Firmware BRDQBRQBC=SH22 documentation - qbcontrol main board

Generated by Doxygen 1.8.13

Contents

Chapter 1

Firmware

This is the firmware of the qbcontrol main board for SoftHand2 terminal device.

Version

1.0

This is the firmware of the SoftHand2. It can control two motors and read their encoders. 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:

Chapter 3

File Index

3.1 File List

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

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 Fields

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

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

- **globals.h**

4.3 st_meas Struct Reference

Data Fields

- int16 **vel** [NUM_OF_SENSORS]
- int32 **curr** [NUM_OF_MOTORS]
- int32 **pos** [NUM_OF_SENSORS]
- int16 **rot** [NUM_OF_SENSORS]
- int32 **emg** [NUM_OF_EMGS]
- int16 **joystick** [NUM_OF_MOTORS]

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

- **globals.h**

4.4 st_mem Struct Reference

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**
- int16 **current_limit**
- 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]
- uint8 **baud_rate**
- uint8 **watchdog_period**
- int32 **max_step_neg**
- int32 **max_step_pos**
- 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**
- float **curr_lookup** [6]
- uint8 **activate_pwm_rescaling**
- uint16 **closure_speed**
- int16 **joystick_threshold**
- uint16 **joystick_gains** [2]

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

- **globals.h**

4.5 st_ref Struct Reference

Data Fields

- int32 **pos** [NUM_OF_MOTORS]
- uint8 **onoff**

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

- **globals.h**

Chapter 5

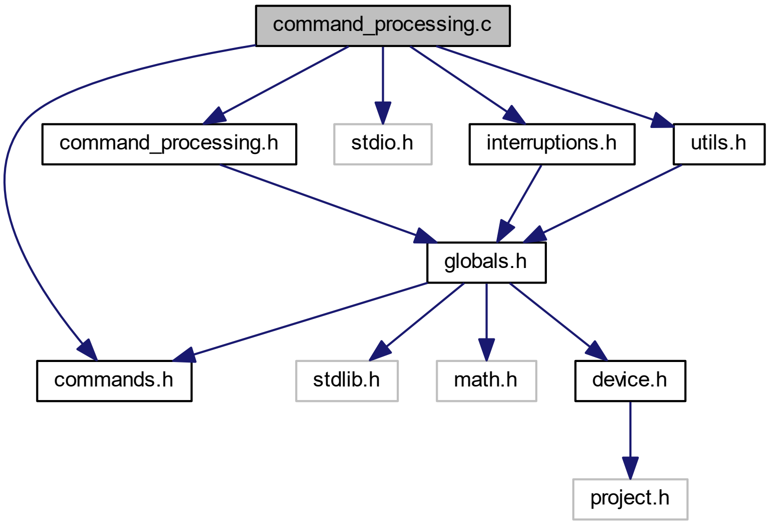
File Documentation

5.1 command_processing.c File Reference

Command processing functions.

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

Include dependency graph for command_processing.c:



Functions

- void **commProcess** ()

- void **infoGet** (uint16 info_type)
- void **setZeros** ()
- void **get_param_list** (uint16 index)
- void **infoPrepare** (unsigned char ∗info_string)
- void **commWrite_old_id** (uint8 ∗packet_data, uint16 packet_lenght, uint8 old_id)
- void **commWrite** (uint8 ∗packet_data, const uint16 packet_lenght)
- void **sendAcknowledgment** (const uint8 value)
- uint8 **memStore** (int displacement)
- void **memRecall** (void)
- uint8 **memRestore** (void)
- uint8 **memInit** (void)
- void **cmd_get_measurements** ()
- void **cmd_get_inputs** ()
- void **cmd_get_currents** ()
- void **cmd_get_curr_and_meas** ()
- void **cmd_set_inputs** ()
- void **cmd_set_pos_stiff** ()
- void **cmd_get_velocities** ()
- void **cmd_get_joystick** ()
- void **cmd_get_emg** ()
- void **cmd_activate** ()
- void **cmd_set_watchdog** ()
- void **cmd_get_activate** ()
- void **cmd_ping** ()
- void **cmd_store_params** ()
- void **cmd_set_baudrate** ()

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_get_measurements()

```
void cmd_get_measurements ( )
```

Bunch of functions used on request from UART communication

5.1.2.2 memInit()

```
uint8 memInit (
             void  )
```

This function initialize memory when eeprom is compromised.

5.1.2.3 memRecall()

```
void memRecall (
             void  )
```

This function loads user settings from the eeprom.

5.1.2.4 memRestore()

```
uint8 memRestore (
             void  )
```

This function loads default settings from the eeprom.

5.1.2.5 memStore()

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

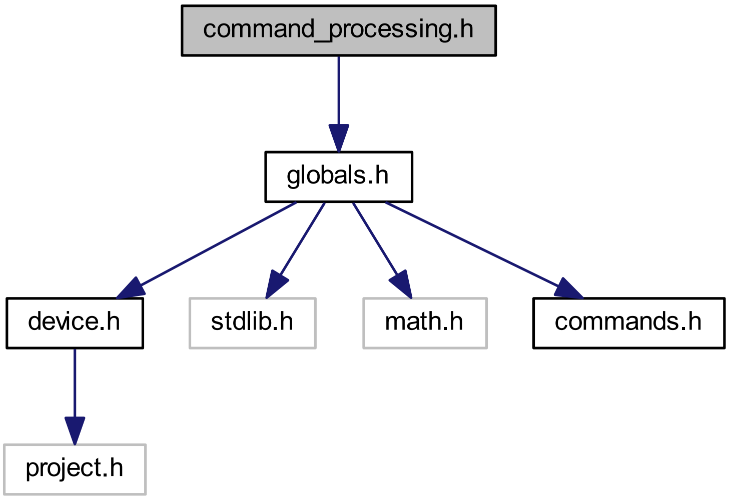
This function stores current memory settings on the eeprom with the specified displacement

5.2 command_processing.h File Reference

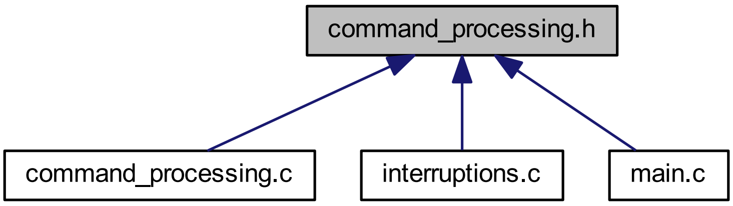
Command processing functions.

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

Include dependency graph for command_processing.h:



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



Functions

- void **setZeros** (void)
- void **get_param_list** (uint16 index)
- void **infoPrepare** (unsigned char ∗)
- void **infoGet** (uint16)
- void **commProcess** ()

- void **commWrite** (uint8 ∗, const uint16)
- void **commWrite_old_id** (uint8 ∗, const uint16, uint8)
- uint8 **memStore** (int)
- void **sendAcknowledgment** (const uint8)
- void **memRecall** (void)
- uint8 **memRestore** (void)
- uint8 **memInit** (void)
- void **cmd_get_measurements** ()
- void **cmd_get_inputs** ()
- void **cmd_get_currents** ()
- void **cmd_get_curr_and_meas** ()
- void **cmd_set_inputs** ()
- void **cmd_set_pos_stiff** ()
- void **cmd_get_velocities** ()
- void **cmd_get_joystick** ()
- void **cmd_get_emg** ()
- void **cmd_activate** ()
- void **cmd_set_watchdog** ()
- void **cmd_get_activate** ()
- void **cmd_ping** ()
- void **cmd_store_params** ()
- void **cmd_set_baudrate** ()

5.2.1 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.

5.2.2 Function Documentation

5.2.2.1 cmd_get_measurements()

```
void cmd_get_measurements ( )
```

Bunch of functions used on request from UART communication

5.2.2.2 memInit()

```
uint8 memInit (
             void  )
```

This function initialize memory when eeprom is compromised.

5.2.2.3 memRecall()

```
void memRecall (
             void  )
```

This function loads user settings from the eeprom.

5.2.2.4 memRestore()

```
uint8 memRestore (
             void  )
```

This function loads default settings from the eeprom.

5.2.2.5 memStore()

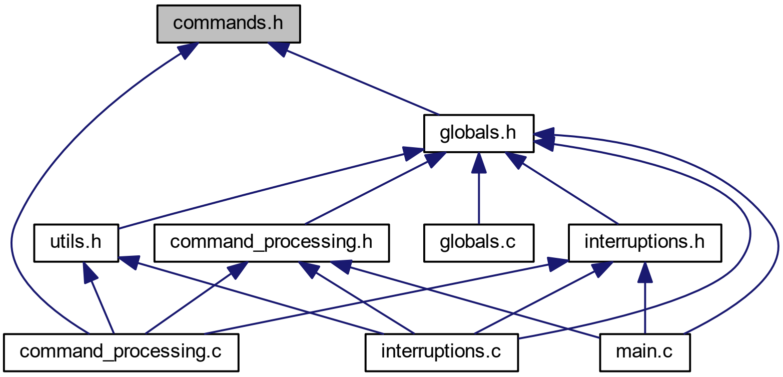
```
uint8 memStore (
             int displacement )
```

This function stores current memory settings on the eeprom with the specified displacement

5.3 commands.h File Reference

Definitions for qbMove and qbHand commands, parameters and packages.

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



Macros

QB Move Information Strings

in the get_param_list package

- #define **INFO_ALL** 0

  All system information.

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
- #define **PARAM_MENU_SLOT** 150

  in the get_param_list package
- 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,
  PARAM_JOYSTICK_THRESHOLD = 26, **PARAM_JOYSTICK_GAINS** = 27 }
- 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, **INPUT_MODE_JOYSTICK** = 6 }
- 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 qbMove and qbHand 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 SoftHand2 firmware, in its libraries and applications. It contains all definitions that are necessary for the contruction of communication packages.

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

5.3.2 Macro Definition Documentation

5.3.2.1 PARAM_BYTE_SLOT

```
#define PARAM_BYTE_SLOT 50
```

Number of bytes reserved to a param information

5.3.2.2 PARAM_MENU_SLOT

```
#define PARAM_MENU_SLOT 150
```

in the get_param_list package

Number of bytes reserved to a param menu

5.3.3 Enumeration Type Documentation

5.3.3.1 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 defalut 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
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)

Enumerator

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)

5.3.3.2 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.

5.3.3.3 qbmove_input_mode

enum **qbmove_input_mode**

Enumerator

INPUT_MODE_EXTERNAL	References through external commands (default)
INPUT_MODE_ENCODER3	Encoder 3 drives all inputs.
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
INPUT_MODE_JOYSTICK	Joystick input mode.

5.3.3.4 qbmove_parameter

enum **qbmove_parameter**

Enumerator

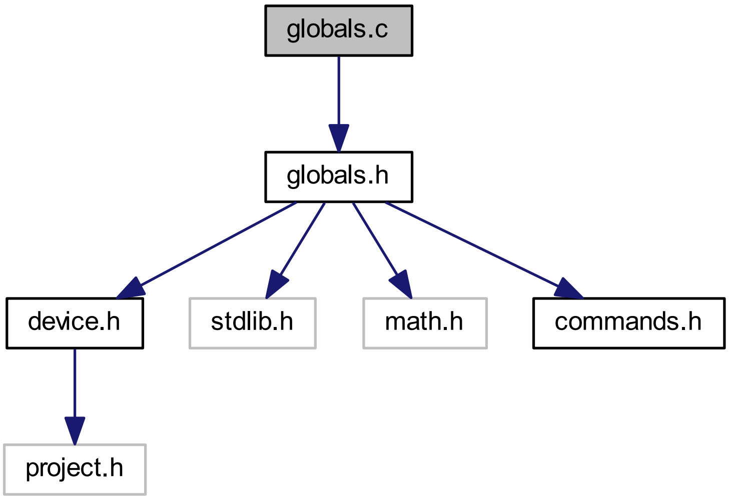
PARAM_ID	Device's ID number.
PARAM_PID_CONTROL	PID parameters.
PARAM_STARTUP_ACTIVATION	Start up activation byte.
PARAM_INPUT_MODE	Input mode.
PARAM_CONTROL_MODE	Choose the kind of control between position control, current control, direct PWM value or current+position control
PARAM_MEASUREMENT_OFFSET	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.
PARAM_JOYSTICK_THRESHOLD	Joystick threshold for joystick control.
PARAM_JOYSTICK_GAINS	Gains set to regulate Joystick readings.

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**
- uint32 **timer_value**
- uint32 **timer_value0**
- int32 **dev_tension**
- uint8 **dev_pwm_limit**
- uint8 **calibration_flag**
- CYBIT **reset_last_value_flag**
- CYBIT **tension_valid**
- CYBIT **interrupt_flag**
- CYBIT **watchdog_flag**
- int16 **ADC_buf** [3]
- int8 **pwm_sign** [NUM_OF_MOTORS]

5.4.1 Detailed Description

Global variables.

Date

October 01, 2017

Author

Centro "E.Piaggio"

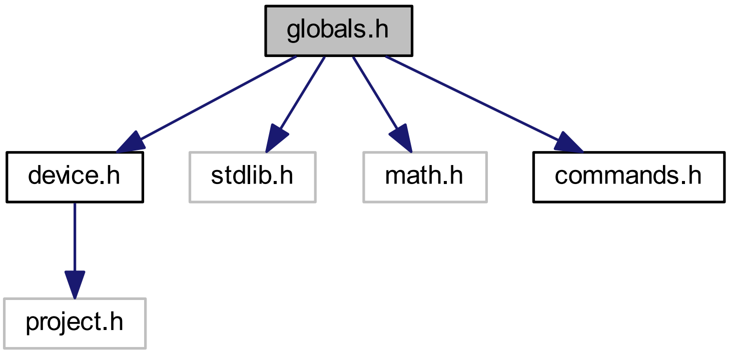
Copyright

5.5 globals.h File Reference

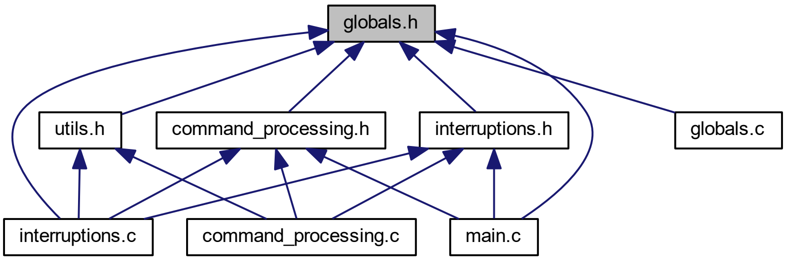
Global definitions and macros are set in this file.

```
#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**
- struct **st_meas**
- struct **st_data**
- struct **st_calib**

  Hand calibration structure.
- struct **st_mem**

Macros

- #define **VERSION** "TWO MOTORS HAND v6.1.0"
- #define **NUM_OF_MOTORS** 2
- #define **NUM_OF_SENSORS** 4
- #define **NUM_OF_EMGS** 2
- #define **NUM_OF_ANALOG_INPUTS** 7
- #define **NUM_OF_PARAMS** 24
- #define **PWM_MAX_VALUE** 100
- #define **PWM_DEAD** 0
- #define **CALIBRATION_DIV** 100
- #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 **FALSE** 0
- #define **TRUE** 1
- #define **DEFAULT_EEPROM_DISPLACEMENT** 8
- #define **MAX_WATCHDOG_TIMER** 250
- #define **PWM_MAX_VALUE** 100
- #define **ANTI_WINDUP** 1000
- #define **DEFAULT_CURRENT_LIMIT** 1000
- #define **CURRENT_HYSTERESIS** 10
- #define **EMG_SAMPLE_TO_DISCARD** 500
- #define **SAMPLES_FOR_MEAN** 100
- #define **SAMPLES_FOR_EMG_MEAN** 1000
- #define **SAMPLES_FOR_JOYSTICK_MEAN** 200
- #define **JOYSTICK_SAMPLE_TO_DISCARD** 100
- #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

Typedefs

- typedef enum **emg_status** **joystick_status**

Enumerations

- enum **calibration_status** {
  STOP = 0, **START** = 1, **CONTINUE_1** = 2, **CONTINUE_2** = 3,
  PAUSE_1 = 4, **PAUSE_2** = 5 }
- enum **emg_status** {
  NORMAL = 0, **RESET** = 1, **DISCARD** = 2, **SUM_AND_MEAN** = 3,
  WAIT = 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**
- int32 **dev_tension**
- uint8 **dev_pwm_limit**
- uint8 **calibration_flag**
- CYBIT **reset_last_value_flag**
- CYBIT **tension_valid**
- CYBIT **interrupt_flag**
- CYBIT **watchdog_flag**
- int16 **ADC_buf** [3]
- int8 **pwm_sign** [NUM_OF_MOTORS]

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 CURR_INTEGRAL_SAT_LIMIT

```
#define CURR_INTEGRAL_SAT_LIMIT 100000
```

Anti windup on current control.

5.5.2.3 CURRENT_HYSTERESIS

```
#define CURRENT_HYSTERESIS 10
```

milliAmperes of hysteresis for current control.

5.5.2.4 DEFAULT_CURRENT_LIMIT

```
#define DEFAULT_CURRENT_LIMIT 1000
```

Default Current limit, 0 stands for unlimited.

5.5.2.5 EMG_SAMPLE_TO_DISCARD

```
#define EMG_SAMPLE_TO_DISCARD 500
```

Number of sample to discard before calibration.

5.5.2.6 LOOKUP_DIM

```
#define LOOKUP_DIM 6
```

Dimension of the current lookup table.

5.5.2.7 POS_INTEGRAL_SAT_LIMIT

```
#define POS_INTEGRAL_SAT_LIMIT 50000000
```

Anti windup on position control.

5.5.2.8 PWM_MAX_VALUE [1/2]

```
#define PWM_MAX_VALUE 100
```

Maximum value of the PWM signal.

5.5.2.9 PWM_MAX_VALUE [2/2]

```
#define PWM_MAX_VALUE 100
```

Maximum value of the PWM signal.

5.5.2.10 SAMPLES_FOR_EMG_MEAN

```
#define SAMPLES_FOR_EMG_MEAN 1000
```

Number of samples used to mean emg values.

5.5.2.11 SAMPLES_FOR_MEAN

```
#define SAMPLES_FOR_MEAN 100
```

Number of samples used to mean current values.

5.5.3 Typedef Documentation

5.5.3.1 joystick_status

```
typedef enum emg_status joystick_status
```

EMG and joystick status enumeration

5.5.4 Enumeration Type Documentation

5.5.4.1 emg_status

```
enum emg_status
```

Enumerator

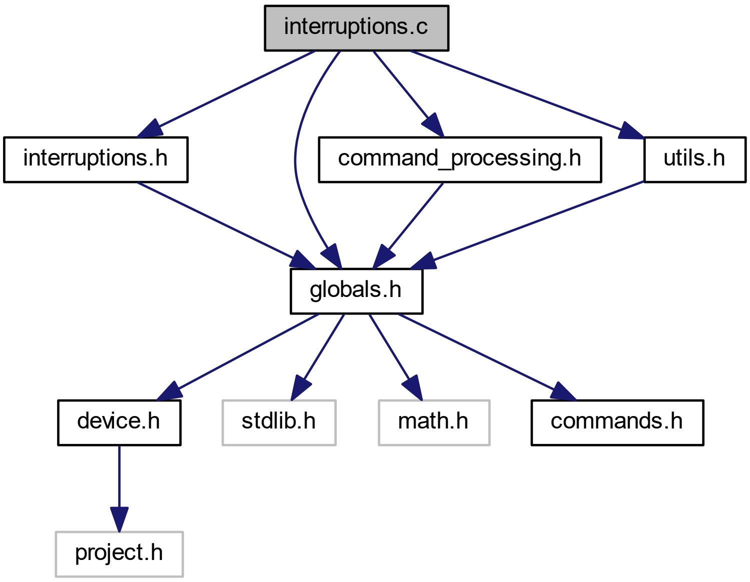
NORMAL	Normal execution
RESET	Reset analog measurements
DISCARD	Discard first samples to obtain a correct value
SUM_AND_MEAN	Sum and mean a definite value of samples
WAIT	The second emg waits until the first emg has a valid value

5.6 interruptions.c File Reference

Interruption functions are in this file.

```
#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)
- void **interrupt_manager** ()
- void **function_scheduler** (void)
- void **motor_control** (const uint8 idx)
- void **analog_read_end** ()
- void **encoder_reading** (const uint8 idx)
- void **pwm_limit_search** ()

Variables

- CYCODE uint8 **pwm_preload_values** [29]

5.6.1 Detailed Description

Interruption functions are in this file.

Date

October 01, 2017

Author

Centro "E.Piaggio"

Copyright

5.6.2 Variable Documentation

5.6.2.1 pwm_preload_values

```
CYCODE uint8 pwm_preload_values[29]
```

Initial value:

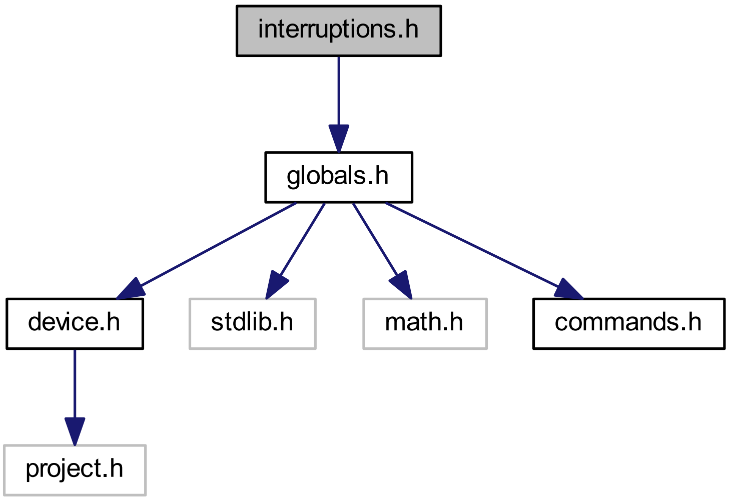
```
= {100,
                                         83,
                                         78,
                                         76,
                                         74,
                                         72,
                                         70,
                                         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

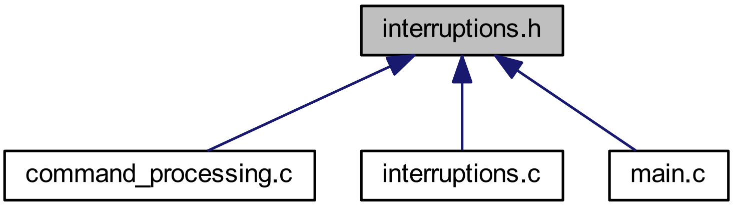
Interruptions header file.

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

Include dependency graph for interruptions.h:



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



Functions

- **CY_ISR_PROTO** (ISR_RS485_RX_ExInterrupt)
- **CY_ISR_PROTO** (ISR_WATCHDOG_Handler)
- void **function_scheduler** (void)
- void **encoder_reading** (const uint8)
- void **motor_control** (const uint8)
- void **analog_read_end** ()
- void **calibration** (void)
- void **pwm_limit_search** ()
- void **interrupt_manager** ()

5.7.1 Detailed Description

Interruptions header file.

Date

October 01, 2017

Author

Centro "E.Piaggio"

Copyright

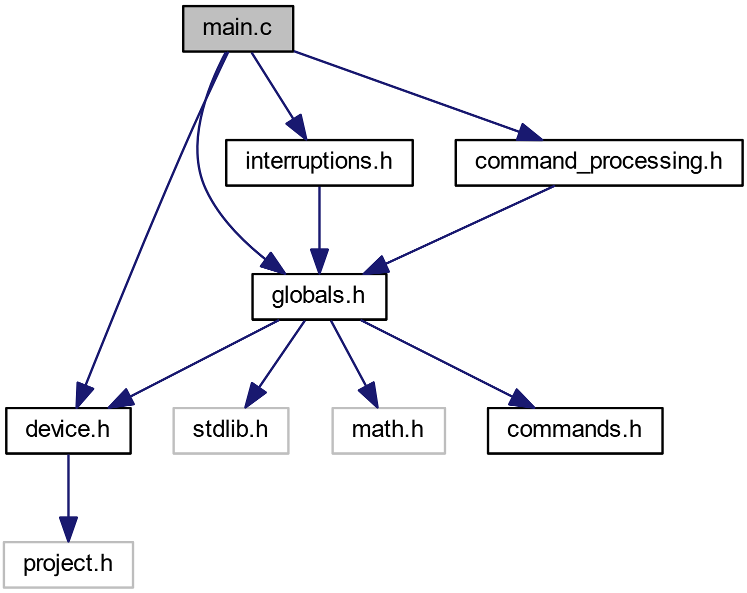
(C) 2012-2016 qbrobotics. All rights reserved.
(C) 2017 Centro "E.Piaggio". All rights reserved.

5.8 main.c File Reference

Firmware main file.

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

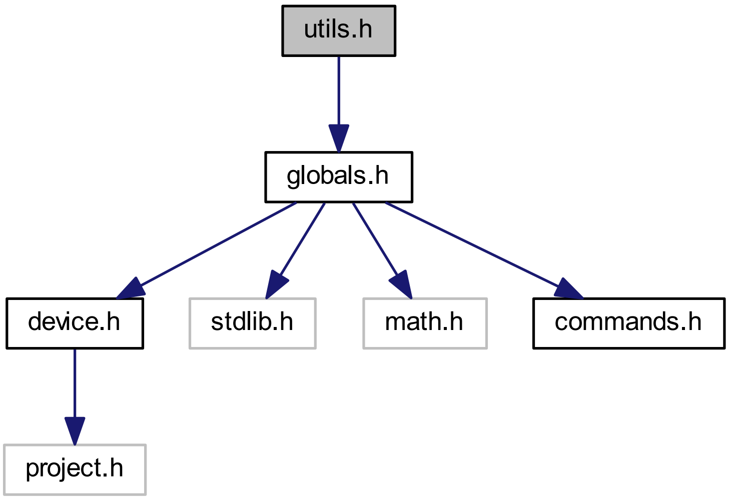
(C) 2012-2016 qbrobotics. All rights reserved.
(C) 2017 Centro "E.Piaggio". All rights reserved.

5.9 utils.h File Reference

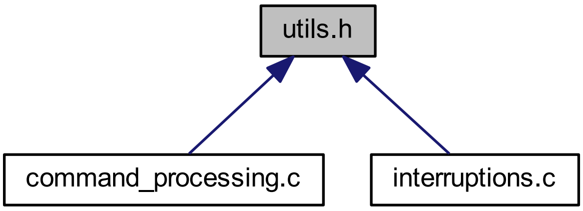
Definition of utility functions.

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

Include dependency graph for utils.h:



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



Macros

- #define **ALPHA** 3

  Voltage and current filters constants.
- #define **BETA** 50

  Emg filters constant.
- #define **GAMMA** 32

  Velocity filters constant.
- #define **DELTA** 32

  Acceleration filters constant.
- #define **SIGN**(A) (((A) >= 0) ? (1) : (-1))

Functions

- int32 **filter_i1** (int32 value)
- int32 **filter_i2** (int32 value)
- int32 **filter_vel_1** (int32 value)
- int32 **filter_vel_2** (int32 value)
- int32 **filter_vel_3** (int32 value)
- int32 **filter_ch1** (int32 value)
- int32 **filter_ch2** (int32 value)
- uint8 **LCRChecksum** (uint8 ∗data_array, const uint8 data_length)
- CYBIT **check_enc_data** (const uint32 ∗)
- uint32 **my_mod** (int32 val, int32 divisor)
- int **calc_turns_fcn** (const int32 pos1, const int32 pos2)
- void **calibration** ()

5.9.1 Detailed Description

Definition of utility functions.

Declaration of utility functions.

Date

October 01, 2017

Author

Centro "E.Piaggio"

Copyright

Index