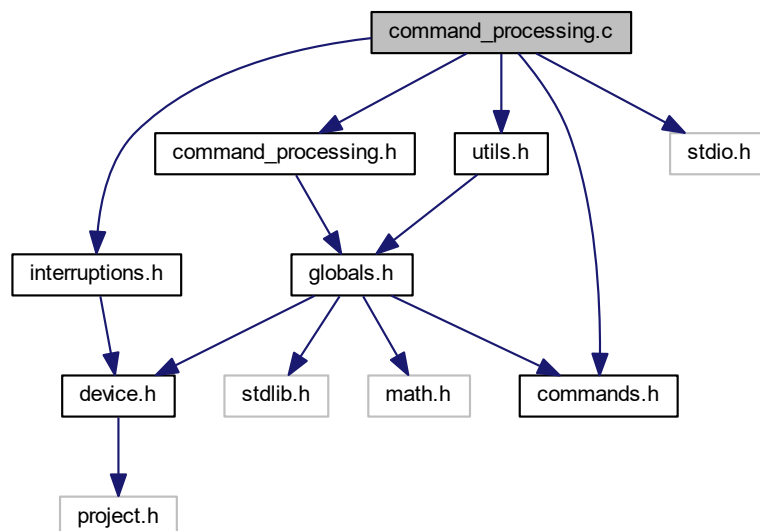
# File Documentation

### 5.1 command\_processing.c File Reference

Command processing functions.

```
#include <command_processing.h>
#include <interruptions.h>
#include <stdio.h>
#include <utils.h>
#include "commands.h"
```

Include dependency graph for command\_processing.c:



## Functions

- void **commProcess** (void)
- void **infoSend** (void)
- void **infoGet** (uint16 info\_type)
- void **get\_param\_list** (uint16 index)
- void **setZeros** ()
- void **prepare\_generic\_info** (char \*info\_string)
- void **prepare\_counter\_info** (char \*info\_string)
- void **commWrite\_old\_id** (uint8 \*packet\_data, uint16 packet\_lenght, uint8 old\_id)
- void **commWrite** (uint8 \*packet\_data, uint16 packet\_lenght)
- void **commWrite\_to\_cuff** (uint8 \*packet\_data, uint16 packet\_lenght)
- uint8 **LCRChecksum** (uint8 \*data\_array, uint8 data\_length)
- void **sendAcknowledgment** (uint8 value)
- uint8 **memStore** (int displacement)
- void **memRecall** (void)
- uint8 **memRestore** (void)
- uint8 **memInit** (void)
- void **cmd\_get\_measurements** ()
- void **cmd\_get\_velocities** ()
- void **cmd\_get\_accelerations** ()
- void **cmd\_set\_inputs** ()
- void **cmd\_activate** ()
- void **cmd\_get\_activate** ()
- void **cmd\_get\_curr\_and\_meas** ()
- void **cmd\_get\_currents** ()
- void **cmd\_get\_currents\_for\_cuff** ()
- void **cmd\_set\_baudrate** ()
- void **cmd\_ping** ()
- void **cmd\_set\_watchdog** ()
- void **cmd\_get\_inputs** ()
- void **cmd\_store\_params** ()
- void **cmd\_get\_emg** ()

## Variables

- reg8 \* **EEPROM\_ADDR** = (reg8 \*) CYDEV\_EE\_BASE

### 5.1.1 Detailed Description

Command processing functions.

#### Date

October 01, 2017

#### Author

*Centro "E.Piaggio"*

#### Copyright

(C) 2012-2016 qbrobotics. All rights reserved.  
(C) 2017 Centro "E.Piaggio". All rights reserved.

## 5.1.2 Function Documentation

### 5.1.2.1 cmd\_activate()

```
void cmd_activate ( )
```

This function activates the board

### 5.1.2.2 cmd\_get\_accelerations()

```
void cmd_get_accelerations ( )
```

This function gets the encoders accelerations and puts them in the package to be sent.

### 5.1.2.3 cmd\_get\_activate()

```
void cmd_get_activate ( )
```

This function gets the board activation status and puts it in the package to be sent.

### 5.1.2.4 cmd\_get\_curr\_and\_meas()

```
void cmd_get_curr_and_meas ( )
```

This function gets the motor current and measures and puts it in the package to be sent.

### 5.1.2.5 cmd\_get\_currents()

```
void cmd_get_currents ( )
```

This function gets the motor current and puts it in the package to be sent.

### 5.1.2.6 cmd\_get\_currents\_for\_cuff()

```
void cmd_get_currents_for_cuff ( )
```

This function gets the motor current and puts it in the package to be sent to the Cuff device, using the **comm↔Write\_to\_cuff** (p. 33) function.

### 5.1.2.7 cmd\_get\_emg()

```
void cmd_get_emg ( )
```

This function gets the electromyographic sensors measurements and puts them in the package to be sent.

#### 5.1.2.8 cmd\_get\_inputs()

```
void cmd_get_inputs ( )
```

This function gets the current motor reference inputs and puts them in the package to be sent.

#### 5.1.2.9 cmd\_get\_measurements()

```
void cmd_get_measurements ( )
```

Bunch of functions used on request from UART communication

#### 5.1.2.10 cmd\_get\_velocities()

```
void cmd_get_velocities ( )
```

This function gets the encoders velocities and puts them in the package to be sent.

#### 5.1.2.11 cmd\_ping()

```
void cmd_ping ( )
```

This function is used to ping the device and see if is connected.

#### 5.1.2.12 cmd\_set\_baudrate()

```
void cmd_set_baudrate ( )
```

This function sets the desired communication baudrate. It is possible to select a value equal to 460800 or 2000000.

#### 5.1.2.13 cmd\_set\_inputs()

```
void cmd_set_inputs ( )
```

This function gets the inputs from the received package and sets them as motor reference.

#### 5.1.2.14 cmd\_set\_watchdog()

```
void cmd_set_watchdog ( )
```

This function sets the watchdog timer to the one received from the package. The board automatically deactivate when the time equivalent, to watchdog timer, has passed.

#### 5.1.2.15 cmd\_store\_params()

```
void cmd_store_params ( )
```

This function stores the parameters to the EEPROM memory

#### 5.1.2.16 commProcess()

```
void commProcess ( )
```

This function unpacks the received package, depending on the command received.

#### 5.1.2.17 commWrite()

```
void commWrite (
    uint8 * packet_data,
    uint16 packet_lenght )
```

This function writes on the serial port the package that needs to be sent to the user.

##### Parameters

|                      |   |
|----------------------|---|
| <i>packet_data</i>   | The array of data that must be written. |
| <i>packet_lenght</i> | The lenght of the data array.           |

#### 5.1.2.18 commWrite\_old\_id()

```
void commWrite_old_id (
    uint8 * packet_data,
    uint16 packet_lenght,
    uint8 old_id )
```

This function writes on the serial port the package that needs to be sent to the user. Is used only when a new is set, to communicate back to the APIs that the new ID setting went fine or there was an error.

##### Parameters

|                      |   |
|----------------------|---|
| <i>packet_data</i>   | The array of data that must be written.                 |
| <i>packet_lenght</i> | The lenght of the data array.                           |
| <i>old_id</i>        | The previous id of the board, before setting a new one. |

#### 5.1.2.19 commWrite\_to\_cuff()

```
void commWrite_to_cuff (
```

```
uint8 * packet_data,
uint16 packet_lenght )
```

This function writes on the serial port the package that needs to be sent to the Cuff device. It is used only when a specific device is connected to the hand. The Hand must have ID equal to the one of the Cuff plus one.

#### Parameters

|                      |   |
|----------------------|---|
| <i>packet_data</i>   | The array of data that must be written. |
| <i>packet_lenght</i> | The lenght of the data array.           |

#### 5.1.2.20 get\_param\_list()

```
void get_param_list (
    uint16 index )
```

This function, depending on the **Firmware** (p. 1) received, gets the list of parameters with their values and sends them to user or sets a parameter from all the parameters of the device.

#### Parameters

|              |   |
|--------------|---|
| <i>index</i> | The index of the parameters to be setted. If 0 gets full parameters list. |
|--------------|---|

#### 5.1.2.21 infoGet()

```
void infoGet (
    uint16 info_type )
```

This function sends the firmware information prepared with `prepare_general_info` or **prepare\_counter\_info** (p. 36) through the serial port to the user interface. Is used when the ID is specified.

#### Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <i>info_type</i> | The type of the information needed. |
|------------------|-------------------------------------|

#### 5.1.2.22 infoSend()

```
void infoSend ( )
```

This function sends the firmware information prepared with `infoPrepare` through the serial port to the user interface. Is used when no ID is specified.



#### 5.1.2.23 `LCRChecksum()`

```
uint8 LCRChecksum (
    uint8 * data_array,
    uint8 data_length )
```

This function calculates a checksum of the array to see if the received data is consistent.

##### Parameters

|                    |  |
|--------------------|--|
| <i>data_array</i>  | The array of data that must be checked.        |
| <i>data_length</i> | Lenght of the data array that must be checked. |

##### Returns

The calculated checksum for the relative `data_array`.

#### 5.1.2.24 `memInit()`

```
uint8 memInit ( )
```

This functions initializes the memory. It is used also to restore the the parameters to their default values.

##### Returns

A true value if the memory is correctly initialized, false otherwise.

#### 5.1.2.25 `memRecall()`

```
void memRecall ( )
```

This function loads user's settings from the EEPROM.

#### 5.1.2.26 `memRestore()`

```
uint8 memRestore ( )
```

This function loads default settings from the EEPROM.

##### Returns

A true value if the memory is correctly restored, false otherwise.

#### 5.1.2.27 `memStore()`

```
uint8 memStore (
    int displacement )
```

This function stores the setted parameters to the internal EEPROM memory. It is usually called, by the user, after a parameter is set.

**Parameters**

|                     |   |
|---------------------|---|
| <i>displacement</i> | The address where the parameters will be written. |
|---------------------|---|

**Returns**

A true value if the memory is correctly stored, false otherwise.

**5.1.2.28 prepare\_counter\_info()**

```
void prepare_counter_info (
    char * info_string )
```

This function is used to prepare an information string about the cycles counter of the hand

**Parameters**

|                    |  |
|--------------------|--|
| <i>info_string</i> | An array of chars containing the requested informations. |
|--------------------|--|

**5.1.2.29 prepare\_generic\_info()**

```
void prepare_generic_info (
    char * info_string )
```

This function is used to prepare a generic information string on the device parameters and measurements

**Parameters**

|                    |  |
|--------------------|--|
| <i>info_string</i> | An array of chars containing the requested informations. |
|--------------------|--|

**5.1.2.30 sendAcknowledgment()**

```
void sendAcknowledgment (
    uint8 value )
```

This functions sends an acknowledgment to see if a command has been executed properly or not.

**Parameters**

|              |                                     |
|--------------|-------------------------------------|
| <i>value</i> | An ACK_OK(1) or ACK_ERROR(0) value. |
|--------------|-------------------------------------|

### 5.1.2.31 setZeros()

```
void setZeros ( )
```

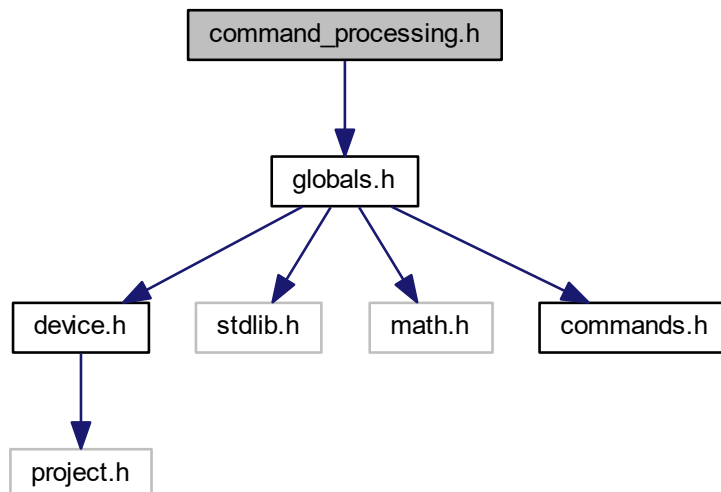
This function sets the encoders zero position.

## 5.2 command\_processing.h File Reference

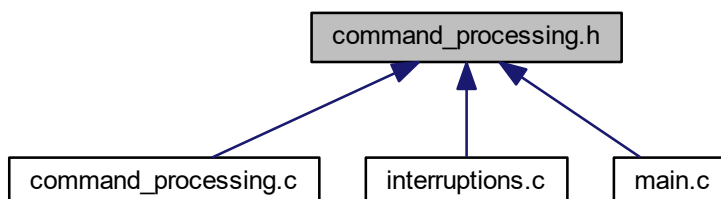
Received commands processing functions.

```
#include <globals.h>
```

Include dependency graph for command\_processing.h:



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



## Functions

### Firmware information functions

- void **prepare\_generic\_info** (char \*info\_string)
- void **prepare\_counter\_info** (char \*info\_string)
- void **infoSend** ()
- void **infoGet** (uint16 info\_type)

### Command receiving and sending functions

- void **commProcess** ()
- void **commWrite\_old\_id** (uint8 \*packet\_data, uint16 packet\_length, uint8 old\_id)
- void **commWrite** (uint8 \*packet\_data, uint16 packet\_length)
- void **commWrite\_to\_cuff** (uint8 \*packet\_data, uint16 packet\_length)

### Memory management functions

- void **get\_param\_list** (uint16 index)
- void **setZeros** ()
- uint8 **memStore** (int displacement)
- void **memRecall** ()
- uint8 **memRestore** ()
- uint8 **memInit** ()

### Utility functions

- uint8 **LCRChecksum** (uint8 \*data\_array, uint8 data\_length)
- void **sendAcknowledgment** (uint8 value)

### Command processing functions

- void **cmd\_activate** ()
- void **cmd\_set\_inputs** ()
- void **cmd\_get\_measurements** ()
- void **cmd\_get\_velocities** ()
- void **cmd\_get\_accelerations** ()
- void **cmd\_get\_currents** ()
- void **cmd\_get\_curr\_and\_meas** ()
- void **cmd\_get\_currents\_for\_cuff** ()
- void **cmd\_get\_emg** ()
- void **cmd\_set\_watchdog** ()
- void **cmd\_get\_activate** ()
- void **cmd\_set\_baudrate** ()
- void **cmd\_get\_inputs** ()
- void **cmd\_store\_params** ()
- void **cmd\_ping** ()

## 5.2.1 Detailed Description

Received commands processing functions.

### Date

October 01, 2017

### Author

*Centro "E.Piaggio"*

### Copyright

(C) 2012-2016 qbrobotics. All rights reserved.  
(C) 2017 Centro "E.Piaggio". All rights reserved.

This file contains all the definitions of the functions used to process the commands sent from the user interfaces (simulink, command line, GUI)

## 5.2.2 Function Documentation

### 5.2.2.1 cmd\_activate()

```
void cmd_activate ( )
```

This function activates the board

### 5.2.2.2 cmd\_get\_accelerations()

```
void cmd_get_accelerations ( )
```

This function gets the encoders accelerations and puts them in the package to be sent.

### 5.2.2.3 cmd\_get\_activate()

```
void cmd_get_activate ( )
```

This function gets the board activation status and puts it in the package to be sent.

### 5.2.2.4 cmd\_get\_curr\_and\_meas()

```
void cmd_get_curr_and_meas ( )
```

This function gets the motor current and measures and puts it in the package to be sent.

### 5.2.2.5 cmd\_get\_currents()

```
void cmd_get_currents ( )
```

This function gets the motor current and puts it in the package to be sent.

### 5.2.2.6 cmd\_get\_currents\_for\_cuff()

```
void cmd_get_currents_for_cuff ( )
```

This function gets the motor current and puts it in the package to be sent to the Cuff device, using the **comm↔Write\_to\_cuff** (p. 33) function.

### 5.2.2.7 cmd\_get\_emg()

```
void cmd_get_emg ( )
```

This function gets the electromyographic sensors measurements and puts them in the package to be sent.

#### 5.2.2.8 cmd\_get\_inputs()

```
void cmd_get_inputs ( )
```

This function gets the current motor reference inputs and puts them in the package to be sent.

#### 5.2.2.9 cmd\_get\_measurements()

```
void cmd_get_measurements ( )
```

This function gets the encoders measurements and puts them in the package to be sent.

Bunch of functions used on request from UART communication

#### 5.2.2.10 cmd\_get\_velocities()

```
void cmd_get_velocities ( )
```

This function gets the encoders velocities and puts them in the package to be sent.

#### 5.2.2.11 cmd\_ping()

```
void cmd_ping ( )
```

This function is used to ping the device and see if is connected.

#### 5.2.2.12 cmd\_set\_baudrate()

```
void cmd_set_baudrate ( )
```

This function sets the desired communication baudrate. It is possible to select a value equal to 460800 or 2000000.

#### 5.2.2.13 cmd\_set\_inputs()

```
void cmd_set_inputs ( )
```

This function gets the inputs from the received package and sets them as motor reference.

#### 5.2.2.14 cmd\_set\_watchdog()

```
void cmd_set_watchdog ( )
```

This function sets the watchdog timer to the one received from the package. The board automatically deactivate when the time equivalent, to watchdog timer, has passed.

#### 5.2.2.15 cmd\_store\_params()

```
void cmd_store_params ( )
```

This function stores the parameters to the EEPROM memory

#### 5.2.2.16 commProcess()

```
void commProcess ( )
```

This function unpacks the received package, depending on the command received.

#### 5.2.2.17 commWrite()

```
void commWrite (
    uint8 * packet_data,
    uint16 packet_lenght )
```

This function writes on the serial port the package that needs to be sent to the user.

##### Parameters

|                      |   |
|----------------------|---|
| <i>packet_data</i>   | The array of data that must be written. |
| <i>packet_lenght</i> | The lenght of the data array.           |

#### 5.2.2.18 commWrite\_old\_id()

```
void commWrite_old_id (
    uint8 * packet_data,
    uint16 packet_lenght,
    uint8 old_id )
```

This function writes on the serial port the package that needs to be sent to the user. Is used only when a new is set, to communicate back to the APIs that the new ID setting went fine or there was an error.

##### Parameters

|                      |   |
|----------------------|---|
| <i>packet_data</i>   | The array of data that must be written.                 |
| <i>packet_lenght</i> | The lenght of the data array.                           |
| <i>old_id</i>        | The previous id of the board, before setting a new one. |

#### 5.2.2.19 commWrite\_to\_cuff()

```
void commWrite_to_cuff (
```

```
uint8 * packet_data,  
uint16 packet_lenght )
```

This function writes on the serial port the package that needs to be sent to the Cuff device. It is used only when a specific device is connected to the hand. The Hand must have ID equal to the one of the Cuff plus one.

#### Parameters

|                      |   |
|----------------------|---|
| <i>packet_data</i>   | The array of data that must be written. |
| <i>packet_lenght</i> | The lenght of the data array.           |

#### 5.2.2.20 get\_param\_list()

```
void get_param_list (  
    uint16 index )
```

This function, depending on the **Firmware** (p. 1) received, gets the list of parameters with their values and sends them to user or sets a parameter from all the parameters of the device.

#### Parameters

|              |   |
|--------------|---|
| <i>index</i> | The index of the parameters to be setted. If 0 gets full parameters list. |
|--------------|---|

#### 5.2.2.21 infoGet()

```
void infoGet (  
    uint16 info_type )
```

This function sends the firmware information prepared with `prepare_general_info` or **`prepare_counter_info`** (p. 36) through the serial port to the user interface. Is used when the ID is specified.

#### Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <i>info_type</i> | The type of the information needed. |
|------------------|-------------------------------------|

#### 5.2.2.22 infoSend()

```
void infoSend ( )
```

This function sends the firmware information prepared with `infoPrepare` through the serial port to the user interface. Is used when no ID is specified.



### 5.2.2.23 LCRChecksum()

```
uint8 LCRChecksum (
    uint8 * data_array,
    uint8 data_length )
```

This function calculates a checksum of the array to see if the received data is consistent.

#### Parameters

|                    |  |
|--------------------|--|
| <i>data_array</i>  | The array of data that must be checked.        |
| <i>data_length</i> | Lenght of the data array that must be checked. |

#### Returns

The calculated checksum for the relative data\_array.

### 5.2.2.24 memInit()

```
uint8 memInit ( )
```

This functions initializes the memory. It is used also to restore the the parameters to their default values.

#### Returns

A true value if the memory is correctly initialized, false otherwise.

### 5.2.2.25 memRecall()

```
void memRecall ( )
```

This function loads user's settings from the EEPROM.

### 5.2.2.26 memRestore()

```
uint8 memRestore ( )
```

This function loads default settings from the EEPROM.

#### Returns

A true value if the memory is correctly restored, false otherwise.

### 5.2.2.27 memStore()

```
uint8 memStore (
    int displacement )
```

This function stores the setted parameters to the internal EEPROM memory. It is usually called, by the user, after a parameter is set.

**Parameters**

|                     |   |
|---------------------|---|
| <i>displacement</i> | The address where the parameters will be written. |
|---------------------|---|

**Returns**

A true value if the memory is correctly stored, false otherwise.

**5.2.2.28 prepare\_counter\_info()**

```
void prepare_counter_info (
    char * info_string )
```

This function is used to prepare an information string about the cycles counter of the hand

**Parameters**

|                    |  |
|--------------------|--|
| <i>info_string</i> | An array of chars containing the requested informations. |
|--------------------|--|

**5.2.2.29 prepare\_generic\_info()**

```
void prepare_generic_info (
    char * info_string )
```

This function is used to prepare a generic information string on the device parameters and measurements

**Parameters**

|                    |  |
|--------------------|--|
| <i>info_string</i> | An array of chars containing the requested informations. |
|--------------------|--|

**5.2.2.30 sendAcknowledgment()**

```
void sendAcknowledgment (
    uint8 value )
```

This functions sends an acknowledgment to see if a command has been executed properly or not.

**Parameters**

|              |                                     |
|--------------|-------------------------------------|
| <i>value</i> | An ACK_OK(1) or ACK_ERROR(0) value. |
|--------------|-------------------------------------|

## 5.2.2.31 setZeros()

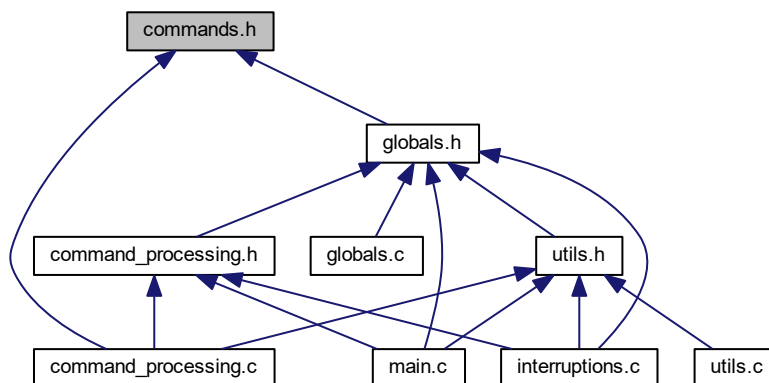
```
void setZeros ( )
```

This function sets the encoders zero position.

## 5.3 commands.h File Reference

Definitions for SoftHand commands, parameters and packages.

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



## Macros

## QB Move Information Strings

- #define **INFO\_ALL** 0  
*Generic device informations.*
- #define **CYCLES\_INFO** 1  
*Cycles counter informations.*

## Enumerations

## qbMove and qbHand Commands

- enum **qbmove\_command** {  
**CMD\_PING** = 0, **CMD\_SET\_ZEROS** = 1, **CMD\_STORE\_PARAMS** = 3, **CMD\_STORE\_DEFAULT\_P**↵  
**ARAMS** = 4,  
**CMD\_RESTORE\_PARAMS** = 5, **CMD\_GET\_INFO** = 6, **CMD\_SET\_VALUE** = 7, **CMD\_GET\_VALUE** =  
8,  
**CMD\_BOOTLOADER** = 9, **CMD\_INIT\_MEM** = 10, **CMD\_CALIBRATE** = 11, **CMD\_GET\_PARAM\_LIST**  
= 12,  
**CMD\_HAND\_CALIBRATE** = 13, **CMD\_ACTIVATE** = 128, **CMD\_GET\_ACTIVATE** = 129, **CMD\_SET**↵  
**\_INPUTS** = 130,  
**CMD\_GET\_INPUTS** = 131, **CMD\_GET\_MEASUREMENTS** = 132, **CMD\_GET\_CURRENTS** = 133, **C**↵  
**MD\_GET\_CURR\_AND\_MEAS** = 134,  
**CMD\_SET\_POS\_STIFF** = 135, **CMD\_GET\_EMG** = 136, **CMD\_GET\_VELOCITIES** = 137, **CMD\_GET**↵  
**\_COUNTERS** = 138,  
**CMD\_GET\_ACCEL** = 139, **CMD\_GET\_CURR\_DIFF** = 140, **CMD\_SET\_CURR\_DIFF** = 141, **CMD\_S**↵  
**ET\_CUFF\_INPUTS** = 142,  
**CMD\_SET\_WATCHDOG** = 143, **CMD\_SET\_BAUDRATE** = 144, **CMD\_EXT\_DRIVE** = 145, **CMD\_G**↵  
**ET\_JOYSTICK** = 146 }

### qbMove and qbHand Parameters

- #define **PARAM\_BYTE\_SLOT** 50  
*Number of bytes reserved to a param information.*
- #define **PARAM\_MENU\_SLOT** 150  
*Number of bytes reserved to a param menu.*
- enum **qbmove\_parameter** {  
**PARAM\_ID** = 0, **PARAM\_PID\_CONTROL** = 1, **PARAM\_STARTUP\_ACTIVATION** = 2, **PARAM\_INPU**↵  
**T\_MODE** = 3,  
**PARAM\_CONTROL\_MODE** = 4, **PARAM\_MEASUREMENT\_OFFSET** = 5, **PARAM\_MEASUREMENT**↵  
**\_MULTIPLIER** = 6, **PARAM\_POS\_LIMIT\_FLAG** = 7,  
**PARAM\_POS\_LIMIT** = 8, **PARAM\_MAX\_STEP\_POS** = 9, **PARAM\_MAX\_STEP\_NEG** = 10, **PARAM**↵  
**\_POS\_RESOLUTION** = 11,  
**PARAM\_CURRENT\_LIMIT** = 12, **PARAM\_EMG\_CALIB\_FLAG** = 13, **PARAM\_EMG\_THRESHOLD** = 14,  
**PARAM\_EMG\_MAX\_VALUE** = 15,  
**PARAM\_EMG\_SPEED** = 16, **PARAM\_PID\_CURR\_CONTROL** = 18, **PARAM\_DOUBLE\_ENC\_ON\_OFF**  
= 19, **PARAM\_MOT\_HANDLE\_RATIO** = 20,  
**PARAM\_MOTOR\_SUPPLY** = 21, **PARAM\_CURRENT\_LOOKUP** = 23, **PARAM\_DL\_POS\_PID** = 24, **P**↵  
**ARAM\_DL\_CURR\_PID** = 25 }
- enum **qbmove\_resolution** {  
**RESOLUTION\_360** = 0, **RESOLUTION\_720** = 1, **RESOLUTION\_1440** = 2, **RESOLUTION\_2880** = 3,  
**RESOLUTION\_5760** = 4, **RESOLUTION\_11520** = 5, **RESOLUTION\_23040** = 6, **RESOLUTION\_46080** = 7,  
**RESOLUTION\_92160** = 8 }
- enum **qbmove\_input\_mode** {  
**INPUT\_MODE\_EXTERNAL** = 0, **INPUT\_MODE\_ENCODER3** = 1, **INPUT\_MODE\_EMG\_PROPORTION**↵  
**AL** = 2, **INPUT\_MODE\_EMG\_INTEGRAL** = 3,  
**INPUT\_MODE\_EMG\_FCFS** = 4, **INPUT\_MODE\_EMG\_FCFS\_ADV** = 5 }
- enum **qbmove\_control\_mode** { **CONTROL\_ANGLE** = 0, **CONTROL\_PWM** = 1, **CONTROL\_CURRENT**  
= 2, **CURR\_AND\_POS\_CONTROL** = 3 }
- enum **motor\_supply\_tipe** { **MAXON\_24V** = 0, **MAXON\_12V** = 1 }
- enum **acknowledgment\_values** { **ACK\_ERROR** = 0, **ACK\_OK** = 1 }
- enum **data\_types** {  
**TYPE\_FLAG** = 0, **TYPE\_INT8** = 1, **TYPE\_UINT8** = 2, **TYPE\_INT16** = 3,  
**TYPE\_UINT16** = 4, **TYPE\_INT32** = 5, **TYPE\_UINT32** = 6, **TYPE\_FLOAT** = 7,  
**TYPE\_DOUBLE** = 8 }

### 5.3.1 Detailed Description

Definitions for SoftHand commands, parameters and packages.

#### Date

October 01, 2017

#### Author

Centro "E.Piaggio"

#### Copyright

(C) 2012-2016 qbrobotics. All rights reserved.  
(C) 2017 Centro "E.Piaggio". All rights reserved.

This file is included in the firmware, in its libraries and applications. It contains all definitions that are necessary for the construction of communication packages.

It includes definitions for all of the device commands, parameters and also the size of answer packages.

### 5.3.2 Enumeration Type Documentation

#### 5.3.2.1 qbmove\_command

enum **qbmove\_command**

#### Enumerator

|                          |   |
|--------------------------|---|
| CMD_PING                 | Asks for a ping message.  |
| CMD_SET_ZEROS            | Command for setting the encoders zero position.   |
| CMD_STORE_PARAMS         | Stores all parameters in memory and loads them.   |
| CMD_STORE_DEFAULT_PARAMS | Store current parameters as factory parameters.   |
| CMD_RESTORE_PARAMS       | Restore default factory parameters.   |
| CMD_GET_INFO             | Asks for a string of information about.   |
| CMD_SET_VALUE            | Not Used.   |
| CMD_GET_VALUE            | Not Used.   |
| CMD_BOOTLOADER           | Sets the bootloader modality to update the firmware.  |
| CMD_INIT_MEM             | Initialize the memory with the default values.  |
| CMD_CALIBRATE            | Starts the stiffness calibration of the qbMove or the hand closure and opening calibration. |
| CMD_GET_PARAM_LIST       | Command to get the parameters list or to set a defined value chosen by the use.             |
| CMD_HAND_CALIBRATE       | Starts a series of opening and closures of the hand.  |
| CMD_ACTIVATE             | Command for activating/deactivating the device.   |
| CMD_GET_ACTIVATE         | Command for getting device activation state.  |

## Enumerator

|                       |   |
|-----------------------|---|
| CMD_SET_INPUTS        | Command for setting reference inputs.   |
| CMD_GET_INPUTS        | Command for getting reference inputs.   |
| CMD_GET_MEASUREMENTS  | Command for asking device's position measurements.  |
| CMD_GET_CURRENTS      | Command for asking device's current measurements.   |
| CMD_GET_CURR_AND_MEAS | Command for asking device's measurements and currents.  |
| CMD_SET_POS_STIFF     | Not used in the softhand firmware.  |
| CMD_GET_EMG           | Command for asking device's emg sensors measurements.   |
| CMD_GET_VELOCITIES    | Command for asking device's velocity measurements.  |
| CMD_GET_COUNTERS      | Command for asking device's counters (mostly used for debugging sent commands).                                 |
| CMD_GET_ACCEL         | Command for asking device's acceleration measurements.  |
| CMD_GET_CURR_DIFF     | Command for asking device's current difference between a measured one and an estimated one (Only for SoftHand). |
| CMD_SET_CURR_DIFF     | Command used to set current difference modality (Only for Cuff device).   |
| CMD_SET_CUFF_INPUTS   | Command used to set Cuff device inputs (Only for Cuff device).  |
| CMD_SET_WATCHDOG      | Command for setting watchdog timer or disable it.   |
| CMD_SET_BAUDRATE      | Command for setting baudrate of communication.  |
| CMD_EXT_DRIVE         | Command to set the actual measurements as inputs to another device (Only for Armslider device).                 |
| CMD_GET_JOYSTICK      | Command to get the joystick measurements (Only for devices driven by a joystick).                               |

## 5.3.2.2 qbmove\_control\_mode

```
enum qbmove_control_mode
```

## Enumerator

|                      |                               |
|----------------------|-------------------------------|
| CONTROL_ANGLE        | Classic position control.     |
| CONTROL_PWM          | Direct PWM value.             |
| CONTROL_CURRENT      | Current control.              |
| CURR_AND_POS_CONTROL | Current and position control. |

## 5.3.2.3 qbmove\_input\_mode

```
enum qbmove_input_mode
```

## Enumerator

|                     |   |
|---------------------|---|
| INPUT_MODE_EXTERNAL | References through external commands (default). |
| INPUT_MODE_ENCODER3 | Encoder 3 drives all inputs.                    |

## Enumerator

|                             |   |
|-----------------------------|---|
| INPUT_MODE_EMG_PROPORTIONAL | Use EMG measure to proportionally drive the position of the motor 1.  |
| INPUT_MODE_EMG_INTEGRAL     | Use 2 EMG signals to drive motor position.  |
| INPUT_MODE_EMG_FCFS         | Use 2 EMG. First reaching threshold wins and its value defines hand closure.  |
| INPUT_MODE_EMG_FCFS_ADV     | Use 2 EMG. First reaching threshold wins and its value defines hand closure Wait for both EMG to lower under threshold. |

## 5.3.2.4 qbmove\_parameter

```
enum qbmove_parameter
```

## Enumerator

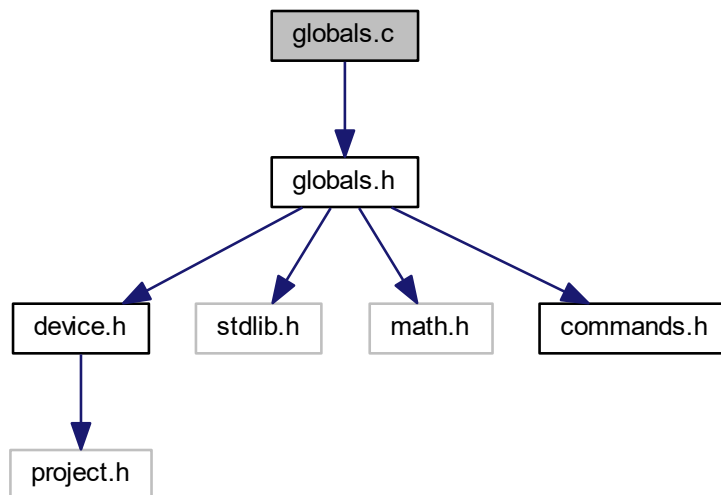
|                              |   |
|------------------------------|---|
| PARAM_ID                     | Device's ID number.   |
| PARAM_PID_CONTROL            | PID parameters.   |
| PARAM_STARTUP_ACTIVATION     | Start up activation byte.   |
| PARAM_INPUT_MODE             | Input mode.   |
| PARAM_CONTROL_MODE           | Choose the kind of control between position control, current control, direct PWM value or current+position control. |
| PARAM_MEASUREMENT_OFFSET     | Adds a constant offset to the measurements.   |
| PARAM_MEASUREMENT_MULTIPLIER | Adds a multiplier to the measurements.  |
| PARAM_POS_LIMIT_FLAG         | Enable/disable position limiting.   |
| PARAM_POS_LIMIT              | Position limit values   int32   int32   int32   int32     INF_LIM_1   SUP_LIM_1   INF_LIM_2   SUP_LIM_2             |
| PARAM_MAX_STEP_POS           | Used to slow down movements for positive values.  |
| PARAM_MAX_STEP_NEG           | Used to slow down movements for negative values.  |
| PARAM_POS_RESOLUTION         | Angle resolution for inputs and measurements. Used during communication.  |
| PARAM_CURRENT_LIMIT          | Limit for absorbed current.   |
| PARAM_EMG_CALIB_FLAG         | Enable calibration on startup.  |
| PARAM_EMG_THRESHOLD          | Minimum value to have effect.   |
| PARAM_EMG_MAX_VALUE          | Maximum value of EMG.   |
| PARAM_EMG_SPEED              | Closure speed when using EMG.   |
| PARAM_PID_CURR_CONTROL       | PID current control.  |
| PARAM_DOUBLE_ENC_ON_OFF      | Double Encoder Y/N.   |
| PARAM_MOT_HANDLE_RATIO       | Multiplier between handle and motor.  |
| PARAM_MOTOR_SUPPLY           | Motor supply voltage of the hand.   |
| PARAM_CURRENT_LOOKUP         | Table of values used to calculate an estimated current of the SoftHand.   |
| PARAM_DL_POS_PID             | Double loop position PID.   |
| PARAM_DL_CURR_PID            | Double loop current PID.  |

## 5.4 globals.c File Reference

Global variables.

```
#include <globals.h>
```

Include dependency graph for globals.c:



### Variables

- struct **st\_ref** `g_ref` `g_refNew` `g_refOld`
- struct **st\_meas** `g_meas` `g_measOld`
- struct **st\_data** `g_rx`
- struct **st\_mem** `g_mem` `c_mem`
- struct **st\_calib** `calib`
- float **tau\_feedback**
- uint32 **timer\_value**
- uint32 **timer\_value0**
- float **cycle\_time**
- int32 **dev\_tension**
- uint8 **dev\_pwm\_limit**
- uint8 **dev\_pwm\_sat** = 100
- int32 **dev\_tension\_f**
- int32 **pow\_tension**
- **counter\_status** CYDATA **cycles\_status** = NONE
- **emg\_status** CYDATA **emg\_1\_status** = RESET
- **emg\_status** CYDATA **emg\_2\_status** = RESET
- CYBIT **reset\_last\_value\_flag**
- CYBIT **tension\_valid**
- CYBIT **interrupt\_flag** = FALSE
- CYBIT **watchdog\_flag** = FALSE
- CYBIT **cycles\_interrupt\_flag** = FALSE



- CYBIT **can\_write** = TRUE
- uint8 **rest\_enabled**
- uint8 **forced\_open**
- int16 **ADC\_buf** [4]
- int8 **pwm\_sign**
- int32 **rest\_pos\_curr\_ref**

### 5.4.1 Detailed Description

Global variables.

#### Date

October 01, 2017

#### Author

*Centro "E.Piaggio"*

#### Copyright

(C) 2012-2016 qbrobotics. All rights reserved.  
(C) 2017 Centro "E.Piaggio". All rights reserved.

### 5.4.2 Variable Documentation

#### 5.4.2.1 c\_mem

```
struct st_mem g_mem c_mem
```

Memory parameters.

#### 5.4.2.2 calib

```
struct st_calib calib
```

Calibration variables.

#### 5.4.2.3 can\_write

```
CYBIT can_write = TRUE
```

Write to EEPROM flag.

#### 5.4.2.4 cycle\_time

```
float cycle_time
```

Variable used to calculate how much time a cycle takes.

#### 5.4.2.5 cycles\_interrupt\_flag

```
CYBIT cycles_interrupt_flag = FALSE
```

Cycles timer interrupt flag enabler.

#### 5.4.2.6 cycles\_status

```
counter_status CYDATA cycles_status = NONE
```

Cycles counter state machine status.

#### 5.4.2.7 dev\_pwm\_limit

```
uint8 dev_pwm_limit
```

Device pwm limit. It may change during execution.

#### 5.4.2.8 dev\_pwm\_sat

```
uint8 dev_pwm_sat = 100
```

Device pwm saturation. By default the saturation value must not exceed 100.

#### 5.4.2.9 dev\_tension

```
int32 dev_tension
```

Power supply tension.

#### 5.4.2.10 dev\_tension\_f

```
int32 dev_tension_f
```

Filtered power supply tension.

#### 5.4.2.11 emg\_1\_status

```
emg_status CYDATA emg_1_status = RESET
```

First EMG sensor status.

#### 5.4.2.12 emg\_2\_status

```
emg_status CYDATA emg_2_status = RESET
```

Second EMG sensor status.

#### 5.4.2.13 forced\_open

```
uint8 forced_open
```

Forced open flag (used in position with rest position control).

#### 5.4.2.14 g\_measOld

```
struct st_meas g_meas g_measOld
```

Measurements.

#### 5.4.2.15 g\_refOld

```
struct st_ref g_ref g_refNew g_refOld
```

Reference variables.

#### 5.4.2.16 g\_rx

```
struct st_data g_rx
```

Incoming/Outcoming data.

#### 5.4.2.17 interrupt\_flag

```
CYBIT interrupt_flag = FALSE
```

Interrupt flag enabler.

#### 5.4.2.18 pow\_tension

```
int32 pow_tension
```

Computed power supply tension.

#### 5.4.2.19 pwm\_sign

```
int8 pwm_sign
```

ADC measurements buffer. Sign of pwm driven. Used to obtain current sign.

#### 5.4.2.20 reset\_last\_value\_flag

```
CYBIT reset_last_value_flag
```

This flag is set when the encoders last values must be resetted.

#### 5.4.2.21 rest\_enabled

```
uint8 rest_enabled
```

Rest position flag.

#### 5.4.2.22 rest\_pos\_curr\_ref

```
int32 rest_pos_curr_ref
```

Rest position current reference.

#### 5.4.2.23 tau\_feedback

```
float tau_feedback
```

Torque feedback.

#### 5.4.2.24 tension\_valid

```
CYBIT tension_valid
```

Tension validation bit.

#### 5.4.2.25 timer\_value

```
uint32 timer_value
```

End time of the firmware main loop.

#### 5.4.2.26 timer\_value0

```
uint32 timer_value0
```

Start time of the firmware main loop.

#### 5.4.2.27 watchdog\_flag

```
CYBIT watchdog_flag = FALSE
```

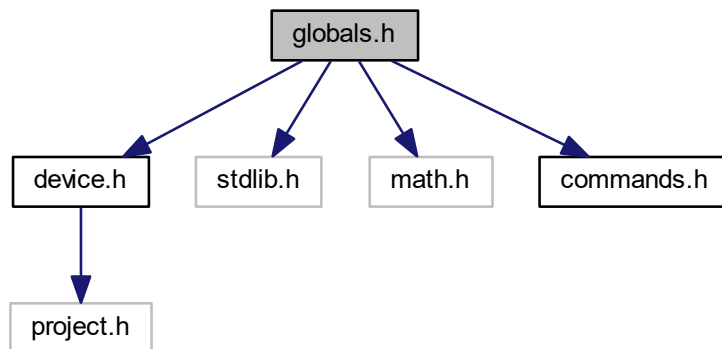
Watchdog flag enabler.

## 5.5 globals.h File Reference

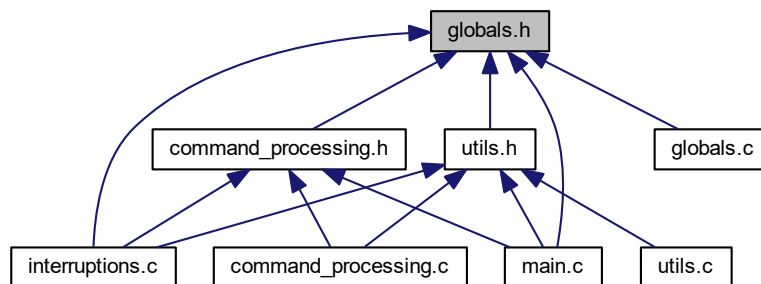
Global definitions and macros are set in this file.

```
#include <device.h>
#include "stdlib.h"
#include "math.h"
#include "commands.h"
```

Include dependency graph for globals.h:



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



### Data Structures

- struct **st\_ref**  
*Motor Reference structure.*
- struct **st\_meas**  
*Measurements structure.*
- struct **st\_data**

*Data sent/received structure.*

- struct **st\_mem**

*EEPROM stored structure.*

- struct **st\_dev**

*Device related structure.*

- struct **st\_calib**

*Hand calibration structure.*

## Macros

- #define **VERSION** "SH-PRO v6.1.6 Centro Piaggio"
- #define **NUM\_OF\_MOTORS** 2
- #define **NUM\_OF\_SENSORS** 3
- #define **NUM\_OF\_EMGS** 2
- #define **NUM\_OF\_ANALOG\_INPUTS** 4
- #define **NUM\_OF\_PARAMS** 29
- #define **CALIBRATION\_DIV** 10
- #define **DIV\_INIT\_VALUE** 1
- #define **DMA\_BYTES\_PER\_BURST** 2
- #define **DMA\_REQUEST\_PER\_BURST** 1
- #define **DMA\_SRC\_BASE** (CYDEV\_PERIPH\_BASE)
- #define **DMA\_DST\_BASE** (CYDEV\_SRAM\_BASE)
- #define **WAIT\_START** 0
- #define **WAIT\_ID** 1
- #define **WAIT\_LENGTH** 2
- #define **RECEIVE** 3
- #define **UNLOAD** 4
- #define **STATE\_INACTIVE** 0
- #define **STATE\_ACTIVE** 1
- #define **COUNTER\_INC** 2
- #define **FALSE** 0
- #define **TRUE** 1
- #define **DEFAULT\_EEPROM\_DISPLACEMENT** 50
- #define **EEPROM\_BYTES\_ROW** 16
- #define **EEPROM\_COUNTERS\_ROWS** 5
- #define **MAX\_WATCHDOG\_TIMER** 250
- #define **PWM\_MAX\_VALUE** 100
- #define **ANTI\_WINDUP** 1000
- #define **DEFAULT\_CURRENT\_LIMIT** 1500
- #define **CURRENT\_HYSTERESIS** 10
- #define **EMG\_SAMPLE\_TO\_DISCARD** 500
- #define **SAMPLES\_FOR\_MEAN** 100
- #define **SAMPLES\_FOR\_EMG\_MEAN** 1000
- #define **CALIB\_DECIMATION** 1
- #define **NUM\_OF\_CLOSURES** 5
- #define **POS\_INTEGRAL\_SAT\_LIMIT** 50000000
- #define **CURR\_INTEGRAL\_SAT\_LIMIT** 100000
- #define **MIN\_CURR\_SAT\_LIMIT** 30
- #define **LOOKUP\_DIM** 6

## Enumerations

- enum **emg\_status** {  
**NORMAL** = 0, **RESET** = 1, **DISCARD** = 2, **SUM\_AND\_MEAN** = 3,  
**WAIT** = 4, **WAIT\_EoC** = 5 }
- enum **counter\_status** {  
**PREPARE\_DATA** = 0, **WRITE\_CYCLES** = 1, **WAIT\_QUERY** = 2, **WRITE\_END** = 3,  
**NONE** = 4 }

## Variables

- struct **st\_ref** g\_ref g\_refNew g\_refOld
- struct **st\_meas** g\_meas g\_measOld
- struct **st\_data** g\_rx
- struct **st\_mem** g\_mem c\_mem
- struct **st\_calib** calib
- uint32 **timer\_value**
- uint32 **timer\_value0**
- float **cycle\_time**
- int32 **dev\_tension**
- uint8 **dev\_pwm\_sat**
- uint8 **dev\_pwm\_limit**
- int32 **dev\_tension\_f**
- int32 **pow\_tension**
- CYBIT **reset\_last\_value\_flag**
- CYBIT **tension\_valid**
- CYBIT **interrupt\_flag**
- CYBIT **watchdog\_flag**
- float **tau\_feedback**
- CYBIT **cycles\_interrupt\_flag**
- CYBIT **can\_write**
- uint8 **rest\_enabled**
- uint8 **forced\_open**
- **counter\_status** CYDATA **cycles\_status**
- **emg\_status** CYDATA **emg\_1\_status**
- **emg\_status** CYDATA **emg\_2\_status**
- int16 **ADC\_buf** [4]
- int8 **pwm\_sign**
- int32 **rest\_pos\_curr\_ref**

### 5.5.1 Detailed Description

Global definitions and macros are set in this file.

#### Date

October 01, 2017

#### Author

*Centro "E.Piaggio"*

#### Copyright

(C) 2012-2016 qbrobotics. All rights reserved.  
 (C) 2017 Centro "E.Piaggio". All rights reserved.

## 5.5.2 Macro Definition Documentation

### 5.5.2.1 ANTI\_WINDUP

```
#define ANTI_WINDUP 1000
```

Anti windup saturation.

### 5.5.2.2 CALIBRATION\_DIV

```
#define CALIBRATION_DIV 10
```

Frequency divisor for hand calibration (100Hz).

### 5.5.2.3 COUNTER\_INC

```
#define COUNTER_INC 2
```

Counter cycle increment.

### 5.5.2.4 CURR\_INTEGRAL\_SAT\_LIMIT

```
#define CURR_INTEGRAL_SAT_LIMIT 100000
```

Anti windup on current control.

### 5.5.2.5 CURRENT\_HYSTERESIS

```
#define CURRENT_HYSTERESIS 10
```

milliAmperes of hysteresis for current control.

### 5.5.2.6 DEFAULT\_CURRENT\_LIMIT

```
#define DEFAULT_CURRENT_LIMIT 1500
```

Default Current limit, 0 stands for unlimited.

### 5.5.2.7 DEFAULT\_EEPROM\_DISPLACEMENT

```
#define DEFAULT_EEPROM_DISPLACEMENT 50
```

Number of pages occupied by the EEPROM.



#### 5.5.2.8 DIV\_INIT\_VALUE

```
#define DIV_INIT_VALUE 1
```

Initial value for hand counter calibration.

#### 5.5.2.9 EEPROM\_BYTES\_ROW

```
#define EEPROM_BYTES_ROW 16
```

EEPROM number of bytes per row.

#### 5.5.2.10 EEPROM\_COUNTERS\_ROWS

```
#define EEPROM_COUNTERS_ROWS 5
```

EEPROM number of rows dedicated to store counters.

#### 5.5.2.11 EMG\_SAMPLE\_TO\_DISCARD

```
#define EMG_SAMPLE_TO_DISCARD 500
```

Number of sample to discard before calibration.

#### 5.5.2.12 LOOKUP\_DIM

```
#define LOOKUP_DIM 6
```

Dimension of the current lookup table.

#### 5.5.2.13 MAX\_WATCHDOG\_TIMER

```
#define MAX_WATCHDOG_TIMER 250
```

num \* 2 [cs].

#### 5.5.2.14 NUM\_OF\_ANALOG\_INPUTS

```
#define NUM_OF_ANALOG_INPUTS 4
```

Total number of analogic inputs.

#### 5.5.2.15 NUM\_OF\_EMGS

```
#define NUM_OF_EMGS 2
```

Number of emg channels.

#### 5.5.2.16 NUM\_OF\_MOTORS

```
#define NUM_OF_MOTORS 2
```

Number of motors.

#### 5.5.2.17 NUM\_OF\_PARAMS

```
#define NUM_OF_PARAMS 29
```

Number of parameters saved in the EEPROM.

#### 5.5.2.18 NUM\_OF\_SENSORS

```
#define NUM_OF_SENSORS 3
```

Number of encoders.

#### 5.5.2.19 POS\_INTEGRAL\_SAT\_LIMIT

```
#define POS_INTEGRAL_SAT_LIMIT 50000000
```

Anti windup on position control.

#### 5.5.2.20 PWM\_MAX\_VALUE

```
#define PWM_MAX_VALUE 100
```

Maximum value of the PWM signal.

#### 5.5.2.21 RECEIVE

```
#define RECEIVE 3
```

Package data receiving status.

#### 5.5.2.22 SAMPLES\_FOR\_EMG\_MEAN

```
#define SAMPLES_FOR_EMG_MEAN 1000
```

Number of samples used to mean emg values.

#### 5.5.2.23 SAMPLES\_FOR\_MEAN

```
#define SAMPLES_FOR_MEAN 100
```

Number of samples used to mean current values.

#### 5.5.2.24 STATE\_ACTIVE

```
#define STATE_ACTIVE 1
```

Closed SoftHand position / EMG Active.

#### 5.5.2.25 STATE\_INACTIVE

```
#define STATE_INACTIVE 0
```

Open SoftHand position / EMG Inactive.

#### 5.5.2.26 UNLOAD

```
#define UNLOAD 4
```

Package data flush status.

#### 5.5.2.27 WAIT\_ID

```
#define WAIT_ID 1
```

Package ID waiting status.

#### 5.5.2.28 WAIT\_LENGTH

```
#define WAIT_LENGTH 2
```

Package lenght waiting status.

#### 5.5.2.29 WAIT\_START

```
#define WAIT_START 0
```

Package start waiting status.

### 5.5.3 Enumeration Type Documentation

#### 5.5.3.1 counter\_status

```
enum counter_status
```

## Enumerator

|              |   |
|--------------|---|
| PREPARE_DATA | Prepare data to be written on EEPROM.   |
| WRITE_CYCLES | Cycles writing on EEPROM is enabled and control is passed to query.                       |
| WAIT_QUERY   | Wait until EEPROM_Query() has finished writing on EEPROM and then disable cycles writing. |
| WRITE_END    | End of EEPROM writing.  |
| NONE         | Cycles writing on EEPROM is disabled.   |

## 5.5.3.2 emg\_status

```
enum emg_status
```

## Enumerator

|              |  |
|--------------|--|
| NORMAL       | Normal execution   |
| RESET        | Reset analog measurements                                  |
| DISCARD      | Discard first samples to obtain a correct value            |
| SUM_AND_MEAN | Sum and mean a definite value of samples                   |
| WAIT         | The second emg waits until the first emg has a valid value |
| WAIT_EoC     | The second emg waits for end of calibration                |

## 5.5.4 Variable Documentation

## 5.5.4.1 c\_mem

```
struct st_mem g_mem c_mem
```

Memory parameters.

## 5.5.4.2 calib

```
struct st_calib calib
```

Calibration variables.

## 5.5.4.3 can\_write

```
CYBIT can_write
```

Write to EEPROM flag.

#### 5.5.4.4 cycle\_time

float cycle\_time

Variable used to calculate how much time a cycle takes.

#### 5.5.4.5 cycles\_interrupt\_flag

CYBIT cycles\_interrupt\_flag

Cycles timer interrupt flag enabler.

#### 5.5.4.6 cycles\_status

**counter\_status** CYDATA cycles\_status

Cycles counter state machine status.

#### 5.5.4.7 dev\_pwm\_limit

uint8 dev\_pwm\_limit

Device pwm limit

Device pwm limit. It may change during execution.

#### 5.5.4.8 dev\_pwm\_sat

uint8 dev\_pwm\_sat

Device pwm limit saturation

Device pwm saturation. By default the saturation value must not exceed 100.

#### 5.5.4.9 dev\_tension

int32 dev\_tension

Power supply tension

Power supply tension.

#### 5.5.4.10 dev\_tension\_f

int32 dev\_tension\_f

Filtered power supply tension

Filtered power supply tension.

#### 5.5.4.11 emg\_1\_status

```
emg_status CYDATA emg_1_status
```

First EMG sensor status.

#### 5.5.4.12 emg\_2\_status

```
emg_status CYDATA emg_2_status
```

Second EMG sensor status.

#### 5.5.4.13 forced\_open

```
uint8 forced_open
```

Forced open flag (used in position with rest position control).

#### 5.5.4.14 g\_measOld

```
struct st_meas g_meas g_measOld
```

Measurements.

#### 5.5.4.15 g\_refOld

```
struct st_ref g_ref g_refNew g_refOld
```

Reference variables.

#### 5.5.4.16 g\_rx

```
struct st_data g_rx
```

Incoming/Outcoming data.

#### 5.5.4.17 interrupt\_flag

```
CYBIT interrupt_flag
```

Interrupt flag enabler.

#### 5.5.4.18 pow\_tension

```
int32 pow_tension
```

Computed power supply tension.

#### 5.5.4.19 pwm\_sign

```
int8 pwm_sign
```

ADC measurements buffer. Sign of pwm driven. Used to obtain current sign.

#### 5.5.4.20 reset\_last\_value\_flag

```
CYBIT reset_last_value_flag
```

This flag is set when the encoders last values must be resetted.

#### 5.5.4.21 rest\_enabled

```
uint8 rest_enabled
```

Rest position flag.

#### 5.5.4.22 rest\_pos\_curr\_ref

```
int32 rest_pos_curr_ref
```

Rest position current reference.

#### 5.5.4.23 tau\_feedback

```
float tau_feedback
```

Torque feedback.

#### 5.5.4.24 tension\_valid

```
CYBIT tension_valid
```

Tension validation bit.

#### 5.5.4.25 timer\_value

```
uint32 timer_value
```

End time of the firmware main loop.

#### 5.5.4.26 timer\_value0

```
uint32 timer_value0
```

Start time of the firmware main loop.

#### 5.5.4.27 watchdog\_flag

CYBIT watchdog\_flag

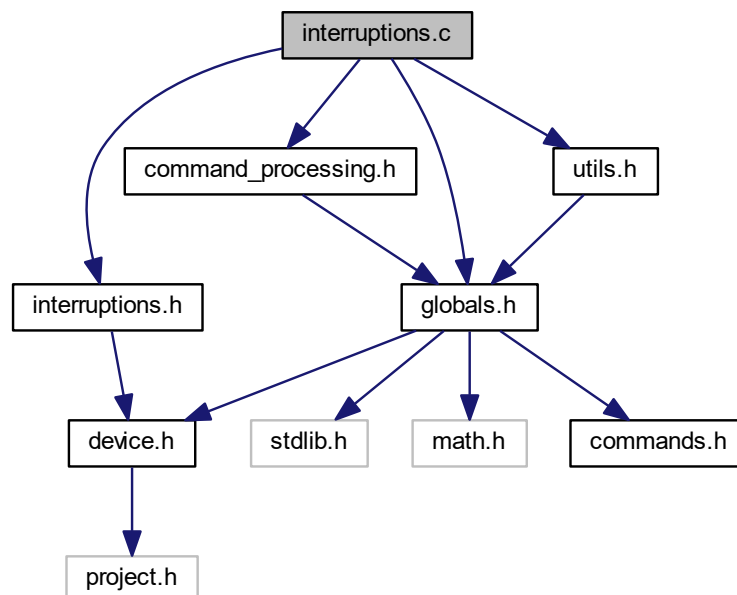
Watchdog flag enabler.

## 5.6 interruptions.c File Reference

Interrupt handling and firmware core functions.

```
#include <interruptions.h>
#include <command_processing.h>
#include <globals.h>
#include <utils.h>
```

Include dependency graph for interruptions.c:



### Functions

- **CY\_ISR** (ISR\_WATCHDOG\_Handler)
- **CY\_ISR** (ISR\_RS485\_RX\_ExInterrupt)
- **CY\_ISR** (ISR\_CYCLES\_Handler)
- void **interrupt\_manager** ()
- void **function\_scheduler** (void)
- void **motor\_control** ()
- void **encoder\_reading** (const uint8 idx)
- void **analog\_read\_end** ()
- void **overcurrent\_control** ()
- void **pwm\_limit\_search** ()
- void **cycles\_counter\_update** ()
- void **save\_cycles\_eeprom** ()



## Variables

- static const uint8 **pwm\_preload\_values** [29]
- CYCODE uint8 **pwm\_preload\_values\_6v** [32]

### 5.6.1 Detailed Description

Interruption handling and firmware core functions.

#### Date

October 01, 2017

#### Author

*Centro "E.Piaggio"*

#### Copyright

(C) 2012-2016 qbrobotics. All rights reserved.  
(C) 2017 Centro "E.Piaggio". All rights reserved.

### 5.6.2 Function Documentation

#### 5.6.2.1 analog\_read\_end()

```
void analog_read_end ( )
```

This function executes and terminates the analog readings.

#### 5.6.2.2 cycles\_counter\_update()

```
void cycles_counter_update ( )
```

This function increases the cycles counters variables, depending on SoftHand position and the current absorbed by the motor.

#### 5.6.2.3 encoder\_reading()

```
void encoder_reading (
    const uint8 index )
```

This functions reads the value from the encoder pointed by index.

**Parameters**

|              |  |
|--------------|--|
| <i>index</i> | The number of the encoder that must be read. |
|--------------|--|

**5.6.2.4 function\_scheduler()**

```
void function_scheduler (
    void )
```

This function schedules the other functions in an order that optimizes the controller usage.

**5.6.2.5 interrupt\_manager()**

```
void interrupt_manager ( )
```

This function is called in predefined moments during firmware execution in order to unpack the received package.

**5.6.2.6 motor\_control()**

```
void motor_control ( )
```

This function controls the motor direction and velocity, depending on the input and control modality set.

**5.6.2.7 overcurrent\_control()**

```
void overcurrent_control ( )
```

This function increases or decreases the pwm maximum value, depending on the current absorbed by the motor.

**5.6.2.8 pwm\_limit\_search()**

```
void pwm_limit_search ( )
```

This function scales the pwm value of the motor, depending on the power supply voltage, in order to not make the motor wind too fast.

**5.6.2.9 save\_cycles\_eeprom()**

```
void save_cycles_eeprom ( )
```

This function saves cycles counters variables into EEPROM memory.

### 5.6.3 Variable Documentation

#### 5.6.3.1 pwm\_preload\_values

```
const uint8 pwm_preload_values[29] [static]
```

**Initial value:**

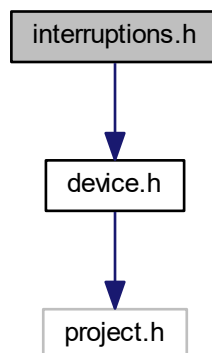
```
= {100,
                                     83,
                                     78,
                                     76,
                                     74,
                                     72,
                                     70,
                                     68,
                                     67,
                                     65,
                                     64,
                                     63,
                                     62,
                                     61,
                                     60,
                                     59,
                                     58,
                                     57,
                                     56,
                                     56,
                                     55,
                                     54,
                                     54,
                                     53,
                                     52,
                                     52,
                                     52,
                                     51,
                                     51}
```

## 5.7 interruptions.h File Reference

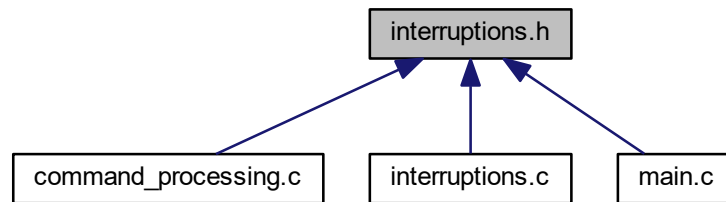
Interruptions header file.

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

Include dependency graph for interruptions.h:



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



## Functions

- void **save\_cycles\_eeprom** ()

### Interruptions

- **CY\_ISR\_PROTO** (ISR\_RS485\_RX\_ExInterrupt)
- **CY\_ISR\_PROTO** (ISR\_WATCHDOG\_Handler)
- **CY\_ISR\_PROTO** (ISR\_CYCLES\_Handler)

### General function scheduler

- void **function\_scheduler** (void)

### Encoder reading function

- void **encoder\_reading** (const uint8 index)

### Motor control function

- void **motor\_control** ()

### Analog readings

- void **analog\_read\_end** ()

### Interrupt manager

- void **interrupt\_manager** ()

### Utility functions

- void **pwm\_limit\_search** ()
- void **overcurrent\_control** ()
- void **cycles\_counter\_update** ()

### 5.7.1 Detailed Description

Interruptions header file.

#### Date

October 01, 2017

#### Author

*Centro "E.Piaggio"*

#### Copyright

(C) 2012-2016 qbrobotics. All rights reserved.

(C) 2017 Centro "E.Piaggio". All rights reserved.

### 5.7.2 Function Documentation

#### 5.7.2.1 analog\_read\_end()

```
void analog_read_end ( )
```

This function executes and terminates the analog readings.

#### 5.7.2.2 CY\_ISR\_PROTO() [1/3]

```
CY_ISR_PROTO (
    ISR_RS485_RX_ExInterrupt )
```

This interruption sets a flag to let the firmware know that a communication interruption is pending and needs to be handled. The interruption will be handled in predefined moments during the firmware execution. When this interruption is handled, it unpacks the package received on the RS485 communication bus.

#### 5.7.2.3 CY\_ISR\_PROTO() [2/3]

```
CY_ISR_PROTO (
    ISR_WATCHDOG_Handler )
```

This interruption sets a flag to let the firmware know that a watchdog interruption is pending and needs to be handled. The interruption will be handled in predefined moments during the firmware execution. When this interruption is handled, it deactivates the board because the watchdog timer has expired.

#### 5.7.2.4 CY\_ISR\_PROTO() [3/3]

```
CY_ISR_PROTO (
    ISR_CYCLES_Handler )
```

This interruption sets a flag to let the firmware know that a cycles timer interruption is pending and needs to be handled. The interruption will be handled in predefined moments during the firmware execution. When this interruption is handled, it updates cycles counters.

#### 5.7.2.5 cycles\_counter\_update()

```
void cycles_counter_update ( )
```

This function increases the cycles counters variables, depending on SoftHand position and the current absorbed by the motor.

#### 5.7.2.6 encoder\_reading()

```
void encoder_reading (
    const uint8 index )
```

This functions reads the value from the encoder pointed by index.

##### Parameters

|              |  |
|--------------|--|
| <i>index</i> | The number of the encoder that must be read. |
|--------------|--|

#### 5.7.2.7 function\_scheduler()

```
void function_scheduler (
    void )
```

This function schedules the other functions in an order that optimizes the controller usage.

#### 5.7.2.8 interrupt\_manager()

```
void interrupt_manager ( )
```

This function is called in predefined moments during firmware execution in order to unpack the received package.

#### 5.7.2.9 motor\_control()

```
void motor_control ( )
```

This function controls the motor direction and velocity, depending on the input and control modality set.

#### 5.7.2.10 overcurrent\_control()

```
void overcurrent_control ( )
```

This function increases or decreases the pwm maximum value, depending on the current absorbed by the motor.

#### 5.7.2.11 pwm\_limit\_search()

```
void pwm_limit_search ( )
```

This function scales the pwm value of the motor, depending on the power supply voltage, in order to not make the motor wind too fast.

#### 5.7.2.12 save\_cycles\_eeprom()

```
void save_cycles_eeprom ( )
```

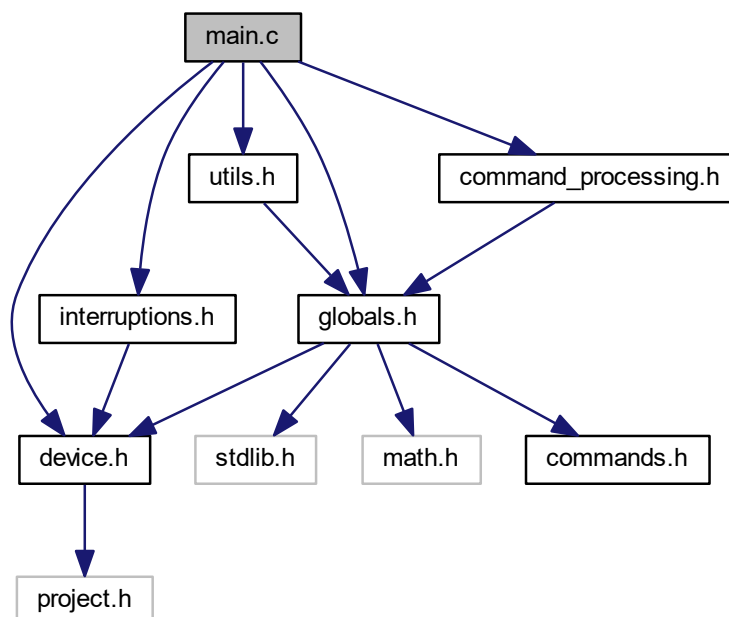
This function saves cycles counters variables into EEPROM memory.

## 5.8 main.c File Reference

Firmware main file.

```
#include <device.h>
#include <globals.h>
#include <interruptions.h>
#include <command_processing.h>
#include <utils.h>
```

Include dependency graph for main.c:



## Functions

- int **main** ()

### 5.8.1 Detailed Description

Firmware main file.

#### Date

October 01, 2017

#### Author

*Centro "E.Piaggio"*

#### Copyright

(C) 2012-2016 qprobotics. All rights reserved.  
(C) 2017 Centro "E.Piaggio". All rights reserved.

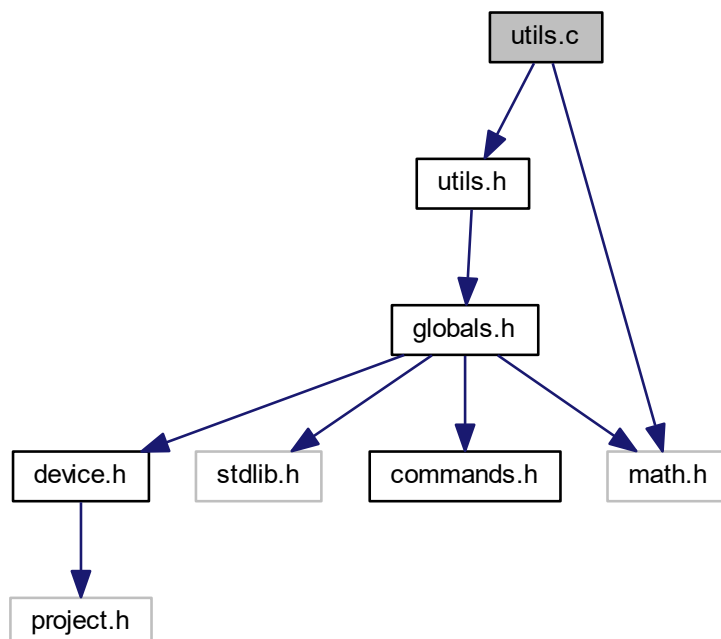
## 5.9 utils.c File Reference

Definition of utility functions.

```
#include <utils.h>
```

```
#include <math.h>
```

Include dependency graph for utils.c:





## Macros

- `#define N1 15`
- `#define N2 14`
- `#define I1 1`  
*First wheel invariant value.*
- `#define I2 (-1)`  
*Second wheel invariant value.*
- `#define M 65536`  
*Number of encoder ticks per turn.*

## Functions

- `int32 curr_estim` (int32 pos, int32 vel, int32 ref)
- `int32 filter_v` (int32 new\_value)
- `int32 filter_i1` (int32 new\_value)
- `int32 filter_ch1` (int32 new\_value)
- `int32 filter_ch2` (int32 new\_value)
- `int32 filter_vel_1` (int32 new\_value)
- `int32 filter_vel_2` (int32 new\_value)
- `int32 filter_vel_3` (int32 new\_value)
- `int32 filter_acc_1` (int32 new\_value)
- `int32 filter_acc_2` (int32 new\_value)
- `int32 filter_acc_3` (int32 new\_value)
- `int32 filter_curr_diff` (int32 new\_value)
- `int32 filter_voltage` (int32 new\_value)
- `CYBIT check_enc_data` (const uint32 \*value)
- `int my_round` (const double x)
- `uint32 my_mod` (int32 val, int32 divisor)
- `void calibration` (void)
- `int calc_turns_fcn` (const int32 pos1, const int32 pos2)
- `void check_rest_position` (void)
- `void reset_counters` ()

### 5.9.1 Detailed Description

Definition of utility functions.

#### Date

October 01, 2017

#### Author

*Centro "E.Piaggio"*

#### Copyright

(C) 2012-2016 qrobotics. All rights reserved.  
(C) 2017 Centro "E.Piaggio". All rights reserved.

## 5.9.2 Function Documentation

### 5.9.2.1 `calc_turns_fcn()`

```
int calc_turns_fcn (
    const int32 pos1,
    const int32 pos2 )
```

This function is used at startup to reconstruct the correct turn of the shaft connected to the motor. It need two encoders to work.

#### Parameters

|             |                         |
|-------------|-------------------------|
| <i>pos1</i> | First encoder position  |
| <i>pos2</i> | Second encoder position |

#### Returns

Returns the number of turns of motor pulley at startup

### 5.9.2.2 `calibration()`

```
void calibration ( )
```

This function counts a series of hand opening and closing used to execute a calibration of the device.

### 5.9.2.3 `check_enc_data()`

```
CYBIT check_enc_data (
    const uint32 * value )
```

This function controls if the read encoder data is correct or not.

#### Parameters

|              |                                    |
|--------------|------------------------------------|
| <i>value</i> | A pointer to the encoder data read |
|--------------|------------------------------------|

#### Returns

Returns 1 if the read data is correct, 0 otherwise

#### 5.9.2.4 check\_rest\_position()

```
void check_rest_position ( )
```

This function checks for rest position and, in case, gives a position reference to qbhand.

#### 5.9.2.5 curr\_estim()

```
int32 curr_estim (
    int32 pos,
    int32 vel,
    int32 acc )
```

Function used to obtain current estimation through current lookup table.

##### Parameters

|              |                                   |
|--------------|-----------------------------------|
| <i>pos</i>   | Position of the encoder in ticks. |
| <i>vel</i>   | Speed of the encoder.             |
| <i>accel</i> | Acceleration of the encoder       |

##### Returns

Returns an estimation of the motor current, depending on its position, velocity and acceleration.

#### 5.9.2.6 filter\_acc\_1()

```
int32 filter_acc_1 (
    int32 value )
```

Filter on first encoder rotational acceleration. The weighted average between the old value and the new one is executed.

##### Parameters

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

##### Returns

Returns the filtered first encoder rotational acceleration value

#### 5.9.2.7 filter\_acc\_2()

```
int32 filter_acc_2 (
    int32 value )
```

Filter on second encoder rotation acceleration (if present). The weighted average between the old value and the new one is executed.

#### Parameters

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

#### Returns

Returns the filtered second encoder rotational acceleration value

#### 5.9.2.8 filter\_acc\_3()

```
int32 filter_acc_3 (  
    int32 value )
```

Filter on third encoder rotation acceleration (if present). The weighted average between the old value and the new one is executed.

#### Parameters

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

#### Returns

Returns the filtered third encoder rotational acceleration value

#### 5.9.2.9 filter\_ch1()

```
int32 filter_ch1 (  
    int32 value )
```

Filter on the first EMG sensor converted value. The weighted average between the old value and the new one is executed.

#### Parameters

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

#### Returns

Returns the filtered emg sensor value

#### 5.9.2.10 filter\_ch2()

```
int32 filter_ch2 (  
    int32 value )
```

Filter on the second EMG sensor converted value. The weighted average between the old value and the new one is executed.

##### Parameters

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

##### Returns

Returns the filtered emg sensor value

#### 5.9.2.11 filter\_curr\_diff()

```
int32 filter_curr_diff (  
    int32 curr_diff )
```

Low pass filter on current difference between measured and estimated current

##### Parameters

|                  |  |
|------------------|--|
| <i>curr_diff</i> | Difference between the measured current and the estimated one. |
|------------------|--|

##### Returns

Returns the filtered current difference value

#### 5.9.2.12 filter\_i1()

```
int32 filter_i1 (  
    int32 value )
```

Filter on the motor current converted value. The weighted average between the old value and the new one is executed.

##### Parameters

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

**Returns**

Returns the filtered current value

**5.9.2.13 filter\_v()**

```
int32 filter_v (
    int32 new_value )
```

Filter on the converted voltage value. The weighted average between the old value and the new one is executed.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <i>new_value</i> | New value of the filter. |
|------------------|--------------------------|

**Returns**

Returns the filtered voltage value

**5.9.2.14 filter\_vel\_1()**

```
int32 filter_vel_1 (
    int32 value )
```

Filter on first encoder rotational speed. The weighted average between the old value and the new one is executed.

**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

**Returns**

Returns the filtered first encoder rotational speed value

**5.9.2.15 filter\_vel\_2()**

```
int32 filter_vel_2 (
    int32 value )
```

Filter on second encoder rotational speed (if present). The weighted average between the old value and the new one is executed.

**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

**Returns**

Returns the filtered second encoder rotational speed value

**5.9.2.16 filter\_vel\_3()**

```
int32 filter_vel_3 (  
    int32 value )
```

Filter on third encoder rotational speed (if present). The weighted average between the old value and the new one is executed.

**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

**Returns**

Returns the filtered third encoder rotational speed value

**5.9.2.17 filter\_voltage()**

```
int32 filter_voltage (  
    int32 value )
```

Filter on voltage readings. The weighted average between the old value and the new one is executed.

**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

**Returns**

Returns the filtered voltage value

**5.9.2.18 my\_mod()**

```
uint32 my_mod (  
    int32 val,  
    int32 divisor )
```

This function computes the module function, returning positive values regardless of whether the value passed is negative

**Parameters**

|                |   |
|----------------|---|
| <i>val</i>     | The value of which the module needs to be calculated    |
| <i>divisor</i> | The divisor according to which the module is calculated |

**5.9.2.19 my\_round()**

```
int my_round (
    const double x )
```

This function approximates the value passed to the nearest integer

**Parameters**

|          |   |
|----------|---|
| <i>x</i> | The floating point value that needs to be rounded |
|----------|---|

**5.9.2.20 reset\_counters()**

```
void reset_counters ( )
```

This function resets statistics counters

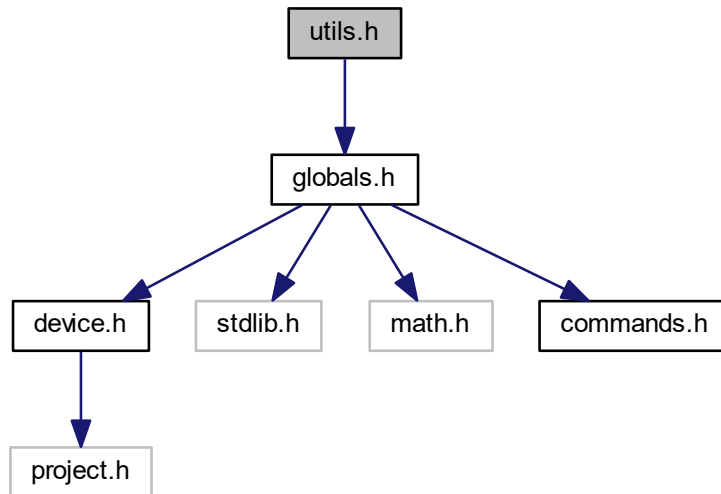
**5.10 utils.h File Reference**

Utility functions declaration.

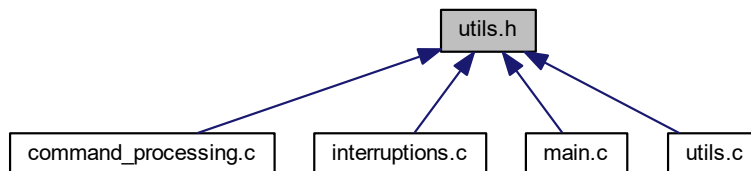


```
#include <globals.h>
```

Include dependency graph for utils.h:



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



## Macros

- `#define TIMER_CLOCK 10000`
- `#define ALPHA 32`  
*Voltage and current filters constant.*
- `#define BETA 50`  
*Emg filters constant.*
- `#define GAMMA 128`  
*Velocity filters constant.*
- `#define DELTA 8`  
*Acceleration filters constant.*
- `#define ETA 16`  
*Current residual filter.*

- `#define EPSILON 8`  
*Voltage readings filter.*
- `#define ZMAX 5`
- `#define ZERO_TOL 100`
- `#define REFSPEED 20`
- `#define SIGN(A) (((A) >=0) ? (1) : (-1))`

## Functions

- `int32 filter_voltage (int32 value)`

### Filters

- `int32 filter_v (int32 new_value)`
- `int32 filter_ch1 (int32 value)`
- `int32 filter_ch2 (int32 value)`
- `int32 filter_i1 (int32 value)`
- `int32 filter_vel_1 (int32 value)`
- `int32 filter_vel_2 (int32 value)`
- `int32 filter_vel_3 (int32 value)`
- `int32 filter_acc_1 (int32 value)`
- `int32 filter_acc_2 (int32 value)`
- `int32 filter_acc_3 (int32 value)`

### Estimating current and difference

- `int32 curr_estim (int32 pos, int32 vel, int32 acc)`
- `int32 filter_curr_diff (int32 curr_diff)`

### Utility functions

- `int my_round (const double x)`
- `uint32 my_mod (int32 val, int32 divisor)`
- `CYBIT check_enc_data (const uint32 *value)`
- `int calc_turns_fcn (const int32 pos1, const int32 pos2)`
- `void calibration ()`
- `void check_rest_position ()`
- `void reset_counters ()`

## 5.10.1 Detailed Description

Utility functions declaration.

### Date

October 01, 2017

### Author

*Centro "E.Piaggio"*

### Copyright

(C) 2012-2016 qrobotics. All rights reserved.  
(C) 2017 Centro "E.Piaggio". All rights reserved.

## 5.10.2 Macro Definition Documentation

### 5.10.2.1 REFSPEED

```
#define REFSPEED 20
```

Constant depending on PID values.

### 5.10.2.2 SIGN

```
#define SIGN(  
    A ) ((A) >=0) ? (1) : (-1)
```

Sign calculation function.

### 5.10.2.3 ZERO\_TOL

```
#define ZERO_TOL 100
```

Deadband used to put to zero the virtual position due to the fact that the friction model has errors when the position is near to zero.

### 5.10.2.4 ZMAX

```
#define ZMAX 5
```

Constant useful for current estimation procedure.

## 5.10.3 Function Documentation

### 5.10.3.1 calc\_turns\_fcn()

```
int calc_turns_fcn (  
    const int32 pos1,  
    const int32 pos2 )
```

This function is used at startup to reconstruct the correct turn of the shaft connected to the motor. It need two encoders to work.

#### Parameters

|             |                         |
|-------------|-------------------------|
| <i>pos1</i> | First encoder position  |
| <i>pos2</i> | Second encoder position |

**Returns**

Returns the number of turns of motor pulley at startup

**5.10.3.2 calibration()**

```
void calibration ( )
```

This function counts a series of hand opening and closing used to execute a calibration of the device.

**5.10.3.3 check\_enc\_data()**

```
CYBIT check_enc_data (
    const uint32 * value )
```

This function controls if the read encoder data is correct or not.

**Parameters**

|              |                                    |
|--------------|------------------------------------|
| <i>value</i> | A pointer to the encoder data read |
|--------------|------------------------------------|

**Returns**

Returns 1 if the read data is correct, 0 otherwise

**5.10.3.4 check\_rest\_position()**

```
void check_rest_position ( )
```

This function checks for rest position and, in case, gives a position reference to qbhand.

**5.10.3.5 curr\_estim()**

```
int32 curr_estim (
    int32 pos,
    int32 vel,
    int32 acc )
```

Function used to obtain current estimation through current lookup table.

**Parameters**

|              |                                   |
|--------------|-----------------------------------|
| <i>pos</i>   | Position of the encoder in ticks. |
| <i>vel</i>   | Speed of the encoder.             |
| <i>accel</i> | Acceleration of the encoder       |

**Returns**

Returns an estimation of the motor current, depending on its position, velocity and acceleration.

**5.10.3.6 filter\_acc\_1()**

```
int32 filter_acc_1 (  
    int32 value )
```

Filter on first encoder rotational acceleration. The weighted average between the old value and the new one is executed.

**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

**Returns**

Returns the filtered first encoder rotational acceleration value

**5.10.3.7 filter\_acc\_2()**

```
int32 filter_acc_2 (  
    int32 value )
```

Filter on second encoder rotation acceleration (if present). The weighted average between the old value and the new one is executed.

**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

**Returns**

Returns the filtered second encoder rotational acceleration value

**5.10.3.8 filter\_acc\_3()**

```
int32 filter_acc_3 (  
    int32 value )
```

Filter on third encoder rotation acceleration (if present). The weighted average between the old value and the new one is executed.

**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

**Returns**

Returns the filtered third encoder rotational acceleration value

**5.10.3.9 filter\_ch1()**

```
int32 filter_ch1 (  
    int32 value )
```

Filter on the first EMG sensor converted value. The weighted average between the old value and the new one is executed.

**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

**Returns**

Returns the filtered emg sensor value

**5.10.3.10 filter\_ch2()**

```
int32 filter_ch2 (  
    int32 value )
```

Filter on the second EMG sensor converted value. The weighted average between the old value and the new one is executed.

**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

**Returns**

Returns the filtered emg sensor value

#### 5.10.3.11 filter\_curr\_diff()

```
int32 filter_curr_diff (
    int32 curr_diff )
```

Low pass filter on current difference between measured and estimated current

##### Parameters

|                  |  |
|------------------|--|
| <i>curr_diff</i> | Difference between the measured current and the estimated one. |
|------------------|--|

##### Returns

Returns the filtered current difference value

#### 5.10.3.12 filter\_i1()

```
int32 filter_i1 (
    int32 value )
```

Filter on the motor current converted value. The weighted average between the old value and the new one is executed.

##### Parameters

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

##### Returns

Returns the filtered current value

#### 5.10.3.13 filter\_v()

```
int32 filter_v (
    int32 new_value )
```

Filter on the converted voltage value. The weighted average between the old value and the new one is executed.

##### Parameters

|                  |                          |
|------------------|--------------------------|
| <i>new_value</i> | New value of the filter. |
|------------------|--------------------------|

**Returns**

Returns the filtered voltage value

**5.10.3.14 filter\_vel\_1()**

```
int32 filter_vel_1 (
    int32 value )
```

Filter on first encoder rotational speed. The weighted average between the old value and the new one is executed.

**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

**Returns**

Returns the filtered first encoder rotational speed value

**5.10.3.15 filter\_vel\_2()**

```
int32 filter_vel_2 (
    int32 value )
```

Filter on second encoder rotational speed (if present). The weighted average between the old value and the new one is executed.

**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

**Returns**

Returns the filtered second encoder rotational speed value

**5.10.3.16 filter\_vel\_3()**

```
int32 filter_vel_3 (
    int32 value )
```

Filter on third encoder rotational speed (if present). The weighted average between the old value and the new one is executed.



**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

**Returns**

Returns the filtered third encoder rotational speed value

**5.10.3.17 filter\_voltage()**

```
int32 filter_voltage (  
    int32 value )
```

Filter on voltage readings. The weighted average between the old value and the new one is executed.

**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>value</i> | New value of the filter. |
|--------------|--------------------------|

**Returns**

Returns the filtered voltage value

**5.10.3.18 my\_mod()**

```
uint32 my_mod (  
    int32 val,  
    int32 divisor )
```

This function computes the module function, returning positive values regardless of wheter the value passed is negative

**Parameters**

|                |   |
|----------------|---|
| <i>val</i>     | The value of which the module needs to be calculated    |
| <i>divisor</i> | The divisor according to which the module is calculated |

**5.10.3.19 my\_round()**

```
int my_round (  
    const double x )
```

This functions approximates the value passed to the nearest integer

## Parameters

|   |   |
|---|---|
| x | The floating point value that needs to be rounded |
|---|---|

## 5.10.3.20 reset\_counters()

```
void reset_counters ( )
```

This function reset statistics counters



# Index

- ANTI\_WINDUP
  - globals.h, 50
- acc
  - st\_meas, 11
- activ
  - st\_mem, 13
- activate\_pwm\_rescaling
  - st\_mem, 13
- analog\_read\_end
  - interruptions.c, 59
  - interruptions.h, 63
- baud\_rate
  - st\_mem, 13
- buffer
  - st\_data, 8
- c\_mem
  - globals.c, 43
  - globals.h, 54
- CALIBRATION\_DIV
  - globals.h, 50
- COUNTER\_INC
  - globals.h, 50
- CURR\_INTEGRAL\_SAT\_LIMIT
  - globals.h, 50
- CURRENT\_HYSTERESIS
  - globals.h, 50
- CY\_ISR\_PROTO
  - interruptions.h, 63
- calc\_turns\_fcn
  - utils.c, 68
  - utils.h, 77
- calib
  - globals.c, 43
  - globals.h, 54
- calibration
  - utils.c, 68
  - utils.h, 78
- can\_write
  - globals.c, 43
  - globals.h, 54
- check\_enc\_data
  - utils.c, 68
  - utils.h, 78
- check\_rest\_position
  - utils.c, 68
  - utils.h, 78
- cmd\_activate
  - command\_processing.c, 23
  - command\_processing.h, 31
- cmd\_get\_accelerations
  - command\_processing.c, 23
  - command\_processing.h, 31
- cmd\_get\_activate
  - command\_processing.c, 23
  - command\_processing.h, 31
- cmd\_get\_curr\_and\_meas
  - command\_processing.c, 23
  - command\_processing.h, 31
- cmd\_get\_currents
  - command\_processing.c, 23
  - command\_processing.h, 31
- cmd\_get\_currents\_for\_cuff
  - command\_processing.c, 23
  - command\_processing.h, 31
- cmd\_get\_emg
  - command\_processing.c, 23
  - command\_processing.h, 31
- cmd\_get\_inputs
  - command\_processing.c, 23
  - command\_processing.h, 31
- cmd\_get\_measurements
  - command\_processing.c, 24
  - command\_processing.h, 32
- cmd\_get\_velocities
  - command\_processing.c, 24
  - command\_processing.h, 32
- cmd\_ping
  - command\_processing.c, 24
  - command\_processing.h, 32
- cmd\_set\_baudrate
  - command\_processing.c, 24
  - command\_processing.h, 32
- cmd\_set\_inputs
  - command\_processing.c, 24
  - command\_processing.h, 32
- cmd\_set\_watchdog
  - command\_processing.c, 24
  - command\_processing.h, 32
- cmd\_store\_params
  - command\_processing.c, 24
  - command\_processing.h, 32
- commProcess
  - command\_processing.c, 25
  - command\_processing.h, 33
- commWrite
  - command\_processing.c, 25
  - command\_processing.h, 33

- commWrite\_old\_id
  - command\_processing.c, 25
  - command\_processing.h, 33
- commWrite\_to\_cuff
  - command\_processing.c, 25
  - command\_processing.h, 33
- command\_processing.c, 21
  - cmd\_activate, 23
  - cmd\_get\_accelerations, 23
  - cmd\_get\_activate, 23
  - cmd\_get\_curr\_and\_meas, 23
  - cmd\_get\_currents, 23
  - cmd\_get\_currents\_for\_cuff, 23
  - cmd\_get\_emg, 23
  - cmd\_get\_inputs, 23
  - cmd\_get\_measurements, 24
  - cmd\_get\_velocities, 24
  - cmd\_ping, 24
  - cmd\_set\_baudrate, 24
  - cmd\_set\_inputs, 24
  - cmd\_set\_watchdog, 24
  - cmd\_store\_params, 24
- commProcess, 25
- commWrite, 25
- commWrite\_old\_id, 25
- commWrite\_to\_cuff, 25
- get\_param\_list, 26
- infoGet, 26
- infoSend, 26
- LCRChecksum, 26
- memInit, 27
- memRecall, 27
- memRestore, 27
- memStore, 27
- prepare\_counter\_info, 28
- prepare\_generic\_info, 28
- sendAcknowledgment, 28
- setZeros, 29
- command\_processing.h, 29
  - cmd\_activate, 31
  - cmd\_get\_accelerations, 31
  - cmd\_get\_activate, 31
  - cmd\_get\_curr\_and\_meas, 31
  - cmd\_get\_currents, 31
  - cmd\_get\_currents\_for\_cuff, 31
  - cmd\_get\_emg, 31
  - cmd\_get\_inputs, 31
  - cmd\_get\_measurements, 32
  - cmd\_get\_velocities, 32
  - cmd\_ping, 32
  - cmd\_set\_baudrate, 32
  - cmd\_set\_inputs, 32
  - cmd\_set\_watchdog, 32
  - cmd\_store\_params, 32
- commProcess, 33
- commWrite, 33
- commWrite\_old\_id, 33
- commWrite\_to\_cuff, 33
- get\_param\_list, 34
- infoGet, 34
- infoSend, 34
- LCRChecksum, 34
- memInit, 35
- memRecall, 35
- memRestore, 35
- memStore, 35
- prepare\_counter\_info, 36
- prepare\_generic\_info, 36
- sendAcknowledgment, 36
- setZeros, 37
- commands.h, 37
  - qbmove\_command, 39
  - qbmove\_control\_mode, 40
  - qbmove\_input\_mode, 40
  - qbmove\_parameter, 41
- control\_mode
  - st\_mem, 13
- counter\_status
  - globals.h, 53
- curr
  - st\_meas, 11
  - st\_ref, 20
- curr\_estim
  - utils.c, 69
  - utils.h, 78
- curr\_lookup
  - st\_mem, 13
- current\_hist
  - st\_mem, 13
- current\_limit
  - st\_mem, 14
- cycle\_time
  - globals.c, 43
  - globals.h, 54
- cycles\_counter\_update
  - interruptions.c, 59
  - interruptions.h, 64
- cycles\_interrupt\_flag
  - globals.c, 44
  - globals.h, 55
- cycles\_status
  - globals.c, 44
  - globals.h, 55
- DEFAULT\_CURRENT\_LIMIT
  - globals.h, 50
- DEFAULT\_EEPROM\_DISPLACEMENT
  - globals.h, 50
- DIV\_INIT\_VALUE
  - globals.h, 50
- dev\_pwm\_limit
  - globals.c, 44
  - globals.h, 55
- dev\_pwm\_sat
  - globals.c, 44
  - globals.h, 55
- dev\_tension

- globals.c, 44
- globals.h, 55
- dev\_tension\_f
  - globals.c, 44
  - globals.h, 55
- direction
  - st\_calib, 7
- double\_encoder\_on\_off
  - st\_mem, 14
- EEPROM\_BYTES\_ROW
  - globals.h, 51
- EEPROM\_COUNTERS\_ROWS
  - globals.h, 51
- EMG\_SAMPLE\_TO\_DISCARD
  - globals.h, 51
- emg
  - st\_meas, 11
- emg\_1\_status
  - globals.c, 44
  - globals.h, 55
- emg\_2\_status
  - globals.c, 44
  - globals.h, 56
- emg\_calibration\_flag
  - st\_mem, 14
- emg\_counter
  - st\_mem, 14
- emg\_max\_value
  - st\_mem, 14
- emg\_speed
  - st\_mem, 14
- emg\_status
  - globals.h, 54
- emg\_threshold
  - st\_mem, 14
- enabled
  - st\_calib, 7
- encoder\_reading
  - interruptions.c, 59
  - interruptions.h, 64
- filter\_acc\_1
  - utils.c, 69
  - utils.h, 79
- filter\_acc\_2
  - utils.c, 69
  - utils.h, 79
- filter\_acc\_3
  - utils.c, 70
  - utils.h, 79
- filter\_ch1
  - utils.c, 70
  - utils.h, 80
- filter\_ch2
  - utils.c, 70
  - utils.h, 80
- filter\_curr\_diff
  - utils.c, 71
- utils.h, 80
- filter\_i1
  - utils.c, 71
  - utils.h, 81
- filter\_v
  - utils.c, 72
  - utils.h, 81
- filter\_vel\_1
  - utils.c, 72
  - utils.h, 82
- filter\_vel\_2
  - utils.c, 72
  - utils.h, 82
- filter\_vel\_3
  - utils.c, 73
  - utils.h, 82
- filter\_voltage
  - utils.c, 73
  - utils.h, 83
- flag
  - st\_mem, 14
- forced\_open
  - globals.c, 45
  - globals.h, 56
- function\_scheduler
  - interruptions.c, 60
  - interruptions.h, 64
- g\_measOld
  - globals.c, 45
  - globals.h, 56
- g\_refOld
  - globals.c, 45
  - globals.h, 56
- g\_rx
  - globals.c, 45
  - globals.h, 56
- get\_param\_list
  - command\_processing.c, 26
  - command\_processing.h, 34
- globals.c, 42
  - c\_mem, 43
  - calib, 43
  - can\_write, 43
  - cycle\_time, 43
  - cycles\_interrupt\_flag, 44
  - cycles\_status, 44
  - dev\_pwm\_limit, 44
  - dev\_pwm\_sat, 44
  - dev\_tension, 44
  - dev\_tension\_f, 44
  - emg\_1\_status, 44
  - emg\_2\_status, 44
  - forced\_open, 45
  - g\_measOld, 45
  - g\_refOld, 45
  - g\_rx, 45
  - interrupt\_flag, 45
  - pow\_tension, 45

- pwm\_sign, 45
- reset\_last\_value\_flag, 45
- rest\_enabled, 46
- rest\_pos\_curr\_ref, 46
- tau\_feedback, 46
- tension\_valid, 46
- timer\_value, 46
- timer\_value0, 46
- watchdog\_flag, 46
- globals.h, 47
  - ANTI\_WINDUP, 50
  - c\_mem, 54
  - CALIBRATION\_DIV, 50
  - COUNTER\_INC, 50
  - CURR\_INTEGRAL\_SAT\_LIMIT, 50
  - CURRENT\_HYSTERESIS, 50
  - calib, 54
  - can\_write, 54
  - counter\_status, 53
  - cycle\_time, 54
  - cycles\_interrupt\_flag, 55
  - cycles\_status, 55
  - DEFAULT\_CURRENT\_LIMIT, 50
  - DEFAULT\_EEPROM\_DISPLACEMENT, 50
  - DIV\_INIT\_VALUE, 50
  - dev\_pwm\_limit, 55
  - dev\_pwm\_sat, 55
  - dev\_tension, 55
  - dev\_tension\_f, 55
  - EEPROM\_BYTES\_ROW, 51
  - EEPROM\_COUNTERS\_ROWS, 51
  - EMG\_SAMPLE\_TO\_DISCARD, 51
  - emg\_1\_status, 55
  - emg\_2\_status, 56
  - emg\_status, 54
  - forced\_open, 56
  - g\_measOld, 56
  - g\_refOld, 56
  - g\_rx, 56
  - interrupt\_flag, 56
  - LOOKUP\_DIM, 51
  - MAX\_WATCHDOG\_TIMER, 51
  - NUM\_OF\_ANALOG\_INPUTS, 51
  - NUM\_OF\_EMGS, 51
  - NUM\_OF\_MOTORS, 51
  - NUM\_OF\_PARAMS, 52
  - NUM\_OF\_SENSORS, 52
  - POS\_INTEGRAL\_SAT\_LIMIT, 52
  - PWM\_MAX\_VALUE, 52
  - pow\_tension, 56
  - pwm\_sign, 56
  - RECEIVE, 52
  - reset\_last\_value\_flag, 57
  - rest\_enabled, 57
  - rest\_pos\_curr\_ref, 57
  - SAMPLES\_FOR\_EMG\_MEAN, 52
  - SAMPLES\_FOR\_MEAN, 52
  - STATE\_ACTIVE, 52
  - STATE\_INACTIVE, 53
  - tau\_feedback, 57
  - tension\_valid, 57
  - timer\_value, 57
  - timer\_value0, 57
  - UNLOAD, 53
  - WAIT\_ID, 53
  - WAIT\_LENGTH, 53
  - WAIT\_START, 53
  - watchdog\_flag, 57
- id
  - st\_mem, 15
- ind
  - st\_data, 8
- infoGet
  - command\_processing.c, 26
  - command\_processing.h, 34
- infoSend
  - command\_processing.c, 26
  - command\_processing.h, 34
- input\_mode
  - st\_mem, 15
- interrupt\_flag
  - globals.c, 45
  - globals.h, 56
- interrupt\_manager
  - interruptions.c, 60
  - interruptions.h, 64
- interruptions.c, 58
  - analog\_read\_end, 59
  - cycles\_counter\_update, 59
  - encoder\_reading, 59
  - function\_scheduler, 60
  - interrupt\_manager, 60
  - motor\_control, 60
  - overcurrent\_control, 60
  - pwm\_limit\_search, 60
  - pwm\_preload\_values, 61
  - save\_cycles\_eeprom, 60
- interruptions.h, 61
  - analog\_read\_end, 63
  - CY\_ISR\_PROTO, 63
  - cycles\_counter\_update, 64
  - encoder\_reading, 64
  - function\_scheduler, 64
  - interrupt\_manager, 64
  - motor\_control, 64
  - overcurrent\_control, 64
  - pwm\_limit\_search, 65
  - save\_cycles\_eeprom, 65
- k\_d
  - st\_mem, 15
- k\_d\_c
  - st\_mem, 15
- k\_d\_c\_dl
  - st\_mem, 15
- k\_d\_dl



- st\_mem, 15
- k\_i
  - st\_mem, 15
- k\_i\_c
  - st\_mem, 15
- k\_i\_c\_dl
  - st\_mem, 16
- k\_i\_dl
  - st\_mem, 16
- k\_p
  - st\_mem, 16
- k\_p\_c
  - st\_mem, 16
- k\_p\_c\_dl
  - st\_mem, 16
- k\_p\_dl
  - st\_mem, 16
- LCRChecksum
  - command\_processing.c, 26
  - command\_processing.h, 34
- LOOKUP\_DIM
  - globals.h, 51
- length
  - st\_data, 9
- m\_mult
  - st\_mem, 16
- m\_off
  - st\_mem, 16
- MAX\_WATCHDOG\_TIMER
  - globals.h, 51
- main.c, 65
- maint\_day
  - st\_mem, 17
- maint\_month
  - st\_mem, 17
- maint\_year
  - st\_mem, 17
- max\_step\_neg
  - st\_mem, 17
- max\_step\_pos
  - st\_mem, 17
- memInit
  - command\_processing.c, 27
  - command\_processing.h, 35
- memRecall
  - command\_processing.c, 27
  - command\_processing.h, 35
- memRestore
  - command\_processing.c, 27
  - command\_processing.h, 35
- memStore
  - command\_processing.c, 27
  - command\_processing.h, 35
- motor\_control
  - interruptions.c, 60
  - interruptions.h, 64
- motor\_handle\_ratio
  - st\_mem, 17
- my\_mod
  - utils.c, 73
  - utils.h, 83
- my\_round
  - utils.c, 74
  - utils.h, 83
- NUM\_OF\_ANALOG\_INPUTS
  - globals.h, 51
- NUM\_OF\_EMGS
  - globals.h, 51
- NUM\_OF\_MOTORS
  - globals.h, 51
- NUM\_OF\_PARAMS
  - globals.h, 52
- NUM\_OF\_SENSORS
  - globals.h, 52
- onoff
  - st\_ref, 20
- overcurrent\_control
  - interruptions.c, 60
  - interruptions.h, 64
- POS\_INTEGRAL\_SAT\_LIMIT
  - globals.h, 52
- PWM\_MAX\_VALUE
  - globals.h, 52
- pos
  - st\_meas, 11
  - st\_ref, 20
- pos\_lim\_flag
  - st\_mem, 17
- pos\_lim\_inf
  - st\_mem, 17
- pos\_lim\_sup
  - st\_mem, 18
- position\_hist
  - st\_mem, 18
- pow\_tension
  - globals.c, 45
  - globals.h, 56
- prepare\_counter\_info
  - command\_processing.c, 28
  - command\_processing.h, 36
- prepare\_generic\_info
  - command\_processing.c, 28
  - command\_processing.h, 36
- pwm
  - st\_ref, 20
- pwm\_limit
  - st\_dev, 9
- pwm\_limit\_search
  - interruptions.c, 60
  - interruptions.h, 65
- pwm\_preload\_values
  - interruptions.c, 61
- pwm\_sign

- globals.c, 45
- globals.h, 56
- qbmove\_command
  - commands.h, 39
- qbmove\_control\_mode
  - commands.h, 40
- qbmove\_input\_mode
  - commands.h, 40
- qbmove\_parameter
  - commands.h, 41
- RECEIVE
  - globals.h, 52
- REFSPEED
  - utils.h, 77
- ready
  - st\_data, 9
- repetitions
  - st\_calib, 7
- res
  - st\_mem, 18
- reset\_counters
  - utils.c, 74
  - utils.h, 85
- reset\_last\_value\_flag
  - globals.c, 45
  - globals.h, 57
- rest\_counter
  - st\_mem, 18
- rest\_delay
  - st\_mem, 18
- rest\_enabled
  - globals.c, 46
  - globals.h, 57
- rest\_pos
  - st\_mem, 18
- rest\_pos\_curr\_ref
  - globals.c, 46
  - globals.h, 57
- rest\_position\_flag
  - st\_mem, 18
- rest\_ratio
  - st\_mem, 18
- rest\_vel
  - st\_mem, 19
- rot
  - st\_meas, 11
- SAMPLES\_FOR\_EMG\_MEAN
  - globals.h, 52
- SAMPLES\_FOR\_MEAN
  - globals.h, 52
- SIGN
  - utils.h, 77
- STATE\_ACTIVE
  - globals.h, 52
- STATE\_INACTIVE
  - globals.h, 53
- save\_cycles\_eeprom
  - interruptions.c, 60
  - interruptions.h, 65
- sendAcknowledgment
  - command\_processing.c, 28
  - command\_processing.h, 36
- setZeros
  - command\_processing.c, 29
  - command\_processing.h, 37
- speed
  - st\_calib, 8
- st\_calib, 7
  - direction, 7
  - enabled, 7
  - repetitions, 7
  - speed, 8
- st\_data, 8
  - buffer, 8
  - ind, 8
  - length, 9
  - ready, 9
- st\_dev, 9
  - pwm\_limit, 9
  - tension, 10
  - tension\_conv\_factor, 10
  - tension\_valid, 10
- st\_meas, 10
  - acc, 11
  - curr, 11
  - emg, 11
  - pos, 11
  - rot, 11
  - vel, 11
- st\_mem, 12
  - activ, 13
  - activate\_pwm\_rescaling, 13
  - baud\_rate, 13
  - control\_mode, 13
  - curr\_lookup, 13
  - current\_hist, 13
  - current\_limit, 14
  - double\_encoder\_on\_off, 14
  - emg\_calibration\_flag, 14
  - emg\_counter, 14
  - emg\_max\_value, 14
  - emg\_speed, 14
  - emg\_threshold, 14
  - flag, 14
  - id, 15
  - input\_mode, 15
  - k\_d, 15
  - k\_d\_c, 15
  - k\_d\_c\_dl, 15
  - k\_d\_dl, 15
  - k\_i, 15
  - k\_i\_c, 15
  - k\_i\_c\_dl, 16
  - k\_i\_dl, 16

- k\_p, 16
- k\_p\_c, 16
- k\_p\_c\_dl, 16
- k\_p\_dl, 16
- m\_mult, 16
- m\_off, 16
- maint\_day, 17
- maint\_month, 17
- maint\_year, 17
- max\_step\_neg, 17
- max\_step\_pos, 17
- motor\_handle\_ratio, 17
- pos\_lim\_flag, 17
- pos\_lim\_inf, 17
- pos\_lim\_sup, 18
- position\_hist, 18
- res, 18
- rest\_counter, 18
- rest\_delay, 18
- rest\_pos, 18
- rest\_position\_flag, 18
- rest\_ratio, 18
- rest\_vel, 19
- switch\_emg, 19
- total\_time\_on, 19
- total\_time\_rest, 19
- watchdog\_period, 19
- wire\_disp, 19
- st\_ref, 20
  - curr, 20
  - onoff, 20
  - pos, 20
  - pwm, 20
- switch\_emg
  - st\_mem, 19
- tau\_feedback
  - globals.c, 46
  - globals.h, 57
- tension
  - st\_dev, 10
- tension\_conv\_factor
  - st\_dev, 10
- tension\_valid
  - globals.c, 46
  - globals.h, 57
  - st\_dev, 10
- timer\_value
  - globals.c, 46
  - globals.h, 57
- timer\_value0
  - globals.c, 46
  - globals.h, 57
- total\_time\_on
  - st\_mem, 19
- total\_time\_rest
  - st\_mem, 19
- UNLOAD
  - globals.h, 53
- utils.c, 66
  - calc\_turns\_fcn, 68
  - calibration, 68
  - check\_enc\_data, 68
  - check\_rest\_position, 68
  - curr\_estim, 69
  - filter\_acc\_1, 69
  - filter\_acc\_2, 69
  - filter\_acc\_3, 70
  - filter\_ch1, 70
  - filter\_ch2, 70
  - filter\_curr\_diff, 71
  - filter\_i1, 71
  - filter\_v, 72
  - filter\_vel\_1, 72
  - filter\_vel\_2, 72
  - filter\_vel\_3, 73
  - filter\_voltage, 73
  - my\_mod, 73
  - my\_round, 74
  - reset\_counters, 74
- utils.h, 74
  - calc\_turns\_fcn, 77
  - calibration, 78
  - check\_enc\_data, 78
  - check\_rest\_position, 78
  - curr\_estim, 78
  - filter\_acc\_1, 79
  - filter\_acc\_2, 79
  - filter\_acc\_3, 79
  - filter\_ch1, 80
  - filter\_ch2, 80
  - filter\_curr\_diff, 80
  - filter\_i1, 81
  - filter\_v, 81
  - filter\_vel\_1, 82
  - filter\_vel\_2, 82
  - filter\_vel\_3, 82
  - filter\_voltage, 83
  - my\_mod, 83
  - my\_round, 83
  - REFSPEED, 77
  - reset\_counters, 85
  - SIGN, 77
  - ZERO\_TOL, 77
  - ZMAX, 77
- vel
  - st\_meas, 11
- WAIT\_ID
  - globals.h, 53
- WAIT\_LENGTH
  - globals.h, 53
- WAIT\_START
  - globals.h, 53
- watchdog\_flag
  - globals.c, 46

globals.h, 57  
watchdog\_period  
    st\_mem, 19  
wire\_disp  
    st\_mem, 19  
  
ZERO\_TOL  
    utils.h, 77  
ZMAX  
    utils.h, 77