

SPACE HAUC Flight Software

Generated by Doxygen 1.8.13

Contents

Chapter 1

Main Page

SPACE HAUC Flight Code

Last stable tested commit: `release branch`

Current status

The following major features have been implemented:

1. Threaded code (split into different files)
1. `make` code generation system (TODO: Transition to `cmake`)
1. ACS detumble and sunpointing algorithms
1. Serial communication for SITL (Software In The Loop) testing
1. ACS devices have been added for HITL (Hardware In The Loop) testing
1. External data visualization over TCP using a Python frontend
1. External data visualization for Simulink data over Serial + TCP using Python frontend
1. Complete Doxygen documentation and travis build checker support.

make Options:

1. `make`: Invokes `all` which is the default compilation option. Does not pass any arguments to the compiler, hence generates dynamically linked code that runs is compatible with HITL without any sun sensor code.
 2. `make sim_server`: Creates the server code that can read Simulink display output over serial port and publish it over TCP for `geode.py` visualization service.
 3. `make clean`: Delete all the object files and the built code.
 4. `make spotless`: Remove every object file, build directory etc.
-
1. `make doc`: Create doxygen documentation.

Program Options:

Program options are still scattered throughout the program. These options can be passed through the `CFLAGS` variable to `make` (e.g. `make CFLAGS="-DCSS_READY"` will enable coarse sun sensor support in the code). Here is a list of different compile switches that turns on/off different features:

1. `SITL`: Turns on the `sitl_comm` interface for a Software In The Loop test.
2. `PORT`: Requires an input of the form of an integer, assigns port for the DataVis thread.
3. `CSS_READY`: Turns on coarse sun sensor related code in the software for HITL/production.
4. `FSS_READY`: Turns on fine sun sensor related code in the software for HITL/production (partial support).
5. `I2C_BUS`: Requires an input of the form of a string pointing to the absolute path of the I2C device file.
6. `SPIDEV_ACS`: Requires an input of the form of a string pointing to the absolute path of the SPI device file.
7. `ACS_DATALOG`: Writes ACS data to a file.

There is a hidden option in [drivers/tsl2561.c](#) that enables the true low-gain operation of the coarse sun sensors. The true low-gain operation is currently disabled to support the calibration that was last performed on the coarse sun sensors.

Quirks (and TO-DOs)

The following quirks are present in the code as of now:

Serial Communication

1. ~~The Simulink simulator is not a real time system yet (investigating Real-time execution where `Serial` blocks raise errors; using `Packet` blocks may help.)~~ Simulink is running in real time mode using `Packet` output blocks.
1. ~~The lack of true real time implies that the serial data needs to be synchronized to the simulation itself to guarantee a functional data stream without any errors.~~ No synchronization necessary.
1. The baud rate being low (230400 bps == ~1.7 ms for 40 bytes of data) could be a possible reason for the apparent lack of synchronization. In this case, the `sitl_comm` thread should also time (and synchronize itself) to the simulation. Look into such possibilities.
1. Currently due to the synchronization problems the `acs_detumble` thread waits on wakeup from the `sitl_comm` thread to guarantee a basic form of synchronization with the Simulation.
1. For HITL, no such synchronization is necessary and the flight code can operate outside of the realm of Simulink.

ACS Detumble Algorithm

1. Magnetic field is represented in milliGauss to enhance math precision.
1. Omega measurement does not include the second order correction term that uses the MOI and past measurement. This corrected value of omega should be passed through a Bessel filter.
1. Investigate if every sensor reading should be filtered using a low pass filter. Discuss the cutoff frequency for such a filter.
1. Investigate implementation of a Kalman filter instead of a Bessel function.
1. In HITL, due to the noise Bessel filtering is used on B , dB/dt and z which leads to a bias on z . This throws off the detumble determination. Find a better filter/criterion.
1. Investigate the effect of $z < 0$ at initialization.

ACS Sunpointing Algorithm

1. Both FSS and CSS are read. If FSS reading is valid, it is used to determine sun vector.
2. If FSS reading is invalid, CSS readings are used to determine sun vector essentially by subtracting the flux on the negative direction from the positive direction, doing this for all three faces, and then normalizing the resultant vector.
3. Investigate the gain factor in the sunpointing algorithm.

Chapter 2

Class Index

2.1 Class List

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

ads1115	Ads1115 device data structures	??
ads1115_config	Configuration register	??
channel_t	??
data_packet	Union of the datavis_p structure and an array of bytes for transport over TCP using send()	??
datavis_p	DataVis structure for storing current ACS data	??
eps_config2_t	??
eps_config3_t	??
eps_config_t	??
eps_hk_basic_t	??
eps_hk_out_t	??
eps_hk_t	??
eps_hk_vi_t	??
eps_hk_wdt_t	??
hkparam_t	??
lsm9ds1	LSM9DS1 Device Struct	??
helmholtz.lsm9ds1	??
MAG_DATA_RATE	Configuration for magnetometer data rate	??
MAG_DATA_READ	??
MAG_RESET	??
ncv7708	NCV77X8 Device	??
ncv7708_packet	NCV77X8 Data packet (I/O)	??
p31u	??
tca9458a	TCA9458A Device handle	??
tsl2561	TSL2561 Device Handle	??

Chapter 3

File Index

3.1 File List

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

drivers/ads1115.c	ADS1115 I2C Driver function definitions	??
drivers/ads1115.h	ADS1115 I2C Driver function prototypes and data structures	??
drivers/lsm9ds1.c	Function definitions for LSM9DS1 Magnetometer I2C driver	??
drivers/lsm9ds1.h	Function prototypes and data structures for LSM9DS1 Magnetometer I2C driver	??
drivers/ncv7708.c	Function definitions for NCV77X8 SPI Driver (Linux)	??
drivers/ncv7708.h	Function prototypes and data structure for NCV77X8 SPI Driver (Linux)	??
drivers/tca9458a.c	Function definitions for TCA9458A I2C driver	??
drivers/tca9458a.h	Function prototypes and struct declarations for TCA9458A I2C driver	??
drivers/tsl2561.c	TSL2561 I2C driver function definitions	??
drivers/tsl2561.h	TSL2561 I2C driver function and struct declarations	??
include/acs.h	Header file including headers and function prototypes of the Attitude Control System	??
include/bessel.h	Bessel filter implementation for Attitude Control System	??
include/datavis.h	DataVis thread to visualize ACS data over TCP (uses client.py)	??
include/eps_telem.h	GomSpace P31u I2C interface function prototypes and data structures	??
include/main.h	Includes all headers necessary for the core flight software, including ACS, and defines ACS states (which are flight software states), error codes, and relevant error functions	??

include/main_helper.h	Defines vector macros and other helper functions for the flight software	??
include/shflight_consts.h	Describes all constants defined by the preprocessor for the code	??
include/shflight_extrns.h	Extern declaration of the shflight global variables for all threads	??
include/shflight_globals.h	Allocates memory for the global variables used in the flight code for inter-thread communication	??
include/sitl_comm.h	Software-In-The-Loop (SITL) serial communication headers and function prototypes	??
include/uhf.h	EnduroSat UHF Transceiver Interface Code function prototypes (Needs to be written)	??
include/xband.h	SPACE-HAUC X-Band Transceiver function prototypes (Needs to be written)	??
src/acs.c	Attitude Control System related functions	??
src/bessel.c	Bessel filter implementation for Attitude Control System	??
src/eps_telem.c	GomSpace P31u I2C interface function declarations	??
src/main.c	Main() symbol of the SPACE-HAUC Flight Software	??
src/sitl_comm.c	Software-In-The-Loop (SITL) serial communication codes	??
src/uhf.c	UHF interface code	??
src/xband.c	X-Band Radio interface code	??

Chapter 4

Class Documentation

4.1 ads1115 Struct Reference

[ads1115](#) device data structures.

```
#include <ads1115.h>
```

Public Attributes

- int [fd](#)
Device file descriptor.
- char [fname](#) [40]
I2C Bus name.

4.1.1 Detailed Description

[ads1115](#) device data structures.

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

- drivers/[ads1115.h](#)

4.2 ads1115_config Union Reference

Configuration register.

```
#include <ads1115.h>
```

Public Attributes

- ```

struct {
 uint8_t comp_que: 2
 Comparator queue and disable (ADS1114 and ADS1115 only) These bits perform two functions. When set to 11, the comparat
 uint8_t comp_lat: 1
 Latching comparator (ADS1114 and ADS1115 only) This bit controls whether the ALERT/RDY pin latches after being asserted
 uint8_t comp_mode: 1
 Comparator polarity (ADS1114 and ADS1115 only) This bit controls the polarity of the ALERT/RDY pin. This bit serves no funct
 uint8_t comp_pol: 1
 Comparator mode (ADS1114 and ADS1115 only) This bit configures the comparator operating mode. This bit serves no funct
 uint8_t dr: 3
 Data rate These bits control the data rate setting. 000 : 8 SPS 001 : 16 SPS 010 : 32 SPS 011 : 64 SPS 100 : 128 SPS (defa
 uint8_t mode: 1
 Device operating mode This bit controls the operating mode. 0 : Continuous-conversion mode 1 : Single-shot mode or power-
 uint8_t pga: 3
 Programmable gain amplifier configuration These bits set the FSR of the programmable gain amplifier. These bits serve no fun
 uint8_t mux: 3
 Input multiplexer configuration (ADS1115 only) These bits configure the input multiplexer. These bits serve no function on the
 uint8_t os: 1
 Operational status or single-shot conversion start This bit determines the operational status of the device. OS can only be writ
};

```
- ```

uint16_t raw

```

Raw 16 bits corresponding to the config struct.

4.2.1 Detailed Description

Configuration register.

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

- drivers/[ads1115.h](#)

4.3 channel_t Union Reference

Public Attributes

- ```

struct {
 uint8_t V5_1: 1
 uint8_t V5_2: 1
 uint8_t V5_3: 1
 uint8_t V3_1: 1
 uint8_t V3_2: 1
 uint8_t V3_3: 1
 uint8_t qs: 1
 uint8_t qh: 1
};

```

- `uint8_t reg`

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

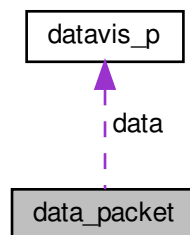
- `include/eps_telem.h`

## 4.4 data\_packet Union Reference

Union of the `datavis_p` structure and an array of bytes for transport over TCP using `send()`.

```
#include <datavis.h>
```

Collaboration diagram for `data_packet`:



### Public Attributes

- `datavis_p data`  
*Data section of the `data_packet` where members of `datavis_p` can be accessed.*
- unsigned char `buf` [`sizeof(datavis_p)`]  
*Byte section of the `data_packet` for transport using `send()`.*

### 4.4.1 Detailed Description

Union of the `datavis_p` structure and an array of bytes for transport over TCP using `send()`.

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

- `include/datavis.h`

## 4.5 datavis\_p Struct Reference

DataVis structure for storing current ACS data.

```
#include <datavis.h>
```

### Public Member Functions

- [DECLARE\\_VECTOR2](#) (B, float)  
*Measured magnetic field.*
- [DECLARE\\_VECTOR2](#) (Bt, float)  
*Calculated value of  $\vec{B}$ .*
- [DECLARE\\_VECTOR2](#) (W, float)  
*Calculated value of  $\vec{\omega}$ .*
- [DECLARE\\_VECTOR2](#) (S, float)  
*Calculated value of sun vector.*

### Public Attributes

- uint8\_t [mode](#)  
*Current system state.*
- uint64\_t [step](#)  
*Current ACS step number.*

#### 4.5.1 Detailed Description

DataVis structure for storing current ACS data.

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

- include/[datavis.h](#)

## 4.6 eps\_config2\_t Struct Reference

### Public Attributes

- uint16\_t **batt\_maxvoltage**
- uint16\_t **batt\_safevoltage**
- uint16\_t **batt\_criticalvoltage**
- uint16\_t **batt\_normalvoltage**
- uint32\_t **reserved1** [2]
- uint8\_t **reserved2** [4]

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

- include/[eps\\_telem.h](#)



## 4.7 eps\_config3\_t Struct Reference

### Public Attributes

- uint8\_t **version**
- uint8\_t **cmd**
- uint8\_t **length**
- uint8\_t **flags**
- uint16\_t **cur\_lim** [8]
- uint8\_t **cur\_ema\_gain**
- uint8\_t **cspwdt\_channel** [2]
- uint8\_t **cspwdt\_address** [2]

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

- include/[eps\\_telem.h](#)

## 4.8 eps\_config\_t Struct Reference

### Public Attributes

- uint8\_t **ppd\_mode**
- uint8\_t [battheater\\_mode](#)  
*Mode for PPT [1 = AUTO, 2 = FIXED].*
- int8\_t [battheater\\_low](#)  
*Mode for battheater [0 = MANUAL, 1 = AUTO].*
- int8\_t [battheater\\_high](#)  
*Turn heater on at [degC].*
- uint8\_t [output\\_normal\\_value](#) [8]  
*Turn off heater at [degC].*
- uint8\_t [output\\_safe\\_value](#) [8]  
*Nominal mode output value.*
- uint16\_t [output\\_initial\\_on\\_delay](#) [8]  
*Safe mode output value.*
- uint16\_t [output\\_initial\\_off\\_delay](#) [8]  
*Output switches: init with these on delays [s].*
- uint16\_t [vboost](#) [3]  
*Output switches: init with these off delays [s].*

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

- include/[eps\\_telem.h](#)

## 4.9 eps\_hk\_basic\_t Struct Reference

### Public Attributes

- uint32\_t **counter\_boot**
- int16\_t **temp** [6]  
*Number of EPS reboots.*
- uint8\_t **bootcause**  
*Temperatures [degC] [0 = TEMP1, TEMP2, TEMP3, TEMP4, BATT0, BATT1].*
- uint8\_t **battmode**  
*Cause of last EPS reset.*
- uint8\_t **pptmode**  
*Mode for battery [0 = initial, 1 = undervoltage, 2 = safemode, 3 = nominal, 4=full].*
- uint16\_t **reserved2**  
*Mode of PPT tracker [1=MPPT, 2=FIXED].*

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

- include/eps\_telem.h

## 4.10 eps\_hk\_out\_t Struct Reference

### Public Attributes

- uint16\_t **curout** [6]
- uint8\_t **output** [8]  
*Current out (switchable outputs) [mA].*
- uint16\_t **output\_on\_delta** [8]  
*Status of outputs\*\*.*
- uint16\_t **output\_off\_delta** [8]  
*Time till power on\*\* [s].*
- uint16\_t **latchup** [6]  
*Time till power off\*\* [s].*

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

- include/eps\_telem.h

## 4.11 eps\_hk\_t Struct Reference

### Public Attributes

- uint16\_t **vboost** [3]  
Voltage of boost converters [mV] [PV1, PV2, PV3].
- uint16\_t **vbatt**  
Voltage of battery [mV].
- uint16\_t **curin** [3]  
Current in [mA].
- uint16\_t **cursun**  
Current from boost converters [mA].
- uint16\_t **reserved1**  
Current out of battery [mA].
- uint16\_t **curout** [6]  
Reserved for future use.
- uint8\_t **output** [8]  
Current out (switchable outputs) [mA].
- uint16\_t **output\_on\_delta** [8]  
Status of outputs\*\*.
- uint16\_t **output\_off\_delta** [8]  
Time till power on\*\* [s].
- uint16\_t **latchup** [6]  
Time till power off\*\* [s].
- uint32\_t **wdt\_i2c\_time\_left**  
Number of latch-ups.
- uint32\_t **wdt\_gnd\_time\_left**  
Time left on I2C wdt [s].
- uint8\_t **wdt\_csp\_pings\_left** [2]  
Time left on I2C wdt [s].
- uint32\_t **counter\_wdt\_i2c**  
Pings left on CSP wdt.
- uint32\_t **counter\_wdt\_gnd**  
Number of WDT I2C reboots.
- uint32\_t **counter\_wdt\_csp** [2]  
Number of WDT GND reboots.
- uint32\_t **counter\_boot**  
Number of WDT CSP reboots.
- int16\_t **temp** [6]  
Number of EPS reboots.
- uint8\_t **bootcause**  
Temperatures [degC] [0 = TEMP1, TEMP2, TEMP3, TEMP4, BP4a, BP4b].
- uint8\_t **battmode**  
Cause of last EPS reset.
- uint8\_t **pptmode**  
Mode for battery [0 = initial, 1 = undervoltage, 2 = safemode, 3 = nominal, 4=full].

- uint16\_t [reserved2](#)  
*Mode of PPT tracker [1=MPPT, 2=FIXED].*

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

- include/[eps\\_telem.h](#)

## 4.12 eps\_hk\_vi\_t Struct Reference

### Public Attributes

- uint16\_t **vboost** [3]
- uint16\_t [vbatt](#)  
*Voltage of boost converters [mV] [PV1, PV2, PV3].*
- uint16\_t [curin](#) [3]  
*Voltage of battery [mV].*
- uint16\_t [cursun](#)  
*Current in [mA].*
- uint16\_t [cursys](#)  
*Current from boost converters [mA].*
- uint16\_t [reserved1](#)  
*Current out of battery [mA].*

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

- include/[eps\\_telem.h](#)

## 4.13 eps\_hk\_wdt\_t Struct Reference

### Public Attributes

- uint32\_t **wdt\_i2c\_time\_left**
- uint32\_t [wdt\\_gnd\\_time\\_left](#)  
*Time left on I2C wdt [s].*
- uint8\_t [wdt\\_csp\\_pings\\_left](#) [2]  
*Time left on I2C wdt [s].*
- uint32\_t [counter\\_wdt\\_i2c](#)  
*Pings left on CSP wdt.*
- uint32\_t [counter\\_wdt\\_gnd](#)  
*Number of WDT I2C reboots.*
- uint32\_t [counter\\_wdt\\_csp](#) [2]  
*Number of WDT GND reboots.*

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

- include/[eps\\_telem.h](#)

## 4.14 hkparam\_t Struct Reference

### Public Attributes

- uint16\_t **pv** [3]
- uint16\_t **pc**
- uint16\_t **bv**
- uint16\_t **sc**
- int16\_t **temp** [4]
- int16\_t **batt\_temp** [2]
- uint16\_t **latchup** [6]
- uint8\_t **reset**
- uint16\_t **bootcount**
- uint16\_t **sw\_errors**
- uint8\_t **ppt\_mode**
- uint8\_t **channel\_status**

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

- include/[eps\\_telem.h](#)

## 4.15 lsm9ds1 Struct Reference

LSM9DS1 Device Struct.

```
#include <lsm9ds1.h>
```

### Public Attributes

- int [accel\\_file](#)  
*File descriptor for accelerometer + gyro.*
- int [mag\\_file](#)  
*File descriptor for magnetometer.*
- char [fname](#) [40]  
*I2C Bus file name.*

### 4.15.1 Detailed Description

LSM9DS1 Device Struct.

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

- drivers/[lsm9ds1.h](#)

## 4.16 helmholtz.lsm9ds1 Class Reference

### Public Member Functions

- def **\_\_init\_\_** (self, busnum, xl\_addr, mag\_addr)
- def **readMag** (self)
- def **\_\_del\_\_** (self)

### Public Attributes

- **sbus**
- **xl\_addr**
- **mag\_addr**

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

- calibration/helmholtz.py

## 4.17 MAG\_DATA\_RATE Struct Reference

Configuration for magnetometer data rate.

```
#include <lsm9ds1.h>
```

### Public Attributes

- uint8\_t **self\_test**: 1  
*Self test enable. Default: 0. (0: disabled, 1: enabled)*
- uint8\_t **fast\_odr**: 1  
*Enables data rates faster than 80 Hz. Default: 0 (0: disabled, 1: enabled)*
- uint8\_t **data\_rate**: 3  
*Sets data rate from the sensor when fast\_odr is disabled.*
- uint8\_t **operative\_mode**: 2  
*X and Y axes operative mode selection. Default value: 00.*
- uint8\_t **temp\_comp**: 1  
*Temperature compensation enable. Default value: 0 (0: temperature compensation disabled; 1: temperature compensation enabled)*

### 4.17.1 Detailed Description

Configuration for magnetometer data rate.

## 4.17.2 Member Data Documentation

### 4.17.2.1 data\_rate

```
uint8_t MAG_DATA_RATE::data_rate
```

Sets data rate from the sensor when fast\_odr is disabled.

Set Data Rate in Hz. 000: 0.625 Hz 001: 1.25 Hz 010: 2.5 Hz 011: 5 Hz 100: 10 Hz (Default) 101: 20 Hz (SPACE HAUC setting) 110: 40 Hz 111: 80 Hz

### 4.17.2.2 operative\_mode

```
uint8_t MAG_DATA_RATE::operative_mode
```

X and Y axes operative mode selection. Default value: 00.

Operative mode for X and Y axes. 00: LP mode (Default) 01: Medium perf 10: High perf 11: Ultra-high perf

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

- [drivers/lsm9ds1.h](#)

## 4.18 MAG\_DATA\_READ Struct Reference

### Public Attributes

- `uint8_t reserved`: 6  
*Reserved, must be 0.*
- `uint8_t bdu`: 1  
*Block data update for magnetic data. 0: Continuous update, 1: Output registers not updated until MSB and LSB has been read.*
- `uint8_t fast_read`: 1  
*FAST\_READ allows reading the high part of DATA OUT only in order to increase reading efficiency. Default: 0 0: FAST\_READ disabled, 1: Enabled.*

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

- [drivers/lsm9ds1.h](#)

## 4.19 MAG\_RESET Struct Reference

### Public Attributes

- `uint8_t reserved`: 2  
*Reserved, must be 0.*
- `uint8_t soft_rst`: 1  
*Configuration registers and user register reset function. (0: default value; 1: reset operation)*
- `uint8_t reboot`: 1  
*Reboot memory content. Default value: 0 (0: normal mode; 1: reboot memory content)*
- `uint8_t reserved2`: 1  
*Reserved, must be 0.*
- `uint8_t full_scale`: 2  
*Full-scale configuration. Default value: 00 00: +/- 4 Gauss 01: +/- 8 Gauss 10: +/- 12 Gauss 11: +/- 16 Gauss.*
- `uint8_t reserved3`: 1  
*Reserved, must be 0.*

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

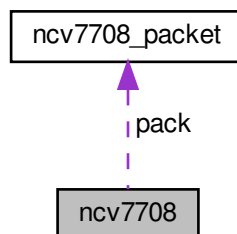
- `drivers/lsm9ds1.h`

## 4.20 ncv7708 Struct Reference

NCV77X8 Device.

```
#include <ncv7708.h>
```

Collaboration diagram for ncv7708:





## Public Attributes

- struct spi\_ioc\_transfer [xfer](#) [1]  
*SPI Transfer IO buffer.*
- int [file](#)  
*File descriptor for SPI bus.*
- \_\_u8 [mode](#)  
*SPI Mode (Mode 0)*
- \_\_u8 [lsb](#)  
*MSB First.*
- \_\_u8 [bits](#)  
*Number of bits per transfer (16)*
- \_\_u32 [speed](#)  
*SPI Bus speed (1 MHz)*
- char [fname](#) [40]  
*SPI device file name.*
- [ncv7708\\_packet](#) \* [pack](#)  
*Pointer to [ncv7708\\_packet](#) for internal consistency.*

### 4.20.1 Detailed Description

NCV77X8 Device.

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

- drivers/[ncv7708.h](#)

## 4.21 ncv7708\_packet Struct Reference

NCV77X8 Data packet (I/O)

```
#include <ncv7708.h>
```

## Public Attributes

-

```

union {
 unsigned short cmd
 Combined bits.
 struct {
 unsigned char ovlo: 1
 over voltage lockout
 unsigned char hbcnf1: 1
 half bridge 1 configuration (1 -> LS1 off and HS1 on, 0 -> LS1 on and HS1 off)
 unsigned char hbcnf2: 1
 unsigned char hbcnf3: 1
 unsigned char hbcnf4: 1
 unsigned char hbcnf5: 1
 unsigned char hbcnf6: 1
 unsigned char hben1: 1
 half bridge 1 enable (1 -> bridge in use, 0 -> bridge not in use)
 unsigned char hben2: 1
 unsigned char hben3: 1
 unsigned char hben4: 1
 unsigned char hben5: 1
 unsigned char hben6: 1
 unsigned char uldsc: 1
 under load detection shutdown
 unsigned char hbse1: 1
 half bridge selection (needs to be set to 0)
 unsigned char srr: 1
 status reset register: 1 -> clear all faults and reset
 }
};

```

•

```

union {
 unsigned short data
 struct {
 unsigned char tw: 1
 thermal warning
 unsigned char hbcr1: 1
 half bridge 1 configuration reporting (mirrors command)
 unsigned char hbcr2: 1
 unsigned char hbcr3: 1
 unsigned char hbcr4: 1
 unsigned char hbcr5: 1
 unsigned char hbcr6: 1
 unsigned char hbst1: 1
 half bridge 1 enable status (mirrors command)
 unsigned char hbst2: 1
 unsigned char hbst3: 1
 unsigned char hbst4: 1
 unsigned char hbst5: 1
 unsigned char hbst6: 1
 unsigned char uld: 1
 under load detection (1 -> fault)
 unsigned char psf: 1
 power supply failure
 unsigned char ocs: 1
 }
};

```

```

 over current shutdown
 }
};

```

### 4.21.1 Detailed Description

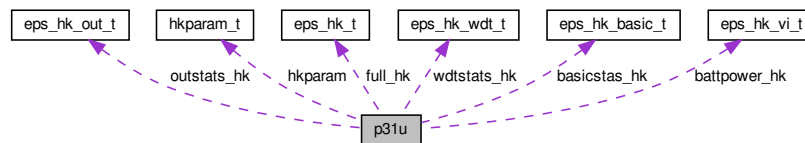
NCV77X8 Data packet (I/O)

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

- [drivers/ncv7708.h](#)

## 4.22 p31u Struct Reference

Collaboration diagram for p31u:



### Public Attributes

- int **file**
- char **fname** [40]  
*I2C File Descriptor.*
- uint8\_t **addr**  
*I2C File Name.*
- [hkparam\\_t](#) **hkparam**  
*Device Address.*
- [eps\\_hk\\_t](#) **full\_hk**  
*hkparam\_t structure memory*
- [eps\\_hk\\_vi\\_t](#) **battpower\_hk**  
*Full housekeeping data.*
- [eps\\_hk\\_out\\_t](#) **outstats\_hk**  
*battery voltage and current data*
- [eps\\_hk\\_wdt\\_t](#) **wdtstats\_hk**  
*Output status and current data.*
- [eps\\_hk\\_basic\\_t](#) **basicstas\_hk**  
*Watchdog status data.*

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

- [include/eps\\_telem.h](#)

## 4.23 tca9458a Struct Reference

TCA9458A Device handle.

```
#include <tca9458a.h>
```

### Public Attributes

- int [fd](#)  
*File descriptor for I2C Bus.*
- char [fname](#) [40]  
*File name for I2C Bus.*
- uint8\_t [channel](#)  
*Current active channel.*

### 4.23.1 Detailed Description

TCA9458A Device handle.

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

- [drivers/tca9458a.h](#)

## 4.24 tsl2561 Struct Reference

TSL2561 Device Handle.

```
#include <tsl2561.h>
```

### Public Attributes

- int [fd](#)  
*File descriptor for I2C bus.*
- char [fname](#) [40]  
*I2C Device name.*

### 4.24.1 Detailed Description

TSL2561 Device Handle.

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

- [drivers/tsl2561.h](#)

## Chapter 5

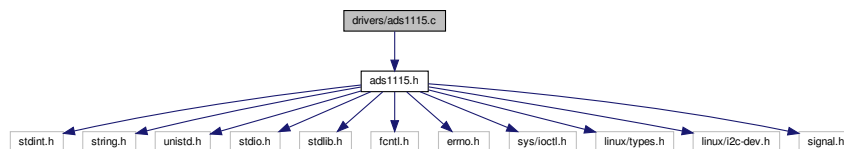
# File Documentation

### 5.1 drivers/ads1115.c File Reference

ADS1115 I2C Driver function definitions.

```
#include "ads1115.h"
```

Include dependency graph for ads1115.c:



### Functions

- int `ads1115_init` (`ads1115` \*dev, uint8\_t s\_address)  
*Initializes an ADS1115 device. Opens the I2C device named in ads1115->fname.*
- int `ads1115_configure` (`ads1115` \*dev, `ads1115_config` m\_con)  
*Configures an ADS1115 device.*
- int `ads1115_read_data` (`ads1115` \*dev, int16\_t \*data)  
*Reads data from the ADC in single shot.*
- int `ads1115_read_cont` (`ads1115` \*dev, int16\_t \*data)  
*Reads data from the ADC in continuous mode.*
- int `ads1115_read_config` (`ads1115` \*dev, uint16\_t \*data)  
*Read current configuration of an ADS1115.*
- void `ads1115_destroy` (`ads1115` \*dev)  
*Powers down ADS1115 device and closes file descriptor.*

### 5.1.1 Detailed Description

ADS1115 I2C Driver function definitions.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.1.2 Function Documentation

#### 5.1.2.1 `ads1115_configure()`

```
int ads1115_configure (
 ads1115 * dev,
 ads1115_config m_con)
```

Configures an ADS1115 device.

#### Parameters

|              |                                                   |
|--------------|---------------------------------------------------|
| <i>dev</i>   | Pointer to <a href="#">ads1115</a> device struct. |
| <i>m_con</i> | Configuration to apply                            |

#### Returns

Returns 1 on success, -1 on failure.

### 5.1.2.2 ads1115\_destroy()

```
void ads1115_destroy (
 ads1115 * dev)
```

Powers down ADS1115 device and closes file descriptor.

#### Parameters

|            |                                                   |
|------------|---------------------------------------------------|
| <i>dev</i> | Pointer to <a href="#">ads1115</a> device struct. |
|------------|---------------------------------------------------|

### 5.1.2.3 ads1115\_init()

```
int ads1115_init (
 ads1115 * dev,
 uint8_t s_address)
```

Initializes an ADS1115 device. Opens the I2C device named in `ads1115->fname`.

#### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <i>dev</i>       | Pointer to <a href="#">ads1115</a> device struct. |
| <i>s_address</i> | 7-bit I2C address                                 |

#### Returns

Returns 1 on success, -1 on failure. Sets `errno`.

### 5.1.2.4 ads1115\_read\_config()

```
int ads1115_read_config (
 ads1115 * dev,
 uint16_t * data)
```

Read current configuration of an ADS1115.

#### Parameters

|             |                                                                   |
|-------------|-------------------------------------------------------------------|
| <i>dev</i>  | Pointer to <a href="#">ads1115</a> device struct.                 |
| <i>data</i> | Pointer to unsigned short ( <code>ads1115_config-&gt;raw</code> ) |

**Returns**

Returns 1 on success, -1 on failure.

**5.1.2.5 ads1115\_read\_cont()**

```
int ads1115_read_cont (
 ads1115 * dev,
 int16_t * data)
```

Reads data from the ADC in continuous mode.

**Parameters**

|             |                                                               |
|-------------|---------------------------------------------------------------|
| <i>dev</i>  | Pointer to <a href="#">ads1115</a> device struct.             |
| <i>data</i> | Pointer to an array of short of length 4 where data is stored |

**Returns**

Returns 1 on success, -1 on failure.

**5.1.2.6 ads1115\_read\_data()**

```
int ads1115_read_data (
 ads1115 * dev,
 int16_t * data)
```

Reads data from the ADC in single shot.

**Parameters**

|             |                                                               |
|-------------|---------------------------------------------------------------|
| <i>dev</i>  | Pointer to <a href="#">ads1115</a> device struct.             |
| <i>data</i> | Pointer to an array of short of length 4 where data is stored |

**Returns**

Returns 1 on success, -1 on failure.

## 5.2 drivers/ads1115.h File Reference

ADS1115 I2C Driver function prototypes and data structures.

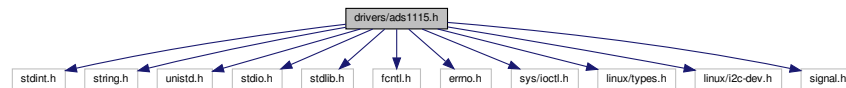


```

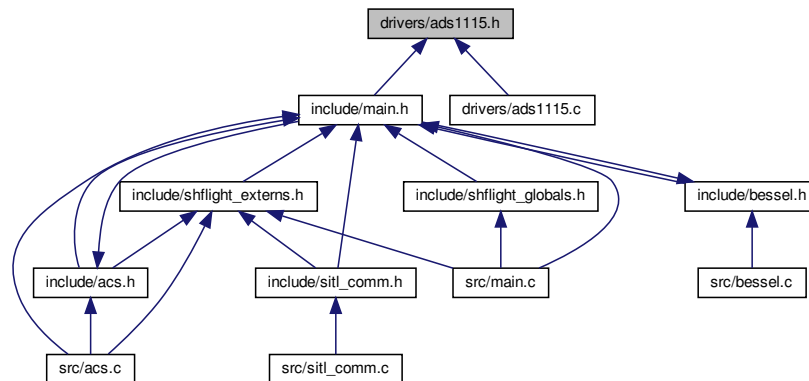
#include <stdint.h>
#include <string.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/ioctl.h>
#include <linux/types.h>
#include <linux/i2c-dev.h>
#include <signal.h>

```

Include dependency graph for ads115.h:



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



## Classes

- union [ads115\\_config](#)  
*Configuration register.*
- struct [ads115](#)  
*ads115 device data structures.*

## Macros

- #define [ADS115\\_S\\_ADDR](#) 0x48

*Default I2C Address.*

- `#define I2C_BUS "/dev/i2c-1"`

*Default I2C Bus.*

- `#define CONVERSION_REG 0x00`
- `#define CONFIG_REG 0x01`
- `#define CONFIG_REG_OS 0x80`
- `#define CONFIG_REG_MUX_0 0x40`
- `#define CONFIG_REG_MUX_1 0x50`
- `#define CONFIG_REG_MUX_2 0x60`
- `#define CONFIG_REG_MUX_3 0x70`

## Functions

- `int ads1115_init (ads1115 *dev, uint8_t s_address)`  
*Initializes an ADS1115 device. Opens the I2C device named in ads1115->fname.*
- `int ads1115_configure (ads1115 *dev, ads1115_config m_con)`  
*Configures an ADS1115 device.*
- `int ads1115_read_data (ads1115 *dev, int16_t *data)`  
*Reads data from the ADC in single shot.*
- `int ads1115_read_cont (ads1115 *dev, int16_t *data)`  
*Reads data from the ADC in continuous mode.*
- `int ads1115_read_config (ads1115 *dev, uint16_t *data)`  
*Read current configuration of an ADS1115.*
- `void ads1115_destroy (ads1115 *dev)`  
*Powers down ADS1115 device and closes file descriptor.*

### 5.2.1 Detailed Description

ADS1115 I2C Driver function prototypes and data structures.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

## 5.2.2 Function Documentation

### 5.2.2.1 ads1115\_configure()

```
int ads1115_configure (
 ads1115 * dev,
 ads1115_config m_con)
```

Configures an ADS1115 device.

#### Parameters

|              |                                                   |
|--------------|---------------------------------------------------|
| <i>dev</i>   | Pointer to <a href="#">ads1115</a> device struct. |
| <i>m_con</i> | Configuration to apply                            |

#### Returns

Returns 1 on success, -1 on failure.

### 5.2.2.2 ads1115\_destroy()

```
void ads1115_destroy (
 ads1115 * dev)
```

Powers down ADS1115 device and closes file descriptor.

#### Parameters

|            |                                                   |
|------------|---------------------------------------------------|
| <i>dev</i> | Pointer to <a href="#">ads1115</a> device struct. |
|------------|---------------------------------------------------|

### 5.2.2.3 ads1115\_init()

```
int ads1115_init (
 ads1115 * dev,
 uint8_t s_address)
```

Initializes an ADS1115 device. Opens the I2C device named in `ads1115->fname`.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <i>dev</i>       | Pointer to <a href="#">ads1115</a> device struct. |
| <i>s_address</i> | 7-bit I2C address                                 |

**Returns**

Returns 1 on success, -1 on failure. Sets errno.

**5.2.2.4 ads1115\_read\_config()**

```
int ads1115_read_config (
 ads1115 * dev,
 uint16_t * data)
```

Read current configuration of an ADS1115.

**Parameters**

|             |                                                   |
|-------------|---------------------------------------------------|
| <i>dev</i>  | Pointer to <a href="#">ads1115</a> device struct. |
| <i>data</i> | Pointer to unsigned short (ads1115_config->raw)   |

**Returns**

Returns 1 on success, -1 on failure.

**5.2.2.5 ads1115\_read\_cont()**

```
int ads1115_read_cont (
 ads1115 * dev,
 int16_t * data)
```

Reads data from the ADC in continuous mode.

**Parameters**

|             |                                                               |
|-------------|---------------------------------------------------------------|
| <i>dev</i>  | Pointer to <a href="#">ads1115</a> device struct.             |
| <i>data</i> | Pointer to an array of short of length 4 where data is stored |

**Returns**

Returns 1 on success, -1 on failure.

**5.2.2.6 ads1115\_read\_data()**

```
int ads1115_read_data (
 ads1115 * dev,
 int16_t * data)
```

Reads data from the ADC in single shot.

**Parameters**

|             |                                                               |
|-------------|---------------------------------------------------------------|
| <i>dev</i>  | Pointer to <a href="#">ads1115</a> device struct.             |
| <i>data</i> | Pointer to an array of short of length 4 where data is stored |

**Returns**

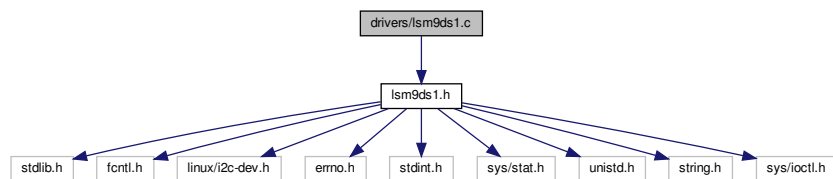
Returns 1 on success, -1 on failure.

**5.3 drivers/lsm9ds1.c File Reference**

Function definitions for LSM9DS1 Magnetometer I2C driver.

```
#include "lsm9ds1.h"
```

Include dependency graph for lsm9ds1.c:

**Functions**

- int `lsm9ds1_init` (`lsm9ds1` \*dev, uint8\_t xl\_addr, uint8\_t mag\_addr)  
*Takes the pointer to the device struct, XL address and M address, returns 1 on success, negative numbers on failure.*
- int `lsm9ds1_config_mag` (`lsm9ds1` \*dev, `MAG_DATA_RATE` datarate, `MAG_RESET` rst, `MAG_DATA_READ` dread)

*Configure the data rate, reset vector and data granularity.*

- int `lsm9ds1_reset_mag` (`lsm9ds1 *dev`)

*Reset the magnetometer memory.*

- int `lsm9ds1_read_mag` (`lsm9ds1 *dev`, short \*B)

*Store the magnetic field readings in the array of shorts, order: X Y Z.*

- int `lsm9ds1_offset_mag` (`lsm9ds1 *dev`, short \*offset)

*Set the mag field offsets using the array, order: X Y Z.*

- void `lsm9ds1_destroy` (`lsm9ds1 *dev`)

*Closes the file descriptors for the mag and accel and frees the allocated memory.*

### 5.3.1 Detailed Description

Function definitions for LSM9DS1 Magnetometer I2C driver.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.3.2 Function Documentation

#### 5.3.2.1 `lsm9ds1_config_mag()`

```
int lsm9ds1_config_mag (
 lsm9ds1 * dev,
 MAG_DATA_RATE datarate,
 MAG_RESET rst,
 MAG_DATA_READ dread)
```

Configure the data rate, reset vector and data granularity.

## Parameters

|                 |                                    |
|-----------------|------------------------------------|
| <i>dev</i>      | Pointer to <a href="#">lsm9ds1</a> |
| <i>datarate</i> |                                    |
| <i>rst</i>      |                                    |
| <i>dread</i>    |                                    |

## Returns

Returns 1 on success, -1 on failure

5.3.2.2 `lsm9ds1_destroy()`

```
void lsm9ds1_destroy (
 lsm9ds1 * dev)
```

Closes the file descriptors for the mag and accel and frees the allocated memory.

## Parameters

|            |                                    |
|------------|------------------------------------|
| <i>dev</i> | Pointer to <a href="#">lsm9ds1</a> |
|------------|------------------------------------|

5.3.2.3 `lsm9ds1_init()`

```
int lsm9ds1_init (
 lsm9ds1 * dev,
 uint8_t xl_addr,
 uint8_t mag_addr)
```

Takes the pointer to the device struct, XL address and M address, returns 1 on success, negative numbers on failure.

## Parameters

|                 |                                                 |
|-----------------|-------------------------------------------------|
| <i>dev</i>      | Pointer to <a href="#">lsm9ds1</a>              |
| <i>xl_addr</i>  | Accelerometer address on I2C Bus (default 0x6b) |
| <i>mag_addr</i> | magnetometer address on I2C Bus (default 0x1e)  |

## Returns

Returns 1 on success, -1 on failure

#### 5.3.2.4 lsm9ds1\_offset\_mag()

```
int lsm9ds1_offset_mag (
 lsm9ds1 * dev,
 short * offset)
```

Set the mag field offsets using the array, order: X Y Z.

##### Parameters

|            |                                                                              |
|------------|------------------------------------------------------------------------------|
| <i>dev</i> | Pointer to <a href="#">lsm9ds1</a>                                           |
| <i>B</i>   | Pointer to an array of short of length 3 where magnetometer offset is stored |

##### Returns

Returns 1 on success, -1 on failure

#### 5.3.2.5 lsm9ds1\_read\_mag()

```
int lsm9ds1_read_mag (
 lsm9ds1 * dev,
 short * B)
```

Store the magnetic field readings in the array of shorts, order: X Y Z.

##### Parameters

|            |                                                                               |
|------------|-------------------------------------------------------------------------------|
| <i>dev</i> | Pointer to <a href="#">lsm9ds1</a>                                            |
| <i>B</i>   | Pointer to an array of short of length 3 where magnetometer reading is stored |

##### Returns

Returns 1 on success, -1 on failure

#### 5.3.2.6 lsm9ds1\_reset\_mag()

```
int lsm9ds1_reset_mag (
 lsm9ds1 * dev)
```

Reset the magnetometer memory.



## Parameters

|            |                                    |
|------------|------------------------------------|
| <i>dev</i> | Pointer to <a href="#">lsm9ds1</a> |
|------------|------------------------------------|

## Returns

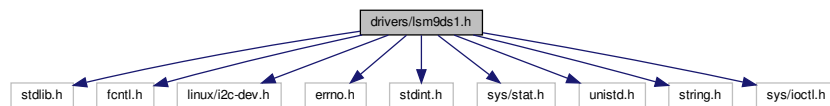
Returns 1 on success, -1 on failure

## 5.4 drivers/lsm9ds1.h File Reference

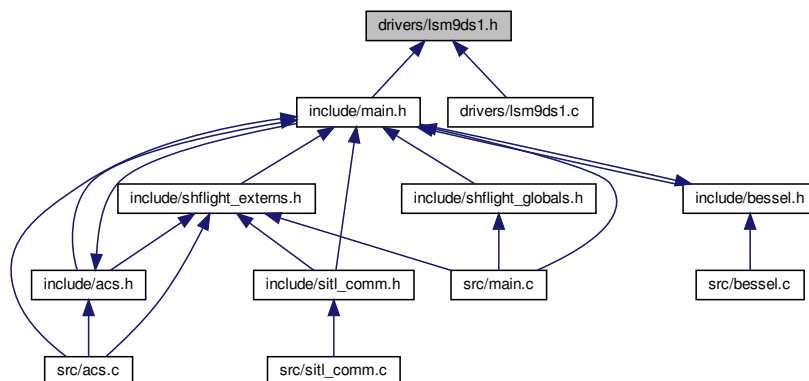
Function prototypes and data structures for LSM9DS1 Magnetometer I2C driver.

```
#include <stdlib.h>
#include <fcntl.h>
#include <linux/i2c-dev.h>
#include <errno.h>
#include <stdint.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
```

Include dependency graph for `lsm9ds1.h`:



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



## Classes

- struct [MAG\\_DATA\\_RATE](#)  
*Configuration for magnetometer data rate.*
- struct [MAG\\_RESET](#)
- struct [MAG\\_DATA\\_READ](#)
- struct [lsm9ds1](#)  
*LSM9DS1 Device Struct.*

## Macros

- `#define MAG\_I2C\_File "/dev/i2c-1"`  
*Default I2C device address.*
- `#define LSM9DS1\_XL\_ADDR 0x6b`  
*Accelerometer address.*
- `#define LSM9DS1\_MAG\_ADDR 0x1e`  
*Magnetometer address.*
- `#define LSM9DS1\_CTRL\_REG1\_G 0x10`  
– *Accelerometer and Gyro registers*
- `#define LSM9DS1\_GYRO\_PD 0x00`  
*Content of the gyro control register for power down.*
- `#define LSM9DS1\_CTRL\_REG5\_XL 0x1f`  
*Acceleration control register.*
- `#define LSM9DS1\_XL\_PD 0x00`  
*Disable outputs.*
- `#define LSM9DS1\_CTRL\_REG6\_XL 0x20`  
– *ODR\_XL[7:5]: Output data rate and power mode, 0 0 0 for power down. FS\_XL[4:3]: Full scale selection. BW\_SC↔AL\_ODR[2:2]: Bandwidth selection, 0 – default, 1 – bandwidth from BW\_XL. BW\_XL[1:0]: Custom bandwidth.*
- `#define MAG\_CTRL\_REG1\_M 0x20`  
*Magnetometer control register 1 address.*
- `#define MAG\_CTRL\_REG2\_M 0x21`  
*Magnetometer control register 2 address.*
- `#define MAG\_CTRL\_REG3\_M 0x22`  
*Magnetometer control register 3 address, write 0x0 to this.*
- `#define MAG\_CTRL\_REG4\_M 0x23`  
*Magnetometer control register 4 address.*
- `#define MAG\_CTRL\_REG4\_DATA 0x0c`  
*Magnetometer control register 4: [11][0 0], ultra high Z performance + little endian register data selection.*
- `#define MAG\_CTRL\_REG5\_M 0x24`  
*Magnetometer control register 5 address.*
- `#define MAG\_WHO\_AM\_I 0x0f`  
*Address of magnetometer ID register.*
- `#define MAG\_IDENT 0b00111101`  
*Magnetometer ID.*

## Enumerations

- enum `MAG_OFFSET_REGISTERS` {  
`MAG_OFFSET_X_REG_L_M = 0x05`, `MAG_OFFSET_X_REG_H_M`, `MAG_OFFSET_Y_REG_L_M`, `MAG_OFFSET_Y_REG_H_M`,  
`MAG_OFFSET_Z_REG_L_M`, `MAG_OFFSET_Z_REG_H_M` }  
*Magnetometer registers.*
- enum `MAG_OUT_DATA` {  
`MAG_OUT_X_L = 0x28`, `MAG_OUT_X_H`, `MAG_OUT_Y_L`, `MAG_OUT_Y_H`,  
`MAG_OUT_Z_L`, `MAG_OUT_Z_H` }  
*Magnetometer measurement register addresses.*

## Functions

- int `lsm9ds1_init` (`lsm9ds1 *`, `uint8_t`, `uint8_t`)  
*Takes the pointer to the device struct, XL address and M address, returns 1 on success, negative numbers on failure.*
- int `lsm9ds1_config_mag` (`lsm9ds1 *`, `MAG_DATA_RATE`, `MAG_RESET`, `MAG_DATA_READ`)  
*Configure the data rate, reset vector and data granularity.*
- int `lsm9ds1_reset_mag` (`lsm9ds1 *`)  
*Reset the magnetometer memory.*
- int `lsm9ds1_read_mag` (`lsm9ds1 *`, `short *`)  
*Store the magnetic field readings in the array of shorts, order: X Y Z.*
- int `lsm9ds1_offset_mag` (`lsm9ds1 *`, `short *`)  
*Set the mag field offsets using the array, order: X Y Z.*
- void `lsm9ds1_destroy` (`lsm9ds1 *`)  
*Closes the file descriptors for the mag and accel and frees the allocated memory.*

### 5.4.1 Detailed Description

Function prototypes and data structures for LSM9DS1 Magnetometer I2C driver.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

## 5.4.2 Macro Definition Documentation

### 5.4.2.1 LSM9DS1\_CTRL\_REG1\_G

```
#define LSM9DS1_CTRL_REG1_G 0x10
```

- Accelerometer and Gyro registers

NOTE: Few registers are used ONLY TO power down the accelerometer and the gyroscope.

## 5.4.3 Enumeration Type Documentation

### 5.4.3.1 MAG\_OFFSET\_REGISTERS

```
enum MAG_OFFSET_REGISTERS
```

Magnetometer registers.

Enumerator

|                      |                                       |
|----------------------|---------------------------------------|
| MAG_OFFSET_X_REG_L_M | Magnetometer X axis offset LOW byte.  |
| MAG_OFFSET_X_REG_H_M | Magnetometer X axis offset HIGH byte. |
| MAG_OFFSET_Y_REG_L_M | Magnetometer Y axis offset LOW byte.  |
| MAG_OFFSET_Y_REG_H_M | Magnetometer Y axis offset HIGH byte. |
| MAG_OFFSET_Z_REG_L_M | Magnetometer Z axis offset LOW byte.  |
| MAG_OFFSET_Z_REG_H_M | Magnetometer Z axis offset HIGH byte. |

### 5.4.3.2 MAG\_OUT\_DATA

```
enum MAG_OUT_DATA
```

Magnetometer measurement register addresses.

## Enumerator

|             |                                            |
|-------------|--------------------------------------------|
| MAG_OUT_X_L | Magnetometer X axis measurement LOW byte.  |
| MAG_OUT_X_H | Magnetometer X axis measurement HIGH byte. |
| MAG_OUT_Y_L | Magnetometer Y axis measurement LOW byte.  |
| MAG_OUT_Y_H | Magnetometer Y axis measurement HIGH byte. |
| MAG_OUT_Z_L | Magnetometer Z axis measurement LOW byte.  |
| MAG_OUT_Z_H | Magnetometer Z axis measurement HIGH byte. |

## 5.4.4 Function Documentation

## 5.4.4.1 lsm9ds1\_config\_mag()

```
int lsm9ds1_config_mag (
 lsm9ds1 * dev,
 MAG_DATA_RATE datarate,
 MAG_RESET rst,
 MAG_DATA_READ dread)
```

Configure the data rate, reset vector and data granularity.

## Parameters

|                 |                                    |
|-----------------|------------------------------------|
| <i>dev</i>      | Pointer to <a href="#">lsm9ds1</a> |
| <i>datarate</i> |                                    |
| <i>rst</i>      |                                    |
| <i>dread</i>    |                                    |

## Returns

Returns 1 on success, -1 on failure

## 5.4.4.2 lsm9ds1\_destroy()

```
void lsm9ds1_destroy (
 lsm9ds1 * dev)
```

Closes the file descriptors for the mag and accel and frees the allocated memory.

## Parameters

|            |                                    |
|------------|------------------------------------|
| <i>dev</i> | Pointer to <a href="#">lsm9ds1</a> |
|------------|------------------------------------|

5.4.4.3 `lsm9ds1_init()`

```
int lsm9ds1_init (
 lsm9ds1 * dev,
 uint8_t xl_addr,
 uint8_t mag_addr)
```

Takes the pointer to the device struct, XL address and M address, returns 1 on success, negative numbers on failure.

## Parameters

|                 |                                                 |
|-----------------|-------------------------------------------------|
| <i>dev</i>      | Pointer to <a href="#">lsm9ds1</a>              |
| <i>xl_addr</i>  | Accelerometer address on I2C Bus (default 0x6b) |
| <i>mag_addr</i> | magnetometer address on I2C Bus (default 0x1e)  |

## Returns

Returns 1 on success, -1 on failure

5.4.4.4 `lsm9ds1_offset_mag()`

```
int lsm9ds1_offset_mag (
 lsm9ds1 * dev,
 short * offset)
```

Set the mag field offsets using the array, order: X Y Z.

## Parameters

|            |                                                                              |
|------------|------------------------------------------------------------------------------|
| <i>dev</i> | Pointer to <a href="#">lsm9ds1</a>                                           |
| <i>B</i>   | Pointer to an array of short of length 3 where magnetometer offset is stored |

## Returns

Returns 1 on success, -1 on failure

#### 5.4.4.5 lsm9ds1\_read\_mag()

```
int lsm9ds1_read_mag (
 lsm9ds1 * dev,
 short * B)
```

Store the magnetic field readings in the array of shorts, order: X Y Z.

##### Parameters

|            |                                                                               |
|------------|-------------------------------------------------------------------------------|
| <i>dev</i> | Pointer to <a href="#">lsm9ds1</a>                                            |
| <i>B</i>   | Pointer to an array of short of length 3 where magnetometer reading is stored |

##### Returns

Returns 1 on success, -1 on failure

#### 5.4.4.6 lsm9ds1\_reset\_mag()

```
int lsm9ds1_reset_mag (
 lsm9ds1 * dev)
```

Reset the magnetometer memory.

##### Parameters

|            |                                    |
|------------|------------------------------------|
| <i>dev</i> | Pointer to <a href="#">lsm9ds1</a> |
|------------|------------------------------------|

##### Returns

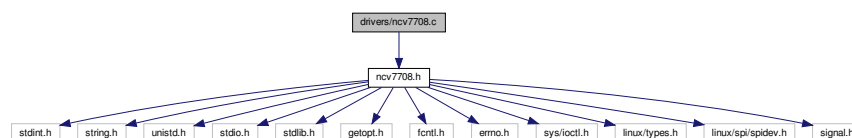
Returns 1 on success, -1 on failure

## 5.5 drivers/ncv7708.c File Reference

Function definitions for NCV77X8 SPI Driver (Linux)

```
#include "ncv7708.h"
```

Include dependency graph for ncv7708.c:





## Functions

- int `ncv7708_init` (`ncv7708` \*dev)  
*Initialize the SPI bus to communicate with the NCV77X8.*
- int `ncv7708_transfer` (`ncv7708` \*dev, uint16\_t \*data, uint16\_t \*cmd)  
*Makes an SPI transaction for a NCV77X8 device.*
- int `ncv7708_xfer` (`ncv7708` \*dev)  
*Makes an SPI transaction using internal data.*
- void `ncv7708_destroy` (`ncv7708` \*dev)  
*Closes SPI bus file descriptor and frees memory allocated for device.*

### 5.5.1 Detailed Description

Function definitions for NCV77X8 SPI Driver (Linux)

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.5.2 Function Documentation

#### 5.5.2.1 `ncv7708_destroy()`

```
void ncv7708_destroy (
 ncv7708 * dev)
```

Closes SPI bus file descriptor and frees memory allocated for device.

**Parameters**

|            |                       |
|------------|-----------------------|
| <i>dev</i> | NCV77X8 Device Handle |
|------------|-----------------------|

**5.5.2.2 ncv7708\_init()**

```
int ncv7708_init (
 ncv7708 * dev)
```

Initialize the SPI bus to communicate with the NCV77X8.

**Parameters**

|            |                       |
|------------|-----------------------|
| <i>dev</i> | NCV77X8 Device Handle |
|------------|-----------------------|

**Returns**

Returns 1 on success, 0 on SPI ioctl failures, -1 on device setup failure.

**5.5.2.3 ncv7708\_transfer()**

```
int ncv7708_transfer (
 ncv7708 * dev,
 uint16_t * data,
 uint16_t * cmd)
```

Makes an SPI transaction for a NCV77X8 device.

**Parameters**

|             |                                            |
|-------------|--------------------------------------------|
| <i>dev</i>  | NCV77X8 Device Handle                      |
| <i>data</i> | Pointer to store 16-bit data read over SPI |
| <i>cmd</i>  | Pointer to 16-bit data sent over SPI       |

**Returns**

1 on success, -1 on failure

#### 5.5.2.4 ncv7708\_xfer()

```
int ncv7708_xfer (
 ncv7708 * dev)
```

Makes an SPI transaction using internal data.

##### Parameters

|            |                       |
|------------|-----------------------|
| <i>dev</i> | NCV77X8 Device Handle |
|------------|-----------------------|

##### Returns

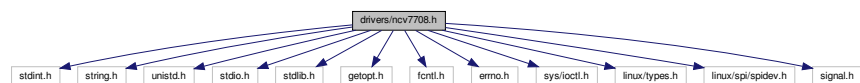
1 on success, -1 on failure

## 5.6 drivers/ncv7708.h File Reference

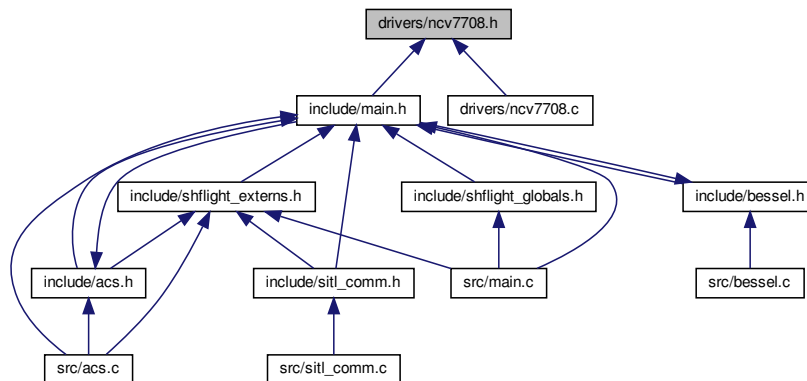
Function prototypes and data structure for NCV77X8 SPI Driver (Linux)

```
#include <stdint.h>
#include <string.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/ioctl.h>
#include <linux/types.h>
#include <linux/spi/spidev.h>
#include <signal.h>
```

Include dependency graph for ncv7708.h:



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



## Classes

- struct [ncv7708\\_packet](#)  
*NCV77X8 Data packet (I/O)*
- struct [ncv7708](#)  
*NCV77X8 Device.*

## Functions

- int [ncv7708\\_init](#) ([ncv7708](#) \*)  
*Initialize the SPI bus to communicate with the NCV77X8.*
- int [ncv7708\\_transfer](#) ([ncv7708](#) \*, uint16\_t \*, uint16\_t \*)  
*Makes an SPI transaction for a NCV77X8 device.*
- int [ncv7708\\_xfer](#) ([ncv7708](#) \*)  
*Makes an SPI transaction using internal data.*
- void [ncv7708\\_destroy](#) ([ncv7708](#) \*)  
*Closes SPI bus file descriptor and frees memory allocated for device.*

### 5.6.1 Detailed Description

Function prototypes and data structure for NCV77X8 SPI Driver (Linux)

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

**Version**

0.1

**Date**

2020-03-19

**Copyright**

Copyright (c) 2020

**5.6.2 Function Documentation****5.6.2.1 ncv7708\_destroy()**

```
void ncv7708_destroy (
 ncv7708 * dev)
```

Closes SPI bus file descriptor and frees memory allocated for device.

**Parameters**

|            |                       |
|------------|-----------------------|
| <i>dev</i> | NCV77X8 Device Handle |
|------------|-----------------------|

**5.6.2.2 ncv7708\_init()**

```
int ncv7708_init (
 ncv7708 * dev)
```

Initialize the SPI bus to communicate with the NCV77X8.

**Parameters**

|            |                       |
|------------|-----------------------|
| <i>dev</i> | NCV77X8 Device Handle |
|------------|-----------------------|

**Returns**

Returns 1 on success, 0 on SPI ioctl failures, -1 on device setup failure.

### 5.6.2.3 ncv7708\_transfer()

```
int ncv7708_transfer (
 ncv7708 * dev,
 uint16_t * data,
 uint16_t * cmd)
```

Makes an SPI transaction for a NCV77X8 device.

#### Parameters

|             |                                            |
|-------------|--------------------------------------------|
| <i>dev</i>  | NCV77X8 Device Handle                      |
| <i>data</i> | Pointer to store 16-bit data read over SPI |
| <i>cmd</i>  | Pointer to 16-bit data sent over SPI       |

#### Returns

1 on success, -1 on failure

### 5.6.2.4 ncv7708\_xfer()

```
int ncv7708_xfer (
 ncv7708 * dev)
```

Makes an SPI transaction using internal data.

#### Parameters

|            |                       |
|------------|-----------------------|
| <i>dev</i> | NCV77X8 Device Handle |
|------------|-----------------------|

#### Returns

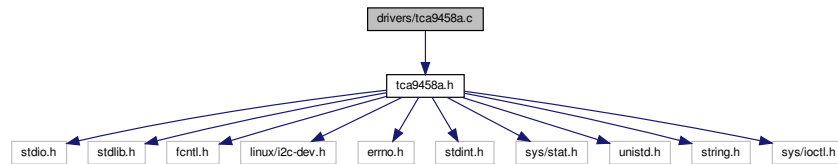
1 on success, -1 on failure

## 5.7 drivers/tca9458a.c File Reference

Function definitions for TCA9458A I2C driver.

```
#include "tca9458a.h"
```

Include dependency graph for tca9458a.c:



## Functions

- int [tca9458a\\_init](#) ([tca9458a](#) \*dev, uint8\_t addr)  
*Initialize a Mux device, returns 1 on success TODO: Implement a scan function at init where it checks all 3 CSS are present on 3 buses?*
- void [tca9458a\\_destroy](#) ([tca9458a](#) \*dev)  
*Disable all outputs, close file descriptor for the I2C Bus.*

### 5.7.1 Detailed Description

Function definitions for TCA9458A I2C driver.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.7.2 Function Documentation

#### 5.7.2.1 [tca9458a\\_destroy\(\)](#)

```
void tca9458a_destroy (
 tca9458a * dev)
```

Disable all outputs, close file descriptor for the I2C Bus.

## Parameters

|            |  |
|------------|--|
| <i>dev</i> |  |
|------------|--|

## 5.7.2.2 tca9458a\_init()

```
int tca9458a_init (
 tca9458a * dev,
 uint8_t addr)
```

Initialize a Mux device, returns 1 on success TODO: Implement a scan function at init where it checks all 3 CSS are present on 3 buses?

## Parameters

|             |                                         |
|-------------|-----------------------------------------|
| <i>dev</i>  |                                         |
| <i>addr</i> | TCA9458A device address (default: 0x70) |

## Returns

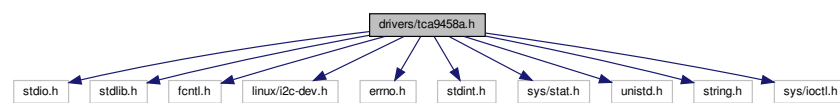
1 on success, -1 on error

## 5.8 drivers/tca9458a.h File Reference

Function prototypes and struct declarations for TCA9458A I2C driver.

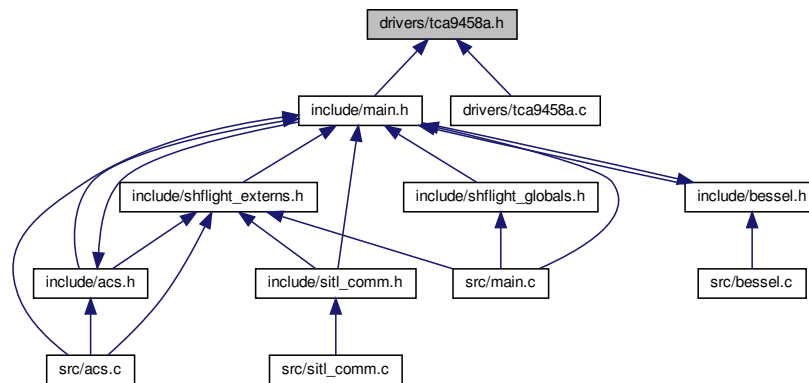
```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <linux/i2c-dev.h>
#include <errno.h>
#include <stdint.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
```

Include dependency graph for tca9458a.h:





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



## Classes

- struct [tca9458a](#)  
*TCA9458A Device handle.*

## Macros

- `#define MUX_I2C_File "/dev/i2c-1"`  
*I2C Device for Mux.*

## Functions

- int [tca9458a\\_init](#) ([tca9458a](#) \*, uint8\_t)  
*Initialize a Mux device, returns 1 on success TODO: Implement a scan function at init where it checks all 3 CSS are present on 3 buses?*
- int [tca9458a\\_set](#) ([tca9458a](#) \*dev, uint8\_t channel\_id)  
*Update active I2C channel (Inlined global symbol)*
- void [tca9458a\\_destroy](#) ([tca9458a](#) \*)  
*Disable all outputs, close file descriptor for the I2C Bus.*

### 5.8.1 Detailed Description

Function prototypes and struct declarations for TCA9458A I2C driver.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

**Version**

0.1

**Date**

2020-03-19

**Copyright**

Copyright (c) 2020

**5.8.2 Function Documentation****5.8.2.1 tca9458a\_destroy()**

```
void tca9458a_destroy (
 tca9458a * dev)
```

Disable all outputs, close file descriptor for the I2C Bus.

**Parameters**

|            |  |
|------------|--|
| <i>dev</i> |  |
|------------|--|

**5.8.2.2 tca9458a\_init()**

```
int tca9458a_init (
 tca9458a * dev,
 uint8_t addr)
```

Initialize a Mux device, returns 1 on success TODO: Implement a scan function at init where it checks all 3 CSS are present on 3 buses?

**Parameters**

|             |                                         |
|-------------|-----------------------------------------|
| <i>dev</i>  |                                         |
| <i>addr</i> | TCA9458A device address (default: 0x70) |

**Returns**

1 on success, -1 on error

**5.8.2.3 tca9458a\_set()**

```
int tca9458a_set (
 tca9458a * dev,
 uint8_t channel_id) [inline]
```

Update active I2C channel (Inlined global symbol)

**Parameters**

|                   |                   |
|-------------------|-------------------|
| <i>dev</i>        |                   |
| <i>channel_id</i> | Channel to enable |

**Returns**

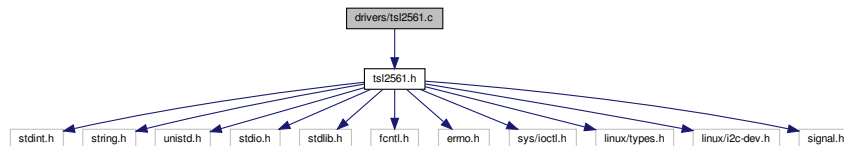
Returns 1 on success, 0 or -1 on error (see write())

**5.9 drivers/tsl2561.c File Reference**

TSL2561 I2C driver function definitions.

```
#include "tsl2561.h"
```

Include dependency graph for tsl2561.c:

**Functions**

- int **tsl2561\_init** (tsl2561 \*dev, uint8\_t s\_address)  
Init function for the TSL2561 device. Default: I2C\_BUS TODO: Fix init + gain, figure out what goes wrong if ID register is read.
- void **tsl2561\_measure** (tsl2561 \*dev, uint32\_t \*measure)

*Read I2C data into the uint32\_t measure var. \ Format: (MSB) broadband | ir (LSB)*

- uint32\_t [tsl2561\\_get\\_lux](#) (uint32\_t measure)

*Calculate lux using value measured using [tsl2561\\_measure\(\)](#)*

- void [tsl2561\\_destroy](#) ([tsl2561](#) \*dev)

*Destroy function for the TSL2561 device. Closes the file descriptor and powers down the device.*

### 5.9.1 Detailed Description

TSL2561 I2C driver function definitions.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.9.2 Function Documentation

#### 5.9.2.1 [tsl2561\\_destroy\(\)](#)

```
void tsl2561_destroy (
 tsl2561 * dev)
```

Destroy function for the TSL2561 device. Closes the file descriptor and powers down the device.

#### Parameters

|            |  |
|------------|--|
| <i>dev</i> |  |
|------------|--|

### 5.9.2.2 tsl2561\_get\_lux()

```
uint32_t tsl2561_get_lux (
 uint32_t measure)
```

Calculate lux using value measured using [tsl2561\\_measure\(\)](#)

#### Parameters

|                |  |
|----------------|--|
| <i>measure</i> |  |
|----------------|--|

#### Returns

Lux value

### 5.9.2.3 tsl2561\_init()

```
int tsl2561_init (
 tsl2561 * dev,
 uint8_t s_address)
```

Init function for the TSL2561 device. Default: I2C\_BUS TODO: Fix init + gain, figure out what goes wrong if ID register is read.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <i>dev</i>       |                                                  |
| <i>s_address</i> | Address for the device, values: 0x29, 0x39, 0x49 |

#### Returns

1 on success, -1 on failure

### 5.9.2.4 tsl2561\_measure()

```
void tsl2561_measure (
 tsl2561 * dev,
 uint32_t * measure)
```

Read I2C data into the uint32\_t measure var.\ Format: (MSB) broadband | ir (LSB)

## Parameters

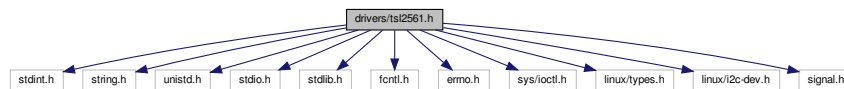
|                |                                                                |
|----------------|----------------------------------------------------------------|
| <i>dev</i>     |                                                                |
| <i>measure</i> | Pointer to unsigned 32 bit integer where measurement is stored |

## 5.10 drivers/tsl2561.h File Reference

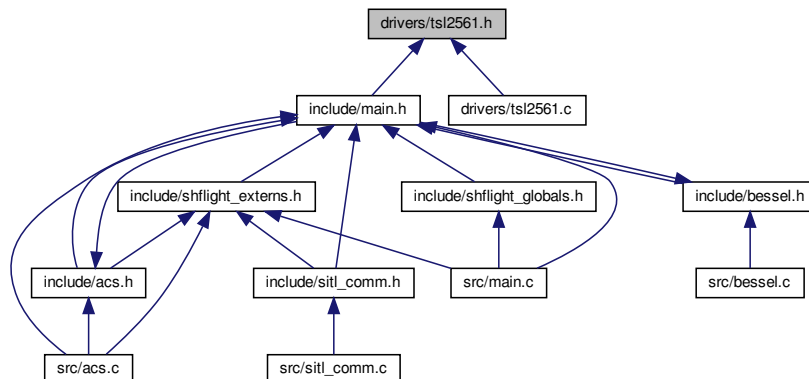
TSL2561 I2C driver function and struct declarations.

```
#include <stdint.h>
#include <string.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/ioctl.h>
#include <linux/types.h>
#include <linux/i2c-dev.h>
#include <signal.h>
```

Include dependency graph for tsl2561.h:



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



## Classes

- struct [tsl2561](#)  
*TSL2561 Device Handle.*

## Macros

- #define [TSL2561\\_VISIBLE](#) 2  
*channel 0 - channel 1*
- #define [TSL2561\\_INFRARED](#) 1  
*channel 1*
- #define [TSL2561\\_FULLSPECTRUM](#) 0  
*channel 0*
- #define [TSL2561\\_ADDR\\_LOW](#) (0x29)  
*Default address (pin pulled low)*
- #define [TSL2561\\_ADDR\\_FLOAT](#) (0x39)  
*Default address (pin left floating)*
- #define [TSL2561\\_ADDR\\_HIGH](#) (0x49)  
*Default address (pin pulled high)*
- #define [TSL2561\\_PACKAGE\\_T\\_FN\\_CL](#)  
*Dual Flat No-Lead package.*
- #define [TSL2561\\_COMMAND\\_BIT](#) (0x80)  
*Must be 1.*
- #define [TSL2561\\_CLEAR\\_BIT](#) (0x40)  
*Clears any pending interrupt (write 1 to clear)*
- #define [TSL2561\\_WORD\\_BIT](#) (0x20)  
*1 = read/write word (rather than byte)*
- #define [TSL2561\\_BLOCK\\_BIT](#) (0x10)  
*1 = using block read/write*
- #define [TSL2561\\_CONTROL\\_POWERON](#) (0x03)  
*Control register setting to turn on.*
- #define [TSL2561\\_CONTROL\\_POWEROFF](#) (0x00)  
*Control register setting to turn off.*
- #define [TSL2561\\_LUX\\_LUXSCALE](#) (14)  
*Scale by  $2^{14}$ .*
- #define [TSL2561\\_LUX\\_RATIOSCALE](#) (9)  
*Scale ratio by  $2^9$ .*
- #define [TSL2561\\_LUX\\_CHSCALE](#) (10)  
*Scale channel values by  $2^{10}$ .*
- #define [TSL2561\\_LUX\\_CHSCALE\\_TINT0](#) (0x7517)  
 *$322/11 * 2^{TSL2561\_LUX\_CHSCALE}$*
- #define [TSL2561\\_LUX\\_CHSCALE\\_TINT1](#) (0x0FE7)  
 *$322/81 * 2^{TSL2561\_LUX\_CHSCALE}$*
- #define [TSL2561\\_LUX\\_K1T](#) (0x0040)  
 *$0.125 * 2^{RATIO\_SCALE}$*
- #define [TSL2561\\_LUX\\_B1T](#) (0x01f2)  
 *$0.0304 * 2^{LUX\_SCALE}$*

- #define TSL2561\_LUX\_M1T (0x01be)  
 $0.0272 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K2T (0x0080)  
 $0.250 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B2T (0x0214)  
 $0.0325 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M2T (0x02d1)  
 $0.0440 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K3T (0x00c0)  
 $0.375 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B3T (0x023f)  
 $0.0351 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M3T (0x037b)  
 $0.0544 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K4T (0x0100)  
 $0.50 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B4T (0x0270)  
 $0.0381 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M4T (0x03fe)  
 $0.0624 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K5T (0x0138)  
 $0.61 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B5T (0x016f)  
 $0.0224 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M5T (0x01fc)  
 $0.0310 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K6T (0x019a)  
 $0.80 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B6T (0x00d2)  
 $0.0128 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M6T (0x00fb)  
 $0.0153 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K7T (0x029a)  
 $1.3 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B7T (0x0018)  
 $0.00146 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M7T (0x0012)  
 $0.00112 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K8T (0x029a)  
 $1.3 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B8T (0x0000)  
 $0.000 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M8T (0x0000)  
 $0.000 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K1C (0x0043)  
 $0.130 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B1C (0x0204)



- $0.0315 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M1C (0x01ad)
- $0.0262 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K2C (0x0085)
- $0.260 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B2C (0x0228)
- $0.0337 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M2C (0x02c1)
- $0.0430 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K3C (0x00c8)
- $0.390 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B3C (0x0253)
- $0.0363 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M3C (0x0363)
- $0.0529 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K4C (0x010a)
- $0.520 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B4C (0x0282)
- $0.0392 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M4C (0x03df)
- $0.0605 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K5C (0x014d)
- $0.65 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B5C (0x0177)
- $0.0229 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M5C (0x01dd)
- $0.0291 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K6C (0x019a)
- $0.80 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B6C (0x0101)
- $0.0157 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M6C (0x0127)
- $0.0180 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K7C (0x029a)
- $1.3 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B7C (0x0037)
- $0.00338 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M7C (0x002b)
- $0.00260 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_K8C (0x029a)
- $1.3 * 2^{\wedge} RATIO\_SCALE$
- #define TSL2561\_LUX\_B8C (0x0000)
- $0.000 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_LUX\_M8C (0x0000)
- $0.000 * 2^{\wedge} LUX\_SCALE$
- #define TSL2561\_AGC\_THI\_13MS (4850)
- Max value at Ti 13ms = 5047.*

- `#define TSL2561_AGC_TLO_13MS (100)`  
*Min value at Ti 13ms = 100.*
- `#define TSL2561_AGC_THI_101MS (36000)`  
*Max value at Ti 101ms = 37177.*
- `#define TSL2561_AGC_TLO_101MS (200)`  
*Min value at Ti 101ms = 200.*
- `#define TSL2561_AGC_THI_402MS (63000)`  
*Max value at Ti 402ms = 65535.*
- `#define TSL2561_AGC_TLO_402MS (500)`  
*Min value at Ti 402ms = 500.*
- `#define TSL2561_CLIPPING_13MS (4900)`  
*Counts that trigger a change in gain/integration.*
- `#define TSL2561_CLIPPING_101MS (37000)`  
*Counts that trigger a change in gain/integration.*
- `#define TSL2561_CLIPPING_402MS (65000)`  
*Counts that trigger a change in gain/integration.*
- `#define TSL2561_DELAY_INTTIME_13MS (15)`  
*Wait 15ms for 13ms integration.*
- `#define TSL2561_DELAY_INTTIME_101MS (120)`  
*Wait 120ms for 101ms integration.*
- `#define TSL2561_DELAY_INTTIME_402MS (450)`  
*Wait 450ms for 402ms integration.*
- `#define I2C_BUS "/dev/i2c-1"`  
*I2C bus name.*
- `#define TSL2561_BLOCK_READ 0x0B`  
*Block read mask.*

## Enumerations

- `enum {`  
`TSL2561_REGISTER_CONTROL = 0x00, TSL2561_REGISTER_TIMING = 0x01, TSL2561_REGISTER_THRESHL←`  
`RESHHOLDL_LOW = 0x02, TSL2561_REGISTER_THRESHHOLDL_HIGH = 0x03,`  
`TSL2561_REGISTER_THRESHHOLDH_LOW = 0x04, TSL2561_REGISTER_THRESHHOLDH_HIGH = 0x05,`  
`TSL2561_REGISTER_INTERRUPT = 0x06, TSL2561_REGISTER_CRC = 0x08,`  
`TSL2561_REGISTER_ID = 0x0A, TSL2561_REGISTER_CHAN0_LOW = 0x0C, TSL2561_REGISTER_CHA←`  
`N0_HIGH = 0x0D, TSL2561_REGISTER_CHAN1_LOW = 0x0E,`  
`TSL2561_REGISTER_CHAN1_HIGH = 0x0F }`  
*TSL2561 I2C Registers.*
- `enum tsl2561IntegrationTime_t { TSL2561_INTEGRATIONTIME_13MS = 0x00, TSL2561_INTEGRATIONTIM←`  
`E_101MS = 0x01, TSL2561_INTEGRATIONTIME_402MS = 0x02 }`  
*Three options for how long to integrate readings for.*
- `enum tsl2561Gain_t { TSL2561_GAIN_1X = 0x00, TSL2561_GAIN_16X = 0x10 }`  
*TSL2561 offers 2 gain settings.*

## Functions

- int `tsl2561_init` (`tsl2561` \*dev, uint8\_t s\_address)  
*Init function for the TSL2561 device. Default: I2C\_BUS TODO: Fix init + gain, figure out what goes wrong if ID register is read.*
- void `tsl2561_measure` (`tsl2561` \*dev, uint32\_t \*measure)  
*Read I2C data into the uint32\_t measure var. Format: (MSB) broadband | ir (LSB)*
- uint32\_t `tsl2561_get_lux` (uint32\_t measure)  
*Calculate lux using value measured using `tsl2561_measure()`*
- void `tsl2561_destroy` (`tsl2561` \*dev)  
*Destroy function for the TSL2561 device. Closes the file descriptor and powers down the device.*
- static void `write8` (int fd, uint8\_t val)  
*write 8 bytes to the device represented by the file descriptor.*
- static void `writecmd8` (int fd, uint8\_t reg, uint8\_t val)  
*Write a command to the register on the device represented by fd.*
- static uint8\_t `read8` (int fd, uint8\_t reg)  
*Read a byte from the specified register on the device represented by fd.*
- static void `write16` (int fd, uint16\_t val)  
*Write 16 bits to the device (very similar to `writecmd8()`)*
- static uint16\_t `read16` (int fd, uint8\_t cmd)  
*Read 2 bytes in LE format from reg on the device represented by fd.*

### 5.10.1 Detailed Description

TSL2561 I2C driver function and struct declarations.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.10.2 Enumeration Type Documentation

#### 5.10.2.1 anonymous enum

anonymous enum

TSL2561 I2C Registers.

**Enumerator**

|                                  |                                     |
|----------------------------------|-------------------------------------|
| TSL2561_REGISTER_CONTROL         | Control/power register.             |
| TSL2561_REGISTER_TIMING          | Set integration time register.      |
| TSL2561_REGISTER_THRESHOLD_LOW   | Interrupt low threshold low-byte.   |
| TSL2561_REGISTER_THRESHOLD_HIGH  | Interrupt low threshold high-byte.  |
| TSL2561_REGISTER_THRESHOLDH_LOW  | Interrupt high threshold low-byte.  |
| TSL2561_REGISTER_THRESHOLDH_HIGH | Interrupt high threshold high-byte. |
| TSL2561_REGISTER_INTERRUPT       | Interrupt settings.                 |
| TSL2561_REGISTER_CRC             | Factory use only.                   |
| TSL2561_REGISTER_ID              | TSL2561 identification setting.     |
| TSL2561_REGISTER_CHAN0_LOW       | Light data channel 0, low byte.     |
| TSL2561_REGISTER_CHAN0_HIGH      | Light data channel 0, high byte.    |
| TSL2561_REGISTER_CHAN1_LOW       | Light data channel 1, low byte.     |
| TSL2561_REGISTER_CHAN1_HIGH      | Light data channel 1, high byte.    |

**5.10.2.2 `tsl2561Gain_t`**

```
enum tsl2561Gain_t
```

TSL2561 offers 2 gain settings.

**Enumerator**

|                  |          |
|------------------|----------|
| TSL2561_GAIN_1X  | No gain. |
| TSL2561_GAIN_16X | 16x gain |

**5.10.2.3 `tsl2561IntegrationTime_t`**

```
enum tsl2561IntegrationTime_t
```

Three options for how long to integrate readings for.

**Enumerator**

|                               |        |
|-------------------------------|--------|
| TSL2561_INTEGRATIONTIME_13MS  | 13.7ms |
| TSL2561_INTEGRATIONTIME_101MS | 101ms  |
| TSL2561_INTEGRATIONTIME_402MS | 402ms  |

### 5.10.3 Function Documentation

#### 5.10.3.1 read16()

```
static uint16_t read16 (
 int fd,
 uint8_t cmd) [inline], [static]
```

Read 2 bytes in LE format from reg on the device represented by fd.

##### Parameters

|            |  |
|------------|--|
| <i>fd</i>  |  |
| <i>cmd</i> |  |

##### Returns

uint16\_t

#### 5.10.3.2 read8()

```
static uint8_t read8 (
 int fd,
 uint8_t reg) [inline], [static]
```

Read a byte from the specified register on the device represented by fd.

##### Parameters

|            |                  |
|------------|------------------|
| <i>fd</i>  |                  |
| <i>reg</i> | Register address |

##### Returns

Byte read over serial

#### 5.10.3.3 tsl2561\_destroy()

```
void tsl2561_destroy (
 tsl2561 * dev)
```

Destroy function for the TSL2561 device. Closes the file descriptor and powers down the device.

#### Parameters

|            |  |
|------------|--|
| <i>dev</i> |  |
|------------|--|

#### 5.10.3.4 `tsl2561_get_lux()`

```
uint32_t tsl2561_get_lux (
 uint32_t measure)
```

Calculate lux using value measured using [tsl2561\\_measure\(\)](#)

#### Parameters

|                |  |
|----------------|--|
| <i>measure</i> |  |
|----------------|--|

#### Returns

Lux value

#### 5.10.3.5 `tsl2561_init()`

```
int tsl2561_init (
 tsl2561 * dev,
 uint8_t s_address)
```

Init function for the TSL2561 device. Default: I2C\_BUS TODO: Fix init + gain, figure out what goes wrong if ID register is read.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <i>dev</i>       |                                                  |
| <i>s_address</i> | Address for the device, values: 0x29, 0x39, 0x49 |

#### Returns

1 on success, -1 on failure

### 5.10.3.6 tsl2561\_measure()

```
void tsl2561_measure (
 tsl2561 * dev,
 uint32_t * measure)
```

Read I2C data into the uint32\_t measure var.\ Format: (MSB) broadband | ir (LSB)

#### Parameters

|                |                                                                |
|----------------|----------------------------------------------------------------|
| <i>dev</i>     |                                                                |
| <i>measure</i> | Pointer to unsigned 32 bit integer where measurement is stored |

### 5.10.3.7 write16()

```
static void write16 (
 int fd,
 uint16_t val) [inline], [static]
```

Write 16 bits to the device (very similar to [writecmd8\(\)](#))

#### Parameters

|            |  |
|------------|--|
| <i>fd</i>  |  |
| <i>val</i> |  |

### 5.10.3.8 write8()

```
static void write8 (
 int fd,
 uint8_t val) [inline], [static]
```

write 8 bytes to the device represented by the file descriptor.

#### Parameters

|            |  |
|------------|--|
| <i>fd</i>  |  |
| <i>val</i> |  |

### 5.10.3.9 writecmd8()

```
static void writecmd8 (
 int fd,
 uint8_t reg,
 uint8_t val) [inline], [static]
```

Write a command to the register on the device represented by fd.

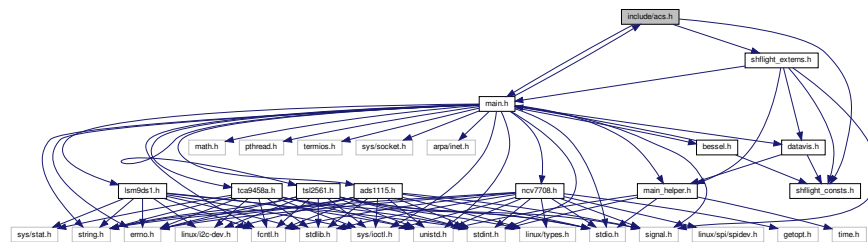
#### Parameters

|            |                                    |
|------------|------------------------------------|
| <i>fd</i>  | File descriptor                    |
| <i>reg</i> | Register address                   |
| <i>val</i> | Value to write at register address |

## 5.11 include/acs.h File Reference

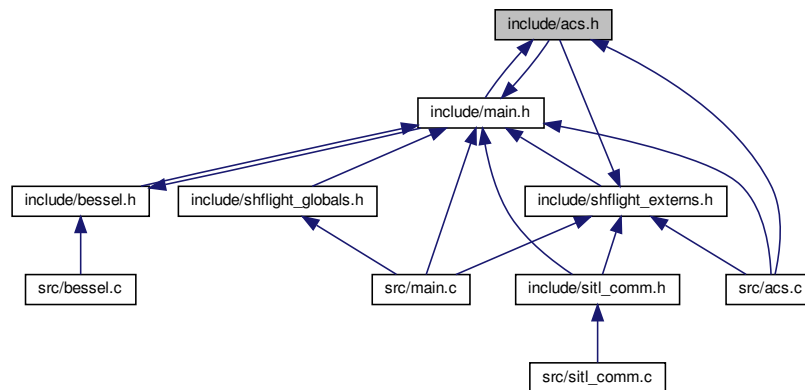
Header file including headers and function prototypes of the Attitude Control System.

```
#include <main.h>
#include <shflight_consts.h>
#include <shflight_extens.h>
Include dependency graph for acs.h:
```





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



## Macros

- `#define HBRIDGE_ENABLE(name) hbridge_enable(x_##name, y_##name, z_##name);`  
Fire magnetorquer in the direction dictated by the input vector.

## Functions

- `int acs_init (void)`  
Initializes the devices required to run the attitude control system.
- `void * acs_thread (void *)`  
Attitude Control System Thread.
- `void acs_destroy (void)`  
Powers down ACS devices and closes relevant file descriptors.
- `void insertionSort (int a1[], int a2[])`  
Sorts the first array and reorders the second array according to the first array.
- `int hbridge_enable (int x, int y, int z)`  
Fire magnetorquer in X, Y, and Z directions using the input integers.
- `int HBRIDGE_DISABLE (int num)`  
Disables magnetorquer in the axis indicated by the input.
- `void getOmega (void)`  
Calculates  $\vec{\omega}$  using  $\vec{B}$  and stores in the circular buffer.
- `void getSVec (void)`  
Calculates sun vector using coarse sun sensor and fine sun sensor measurements. Favors the fine sun sensor measurements if exists. The value is inserted into a circular buffer.
- `int readSensors (void)`  
Reads hardware sensors and puts the values in the global storage, upon which calls the `getOmega()` and `getSVec()` functions to calculate angular speed and sun vector.
- `void checkTransition (void)`  
This function checks if the ACS should transition from one state to the other at every iteration. The function executes only when the  $\vec{\omega}$  and sun vector buffers are full.

### 5.11.1 Detailed Description

Header file including headers and function prototypes of the Attitude Control System.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.11.2 Macro Definition Documentation

#### 5.11.2.1 HBRIDGE\_ENABLE

```
#define HBRIDGE_ENABLE(
 name) hbridge_enable(x_##name, y_##name, z_##name);
```

Fire magnetorquer in the direction dictated by the input vector.

#### Parameters

|             |                          |
|-------------|--------------------------|
| <i>name</i> | Name of the input vector |
|-------------|--------------------------|

### 5.11.3 Function Documentation

#### 5.11.3.1 acs\_init()

```
int acs_init (
 void)
```

Initializes the devices required to run the attitude control system.

This function initializes the target angular momentum using MOI defined in [shflight\\_globals.h](#) and the target angular speed set in [main.c](#). Then this function initializes all the relevant devices for ACS to function.

#### Returns

int 1 on success, error codes defined in SH\_ERRORS on error.

#### 5.11.3.2 acs\_thread()

```
void* acs_thread (
 void *)
```

Attitude Control System Thread.

This thread executes the ACS functions in a loop controlled by the variable `done`, which is controlled by the interrupt handler.

#### Parameters

|           |                                              |
|-----------|----------------------------------------------|
| <i>id</i> | Thread ID passed as a pointer to an integer. |
|-----------|----------------------------------------------|

#### Returns

NULL

#### 5.11.3.3 getOmega()

```
void getOmega (
 void)
```

Calculates  $\omega$  using  $\dot{\vec{B}}$  and stores in the circular buffer.

Calculates current angular speed. Requires current and previous measurements of  $\dot{\vec{B}}$ . The calculated angular speed is put inside the global circular buffer. Sets `W_full` to indicate the buffer becoming full the first time.

#### 5.11.3.4 HBRIDGE\_DISABLE()

```
int HBRIDGE_DISABLE (
 int num)
```

Disables magnetorquer in the axis indicated by the input.

**Parameters**

|            |                                                                                                                                        |
|------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <i>num</i> | Integer, 0 indicates X axis, 1 indicates Y axis, 2 indicates Z axis. In hardware, a number > 2 causes all three torquers to shut down. |
|------------|----------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

int Status of the operation, returns 1 on success.

**5.11.3.5 hbridge\_enable()**

```
int hbridge_enable (
 int x,
 int y,
 int z)
```

Fire magnetorquer in X, Y, and Z directions using the input integers.

**Parameters**

|          |                                                                                                  |
|----------|--------------------------------------------------------------------------------------------------|
| <i>x</i> | Fires in the +X or -X direction depending on the input being +1 or -1, and does nothing if x = 0 |
| <i>y</i> | Fires in the +Y or -Y direction depending on the input being +1 or -1, and does nothing if y = 0 |
| <i>z</i> | Fires in the +Z or -Z direction depending on the input being +1 or -1, and does nothing if z = 0 |

**Returns**

int Status of the operation, returns 1 on success.

**5.11.3.6 insertionSort()**

```
void insertionSort (
 int a1[],
 int a2[])
```

Sorts the first array and reorders the second array according to the first array.

**Parameters**

|           |                                      |
|-----------|--------------------------------------|
| <i>a1</i> | Pointer to integer array to sort.    |
| <i>a2</i> | Pointer to integer array to reorder. |

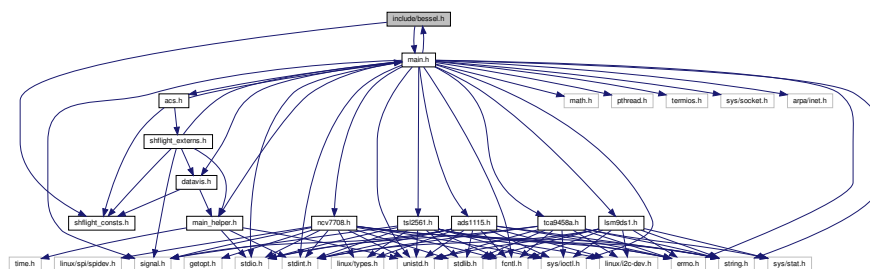
```
int readSensors (
 void)
```

## Returns

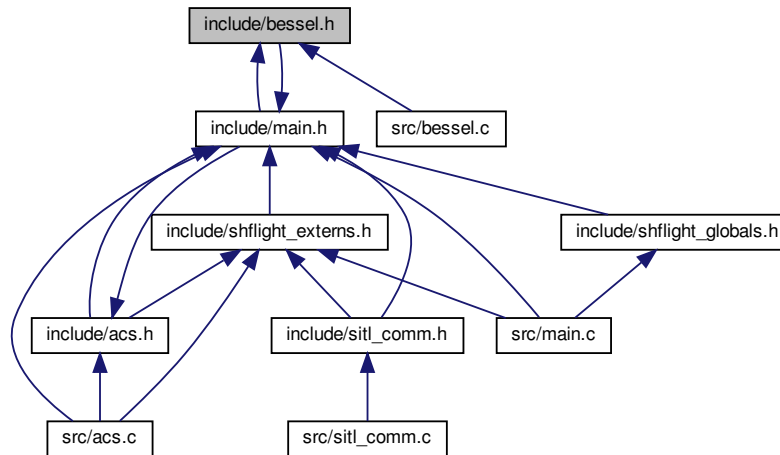
int Returns 1 for success, and -1 for error.

### Bessel filter implementation for Attitude Control System.

```
#include <main.h>
#include <shflight_consts.h>
Include dependency graph for bessel.h:
```



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



## Macros

- `#define BESSEL_MIN_THRESHOLD 0.001`  
*Minimum threshold for Bessel coefficient for filter calculations.*
- `#define BESSEL_FREQ_CUTOFF 5`  
*Cut off frequency for the Bessel filter. cutoff frequency 5 == 5\*DETUMBLE\_TIME\_STEP seconds cycle == 2 Hz at 100ms loop speed.*
- `#define APPLY_DBESSEL(name, index)`  
*Applies double precision Bessel filter on a buffer declared using [DECLARE\\_BUFFER\(\)](#), and stores the filtered value at the current index.*
- `#define APPLY_FBESSEL(name, index)`  
*Applies floating point Bessel filter on a buffer declared using [DECLARE\\_BUFFER\(\)](#), and stores the filtered value at the current index.*

## Functions

- void [calculateBessel](#) (float arr[], int size, int order, float freq\_cutoff)  
*Calculates discrete Bessel filter coefficients for the given order and cutoff frequency.*
- double [dfilterBessel](#) (double arr[], int index)  
*Returns the filtered value at the current index using past values.*
- float [ffilterBessel](#) (float arr[], int index)  
*Returns the filtered value at the current index using past values.*

## Variables

- float [bessel\\_coeff](#) [SH\_BUFFER\_SIZE]  
*Coefficients for the Bessel filter, calculated using [calculateBessel\(\)](#).*

### 5.12.1 Detailed Description

Bessel filter implementation for Attitude Control System.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.12.2 Macro Definition Documentation

#### 5.12.2.1 APPLY\_DBESSEL

```
#define APPLY_DBESSEL(
 name,
 index)
```

#### Value:

```
x_##name[index] = dfilterBessel(x_##name, index); \
y_##name[index] = dfilterBessel(y_##name, index); \
z_##name[index] = dfilterBessel(z_##name, index)
```

Applies double precision Bessel filter on a buffer declared using [DECLARE\\_BUFFER\(\)](#), and stores the filtered value at the current index.

#### Parameters

|              |                                          |
|--------------|------------------------------------------|
| <i>name</i>  | Name of the buffer                       |
| <i>index</i> | Index of the current value in the buffer |

### 5.12.2.2 APPLY\_FBESSEL

```
#define APPLY_FBESSEL(
 name,
 index)
```

#### Value:

```
x_##name[index] = ffilterBessel(x_##name, index); \
y_##name[index] = ffilterBessel(y_##name, index); \
z_##name[index] = ffilterBessel(z_##name, index)
```

Applies floating point Bessel filter on a buffer declared using [DECLARE\\_BUFFER\(\)](#), and stores the filtered value at the current index.

#### Parameters

|              |                                          |
|--------------|------------------------------------------|
| <i>name</i>  | Name of the buffer                       |
| <i>index</i> | Index of the current value in the buffer |

## 5.12.3 Function Documentation

### 5.12.3.1 calculateBessel()

```
void calculateBessel (
 float arr[],
 int size,
 int order,
 float freq_cutoff)
```

Calculates discrete Bessel filter coefficients for the given order and cutoff frequency.

#### Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <i>arr</i>         | Stores the filter coefficients         |
| <i>size</i>        | Size of the filter coefficients array  |
| <i>order</i>       | Order of the Bessel filter             |
| <i>freq_cutoff</i> | Cut-off frequency of the Bessel filter |

### 5.12.3.2 dfilterBessel()

```
double dfilterBessel (
```



```
double arr[],
int index)
```

Returns the filtered value at the current index using past values.

#### Parameters

|              |                                     |
|--------------|-------------------------------------|
| <i>arr</i>   | Input array                         |
| <i>index</i> | Index of current value in the array |

#### Returns

double Filtered value

#### 5.12.3.3 ffilterBessel()

```
float ffilterBessel (
 float arr[],
 int index)
```

Returns the filtered value at the current index using past values.

#### Parameters

|              |                                     |
|--------------|-------------------------------------|
| <i>arr</i>   | Input array                         |
| <i>index</i> | Index of current value in the array |

#### Returns

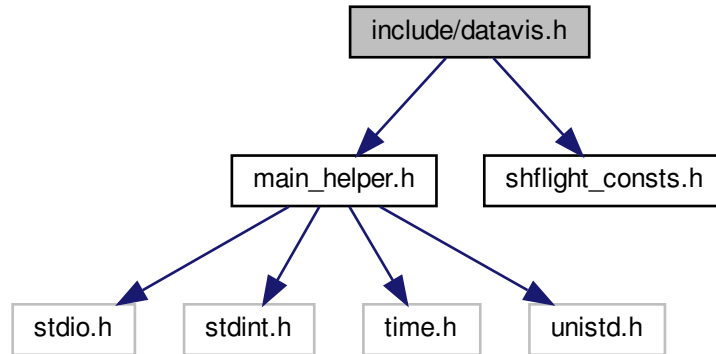
double Filtered value

## 5.13 include/datavis.h File Reference

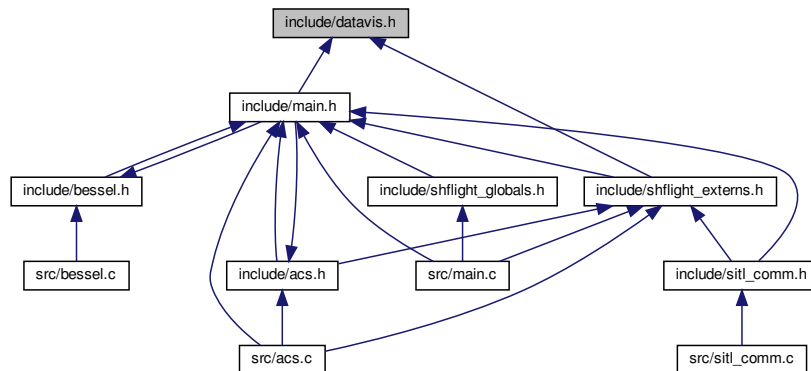
DataVis thread to visualize ACS data over TCP (uses client.py)

```
#include <main_helper.h>
#include <shflight_consts.h>
```

Include dependency graph for `datavis.h`:



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



## Classes

- struct [datavis\\_p](#)  
*DataVis structure for storing current ACS data.*
- union [data\\_packet](#)  
*Union of the [datavis\\_p](#) structure and an array of bytes for transport over TCP using `send()`.*

## Macros

- `#define` **PORT** 12376
- `#define` **PACK\_SIZE** `sizeof(datavis_p)`  
*Size of the [datavis\\_p](#) struct.*

## Typedefs

- typedef struct sockaddr **sk\_sockaddr**

## Functions

- void \* [datavis\\_thread](#) (void \*t)

*DataVis thread, sends data in g\_datavis\_st over TCP. This thread loops over done, and at each wakeup from the ACS thread sends the currently available data over TCP to the listening connection.*

### 5.13.1 Detailed Description

DataVis thread to visualize ACS data over TCP (uses client.py)

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.13.2 Function Documentation

#### 5.13.2.1 datavis\_thread()

```
void* datavis_thread (
 void * t)
```

DataVis thread, sends data in g\_datavis\_st over TCP. This thread loops over done, and at each wakeup from the ACS thread sends the currently available data over TCP to the listening connection.

**Parameters**

|          |                                                 |
|----------|-------------------------------------------------|
| <i>t</i> | Pointer to an integer containing the thread ID. |
|----------|-------------------------------------------------|

**Returns**

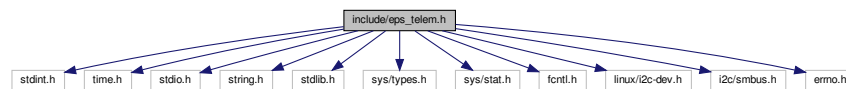
NULL.

## 5.14 include/eps\_telem.h File Reference

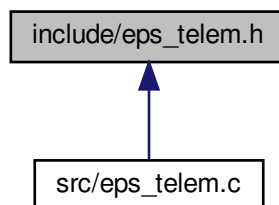
GomSpace P31u I2C interface function prototypes and data structures.

```
#include <stdint.h>
#include <time.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <linux/i2c-dev.h>
#include <i2c/smbus.h>
#include <errno.h>
```

Include dependency graph for eps\_telem.h:



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



## Classes

- struct [hkparam\\_t](#)
- struct [eps\\_hk\\_t](#)
- struct [eps\\_hk\\_vi\\_t](#)
- struct [eps\\_hk\\_out\\_t](#)
- struct [eps\\_hk\\_wdt\\_t](#)
- struct [eps\\_hk\\_basic\\_t](#)
- struct [eps\\_config\\_t](#)
- struct [eps\\_config2\\_t](#)
- struct [eps\\_config3\\_t](#)
- union [channel\\_t](#)
- struct [p31u](#)

## Macros

- #define **EPS\_I2C\_ADDR** 0x7d
- #define **EPS\_I2C\_BUS** "/dev/i2c-0"

## Enumerations

- enum **eps\_xfer\_ret\_t** { **EPS\_I2C\_READ\_FAILED** = -20, **EPS\_I2C\_WRITE\_FAILED**, **EPS\_COMMAND\_FAILED**, **EPS\_COMMAND\_SUCCESS** = 1 }
- enum **eps\_commands** { **PING** = 1, **REBOOT** = 4, **GET\_HK** = 8, **SET\_OUTPUT**, **SET\_SINGLE\_OUTPUT**, **SET\_PV\_VOLT**, **SET\_PV\_AUTO**, **SET\_HEATER**, **RESET\_COUNTERS** = 15, **RESET\_WDT**, **CONFIG\_CMD**, **CONFIG\_GET**, **CONFIG\_SET**, **HARD\_RESET**, **CONFIG2\_CMD**, **CONFIG2\_GET**, **CONFIG2\_SET**, **CONFIG3** = 25 }

## Functions

- void \* **eps\_telem** (void \*id)
- int **p31u\_init** ([p31u](#) \*)
- void **p31u\_destroy** ([p31u](#) \*)
- int **p31u\_xfer** ([p31u](#) \*, char \*, ssize\_t, char \*, ssize\_t)
- int **eps\_ping** ([p31u](#) \*)
- int **eps\_reboot** ([p31u](#) \*)
- int **eps\_get\_hk** ([p31u](#) \*, uint8\_t)
- int **eps\_hk** ([p31u](#) \*)
- int **eps\_set\_output** ([p31u](#) \*, [channel\\_t](#))
- int **eps\_set\_single** ([p31u](#) \*, uint8\_t, uint8\_t, int16\_t)
- int **eps\_set\_pv\_volt** ([p31u](#) \*, uint16\_t, uint16\_t, uint16\_t)
- int **eps\_set\_pv\_mode** ([p31u](#) \*, uint8\_t)
- int **eps\_set\_heater** ([p31u](#) \*, uint8\_t cmd, uint8\_t heater, uint8\_t mode, uint16\_t \*output)
- int **eps\_reset\_counters** ([p31u](#) \*)
- int **eps\_reset\_wdt** ([p31u](#) \*)
- int **eps\_config\_cmd** ([p31u](#) \*, uint8\_t)

- int **eps\_config\_get** (p31u \*)
- int **eps\_config\_set** (p31u \*, eps\_config\_t)
- int **eps\_hard\_reset** (p31u \*)
- int **eps\_config2\_cmd** (p31u \*, uint8\_t)
- int **eps\_config2\_get** (p31u \*)
- int **eps\_config2\_set** (p31u \*, eps\_config2\_t)
- int **eps\_config3** (p31u \*, eps\_config3\_t)

## Variables

- p31u \* **g\_eps**

### 5.14.1 Detailed Description

GomSpace P31u I2C interface function prototypes and data structures.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

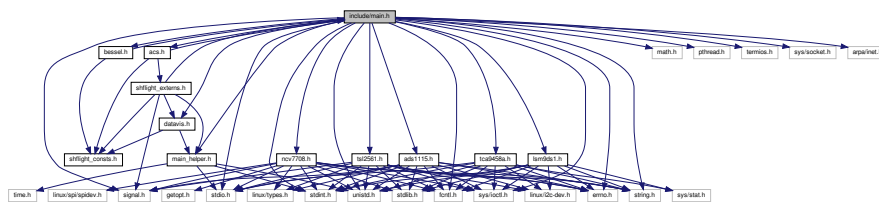
## 5.15 include/main.h File Reference

Includes all headers necessary for the core flight software, including ACS, and defines ACS states (which are flight software states), error codes, and relevant error functions.

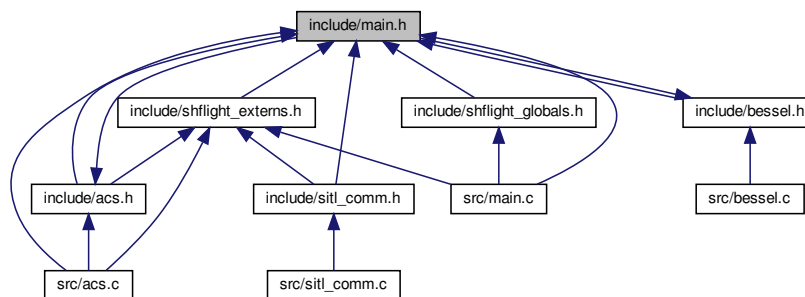
```
#include <stdio.h>
#include <stdint.h>
#include <math.h>
#include <pthread.h>
#include <signal.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
```

```
#include <main_helper.h>
#include <string.h>
#include <termios.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <bessel.h>
#include "ncv7708.h"
#include "lsm9ds1.h"
#include "tsl2561.h"
#include "ads1115.h"
#include "tca9458a.h"
#include <acs.h>
#include <datavis.h>
```

Include dependency graph for main.h:



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



## Macros

- #define **DATAVIS**
- #define **NUM\_SYSTEMS** 2

## Enumerations

- enum `SH_ACS_MODES` {  
`STATE_ACS_DETUMBLE`, `STATE_ACS_SUNPOINT`, `STATE_ACS_NIGHT`, `STATE_ACS_READY`,  
`STATE_XBAND_READY` }

*Describes ACS (system) states.*

- enum [SH\\_ERRORS](#) {  
**ERROR\_MALLOC** = -1, **ERROR\_HBRIDGE\_INIT** = -2, **ERROR\_MUX\_INIT** = -3, **ERROR\_CSS\_INIT** = -4,  
**ERROR\_MAG\_INIT** = -5, **ERROR\_FSS\_INIT** = -6, **ERROR\_FSS\_CONFIG** = -7 }

*Describes possible system errors.*

## Functions

- void [catch\\_sigint](#) (int)  
*SIGINT handler, sets the global variable `done` as 1, so that thread loops can break. Wakes up `sitl_comm` and `datavis` threads to ensure they exit.*
- void [sherror](#) (const char \*)  
*Prints errors specific to shflight in a fashion similar to perror.*

### 5.15.1 Detailed Description

Includes all headers necessary for the core flight software, including ACS, and defines ACS states (which are flight software states), error codes, and relevant error functions.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.15.2 Function Documentation

#### 5.15.2.1 [catch\\_sigint\(\)](#)

```
void catch_sigint (
 int sig)
```

SIGINT handler, sets the global variable `done` as 1, so that thread loops can break. Wakes up `sitl_comm` and `datavis` threads to ensure they exit.



## Parameters

|            |                               |
|------------|-------------------------------|
| <i>sig</i> | Receives the signal as input. |
|------------|-------------------------------|

## 5.15.2.2 sherror()

```
void sherror (
 const char * msg)
```

Prints errors specific to shflight in a fashion similar to perror.

## Parameters

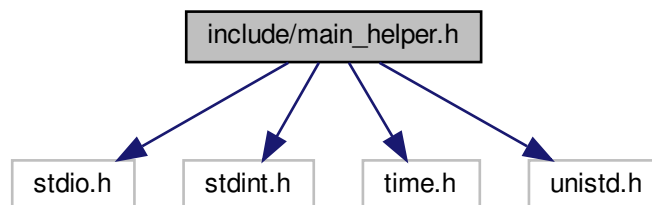
|            |                                                     |
|------------|-----------------------------------------------------|
| <i>msg</i> | Input message to print along with error description |
|------------|-----------------------------------------------------|

## 5.16 include/main\_helper.h File Reference

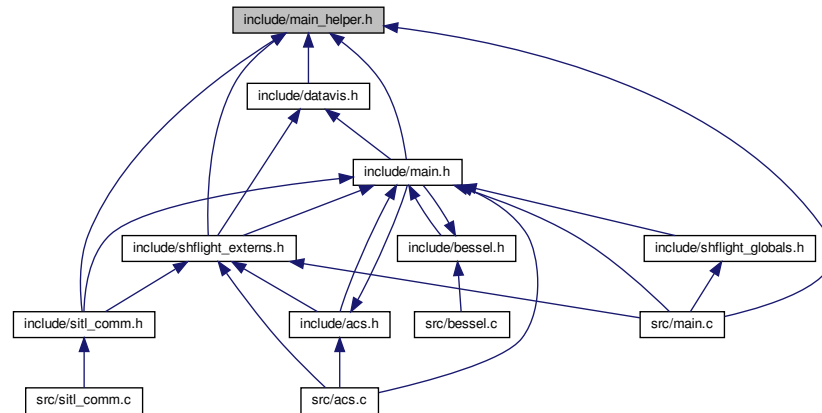
Defines vector macros and other helper functions for the flight software.

```
#include <stdio.h>
#include <stdint.h>
#include <time.h>
#include <unistd.h>
```

Include dependency graph for main\_helper.h:



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



## Macros

- `#define BOOTCOUNT_FNAME "bootcount_fname.txt"`  
Name of the file where bootcount is stored on the file system.
- `#define DECLARE_BUFFER(name, type) type x_##name[SH_BUFFER_SIZE], y_##name[SH_BUFFER_SIZE], z_##name[SH_BUFFER_SIZE]`  
Declares a buffer with name and type. Prepends `x_`, `y_`, `z_` to the names (vector buffer!) This macro allocates three arrays `x_name`, `y_name` and `z_name` of type and size `SH_BUFFER_SIZE`.
- `#define VECTOR_CLEAR(name)`  
Clears a vector.
- `#define DECLARE_VECTOR(name, type) type x_##name = 0, y_##name = 0, z_##name = 0`  
Declares a vector with the name and type. A vector is a three-variable entity with `x_`, `y_`, `z_` prepended to the names. This function initializes the variables to 0, which makes it not ideal for use in extern definitions.
- `#define DECLARE_VECTOR2(name, type) type x_##name, y_##name, z_##name`  
Declares a vector with the name and type. A vector is a three-variable entity with `x_`, `y_`, `z_` prepended to the names. This function does not initialize the variables to 0, which makes it ideal for use in extern definitions.
- `#define FLUSH_BUFFER(name)`  
Flushes a buffer declared using `DECLARE_BUFFER()`. Does not reset index counters or buffer full indicators, which needs to be done by hand on a case by case basis.
- `#define FLUSH_BUFFER_ALL`  
Resets all buffers and resets indices, while not clearing buffer full indicators.
- `#define CROSS_PRODUCT(dest, s1, s2)`  
Calculates cross product of two vectors created using `DECLARE_VECTOR()`. The destination vector must be a different vector from any of the inputs.
- `#define DOT_PRODUCT(s1, s2) (float)(x_##s1 * x_##s2 + y_##s1 * y_##s2 + z_##s1 * z_##s2)`  
Calculates the floating point (32-bit) dot product of two vectors.
- `#define VECTOR_OP(dest, s1, s2, op)`  
Performs a vector operation on the source vectors and stores in destination vector. Since the operations are performed element-by-element, the destination vector can be the same as any of the source vectors.
- `#define VECTOR_MIXED(dest, s1, s2, op)`

*Performs element-by-element operation on a vector with a scalar and stores in the destination vector. Since the operations are performed element-by-element, the scalar can not depend on the source vector.*

- #define **NORMALIZE**(dest, s1)  
*Normalizes the input vector and stores it in the output vector. Works for null vectors as well.*
- #define **NORM**(s) sqrt(**NORM2**(s))  
*Calculates the norm of the input vector in 32-bit floating point.*
- #define **NORM2**(s) x\_##s \* x\_##s + y\_##s \* y\_##s + z\_##s \* z\_##s  
*Calculates the square of the norm of the input vector in 32-bit floating point.*
- #define **INVNORM**(s) q2isqrt(**NORM2**(s))  
*Calculates the inverse norm of the input vector in 32-bit floating point. Does not check for null vectors.*
- #define **MATVECMUL**(dest, s1, s2)  
*Multiplies the input vector by the input matrix (3x3) (left to right).*
- #define **FAVERAGE\_BUFFER**(dest, src, size)  
*Calculates 32-bit float average of an input buffer.*
- #define **DAVERAGE\_BUFFER**(dest, src, size)  
*Calculates double precision average of an input buffer.*

## Functions

- int **bootCount** (void)  
*Function that returns the current bootcount of the system. Returns current boot count, and increases by 1 and stores it in nvmmem. Expected to be invoked only by \_main()*
- float **q2isqrt** (float x)  
*float q2isqrt(float): Returns the inverse square root of a floating point number. Depending on whether MATH\_SQRT is declared, it will use sqrt() function from gcc-math or bit-level hack and 3 rounds of Newton-Raphson to directly calculate inverse square root. The bit-level routine yields consistently better performance and 0.00001% maximum error. Set MATH\_SQRT at compile time to use the sqrt() function.*
- uint64\_t **get\_usec** (void)  
*Returns time elapsed from 1970-1-1, 00:00:00 UTC to now (UTC) in microseconds. Execution time ~18 us on RPi.*
- float **faverage** (float arr[], int size)  
*Calculates floating point average of a float array.*
- double **daverage** (double arr[], int size)  
*Calculates double precision point average of a float array.*

### 5.16.1 Detailed Description

Defines vector macros and other helper functions for the flight software.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

## 5.16.2 Macro Definition Documentation

### 5.16.2.1 CROSS\_PRODUCT

```
#define CROSS_PRODUCT(
 dest,
 s1,
 s2)
```

#### Value:

```
x_##dest = y_##s1 * z_##s2 - z_##s1 * y_##s2; \
y_##dest = z_##s1 * x_##s2 - x_##s1 * z_##s2; \
z_##dest = x_##s1 * y_##s2 - y_##s1 * x_##s2
```

Calculates cross product of two vectors created using [DECLARE\\_VECTOR\(\)](#). The destination vector must be a different vector from any of the inputs.

#### Parameters

|             |                                                                            |
|-------------|----------------------------------------------------------------------------|
| <i>dest</i> | Destination vector name, declared using <a href="#">DECLARE_VECTOR()</a>   |
| <i>s1</i>   | First source vector name, declared using <a href="#">DECLARE_VECTOR()</a>  |
| <i>s2</i>   | Second source vector name, declared using <a href="#">DECLARE_VECTOR()</a> |

### 5.16.2.2 DAVERAGE\_BUFFER

```
#define DAVERAGE_BUFFER(
 dest,
 src,
 size)
```

#### Value:

```
x_##dest = daverage(x_##src, size); \
y_##dest = daverage(y_##src, size); \
z_##dest = daverage(z_##src, size)
```

Calculates double precision average of an input buffer.

#### Parameters

|             |                                                                                                                   |
|-------------|-------------------------------------------------------------------------------------------------------------------|
| <i>dest</i> | Output vector, declared using <a href="#">DECLARE_VECTOR()</a>                                                    |
| <i>src</i>  | Input buffer, declared using <a href="#">DECLARE_BUFFER()</a>                                                     |
| <i>size</i> | Size of the input buffer (equals to SH_BUFFER_SIZE for a buffer declared using <a href="#">DECLARE_BUFFER()</a> ) |

### 5.16.2.3 DECLARE\_BUFFER

```
#define DECLARE_BUFFER(
 name,
 type) type x_##name[SH_BUFFER_SIZE], y_##name[SH_BUFFER_SIZE], z_##name[SH_BUFFER←
_SIZE]
```

Declares a buffer with name and type. Prepends x\_, y\_, z\_ to the names (vector buffer!) This macro allocates three arrays x\_name, y\_name and z\_name of type and size SH\_BUFFER\_SIZE.

#### Parameters

|             |                                                     |
|-------------|-----------------------------------------------------|
| <i>name</i> | Name of the buffer (prepends x_, y_, z_ for vector) |
| <i>type</i> | Data type of the buffer                             |

### 5.16.2.4 DECLARE\_VECTOR

```
#define DECLARE_VECTOR(
 name,
 type) type x_##name = 0, y_##name = 0, z_##name = 0
```

Declares a vector with the name and type. A vector is a three-variable entity with x\_, y\_, z\_ prepended to the names. This function initializes the variables to 0, which makes it not ideal for use in extern definitions.

#### Parameters

|             |                         |
|-------------|-------------------------|
| <i>name</i> | Name of the vector      |
| <i>type</i> | Data type of the vector |

### 5.16.2.5 DECLARE\_VECTOR2

```
#define DECLARE_VECTOR2(
 name,
 type) type x_##name, y_##name, z_##name
```

Declares a vector with the name and type. A vector is a three-variable entity with x\_, y\_, z\_ prepended to the names. This function does not initialize the variables to 0, which makes it ideal for use in extern definitions.

**Parameters**

|             |                         |
|-------------|-------------------------|
| <i>name</i> | Name of the vector      |
| <i>type</i> | Data type of the vector |

**5.16.2.6 DOT\_PRODUCT**

```
#define DOT_PRODUCT(
 s1,
 s2) (float) (x_##s1 * x_##s2 + y_##s1 * y_##s2 + z_##s1 * z_##s2)
```

Calculates the floating point (32-bit) dot product of two vectors.

**Parameters**

|           |                                                                            |
|-----------|----------------------------------------------------------------------------|
| <i>s1</i> | Name of the first vector, declared using <a href="#">DECLARE_VECTOR()</a>  |
| <i>s2</i> | Name of the second vector, declared using <a href="#">DECLARE_VECTOR()</a> |

**5.16.2.7 FAVERAGE\_BUFFER**

```
#define FAVERAGE_BUFFER(
 dest,
 src,
 size)
```

**Value:**

```
x_##dest = faverage(x_##src, size); \
y_##dest = faverage(y_##src, size); \
z_##dest = faverage(z_##src, size)
```

Calculates 32-bit float average of an input buffer.

**Parameters**

|             |                                                                                                                   |
|-------------|-------------------------------------------------------------------------------------------------------------------|
| <i>dest</i> | Output vector, declared using <a href="#">DECLARE_VECTOR()</a>                                                    |
| <i>src</i>  | Input buffer, declared using <a href="#">DECLARE_BUFFER()</a>                                                     |
| <i>size</i> | Size of the input buffer (equals to SH_BUFFER_SIZE for a buffer declared using <a href="#">DECLARE_BUFFER()</a> ) |

## 5.16.2.8 FLUSH\_BUFFER

```
#define FLUSH_BUFFER(
 name)
```

**Value:**

```
for (uint8_t sh_counter = SH_BUFFER_SIZE; sh_counter > 0;) \
{
 sh_counter--;
 x_##name[sh_counter] = 0;
 y_##name[sh_counter] = 0;
 z_##name[sh_counter] = 0;
}
```

Flushes a buffer declared using [DECLARE\\_BUFFER\(\)](#). Does not reset index counters or buffer full indicators, which needs to be done by hand on a case by case basis.

## 5.16.2.9 FLUSH\_BUFFER\_ALL

```
#define FLUSH_BUFFER_ALL
```

**Value:**

```
FLUSH_BUFFER(g_B); \
FLUSH_BUFFER(g_Bt); \
FLUSH_BUFFER(g_W); \
FLUSH_BUFFER(g_S); \
mag_index = -1; \
sol_index = -1; \
bdot_index = -1; \
omega_index = -1; \
q_nightmode = 0; \
omega_ready = -1;
```

Resets all buffers and resets indices, while not clearing buffer full indicators.

## 5.16.2.10 INVNORM

```
#define INVNORM(
 s) q2isqrt(NORM2(s))
```

Calculates the inverse norm of the input vector in 32-bit floating point. Does not check for null vectors.

**Parameters**

|          |                                                               |
|----------|---------------------------------------------------------------|
| <b>s</b> | Input vector, declared using <a href="#">DECLARE_VECTOR()</a> |
|----------|---------------------------------------------------------------|

**Returns**

float Inverse norm of the input vector

**5.16.2.11 MATVECMUL**

```
#define MATVECMUL(
 dest,
 s1,
 s2)
```

**Value:**

```
x_##dest = s1[0][0] * x_##s2 + s1[0][1] * y_##s2 + s1[0][2] * z_##s2; \
y_##dest = s1[1][0] * x_##s2 + s1[1][1] * y_##s2 + s1[1][2] * z_##s2; \
z_##dest = s1[2][0] * x_##s2 + s1[2][1] * y_##s2 + s1[2][2] * z_##s2
```

Multiples the input vector by the input matrix (3x3) (left to right).

**Parameters**

|             |                                                                                                           |
|-------------|-----------------------------------------------------------------------------------------------------------|
| <i>dest</i> | Output vector, declared using <a href="#">DECLARE_VECTOR()</a>                                            |
| <i>s1</i>   | 3 x 3 input matrix                                                                                        |
| <i>s2</i>   | Input vector, declared using <a href="#">DECLARE_VECTOR()</a> . Has to be different from the destination. |

**5.16.2.12 NORM**

```
#define NORM(
 s) sqrt(NORM2(s))
```

Calculates the norm of the input vector in 32-bit floating point.

**Parameters**

|          |                                                               |
|----------|---------------------------------------------------------------|
| <i>s</i> | Input vector, declared using <a href="#">DECLARE_VECTOR()</a> |
|----------|---------------------------------------------------------------|

**Returns**

float Norm of the input vector



## 5.16.2.13 NORM2

```
#define NORM2(
 s) x_##s *x_##s + y_##s *y_##s + z_##s *z_##s
```

Calculates the square of the norm of the input vector in 32-bit floating point.

## Parameters

|          |                                                               |
|----------|---------------------------------------------------------------|
| <i>s</i> | Input vector, declared using <a href="#">DECLARE_VECTOR()</a> |
|----------|---------------------------------------------------------------|

## Returns

float Square of the norm of the input vector

## 5.16.2.14 NORMALIZE

```
#define NORMALIZE(
 dest,
 s1)
```

## Value:

```
for (float sh__temp = INVNORM(s1); sh__temp != 0;) \
{
 x_##dest = x_##s1 * sh__temp;
 y_##dest = y_##s1 * sh__temp;
 z_##dest = z_##s1 * sh__temp;
 break;
}
```

Normalizes the input vector and stores it in the output vector. Works for null vectors as well.

## Parameters

|             |                                                                     |
|-------------|---------------------------------------------------------------------|
| <i>dest</i> | Destination vector, declared using <a href="#">DECLARE_VECTOR()</a> |
| <i>s1</i>   | Source vector, declared using <a href="#">DECLARE_VECTOR()</a>      |

## 5.16.2.15 VECTOR\_CLEAR

```
#define VECTOR_CLEAR(
 name)
```

## Value:

```

x_##name = 0; \
 y_##name = 0; \
 z_##name = 0

```

Clears a vector.

#### Parameters

|             |                    |
|-------------|--------------------|
| <i>name</i> | Name of the vector |
|-------------|--------------------|

#### 5.16.2.16 VECTOR\_MIXED

```

#define VECTOR_MIXED(
 dest,
 s1,
 s2,
 op)

```

#### Value:

```

x_##dest = x_##s1 op s2; \
 y_##dest = y_##s1 op s2; \
 z_##dest = z_##s1 op s2

```

Performs element-by-element operation on a vector with a scalar and stores in the destination vector. Since the operations are performed element-by-element, the scalar can not depend on the source vector.

#### Parameters

|             |                                                                                                                                  |
|-------------|----------------------------------------------------------------------------------------------------------------------------------|
| <i>dest</i> | Destination vector, declared using <a href="#">DECLARE_VECTOR()</a>                                                              |
| <i>s1</i>   | Input vector, declared using <a href="#">DECLARE_VECTOR()</a>                                                                    |
| <i>s2</i>   | Input scalar                                                                                                                     |
| <i>op</i>   | Operation to perform on an element-by-element basis, e.g. +, -, *, /. Note: For division there is no check for division by zero. |

#### 5.16.2.17 VECTOR\_OP

```

#define VECTOR_OP(
 dest,
 s1,
 s2,
 op)

```

#### Value:

```
x_##dest = x_##s1 op x_##s2; \
y_##dest = y_##s1 op y_##s2; \
z_##dest = z_##s1 op z_##s2
```

Performs a vector operation on the source vectors and stores in destination vector. Since the operations are performed element-by-element, the destination vector can be the same as any of the source vectors.

#### Parameters

|             |                                                                                                                                  |
|-------------|----------------------------------------------------------------------------------------------------------------------------------|
| <i>dest</i> | Destination vector, declared using <a href="#">DECLARE_VECTOR()</a>                                                              |
| <i>s1</i>   | First vector, declared using <a href="#">DECLARE_VECTOR()</a>                                                                    |
| <i>s2</i>   | Second vector, declared using <a href="#">DECLARE_VECTOR()</a>                                                                   |
| <i>op</i>   | Operation to perform on an element-by-element basis, e.g. +, -, *, /. Note: For division there is no check for division by zero. |

### 5.16.3 Function Documentation

#### 5.16.3.1 bootCount()

```
int bootCount (
 void)
```

Function that returns the current bootcount of the system. Returns current boot count, and increases by 1 and stores it in nvmm. Expected to be invoked only by `_main()`

#### Returns

int Current boot count (C-style)

#### 5.16.3.2 daverage()

```
double daverage (
 double arr[],
 int size) [inline]
```

Calculates double precision point average of a float array.

#### Parameters

|             |                                              |
|-------------|----------------------------------------------|
| <i>arr</i>  | Pointer to array whose average is calculated |
| <i>size</i> | Length of the input array                    |

**Returns**

double Average of the input array

**5.16.3.3 faverage()**

```
float faverage (
 float arr[],
 int size) [inline]
```

Calculates floating point average of a float array.

**Parameters**

|             |                                              |
|-------------|----------------------------------------------|
| <i>arr</i>  | Pointer to array whose average is calculated |
| <i>size</i> | Length of the input array                    |

**Returns**

float Average of the input array

**5.16.3.4 get\_usec()**

```
uint64_t get_usec (
 void) [inline]
```

Returns time elapsed from 1970-1-1, 00:00:00 UTC to now (UTC) in microseconds. Execution time  $\sim 18$  us on RPi.

**Returns**

uint64\_t Number of microseconds elapsed from epoch.

**5.16.3.5 q2isqrt()**

```
float q2isqrt (
 float x) [inline]
```

float [q2isqrt\(float\)](#): Returns the inverse square root of a floating point number. Depending on whether MATH\_SQRT is declared, it will use sqrt() function from gcc-math or bit-level hack and 3 rounds of Newton-Raphson to directly calculate inverse square root. The bit-level routine yields consistently better performance and 0.00001% maximum error. Set MATH\_SQRT at compile time to use the sqrt() function.

## Parameters

|   |                                                                        |
|---|------------------------------------------------------------------------|
| x | Floating point number (32-bit) whose inverse square root is calculated |
|---|------------------------------------------------------------------------|

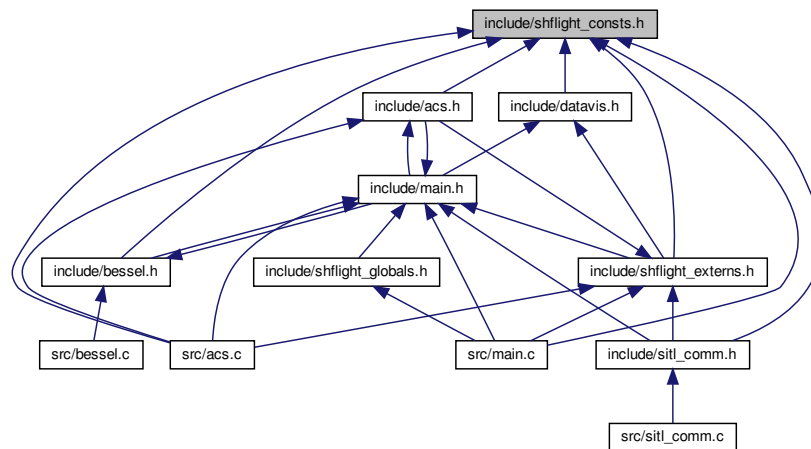
## Returns

float Inverse square root of the input

## 5.17 include/shflight\_consts.h File Reference

Describes all constants defined by the preprocessor for the code.

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



## Macros

- `#define I2C_BUS "/dev/i2c-1"`  
*I2C Bus device file used for ACS sensors.*
- `#define SPIDEV_ACS "/dev/spidev0.0"`  
*SPI device file for H-Bridge (ACS)*
- `#define SH_BUFFER_SIZE 64`  
*Circular buffer size for ACS sensor data.*
- `#define DIPOLE_MOMENT 0.22`  
*Dipole moment of the magnetorquer rods.*
- `#define DETUMBLE_TIME_STEP 100000`  
*ACS loop time period.*
- `#define MEASURE_TIME 20000`  
*ACS `readSensors()` max execute time per cycle.*

- `#define MAX_DETUMBLE_FIRING_TIME (DETUMBLE_TIME_STEP - MEASURE_TIME)`  
*ACS max actuation time per cycle.*
- `#define MIN_DETUMBLE_FIRING_TIME 10000`  
*Minimum magnetorquer firing time.*
- `#define SUNPOINT_DUTY_CYCLE 20000`  
*Sunpointing magnetorquer PWM duty cycle.*
- `#define COARSE_TIME_STEP DETUMBLE_TIME_STEP`  
*Course sun sensing mode loop time for ACS.*
- `#define CSS_MIN_LUX_THRESHOLD 5000 * 0.5`  
*Coarse sun sensor minimum lux threshold for valid measurement.*
- `#define OMEGA_TARGET_LEEWAY z_g_W_target * 0.1`  
*Acceptable leeway of the angular speed target.*
- `#define MIN_SOL_ANGLE 4`  
*Sunpointing angle target (in degrees)*
- `#define MIN_DETUMBLE_ANGLE 4`  
*Detumble angle target (in degrees)*
- `#define PORT 12376`  
*DataVis port number.*
- `#define BESSEL_MIN_THRESHOLD 0.001`  
*Bessel coefficient minimum value threshold for computation.*
- `#define BESSEL_FREQ_CUTOFF 5`  
*Bessel filter cutoff frequency.*

### 5.17.1 Detailed Description

Describes all constants defined by the preprocessor for the code.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

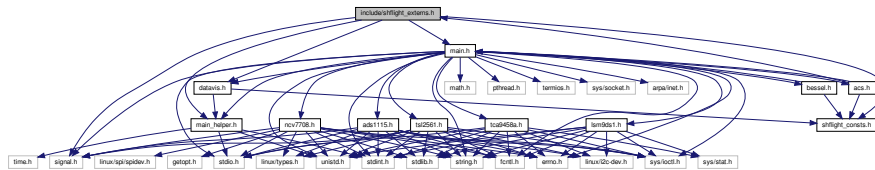
Copyright (c) 2020

## 5.18 include/shflight\_externs.h File Reference

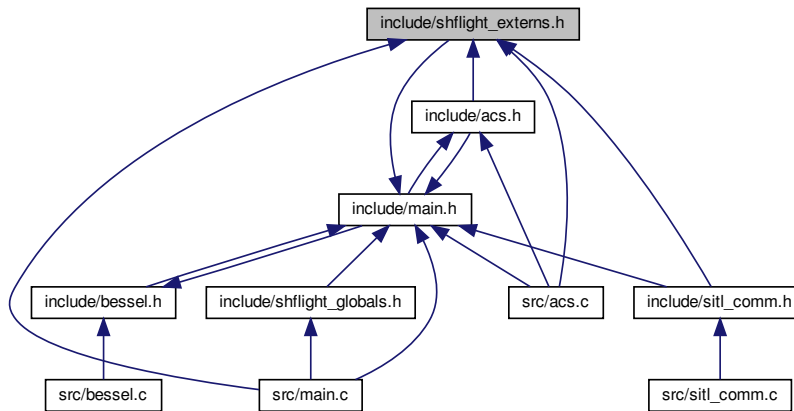
Extern declaration of the shflight global variables for all threads.

```
#include <signal.h>
#include <main.h>
#include <main_helper.h>
#include <shflight_consts.h>
#include <datavis.h>
```

Include dependency graph for shflight\_externs.h:



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



### Functions

- **DECLARE\_VECTOR2** (g\_readB, extern unsigned short)
- **DECLARE\_BUFFER** (g\_W, extern float)
- **DECLARE\_BUFFER** (g\_B, extern double)
- **DECLARE\_BUFFER** (g\_Bt, extern double)
- **DECLARE\_VECTOR2** (g\_L\_target, extern float)
- **DECLARE\_VECTOR2** (g\_W\_target, extern float)
- **DECLARE\_BUFFER** (g\_S, extern float)

## Variables

- pthread\_mutex\_t [serial\\_read](#)  
*Mutex to ensure atomicity of serial data read into the system.*
- pthread\_mutex\_t [serial\\_write](#)  
*Mutex to ensure atomicity of magnetorquer output for serial communication.*
- pthread\_mutex\_t [data\\_check](#)
- pthread\_cond\_t [data\\_available](#)  
*Condition variable to synchronize ACS and Serial thread in SITL.*
- pthread\_mutex\_t [datavis\\_mutex](#)  
*Mutex to ensure atomicity of DataVis and ACS variable access.*
- pthread\_cond\_t [datavis\\_drdy](#)  
*Condition variable used by ACS to signal to DataVis that data is ready.*
- volatile sig\_atomic\_t [done](#)  
*Control variable for thread loops.*
- volatile int [first\\_run](#)  
*This variable is unset by the ACS thread at first execution.*
- unsigned short [g\\_readFS](#) [2]  
*Fine sun sensor angles read over serial.*
- unsigned short [g\\_readCS](#) [9]  
*Coarse sun sensor lux values read over serial.*
- unsigned char [g\\_Fire](#)  
*Magnetorquer command, format: 0b00ZZYYXX, 00 indicates not fired, 01 indicates fire in positive dir, 10 indicates fire in negative dir.*
- [lsm9ds1](#) \* [mag](#)  
*Magnetometer device struct.*
- [ncv7708](#) \* [hbridge](#)  
*H-Bridge device struct.*
- [tca9458a](#) \* [mux](#)  
*I2C Mux device struct.*
- [tsl2561](#) \*\* [css](#)  
*Array of coarse sun sensor device struct.*
- [ads1115](#) \* [adc](#)  
*I2C ADC struct for fine sun sensor.*
- float [g\\_CSS](#) [9]  
*Storage for current coarse sun sensor lux measurements.*
- float [g\\_FSS](#) [2]  
*Storage for current fine sun sensor angle measurements.*
- int [mag\\_index](#)  
*Current index of the  $\vec{B}$  circular buffer.*
- int [omega\\_index](#)  
*Current index of the  $\vec{\omega}$  circular buffer.*
- int [bdot\\_index](#)  
*Current index of the  $\vec{\dot{B}}$  circular buffer.*
- int [sol\\_index](#)  
*Current index of the sun vector circular buffer.*
- int [B\\_full](#)



- Indicates if the  $\vec{B}$  circular buffer is full.*

  - int `Bdot_full`
- Indicates if the  $\vec{B}$  circular buffer is full.*

  - int `W_full`
- Indicates if the  $\vec{\omega}$  circular buffer is full.*

  - int `S_full`
- Indicates if the sun vector circular buffer is full.*

  - uint8\_t `g_night`
- This variable is set by `checkTransition()` if the satellite does not detect the sun.*

  - uint8\_t `g_acs_mode`
- This variable contains the current state of the flight system.*

  - uint8\_t `g_first_detumble`
- This variable is unset when the system is detumbled for the first time after a power cycle.*

  - unsigned long long `acs_ct`
- Counts the number of cycles on the ACS thread.*

  - float `bessel_coeff` [`SH_BUFFER_SIZE`]
- Coefficients for the Bessel filter, calculated using `calculateBessel()`.*

  - float `MOI` [3][3]
- Moment of inertia of the satellite (SI).*

  - float `IMOI` [3][3]
- Inverse of the moment of inertia of the satellite (SI).*

  - unsigned long long `g_t_acs`
- Current timestamp after `readSensors()` in ACS thread, used to keep track of time taken by ACS loop.*

  - `data_packet` `g_datavis_st`
- DataVis data structure.*

  - unsigned long long `t_comm`
- SITL communication time.*

  - unsigned long long `comm_time`

### 5.18.1 Detailed Description

Extern declaration of the shflight global variables for all threads.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020



## Variables

- `__thread int sys_status`  
*Thread-local system status variable (similar to errno).*
- `pthread_mutex_t serial_read`  
*Mutex to ensure atomicity of serial data read into the system.*
- `pthread_mutex_t serial_write`  
*Mutex to ensure atomicity of magnetorquer output for serial communication.*
- `pthread_mutex_t data_check`
- `pthread_cond_t data_available`  
*Condition variable to synchronize ACS and Serial thread in SITL.*
- `pthread_mutex_t datavis_mutex`  
*Mutex to ensure atomicity of DataVis and ACS variable access.*
- `pthread_cond_t datavis_drdy`  
*Condition variable used by ACS to signal to DataVis that data is ready.*
- `volatile sig_atomic_t done = 0`  
*Control variable for thread loops.*
- `volatile int first_run = 1`  
*This variable is unset by the ACS thread at first execution.*
- `unsigned short g_readFS [2]`  
*Fine sun sensor angles read over serial.*
- `unsigned short g_readCS [9]`  
*Coarse sun sensor lux values read over serial.*
- `unsigned char g_Fire`  
*Magnetorquer command, format: 0b00ZZYYXX, 00 indicates not fired, 01 indicates fire in positive dir, 10 indicates fire in negative dir.*
- `lsm9ds1 * mag`  
*Magnetometer device struct.*
- `ncv7708 * hbridge`  
*H-Bridge device struct.*
- `tca9458a * mux`  
*I2C Mux device struct.*
- `tsl2561 ** css`  
*Array of coarse sun sensor device struct.*
- `ads1115 * adc`  
*I2C ADC struct for fine sun sensor.*
- `float g_CSS [9]`  
*Storage for current coarse sun sensor lux measurements.*
- `float g_FSS [2]`  
*Storage for current fine sun sensor angle measurements.*
- `int mag_index = -1`  
*Current index of the  $\vec{B}$  circular buffer.*
- `int omega_index = -1`  
*Current index of the  $\vec{\omega}$  circular buffer.*
- `int bdot_index = -1`  
*Current index of the  $\vec{B}$  circular buffer.*
- `int sol_index = -1`

- *Current index of the sun vector circular buffer.*
- int `B_full` = 0  
*Indicates if the  $\vec{B}$  circular buffer is full.*
- int `Bdot_full` = 0  
*Indicates if the  $\vec{B}$  circular buffer is full.*
- int `W_full` = 0  
*Indicates if the  $\vec{\omega}$  circular buffer is full.*
- int `S_full` = 0  
*Indicates if the sun vector circular buffer is full.*
- uint8\_t `g_night` = 0  
*This variable is set by `checkTransition()` if the satellite does not detect the sun.*
- uint8\_t `g_acs_mode` = 0  
*This variable contains the current state of the flight system.*
- uint8\_t `g_first_detumble` = 1  
*This variable is unset when the system is detumbled for the first time after a power cycle.*
- unsigned long long `acs_ct` = 0  
*Counts the number of cycles on the ACS thread.*
- float `MOI` [3][3]  
*Moment of inertia of the satellite (SI).*
- float `IMOI` [3][3]  
*Inverse of the moment of inertia of the satellite (SI).*
- float `bessel_coeff` [SH\_BUFFER\_SIZE]  
*Coefficients for the Bessel filter, calculated using `calculateBessel()`.*
- unsigned long long `g_t_acs`  
*Current timestamp after `readSensors()` in ACS thread, used to keep track of time taken by ACS loop.*
- `data_packet` `g_datavis_st`  
*DataVis data structure.*
- unsigned long long `t_comm` = 0  
*SITL communication time.*
- unsigned long long `comm_time`

### 5.19.1 Detailed Description

Allocates memory for the global variables used in the flight code for inter-thread communication.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

## 5.19.2 Variable Documentation

### 5.19.2.1 IMOI

```
float IMOI[3][3]
```

**Initial value:**

```
= {{15.461398105297564, 0, 0},
 {0, 15.461398105297564, 0},
 {0, 0, 12.623336025344317}}
```

Inverse of the moment of inertia of the satellite (SI).

### 5.19.2.2 MOI

```
float MOI[3][3]
```

**Initial value:**

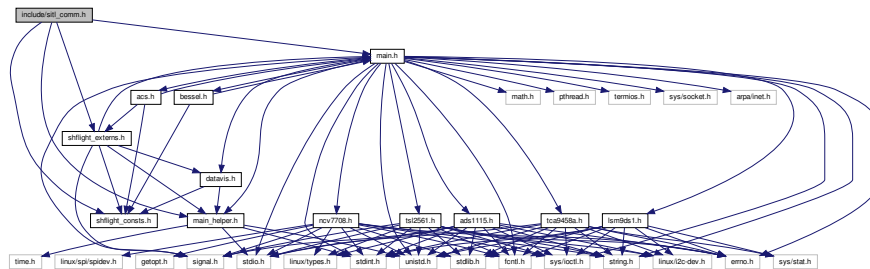
```
= {{0.06467720404, 0, 0},
 {0, 0.06474406267, 0},
 {0, 0, 0.07921836177}}
```

Moment of inertia of the satellite (SI).

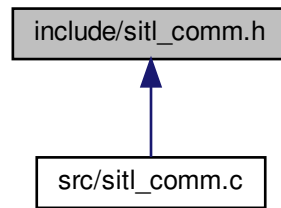
## 5.20 include/sitl\_comm.h File Reference

Software-In-The-Loop (SITL) serial communication headers and function prototypes.

```
#include <main.h>
#include <main_helper.h>
#include <shflight_consts.h>
#include <shflight_extens.h>
Include dependency graph for sitl_comm.h:
```



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



## Functions

- int [set\\_interface\\_attribs](#) (int fd, int speed, int parity)  
*Set speed and parity attributes for the serial device.*
- void [set\\_blocking](#) (int fd, int should\_block)  
*Set the serial device as blocking or non-blocking.*
- int [setup\\_serial](#) (void)  
*Set the up serial device Opens the serial device /dev/ttyS0 (for RPi only)*
- void \* [sitl\\_comm](#) (void \*id)  
*Serial communication thread.*

### 5.20.1 Detailed Description

Software-In-The-Loop (SITL) serial communication headers and function prototypes.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

## 5.20.2 Function Documentation

### 5.20.2.1 set\_blocking()

```
void set_blocking (
 int fd,
 int should_block)
```

Set the serial device as blocking or non-blocking.

#### Parameters

|                     |                                                   |
|---------------------|---------------------------------------------------|
| <i>fd</i>           | Serial device file descriptor                     |
| <i>should_block</i> | 0 for non-blocking, 1 for blocking mode operation |

### 5.20.2.2 set\_interface\_attribs()

```
int set_interface_attribs (
 int fd,
 int speed,
 int parity)
```

Set speed and parity attributes for the serial device.

#### Parameters

|               |                                                                 |
|---------------|-----------------------------------------------------------------|
| <i>fd</i>     | Serial device file descriptor                                   |
| <i>speed</i>  | Baud rate, is a constant of the form B#### defined in termios.h |
| <i>parity</i> | Odd or even parity for the serial device (1, 0)                 |

#### Returns

0 on success, -1 on error

### 5.20.2.3 setup\_serial()

```
int setup_serial (
 void)
```

Set the up serial device Opens the serial device /dev/ttyS0 (for RPi only)

**Returns**

file descriptor to the serial device

**5.20.2.4 sitl\_comm()**

```
void* sitl_comm (
 void * id)
```

Serial communication thread.

Communicates with the environment simulator over serial port. The serial communication happens at 230400 bps, and this thread is intended to loop at 200 Hz. The thread reads the packet over serial (packet format: [0xa0 x 10] [uint8 x 28] [0xb0 x 2]). The thread synchronizes to the 0xa0 in the beginning and checks for the 0xb0 at the end at each iteration. The data is read into global variables, and the magnetorquer command is read out. All read-writes are atomic.

**Parameters**

|           |                                            |
|-----------|--------------------------------------------|
| <i>id</i> | Pointer to an int that specifies thread ID |
|-----------|--------------------------------------------|

**Returns**

NULL

**5.21 include/uhf.h File Reference**

EnduroSat UHF Transceiver Interface Code function prototypes (Needs to be written)

**Functions**

- void \* [uhf](#) (void \*id)  
*UHF main thread.*

**5.21.1 Detailed Description**

EnduroSat UHF Transceiver Interface Code function prototypes (Needs to be written)

**Author**

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))



**Version**

0.1

**Date**

2020-03-19

**Copyright**

Copyright (c) 2020

**5.21.2 Function Documentation****5.21.2.1 uhf()**

```
void* uhf (
 void * id)
```

UHF main thread.

**Parameters**

|           |                                          |
|-----------|------------------------------------------|
| <i>id</i> | Pointer to integer containing thread ID. |
|-----------|------------------------------------------|

**Returns**

NULL

**5.22 include/xband.h File Reference**

SPACE-HAUC X-Band Transceiver function prototypes (Needs to be written)

**Functions**

- void \* [xband](#) (void \*id)  
*X-band thread.*

### 5.22.1 Detailed Description

SPACE-HAUC X-Band Transceiver function prototypes (Needs to be written)

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.22.2 Function Documentation

#### 5.22.2.1 xband()

```
void* xband (
 void * id)
```

X-band thread.

#### Parameters

|           |                                         |
|-----------|-----------------------------------------|
| <i>id</i> | Pointer to integer containing thread ID |
|-----------|-----------------------------------------|

#### Returns

NULL

### 5.23 src/acs.c File Reference

Attitude Control System related functions.



### 5.23.1 Detailed Description

Attitude Control System related functions.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.23.2 Function Documentation

#### 5.23.2.1 `acs_init()`

```
int acs_init (
 void)
```

Initializes the devices required to run the attitude control system.

This function initializes the target angular momentum using MOI defined in [shflight\\_globals.h](#) and the target angular speed set in [main.c](#). Then this function initializes all the relevant devices for ACS to function.

#### Returns

int 1 on success, error codes defined in SH\_ERRORS on error.

#### 5.23.2.2 `acs_thread()`

```
void* acs_thread (
 void *)
```

Attitude Control System Thread.

This thread executes the ACS functions in a loop controlled by the variable `done`, which is controlled by the interrupt handler.

## Parameters

|           |                                              |
|-----------|----------------------------------------------|
| <i>id</i> | Thread ID passed as a pointer to an integer. |
|-----------|----------------------------------------------|

## Returns

NULL

## 5.23.2.3 detumbleAction()

```
static void detumbleAction (
 void) [inline], [static]
```

This function executes the detumble algorithm.

The detumble algorithm calculates the direction and time for which the magnetorquers fire. The direction is determined by first calculating the vector  $\hat{B} \times L_0 - L$ , which is a unit vector, and then checking which of the components have a magnitude greater than 0.01. A component with magnitude greater than 0.01 indicates that torquer can be fired, in the direction indicated by the sign of the component. Further, the torque that is generated by the firing decision is estimated for the current value of the magnetic field by calculating  $\vec{\tau} = \vec{m}u \times \vec{B}$ , where  $\vec{m}u$  is calculated by multiplying the firing direction vector with the dipole moment of the magnetorquers ( $0.21 \text{ A} \cdot \text{m}^2$ ). Then for each direction, the firing time is estimated by  $t_i = \frac{\Delta L_i}{\tau_i}$ . The torquer in any direction is fired only if the firing time is greater than 5 ms, and any torquer is fired for at most the allowed firing time. At the end of the action, all torquers are turned off for the next magnetic field measurement.

## 5.23.2.4 getOmega()

```
void getOmega (
 void)
```

Calculates  $\omega$  using  $\dot{\vec{B}}$  and stores in the circular buffer.

Calculates current angular speed. Requires current and previous measurements of  $\dot{\vec{B}}$ . The calculated angular speed is put inside the global circular buffer. Sets W\_full to indicate the buffer becoming full the first time.

## 5.23.2.5 HBRIDGE\_DISABLE()

```
int HBRIDGE_DISABLE (
 int num)
```

Disables magnetorquer in the axis indicated by the input.

**Parameters**

|            |                                                                                                                                        |
|------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <i>num</i> | Integer, 0 indicates X axis, 1 indicates Y axis, 2 indicates Z axis. In hardware, a number > 2 causes all three torquers to shut down. |
|------------|----------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

int Status of the operation, returns 1 on success.

**5.23.2.6 hbridge\_enable()**

```
int hbridge_enable (
 int x,
 int y,
 int z)
```

Fire magnetorquer in X, Y, and Z directions using the input integers.

**Parameters**

|          |                                                                                                  |
|----------|--------------------------------------------------------------------------------------------------|
| <i>x</i> | Fires in the +X or -X direction depending on the input being +1 or -1, and does nothing if x = 0 |
| <i>y</i> | Fires in the +Y or -Y direction depending on the input being +1 or -1, and does nothing if y = 0 |
| <i>z</i> | Fires in the +Z or -Z direction depending on the input being +1 or -1, and does nothing if z = 0 |

**Returns**

int Status of the operation, returns 1 on success.

**5.23.2.7 insertionSort()**

```
void insertionSort (
 int a1[],
 int a2[])
```

Sorts the first array and reorders the second array according to the first array.

**Parameters**

|           |                                      |
|-----------|--------------------------------------|
| <i>a1</i> | Pointer to integer array to sort.    |
| <i>a2</i> | Pointer to integer array to reorder. |

## 5.23.2.8 readSensors()

```
int readSensors (
 void)
```

Reads hardware sensors and puts the values in the global storage, upon which calls the [getOmega\(\)](#) and [getSVec\(\)](#) functions to calculate angular speed and sun vector.

## Returns

int Returns 1 for success, and -1 for error.

## 5.23.2.9 sunpointAction()

```
static void sunpointAction (
 void) [inline], [static]
```

This function executes the sunpointing algorithm.

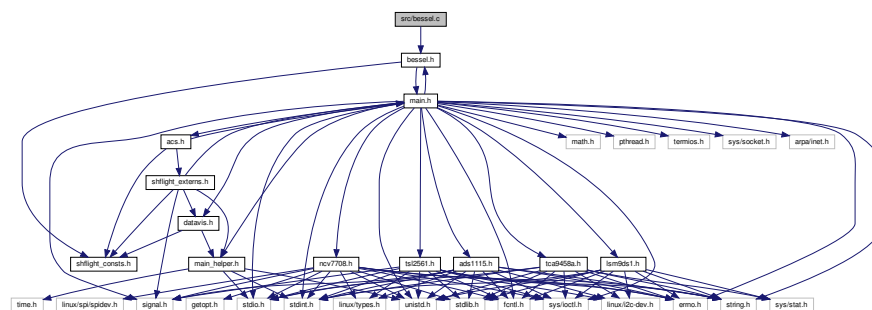
The sunpointing algorithm calculates the duty cycle of the Z-magnetorquer firing. The duty cycle is determined by calculating the vector  $(\hat{S}(\hat{S} \cdot \hat{B})) \times ((\hat{L}(\hat{L} \cdot \hat{B}))$ . The Z component of this vector upon normalization specifies the duty cycle. However, due to lowering of efficiency as the spacecraft aligns with the sun, the gain is increased.

## 5.24 src/bessel.c File Reference

Bessel filter implementation for Attitude Control System.

```
#include <bessel.h>
```

Include dependency graph for `bessel.c`:



## Functions

- static float [factorial](#) (int i)  
*Calculates factorial of the input. This function is inlined, and is available only in the scope of [bessel.c](#).*
- void [calculateBessel](#) (float arr[], int size, int order, float freq\_cutoff)  
*Calculates discrete Bessel filter coefficients for the given order and cutoff frequency.*
- double [dfilterBessel](#) (double arr[], int index)  
*Returns the filtered value at the current index using past values.*
- float [ffilterBessel](#) (float arr[], int index)  
*Returns the filtered value at the current index using past values.*

### 5.24.1 Detailed Description

Bessel filter implementation for Attitude Control System.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.24.2 Function Documentation

#### 5.24.2.1 [calculateBessel\(\)](#)

```
void calculateBessel (
 float arr[],
 int size,
 int order,
 float freq_cutoff)
```

Calculates discrete Bessel filter coefficients for the given order and cutoff frequency.



## Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <i>arr</i>         | Stores the filter coefficients         |
| <i>size</i>        | Size of the filter coefficients array  |
| <i>order</i>       | Order of the Bessel filter             |
| <i>freq_cutoff</i> | Cut-off frequency of the Bessel filter |

## 5.24.2.2 dfilterBessel()

```
double dfilterBessel (
 double arr[],
 int index)
```

Returns the filtered value at the current index using past values.

## Parameters

|              |                                     |
|--------------|-------------------------------------|
| <i>arr</i>   | Input array                         |
| <i>index</i> | Index of current value in the array |

## Returns

double Filtered value

## 5.24.2.3 factorial()

```
static float factorial (
 int i) [inline], [static]
```

Calculates factorial of the input. This function is inlined, and is available only in the scope of [bessel.c](#).

## Parameters

|          |       |
|----------|-------|
| <i>i</i> | Input |
|----------|-------|

## Returns

float Factorial of input

#### 5.24.2.4 ffilterBessel()

```
float ffilterBessel (
 float arr[],
 int index)
```

Returns the filtered value at the current index using past values.

##### Parameters

|              |                                     |
|--------------|-------------------------------------|
| <i>arr</i>   | Input array                         |
| <i>index</i> | Index of current value in the array |

##### Returns

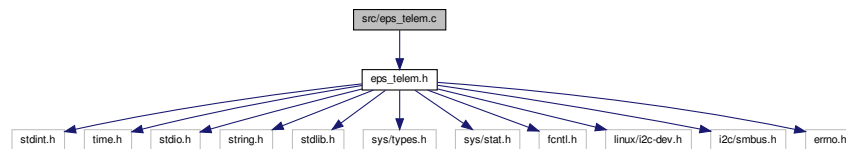
double Filtered value

## 5.25 src/eps\_telem.c File Reference

GomSpace P31u I2C interface function declarations.

```
#include <eps_telem.h>
```

Include dependency graph for eps\_telem.c:



## Functions

- void \* **eps\_telem** (void \*id)
- int **p31u\_init** (p31u \*dev)
- void **p31u\_destroy** (p31u \*dev)
- int **p31u\_xfer** (p31u \*dev, char \*out, ssize\_t outsize, char \*in, ssize\_t insize)
- int **eps\_ping** (p31u \*dev)
- int **eps\_reboot** (p31u \*dev)
- int **eps\_get\_hk** (p31u \*dev, uint8\_t mode)
- int **eps\_hk** (p31u \*dev)
- int **eps\_set\_output** (p31u \*dev, channel\_t channels)
- int **eps\_set\_single** (p31u \*dev, uint8\_t channel, uint8\_t value, int16\_t delay)
- int **eps\_reset\_wdt** (p31u \*dev)



## Functions

- int `main` (void)  
*Main function executed when shflight.out binary is executed.*
- void `catch_sigint` (int sig)  
*SIGINT handler, sets the global variable `done` as 1, so that thread loops can break. Wakes up `sitl_comm` and `datavis` threads to ensure they exit.*
- void `sherror` (const char \*msg)  
*Prints errors specific to shflight in a fashion similar to perror.*

### 5.26.1 Detailed Description

`main()` symbol of the SPACE-HAUC Flight Software.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.26.2 Function Documentation

#### 5.26.2.1 `catch_sigint()`

```
void catch_sigint (
 int sig)
```

SIGINT handler, sets the global variable `done` as 1, so that thread loops can break. Wakes up `sitl_comm` and `datavis` threads to ensure they exit.

#### Parameters

|                  |                               |
|------------------|-------------------------------|
| <code>sig</code> | Receives the signal as input. |
|------------------|-------------------------------|

### 5.26.2.2 main()

```
int main (
 void)
```

Main function executed when shflight.out binary is executed.

#### Returns

int returns 0 on success, -1 on failure, error code on thread init failures

### 5.26.2.3 sherror()

```
void sherror (
 const char * msg)
```

Prints errors specific to shflight in a fashion similar to perror.

#### Parameters

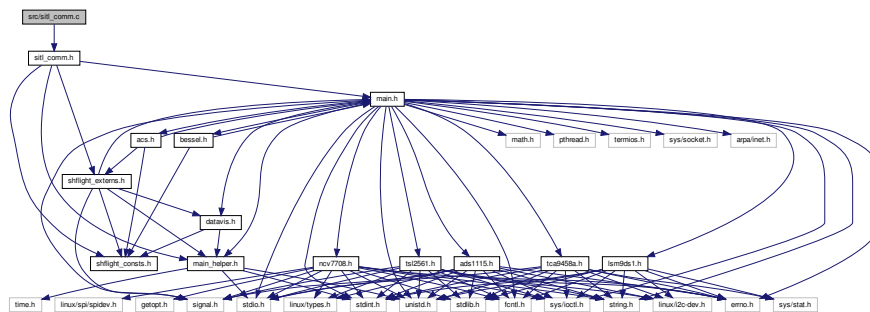
|            |                                                     |
|------------|-----------------------------------------------------|
| <i>msg</i> | Input message to print along with error description |
|------------|-----------------------------------------------------|

## 5.27 src/sitl\_comm.c File Reference

Software-In-The-Loop (SITL) serial communication codes.

```
#include <sitl_comm.h>
```

Include dependency graph for sitl\_comm.c:



## Functions

- int [set\\_interface\\_attribs](#) (int fd, int speed, int parity)  
*Set speed and parity attributes for the serial device.*
- void [set\\_blocking](#) (int fd, int should\_block)  
*Set the serial device as blocking or non-blocking.*
- int [setup\\_serial](#) (void)  
*Set the up serial device Opens the serial device /dev/ttyS0 (for RPi only)*
- void \* [sitl\\_comm](#) (void \*id)  
*Serial communication thread.*

### 5.27.1 Detailed Description

Software-In-The-Loop (SITL) serial communication codes.

#### Author

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

#### Version

0.1

#### Date

2020-03-19

#### Copyright

Copyright (c) 2020

### 5.27.2 Function Documentation

#### 5.27.2.1 [set\\_blocking\(\)](#)

```
void set_blocking (
 int fd,
 int should_block)
```

Set the serial device as blocking or non-blocking.

**Parameters**

|                     |                                                   |
|---------------------|---------------------------------------------------|
| <i>fd</i>           | Serial device file descriptor                     |
| <i>should_block</i> | 0 for non-blocking, 1 for blocking mode operation |

**5.27.2.2 set\_interface\_attribs()**

```
int set_interface_attribs (
 int fd,
 int speed,
 int parity)
```

Set speed and parity attributes for the serial device.

**Parameters**

|               |                                                                              |
|---------------|------------------------------------------------------------------------------|
| <i>fd</i>     | Serial device file descriptor                                                |
| <i>speed</i>  | Baud rate, is a constant of the form B#### defined in <code>termios.h</code> |
| <i>parity</i> | Odd or even parity for the serial device (1, 0)                              |

**Returns**

0 on success, -1 on error

**5.27.2.3 setup\_serial()**

```
int setup_serial (
 void)
```

Set the up serial device Opens the serial device `/dev/ttyS0` (for RPi only)

**Returns**

file descriptor to the serial device

#### 5.27.2.4 sitl\_comm()

```
void* sitl_comm (
 void * id)
```

Serial communication thread.

Communicates with the environment simulator over serial port. The serial communication happens at 230400 bps, and this thread is intended to loop at 200 Hz. The thread reads the packet over serial (packet format: [0xa0 x 10] [uint8 x 28] [0xb0 x 2]). The thread synchronizes to the 0xa0 in the beginning and checks for the 0xb0 at the end at each iteration. The data is read into global variables, and the magnetorquer command is read out. All read-writes are atomic.



**Parameters**

|           |                                            |
|-----------|--------------------------------------------|
| <i>id</i> | Pointer to an int that specifies thread ID |
|-----------|--------------------------------------------|

**Returns**

NULL

## 5.28 src/uhf.c File Reference

UHF interface code.

### 5.28.1 Detailed Description

UHF interface code.

**Author**

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

**Version**

0.1

**Date**

2020-03-19

**Copyright**

Copyright (c) 2020

## 5.29 src/xband.c File Reference

X-Band Radio interface code.

### 5.29.1 Detailed Description

X-Band Radio interface code.

**Author**

Sunip K. Mukherjee ([sunipkmukherjee@gmail.com](mailto:sunipkmukherjee@gmail.com))

**Version**

0.1

**Date**

2020-03-19

**Copyright**

Copyright (c) 2020

