# My Project

Generated by Doxygen 1.8.13

# Contents

# Chapter 1

# File Index

## 1.1 File List

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

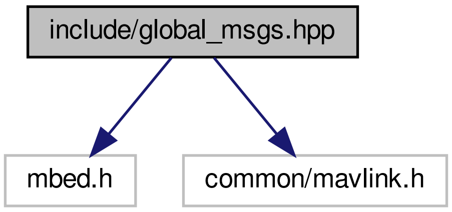
# Chapter 2

# File Documentation

## 2.1 include/global_msgs.hpp File Reference

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

Include dependency graph for global_msgs.hpp:



### Variables

- mavlink_attitude_t att

### 2.1.1 Variable Documentation

#### 2.1.1.1 att

```
mavlink_attitude_t att
```

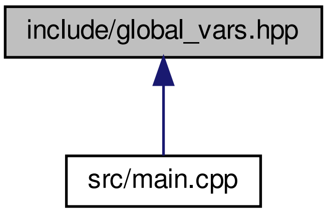
Declaration and definition of used mavlink structs

## 2.2 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
- Semaphore **semDecode**
- Semaphore **semEncode**
- Servo **servo1**
- EventQueue **queue**

### 2.2.1 Detailed Description

Header gathering the global variables used in the Firmware.

Global variables are chosen upon uORB because yes.

### 2.2.2 Variable Documentation

#### 2.2.2.1 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

#### 2.2.2.2 feedback_control_Y

```
ExtY_feedback_control_T feedback_control_Y
```

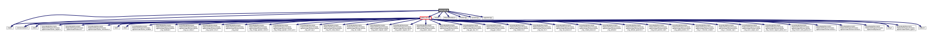
External outputs

## 2.3 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 dependency graph for main.cpp:



### Functions

- Serial **serial** (USBTX, USBRX)
- Thread **ControllerInit** (osPriorityHigh, 8092, nullptr, cntrInit_thread_name)
- Thread **SensorInit** (osPriorityNormal, 8092, nullptr, sensInit_thread_name)
- Thread **OutputPortInit** (osPriorityNormal, 8092, nullptr, outportInit_thread_name)
- Semaphore **semDecode** (0)
- Semaphore **semEncode** (0)
- Servo **servo1** (PTC10)
- int **main** ()

### Variables

- const char ∗ **UDP_thread_name** = "UDPIO"
- const char ∗ **cntrlnit_thread_name** = "cntrInit"
- const char ∗ **sensInit_thread_name** = "sensInit"
- const char ∗ **outportInit_thread_name** = "outportInit"
- FileHandle ∗ **fh** = &serial
- EventQueue **queue**
- ExtU_feedback_control_T feedback_control_U
- ExtY_feedback_control_T feedback_control_Y

### 2.3.1 Detailed Description

Entry point for mbed OS.

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

### 2.3.2 Variable Documentation

#### 2.3.2.1 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

#### 2.3.2.2 feedback_control_Y

```
ExtY_feedback_control_T feedback_control_Y
```

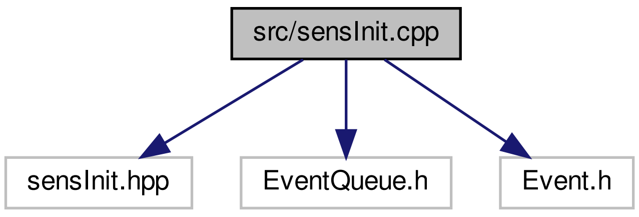
External outputs

## 2.4 src/sensInit.cpp File Reference

Thread initializing sensors read at the given frequency.

```
#include "sensInit.hpp"
#include "EventQueue.h"
#include "Event.h"
```

Include dependency graph for sensInit.cpp:



### Functions

- FXOS8700CQ **accmag** (PTE25, PTE24)
- Thread **SensorRead** (osPriorityNormal, 8092, nullptr,"sensRead")
- void **sensInit** ()
- void **postSensorEvent** (void)
- void **AccMagRead** (void)

#### Variables

- Event< void(void)> accmagreadEvent & **queue**
- Data **accmagValues**

### 2.4.1 Detailed Description

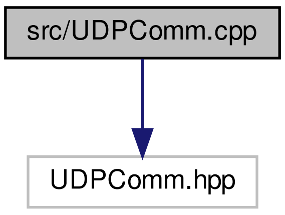
Thread initializing sensors read at the given frequency.

Creates a timer that calls an interrupt with the given frequency

## 2.5 src/UDPComm.cpp File Reference

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

```
#include <UDPComm.hpp>
```
Include dependency graph for UDPComm.cpp:



#### Functions

- void **UDPComm** (void)

#### Variables

- uint8_t **in_data** [MAVLINK_MAX_PACKET_LEN]
- uint8_t **out_buf** [MAVLINK_MAX_PACKET_LEN]
- mavlink_message_t **pos_decoded**
- mavlink_message_t **msg**
- mavlink_status_t **status**
- mavlink_attitude_t att
- uint8_t **SYS_ID** = 1
- uint8_t **COMP_ID** = 1

### 2.5.1 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.

### 2.5.2 Variable Documentation

#### 2.5.2.1 att

```
mavlink_attitude_t att
```

Declaration and definition of used mavlink structs