My Project

Generated by Doxygen 1.8.13

Contents

Chapter 1

Firmware alpha

Processor-In-the-Loop (PIL)

Processor-In-the-Loop validation allows the developer to test the software directly on the target board. The board is connected with an external PC on which is running the physical model of the robot as well as a model of its actuators and sensors. This firmware supports communication using the **UDP** protocol to talk with the external PC. The connection properties (e.g. IP, Port...) can be modified in the file *UDPComm.cpp*:

```
{c++}
static const char*          mbedIP       = "192.168.1.55";      // This board IP seen from the network
static const char*          mbedMask     = "255.255.255.0";     // Mask
static const char*          mbedGateway  = "192.168.1.254";     // Gateway

static const char*          recvIP = "192.168.1.203";           // External PC IP */ "192.168.1.249";
static const char*          localIP = "0.0.0.0";                // Local IP on which the server of the
       board listens
```

The messaging protocol used is `Mavlink v2.0` since it is lightweight, open-source and widespread.

PIL with Matlab/Simulink®

It is possible to employ Simulink to create the robot model

Code generation with Matlab/Simulink®

For *Rapid Prototyping* purposes, code generation is a valid tool. The Firmware is compatible with C/C++ code automatically generated from Matlab/Simulink®. Here it is shown which are the settings used in the **Configuration parameter pane**.

Chapter 2

File Index

2.1 File List

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

Chapter 3

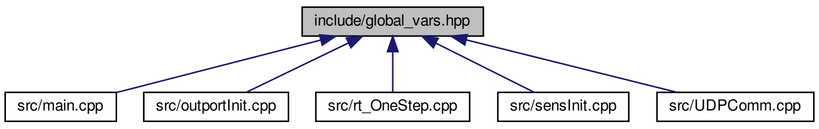
File Documentation

3.1 include/global_vars.hpp File Reference

Header gathering the global variables used in the Firmware.

```
#include <mbed.h>
#include "common/mavlink.h"
#include "feedback_control.h"
#include "Servo.h"
```

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



Variables

- ExtU_feedback_control_T feedback_control_U
- ExtY_feedback_control_T feedback_control_Y

3.1.1 Detailed Description

Header gathering the global variables used in the Firmware.

Global variables are chosen upon uORB because yes.

3.1.2 Variable Documentation

3.1.2.1 feedback_control_U

```
ExtU_feedback_control_T feedback_control_U
```

The I/O variables of the controller are used as extern since their name is the same if a Simulink project is used. This allows to keep the Firmware unchanged.External inputs

3.1.2.2 feedback_control_Y

```
ExtY_feedback_control_T feedback_control_Y
```

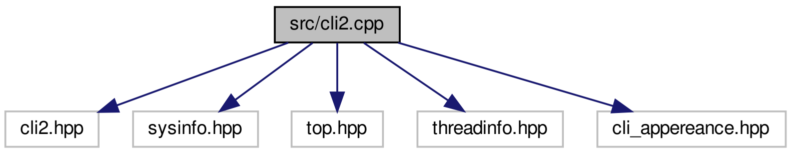
External outputs

3.2 src/cli2.cpp File Reference

Command line interface for hardware resource inspection.

```
#include "cli2.hpp"
#include "sysinfo.hpp"
#include "top.hpp"
#include "threadinfo.hpp"
#include "cli_appereance.hpp"
```

Include dependency graph for cli2.cpp:



3.2.1 Detailed Description

Command line interface for hardware resource inspection.

The command line interface (cli) allows the developer to check the CPU and memory usage, the number of active threads and display the main variables. The available commands are:

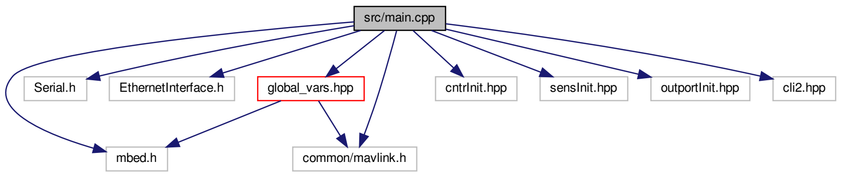
- top Display CPU usage;
- info Display hardware information;
- thread Show active threads state, priority and their resources;
- clear Clear screen;
- help Show available commands.

3.3 src/main.cpp File Reference

Entry point for mbed OS.

```
#include <mbed.h>
#include <Serial.h>
#include <EthernetInterface.h>
#include "global_vars.hpp"
#include "common/mavlink.h"
#include "cntrInit.hpp"
#include "sensInit.hpp"
#include "outportInit.hpp"
#include "cli2.hpp"
```

Include dependency graph for main.cpp:



Variables

- ExtU_feedback_control_T feedback_control_U
- ExtY_feedback_control_T feedback_control_Y

3.3.1 Detailed Description

Entry point for mbed OS.

This script creates and spawns threads and declare global variables defined in the header global_vars.hpp.

3.3.2 Variable Documentation

3.3.2.1 feedback_control_U

```
ExtU_feedback_control_T feedback_control_U
```

The I/O variables of the controller are used as extern since their name is the same if a Simulink project is used. This allows to keep the Firmware unchanged.External inputs

3.3.2.2 feedback_control_Y

```
ExtY_feedback_control_T feedback_control_Y
```

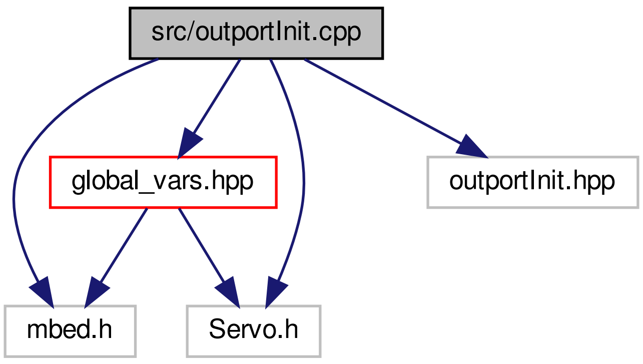
External outputs

3.4 src/outportInit.cpp File Reference

This thread creates the periodic task sending the pulse-width modulated signals to the enabled pins.

```
#include <mbed.h>
#include "Servo.h"
#include "global_vars.hpp"
#include "outportInit.hpp"
```

Include dependency graph for outportInit.cpp:



Functions

- void outportInit ()
- void postServoEvent (void)
- void Servo1Write (void)

3.4.1 Detailed Description

This thread creates the periodic task sending the pulse-width modulated signals to the enabled pins.

Author

Davide Carminati

The script uses events and a queue to pile periodic tasks. The queue contains both read-from and write-to pins events.

3.4.2 Function Documentation

3.4.2.1 outportInit()

```
void outportInit ( )
```

Initialization of the servomotor, spawn of the event posting thread and dispatch queue. The Servo::calibrate() method accepts as first input a value IN SECONDS.

3.4.2.2 postServoEvent()

```
void postServoEvent (
             void  )
```

The period and the initial delay of the PWM write event are set. Then it is posted in the queue.

3.4.2.3 Servo1Write()

```
void Servo1Write (
             void  )
```

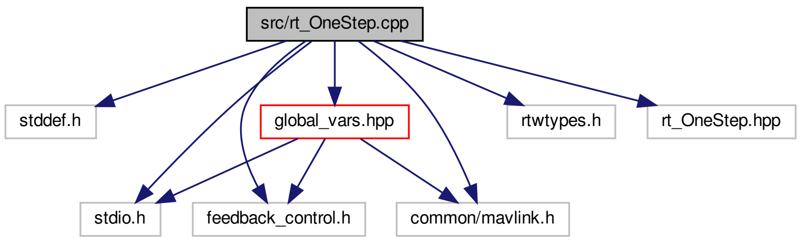
The event is simply a method writing the computed duty cycle to the enabled pin port. The duty cycle is directly dependent on the output of the controller thread: note that the extern variable feedback_control_Y is present.

3.5 src/rt_OneStep.cpp File Reference

It performs a time step of the controller algorithm checking whether it respects the Real-Time deadline.

```
#include <stddef.h>
#include <stdio.h>
#include "feedback_control.h"
#include "rtwtypes.h"
#include "global_vars.hpp"
#include "common/mavlink.h"
#include "rt_OneStep.hpp"
```

Include dependency graph for rt_OneStep.cpp:



3.5.1 Detailed Description

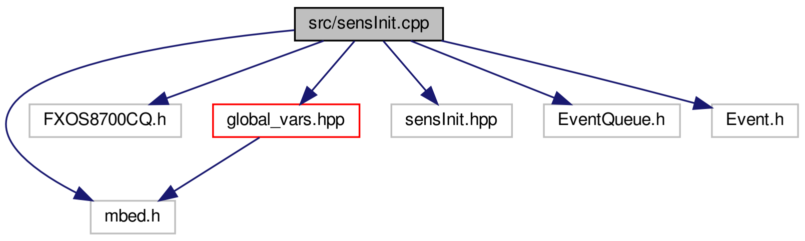
It performs a time step of the controller algorithm checking whether it respects the Real-Time deadline.

3.6 src/sensInit.cpp File Reference

Thread initializing sensors read at the given frequency.

```
#include <mbed.h>
#include "FXOS8700CQ.h"
#include "global_vars.hpp"
#include "sensInit.hpp"
#include "EventQueue.h"
#include "Event.h"
```

Include dependency graph for sensInit.cpp:



Macros

- #define FXOS8700CQ_FREQ 50

  Frequency at which the sensor is interrogated.

3.6.1 Detailed Description

Thread initializing sensors read at the given frequency.

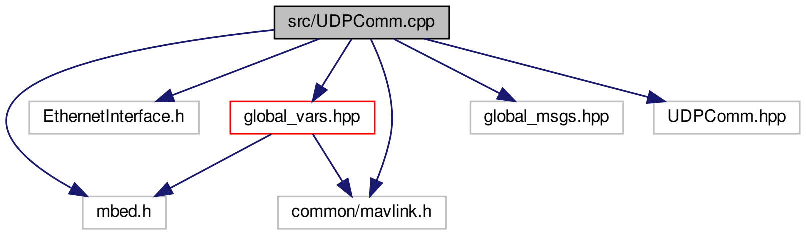
Creates a timer that calls an interrupt with the given frequency

3.7 src/UDPComm.cpp File Reference

Enables UDP communication between the target board and an external PC to perform Processor-In-the-Loop.

```
#include <mbed.h>
#include <EthernetInterface.h>
#include "common/mavlink.h"
#include "global_vars.hpp"
#include "global_msgs.hpp"
#include <UDPComm.hpp>
```

Include dependency graph for UDPComm.cpp:



Variables

- uint8_t in_data [MAVLINK_MAX_PACKET_LEN]
- uint8_t out_buf [MAVLINK_MAX_PACKET_LEN]

3.7.1 Detailed Description

Enables UDP communication between the target board and an external PC to perform Processor-In-the-Loop.

Processor-In-the-Loop validation uses the external PC to simulate the physical model, its actuators and its sensors. The target board runs the firmware with full functionalities plus this thread that is responsible for sending/receiving simulation data to/from the external PC. The target board controls the model just as it was controlling the physical system.

Note

The used communication protocol is Mavlink v2.0.

3.7.2 Variable Documentation

3.7.2.1 in_data

```
uint8_t in_data[MAVLINK_MAX_PACKET_LEN]
```

Buffer in which incoming data is stored

3.7.2.2 out_buf

```
uint8_t out_buf[MAVLINK_MAX_PACKET_LEN]
```

Buffer in which outgoing data is stored