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 genrates 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 \$\$ which leads to a bias on \$ z\$. This throws off the detumble determination. Find a better filter/criterion.

1. Investigate the effect of \$ < 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:

Chapter 3

File Index

3.1 File List

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

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)
  - uint8_t comp_lat: 1
    Latching comparator (ADS1114 and ADS1115 only)
  - uint8_t comp_mode: 1
    Comparator polarity (ADS1114 and ADS1115 only)
  - uint8_t comp_pol: 1
    Comparator mode (ADS1114 and ADS1115 only)
  - 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.

4.2.2 Member Data Documentation

4.2.2.1 comp_lat

```
uint8_t ads1115_config::comp_lat
```

Latching comparator (ADS1114 and ADS1115 only)

This bit controls whether the ALERT/RDY pin latches after being asserted or clears after conversions are within the margin of the upper and lower threshold values. This bit serves no function on the ADS1113. 0 : Nonlatching comparator . The ALERT/RDY pin does not latch when asserted (default). 1 : Latching comparator. The asserted ALERT/RDY pin remains latched until conversion data are read by the master or an appropriate SMBus alert response is sent by the master. The device responds with its address, and it is the lowest address currently asserting the ALERT/RDY bus line.

4.2.2.2 comp_mode

```
uint8_t ads1115_config::comp_mode
```

Comparator polarity (ADS1114 and ADS1115 only)

This bit controls the polarity of the ALERT/RDY pin. This bit serves no function on the ADS1113. 0 : Active low (default) 1 : Active high

4.2.2.3 comp_pol

```
uint8_t ads1115_config::comp_pol
```

Comparator mode (ADS1114 and ADS1115 only)

This bit configures the comparator operating mode. This bit serves no function on the ADS1113. 0 : Traditional comparator (default) 1 : Window comparator

4.2.2.4 comp_que

```
uint8_t ads1115_config::comp_que
```

Comparator queue and disable (ADS1114 and ADS1115 only)

These bits perform two functions. When set to 11, the comparator is disabled and the ALERT/RDY pin is set to a high-impedance state. When set to any other value, the ALERT/RDY pin and the comparator function are enabled, and the set value determines the number of successive conversions exceeding the upper or lower threshold required before asserting the ALERT/RDY pin. These bits serve no function on the ADS1113. 00 : Assert after one conversion 01 : Assert after two conversions 10 : Assert after four conversions 11 : Disable comparator and set ALERT/RDY pin to high-impedance (default)

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:

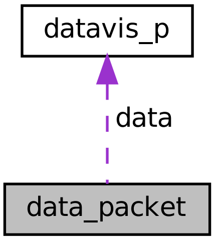
- include/eps_telem.h

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

Internal data structure of a DataVis packet.

```
#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

Internal data structure of a DataVis packet.

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]
- 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].
- 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.

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

4.17.2.3 temp_comp

```
uint8_t MAG_DATA_RATE::temp_comp
```

Temperature compensation enable.

Default value: 0

0: Temperature compensation disabled 1: Temperature compensation enabled

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

- drivers/lsm9ds1.h

4.18 MAG_DATA_READ Struct Reference

Configures data updating method of the magnetometer.

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

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: FAS↵ T_READ disabled, 1: Enabled.

4.18.1 Detailed Description

Configures data updating method of the magnetometer.

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

- drivers/lsm9ds1.h

4.19 MAG_RESET Struct Reference

Reset or configure scale of Magnetometer.

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

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.

4.19.1 Detailed Description

Reset or configure scale of Magnetometer.

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

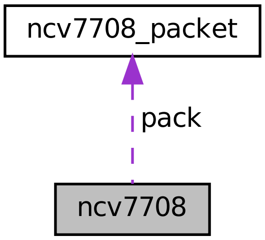
- drivers/lsm9ds1.h

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 hbsel: 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

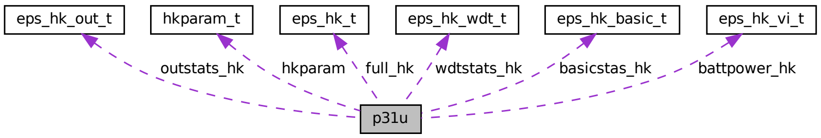
NCV77X8 Data packet (I/O)

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

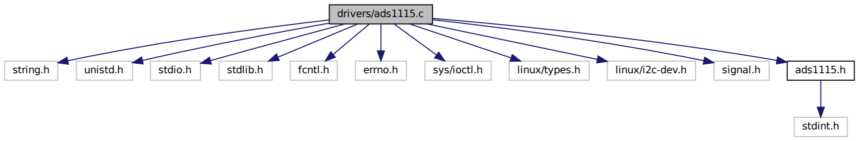
File Documentation

5.1 drivers/ads1115.c File Reference

ADS1115 I2C Driver function definitions.

```
#include <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 "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)

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

dev	Pointer to ads1115 device struct.
m_con	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

dev	Pointer to ads1115 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

dev	Pointer to ads1115 device struct.
s_address	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

dev	Pointer to ads1115 device struct.
data	Pointer to unsigned short (ads1115_config->raw)

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

dev	Pointer to ads1115 device struct.
data	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

dev	Pointer to ads1115 device struct.
data	Pointer to an array of short of length 4 where data is stored

Returns

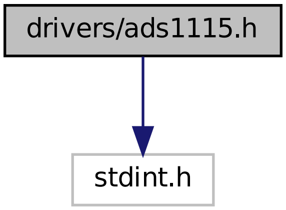
Returns 1 on success, -1 on failure.

5.2 drivers/ads1115.h File Reference

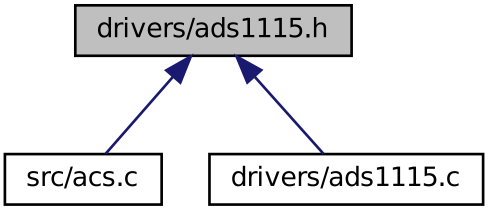
ADS1115 I2C Driver function prototypes and data structures.

```
#include <stdint.h>
```

Include dependency graph for ads1115.h:



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



Classes

- union ads1115_config

  Configuration register.

- struct ads1115

  ads1115 device data structures.

Macros

- #define ADS1115_S_ADDR 0x48

  Default I2C Address.
- #define I2C_BUS "/dev/i2c-1"

  Default I2C Bus.
- #define CONVERSION_REG 0x00

  ADC conversion register.
- #define CONFIG_REG 0x01

  ADC configuration register.

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)

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

dev	Pointer to ads1115 device struct.
m_con	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

dev	Pointer to ads1115 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

dev	Pointer to ads1115 device struct.
s_address	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

dev	Pointer to ads1115 device struct.
data	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

dev	Pointer to ads1115 device struct.
data	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

dev	Pointer to ads1115 device struct.
data	Pointer to an array of short of length 4 where data is stored

Returns

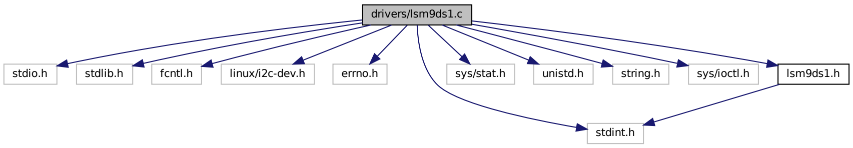
Returns 1 on success, -1 on failure.

5.3 drivers/lsm9ds1.c File Reference

Function definitions for LSM9DS1 Magnetometer I2C driver.

```
#include <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 "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)

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

dev	Pointer to lsm9ds1
datarate	
rst	
dread	

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

dev	Pointer to lsm9ds1

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

dev	Pointer to lsm9ds1
xl_addr	Accelerometer address on I2C Bus (default 0x6b)
mag_addr	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

dev	Pointer to lsm9ds1
offset	Pointer to an array of shorts 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

dev	Pointer to lsm9ds1
B	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

dev	Pointer to lsm9ds1

Returns

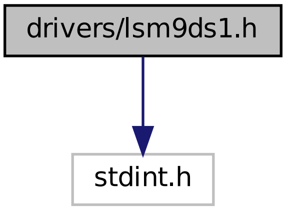
Returns 1 on success, -1 on failure

5.4 drivers/lsm9ds1.h File Reference

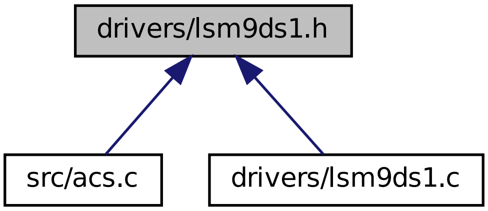
Function prototypes and data structures for LSM9DS1 Magnetometer I2C driver.

```
#include <stdint.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

  Reset or configure scale of Magnetometer.
- struct MAG_DATA_READ

  Configures data updating method of the magnetometer.
- 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_O↩
FFSET_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)

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

dev	Pointer to lsm9ds1
datarate	
rst	
dread	

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

dev	Pointer to lsm9ds1

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

dev	Pointer to lsm9ds1
xl_addr	Accelerometer address on I2C Bus (default 0x6b)
mag_addr	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

dev	Pointer to lsm9ds1
offset	Pointer to an array of shorts 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

dev	Pointer to lsm9ds1
B	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

dev	Pointer to lsm9ds1

Returns

Returns 1 on success, -1 on failure

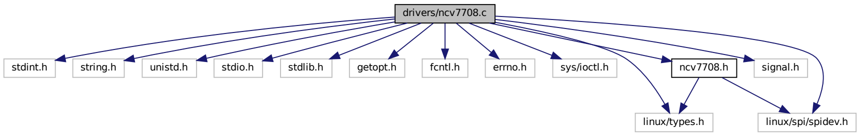
5.5 drivers/ncv7708.c File Reference

Function definitions 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 "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)

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

dev	NCV77X8 Device Handle

5.5.2.2 ncv7708_init()

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

Initialize the SPI bus to communicate with the NCV77X8.

Parameters

dev	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

dev	NCV77X8 Device Handle
data	Pointer to store 16-bit data read over SPI
cmd	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

dev	NCV77X8 Device Handle

Returns

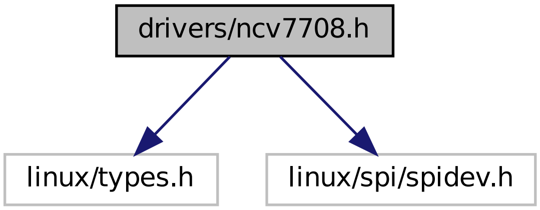
1 on success, -1 on failure

5.6 drivers/ncv7708.h File Reference

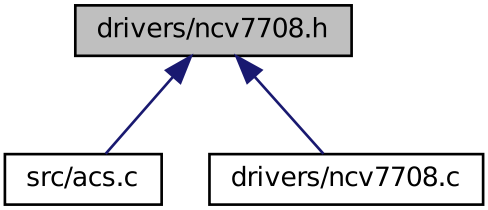
Function prototypes and data structure for NCV77X8 SPI Driver (Linux)

```
#include <linux/types.h>
#include <linux/spi/spidev.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)

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

dev	NCV77X8 Device Handle

5.6.2.2 ncv7708_init()

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

Initialize the SPI bus to communicate with the NCV77X8.

Parameters

dev	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

dev	NCV77X8 Device Handle
data	Pointer to store 16-bit data read over SPI
cmd	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

dev	NCV77X8 Device Handle

Returns

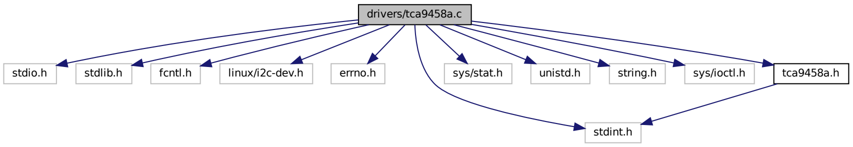
1 on success, -1 on failure

5.7 drivers/tca9458a.c File Reference

Function definitions 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 "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)

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

dev	

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

dev	
addr	TCA9458A device address (default: 0x70)

Returns

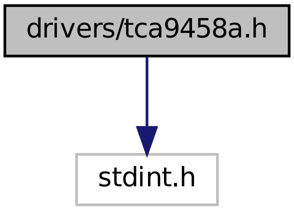
1 on success, -1 on error

5.8 drivers/tca9458a.h File Reference

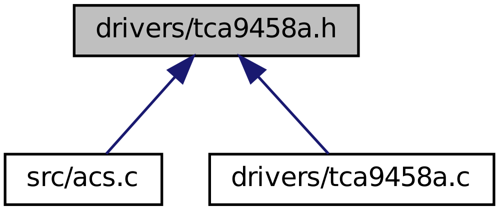
Function prototypes and struct declarations for TCA9458A I2C driver.

```
#include <stdint.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`)

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

dev	

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

dev	
addr	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

dev	
channel_id	Channel to enable

Returns

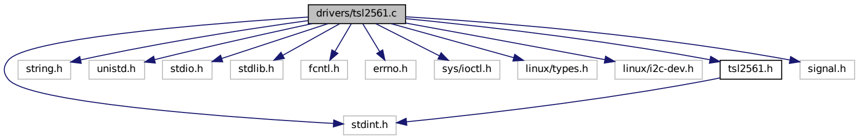
Returns 1 on success, 0 or -1 on error (see write())

5.9 drivers/tsl2561.c File Reference

TSL2561 I2C driver function definitions.

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

Include dependency graph for tsl2561.c:



Functions

- 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.
- 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)

Version

0.1

Date

2020-03-19

Copyright

Copyright (c) 2020

5.9.2 Function Documentation

5.9.2.1 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

fd	
cmd	

Returns

uint16_t

5.9.2.2 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

fd	
reg	Register address

Returns

Byte read over serial

5.9.2.3 tsl2561_destroy()

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

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

Parameters

dev	

5.9.2.4 tsl2561_get_lux()

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

Calculate lux using value measured using tsl2561_measure()

Parameters

measure	

Returns

Lux value

5.9.2.5 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

dev	
s_address	Address for the device, values: 0x29, 0x39, 0x49

Returns

1 on success, -1 on failure

5.9.2.6 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

dev	
measure	Pointer to unsigned 32 bit integer where measurement is stored

5.9.2.7 write16()

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

Write 16 bits to the device (very similar to writecmd8())

Parameters

fd	
val	

5.9.2.8 write8()

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

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

Parameters

fd	
val	

5.9.2.9 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

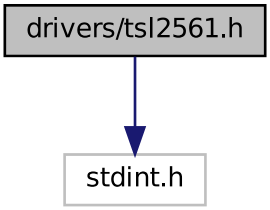
fd	File descriptor
reg	Register address
val	Value to write at register address

5.10 drivers/tsl2561.h File Reference

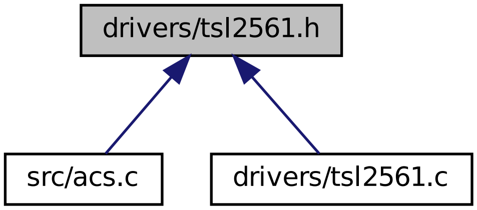
TSL2561 I2C driver function and struct declarations.

```
#include <stdint.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^ LUX_SCALE
- #define TSL2561_LUX_K2T (0x0080)

  0.250 ∗ 2^ RATIO_SCALE
- #define TSL2561_LUX_B2T (0x0214)

  0.0325 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M2T (0x02d1)

  0.0440 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_K3T (0x00c0)

  0.375 ∗ 2^RATIO_SCALE
- #define TSL2561_LUX_B3T (0x023f)

  0.0351 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M3T (0x037b)

  0.0544 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_K4T (0x0100)

  0.50 ∗ 2^RATIO_SCALE
- #define TSL2561_LUX_B4T (0x0270)

  0.0381 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M4T (0x03fe)

  0.0624 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_K5T (0x0138)

  0.61 ∗ 2^RATIO_SCALE
- #define TSL2561_LUX_B5T (0x016f)

  0.0224 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M5T (0x01fc)

  0.0310 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_K6T (0x019a)

  0.80 ∗ 2^RATIO_SCALE
- #define TSL2561_LUX_B6T (0x00d2)

  0.0128 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M6T (0x00fb)

  0.0153 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_K7T (0x029a)

  1.3 ∗ 2^RATIO_SCALE
- #define TSL2561_LUX_B7T (0x0018)

  0.00146 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M7T (0x0012)

  0.00112 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_K8T (0x029a)

  1.3 ∗ 2^RATIO_SCALE
- #define TSL2561_LUX_B8T (0x0000)

  0.000 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M8T (0x0000)

  0.000 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_K1C (0x0043)

  0.130 ∗ 2^RATIO_SCALE
- #define TSL2561_LUX_B1C (0x0204)

  0.0315 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M1C (0x01ad)

  0.0262 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_K2C (0x0085)

  0.260 ∗ 2^RATIO_SCALE

- #define TSL2561_LUX_B2C (0x0228)

  0.0337 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M2C (0x02c1)

  0.0430 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_K3C (0x00c8)

  0.390 ∗ 2^RATIO_SCALE
- #define TSL2561_LUX_B3C (0x0253)

  0.0363 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M3C (0x0363)

  0.0529 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_K4C (0x010a)

  0.520 ∗ 2^RATIO_SCALE
- #define TSL2561_LUX_B4C (0x0282)

  0.0392 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M4C (0x03df)

  0.0605 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_K5C (0x014d)

  0.65 ∗ 2^RATIO_SCALE
- #define TSL2561_LUX_B5C (0x0177)

  0.0229 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M5C (0x01dd)

  0.0291 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_K6C (0x019a)

  0.80 ∗ 2^RATIO_SCALE
- #define TSL2561_LUX_B6C (0x0101)

  0.0157 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M6C (0x0127)

  0.0180 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_K7C (0x029a)

  1.3 ∗ 2^RATIO_SCALE
- #define TSL2561_LUX_B7C (0x0037)

  0.00338 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M7C (0x002b)

  0.00260 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_K8C (0x029a)

  1.3 ∗ 2^RATIO_SCALE
- #define TSL2561_LUX_B8C (0x0000)

  0.000 ∗ 2^LUX_SCALE
- #define TSL2561_LUX_M8C (0x0000)

  0.000 ∗ 2^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_SET {
  TSL2561_REGISTER_CONTROL = 0x00, TSL2561_REGISTER_TIMING = 0x01, TSL2561_REGISTER_THRESHHOLDL_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_CHAN0_HIGH = 0x0D, TSL2561_REGISTER_CHAN1_LOW = 0x0E,
  TSL2561_REGISTER_CHAN1_HIGH = 0x0F }

  TSL2561 I2C Registers.
- enum tsl2561IntegrationTime_t { TSL2561_INTEGRATIONTIME_13MS = 0x00, TSL2561_INTEGRATIONTIME_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.

5.10.1 Detailed Description

TSL2561 I2C driver function and struct declarations.

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.1

Date

2020-03-19

Copyright

Copyright (c) 2020

5.10.2 Enumeration Type Documentation

5.10.2.1 TSL2561_REGISTER_SET

```
enum TSL2561_REGISTER_SET
```

TSL2561 I2C Registers.

Enumerator

TSL2561_REGISTER_CONTROL	Control/power register.
TSL2561_REGISTER_TIMING	Set integration time register.
TSL2561_REGISTER_THRESHHOLDL_LOW	Interrupt low threshold low-byte.
TSL2561_REGISTER_THRESHHOLDL_HIGH	Interrupt low threshold high-byte.
TSL2561_REGISTER_THRESHHOLDH_LOW	Interrupt high threshold low-byte.
TSL2561_REGISTER_THRESHHOLDH_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 tsl2561_destroy()

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

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

Parameters

dev	

5.10.3.2 tsl2561_get_lux()

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

Calculate lux using value measured using tsl2561_measure()

Parameters

measure	

Returns

Lux value

5.10.3.3 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

dev	
s_address	Address for the device, values: 0x29, 0x39, 0x49

Returns

1 on success, -1 on failure

5.10.3.4 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

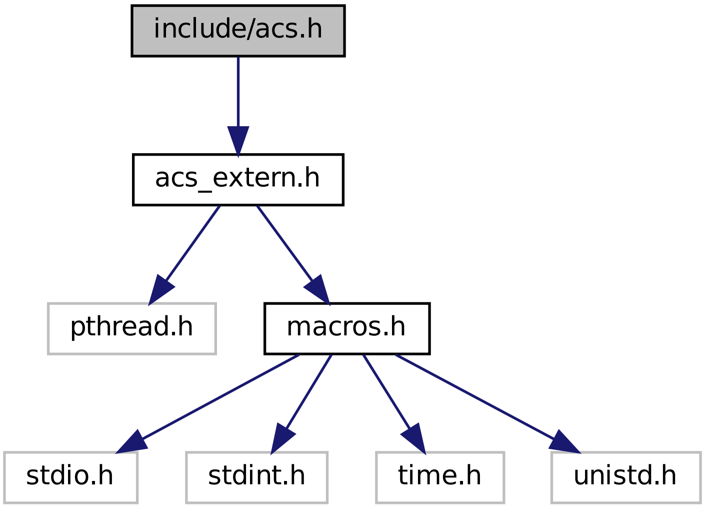
dev	
measure	Pointer to unsigned 32 bit integer where measurement is stored

5.11 include/acs.h File Reference

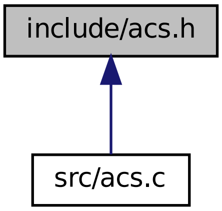
Header file including headers and function prototypes of the Attitude Control System.

```
#include <acs_extern.h>
```

Include dependency graph for acs.h:



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



Macros

- #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 HBRIDGE_ENABLE(name) hbridge_enable(x_##name, y_##name, z_##name);

    Fire magnetorquer in the direction dictated by the input vector.
- #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)

Functions

- int acs_init (void)

    Initializes the devices required to run the attitude control system.
- void ∗ acs_thread (void ∗id)

    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 ω using $\dot{\vec{B}}$ 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)

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

name	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 * id )
```

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

id	Thread ID passed as a pointer to an integer.

Returns

NULL

5.11.3.3 getOmega()

```
void getOmega (
             void  )
```

Calculates ω 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 getSVec()

```
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.

Approximate definition of Pi in case M_PI is not included from math.h

5.11.3.5 HBRIDGE_DISABLE()

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

Disables magnetorquer in the axis indicated by the input.

Parameters

num	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.

5.11.3.6 hbridge_enable()

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

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

Parameters

x	Fires in the +X or -X direction depending on the input being +1 or -1, and does nothing if x = 0
y	Fires in the +Y or -Y direction depending on the input being +1 or -1, and does nothing if y = 0
z	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.

5.11.3.7 insertionSort()

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

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

Parameters

a1	Pointer to integer array to sort.
a2	Pointer to integer array to reorder.

5.11.3.8 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

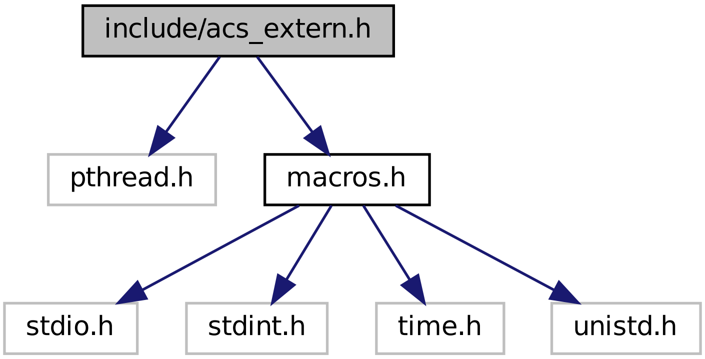
int Returns 1 for success, and -1 for error.

5.12 include/acs_extern.h File Reference

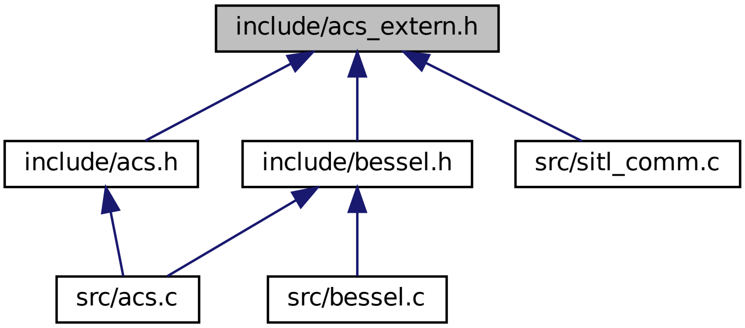
Header file including constants, extern variables and function prototypes that are part of the Attitude Control System, used in other modules.

```
#include <pthread.h>
#include <macros.h>
```

Include dependency graph for acs_extern.h:



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



Macros

- #define SH_BUFFER_SIZE 64

  Circular buffer size for ACS sensor data.

Functions

- **DECLARE_VECTOR2** (g_readB, extern unsigned short)

Variables

- pthread_cond_t data_available

    Condition variable to synchronize ACS and Serial thread in SITL.
- 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.
- volatile int first_run

    This variable is unset by the ACS thread at first execution.

5.12.1 Detailed Description

Header file including constants, extern variables and function prototypes that are part of the Attitude Control System, used in other modules.

Author

Sunip K. Mukherjee (`sunipkmukherjee@gmail.com`)

Version

0.1

Date

2020-03-19

Copyright

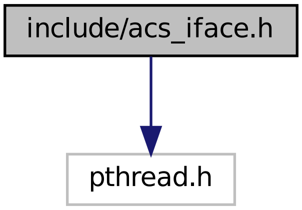
Copyright (c) 2020

5.13 include/acs_iface.h File Reference

Header file including constants, mutexes and function prototypes that initialize, destroy and execute the Attitude Control System module.

```
#include <pthread.h>
```

Include dependency graph for acs_iface.h:



Functions

- int acs_init (void)

    Initializes the devices required to run the attitude control system.
- void acs_destroy (void)

    Powers down ACS devices and closes relevant file descriptors.
- void ∗ acs_thread (void ∗)

    Attitude Control System Thread.

Variables

- pthread_cond_t data_available

    Condition variable to synchronize ACS and Serial thread in SITL.

5.13.1 Detailed Description

Header file including constants, mutexes and function prototypes that initialize, destroy and execute the Attitude Control System module.

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.1

Date

2020-03-19

Copyright

Copyright (c) 2020

5.13.2 Function Documentation

5.13.2.1 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.

5.13.2.2 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

id	Thread ID passed as a pointer to an integer.

Returns

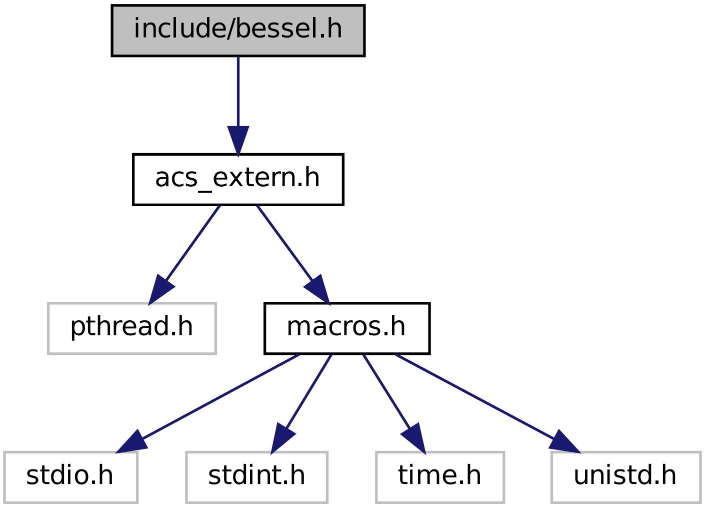
NULL

5.14 include/bessel.h File Reference

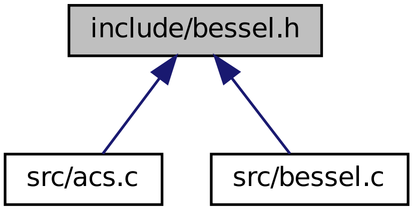
Bessel filter implementation for Attitude Control System.

```
#include <acs_extern.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

  Bessel coefficient minimum value threshold for computation.
- #define BESSEL_FREQ_CUTOFF 5

  Bessel filter cutoff frequency.
- #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().

5.14.1 Detailed Description

Bessel filter implementation for Attitude Control System.

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.1

Date

2020-03-19

Copyright

Copyright (c) 2020

5.14.2 Macro Definition Documentation

5.14.2.1 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

name	Name of the buffer
index	Index of the current value in the buffer

5.14.2.2 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

name	Name of the buffer
index	Index of the current value in the buffer

5.14.3 Function Documentation

5.14.3.1 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

arr	Stores the filter coefficients
size	Size of the filter coefficients array
order	Order of the Bessel filter
freq_cutoff	Cut-off frequency of the Bessel filter

5.14.3.2 dfilterBessel()

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

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

Parameters

arr	Input array
index	Index of current value in the array

Returns

double Filtered value

5.14.3.3 ffilterBessel()

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

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

Parameters

arr	Input array
index	Index of current value in the array

Returns

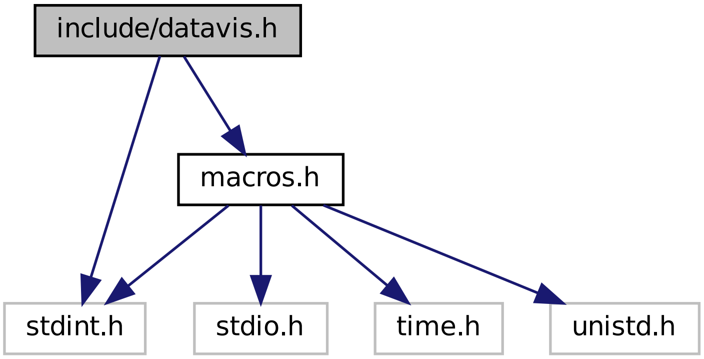
double Filtered value

5.15 include/datavis.h File Reference

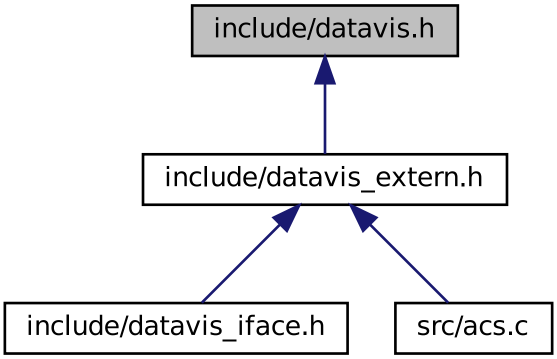
DataVis thread to visualize ACS data over TCP (uses client.py)

```
#include <stdint.h>
#include <macros.h>
```

Include dependency graph for datavis.h:



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



Classes

- struct datavis_p

    Internal data structure of a DataVis packet.

- union data_packet

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

Macros

- #define PORT 12376

    TCP port on which DataVis transmission can be accessed.

- #define PACK_SIZE sizeof(datavis_p)

    Size of the datavis_p struct.

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.

5.15.1 Detailed Description

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

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.1

Date

2020-03-19

Copyright

Copyright (c) 2020

5.15.2 Function Documentation

5.15.2.1 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

t	Pointer to an integer containing the thread ID.

Returns

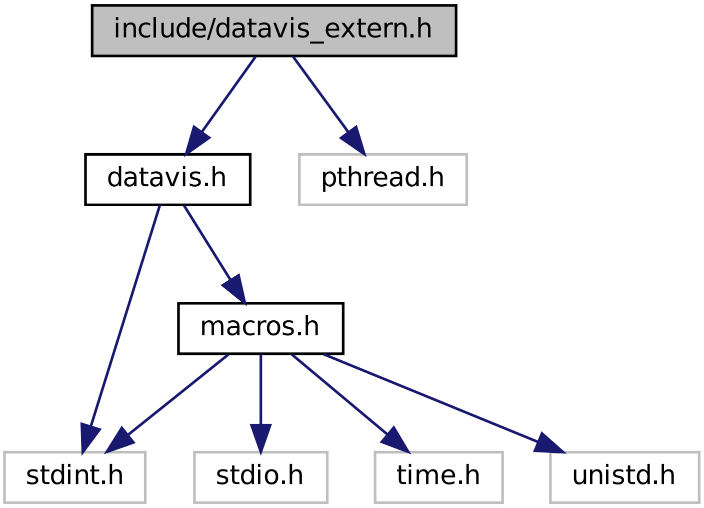
NULL.

5.16 include/datavis_extern.h File Reference

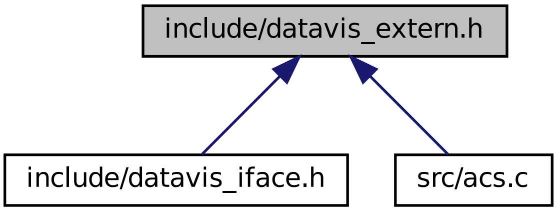
DataVis thread externs for other modules.

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

Include dependency graph for datavis_extern.h:



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



Variables

- data_packet g_datavis_st

  DataVis data structure.
- pthread_cond_t datavis_drdy

  Condition variable used by ACS to signal to DataVis that data is ready.

5.16.1 Detailed Description

DataVis thread externs for other modules.

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Date

2020-03-19

Copyright

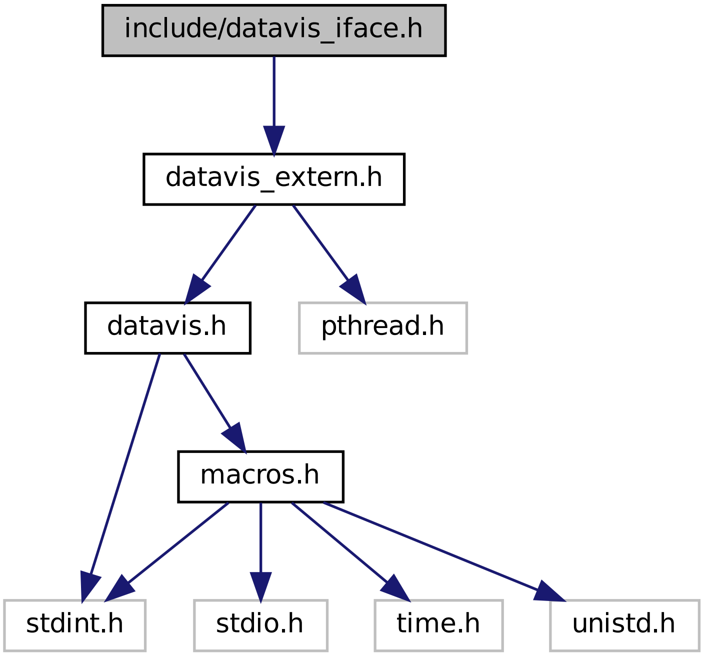
Copyright (c) 2020

5.17 include/datavis_iface.h File Reference

DataVis thread externs for main.

```
#include <datavis_extern.h>
```

Include dependency graph for datavis_iface.h:



Functions

- void ∗ datavis_thread (void ∗)

  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.

5.17.1 Detailed Description

DataVis thread externs for main.

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Date

2020-03-19

Copyright

Copyright (c) 2020

5.17.2 Function Documentation

5.17.2.1 datavis_thread()

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

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

t	Pointer to an integer containing the thread ID.

Returns

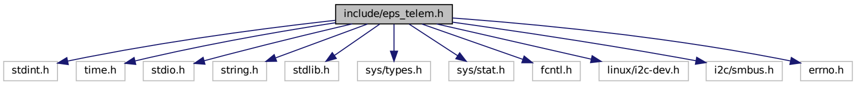
NULL.

5.18 include/eps_telem.h File Reference

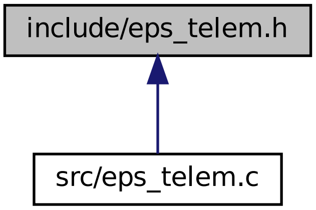
GomSpace P31u I2C interface function prototypes and data structures.

```
#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**

5.18.1 Detailed Description

GomSpace P31u I2C interface function prototypes and data structures.

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.1

Date

2020-03-19

Copyright

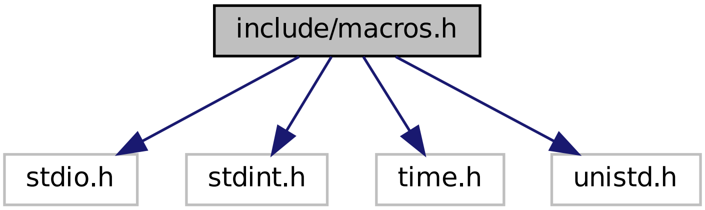
Copyright (c) 2020

5.19 include/macros.h File Reference

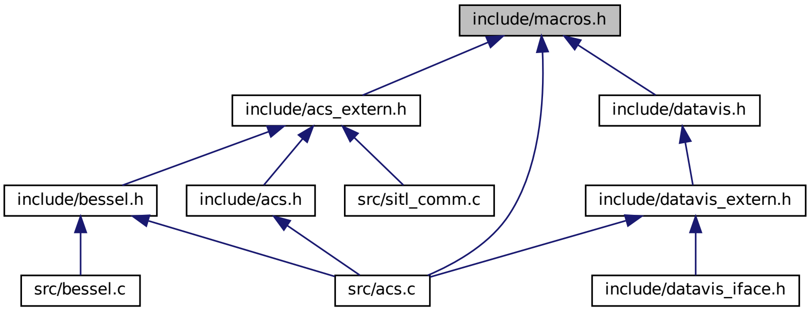
Defines vector macros and other helper functions for the flight software.

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

Include dependency graph for macros.h:



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



Macros

- #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)

  Muliplies 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

- 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.

5.19.1 Detailed Description

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

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.2

Date

2020-03-19

Copyright

Copyright (c) 2020

5.19.2 Macro Definition Documentation

5.19.2.1 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

dest	Destination vector name, declared using DECLARE_VECTOR()
s1	First source vector name, declared using DECLARE_VECTOR()
s2	Second source vector name, declared using DECLARE_VECTOR()

5.19.2.2 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

dest	Output vector, declared using DECLARE_VECTOR()
src	Input buffer, declared using DECLARE_BUFFER()
size	Size of the input buffer (equals to SH_BUFFER_SIZE for a buffer declared using DECLARE_BUFFER())

5.19.2.3 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

name	Name of the buffer (prepends x_, y_, z_ for vector)
type	Data type of the buffer

5.19.2.4 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

name	Name of the vector
type	Data type of the vector

5.19.2.5 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

name	Name of the vector
type	Data type of the vector

5.19.2.6 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

s1	Name of the first vector, declared using DECLARE_VECTOR()
s2	Name of the second vector, declared using DECLARE_VECTOR()

5.19.2.7 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

dest	Output vector, declared using DECLARE_VECTOR()
src	Input buffer, declared using DECLARE_BUFFER()
size	Size of the input buffer (equals to SH_BUFFER_SIZE for a buffer declared using DECLARE_BUFFER())

5.19.2.8 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.

5.19.2.9 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;    \
    g_nightmode = 0;     \
    omega_ready = -1;
```

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

5.19.2.10 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

s	Input vector, declared using DECLARE_VECTOR()

Returns

float Inverse norm of the input vector

5.19.2.11 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
```

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

Parameters

dest	Output vector, declared using DECLARE_VECTOR()
s1	3 x 3 input matrix
s2	Input vector, declared using DECLARE_VECTOR(). Has to be different from the destination.

5.19.2.12 NORM

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

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

Parameters

s	Input vector, declared using DECLARE_VECTOR()

Returns

float Norm of the input vector

5.19.2.13 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

s	Input vector, declared using DECLARE_VECTOR()

Returns

float Square of the norm of the input vector

5.19.2.14 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

dest	Destination vector, declared using DECLARE_VECTOR()
s1	Source vector, declared using DECLARE_VECTOR()

5.19.2.15 VECTOR_CLEAR

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

Value:

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

Clears a vector.

Parameters

name	Name of the vector

5.19.2.16 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

dest	Destination vector, declared using DECLARE_VECTOR()
s1	Input vector, declared using DECLARE_VECTOR()
s2	Input scalar
op	Operation to perform on an element-by-element basis, e.g. +, -, $*$, /. Note: For division there is no check for division by zero.

5.19.2.17 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

dest	Destination vector, declared using DECLARE_VECTOR()
s1	First vector, declared using DECLARE_VECTOR()
s2	Second vector, declared using DECLARE_VECTOR()
op	Operation to perform on an element-by-element basis, e.g. +, -, ∗, /. Note: For division there is no check for division by zero.

5.19.3 Function Documentation

5.19.3.1 daverage()

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

Calculates double precision point average of a float array.

Parameters

arr	Pointer to array whose average is calculated
size	Length of the input array

Returns

double Average of the input array

5.19.3.2 faverage()

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

Calculates floating point average of a float array.

Parameters

arr	Pointer to array whose average is calculated
size	Length of the input array

Returns

float Average of the input array

5.19.3.3 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 ∼18 us on RPi.

Returns

uint64_t Number of microseconds elapsed from epoch.

5.19.3.4 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

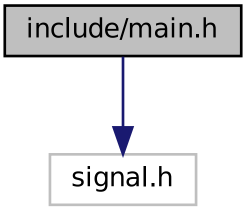
float Inverse square root of the input

5.20 include/main.h File Reference

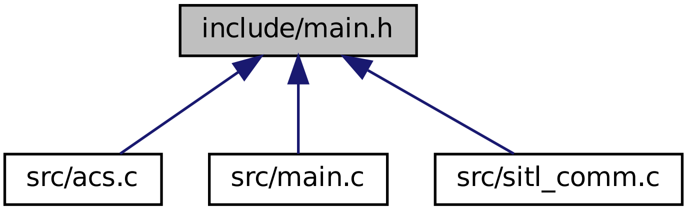
Includes all headers necessary for the core flight software, including ACS, and defines ACS states (which are flight software states), error codes, and relevant error functions.

`#include <signal.h>`

Include dependency graph for main.h:



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



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 sherror (const char ∗)

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

Variables

- __thread int sys_status

  Thread-local system status variable (similar to errno).
- volatile sig_atomic_t done

  Control variable for thread loops.
- int sys_boot_count

  System variable containing the current boot count of the system. This variable is provided to all modules by main.

5.20.1 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)

Version

0.1

Date

2020-03-19

Copyright

Copyright (c) 2020

5.20.2 Function Documentation

5.20.2.1 sherror()

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

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

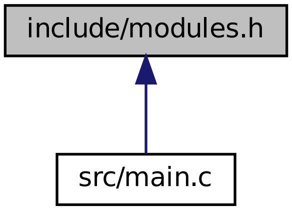
Parameters

msg	Input message to print along with error description

5.21 include/modules.h File Reference

Includes all headers necessary to interface modules with the main program ACS states (which are flight software states), error codes, and relevant error functions.

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



5.21.1 Detailed Description

Includes all headers necessary to interface modules with the main program ACS states (which are flight software states), error codes, and relevant error functions.

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.1

Date

2020-03-19

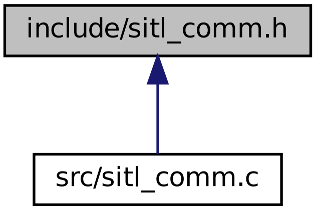
Copyright

Copyright (c) 2020

5.22 include/sitl_comm.h File Reference

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

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



Macros

- #define SITL_COMM_IFACE "/dev/ttyS0"

  File descriptor for SITL comm device.

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.

5.22.1 Detailed Description

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

Author

Sunip K. Mukherjee (`sunipkmukherjee@gmail.com`)

Version

0.2

Date

2020-03-19

Copyright

Copyright (c) 2020

5.22.2 Function Documentation

5.22.2.1 set_blocking()

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

Set the serial device as blocking or non-blocking.

Parameters

fd	Serial device file descriptor
should_block	0 for non-blocking, 1 for blocking mode operation

5.22.2.2 set_interface_attribs()

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

Set speed and parity attributes for the serial device.

Parameters

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

Returns

0 on success, -1 on error

5.22.2.3 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

5.22.2.4 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

id	Pointer to an int that specifies thread ID

Returns

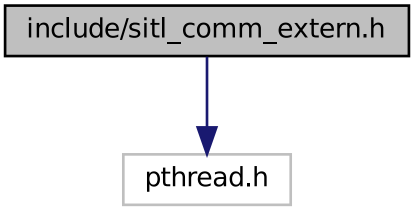
NULL

5.23 include/sitl_comm_extern.h File Reference

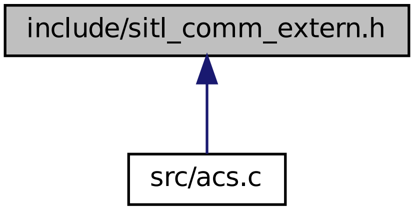
Software-In-The-Loop (SITL) serial communication headers and function prototypes.

```
#include <pthread.h>
```

Include dependency graph for sitl_comm_extern.h:



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



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.
- unsigned long long t_comm

  SITL communication time.
- unsigned long long **comm_time**

5.23.1 Detailed Description

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

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.2

Date

2020-03-19

Copyright

Copyright (c) 2020

5.24 include/sitl_comm_iface.h File Reference

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

Functions

- void * sitl_comm (void *)

  Serial communication thread.

5.24.1 Detailed Description

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

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.2

Date

2020-03-19

Copyright

Copyright (c) 2020

5.24.2 Function Documentation

5.24.2.1 sitl_comm()

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

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

id	Pointer to an int that specifies thread ID

Returns

NULL

5.25 include/uhf.h File Reference

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

Functions

- void ∗ uhf (void ∗id)

  UHF main thread.

5.25.1 Detailed Description

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

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.1

Date

2020-03-19

Copyright

Copyright (c) 2020

5.25.2 Function Documentation

5.25.2.1 uhf()

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

UHF main thread.

Parameters

id	Pointer to integer containing thread ID.

Returns

NULL

5.26 include/xband.h File Reference

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

Functions

- void ∗ xband (void ∗id)

  X-band thread.

5.26.1 Detailed Description

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

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.1

Date

2020-03-19

Copyright

Copyright (c) 2020

5.26.2 Function Documentation

5.26.2.1 xband()

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

X-band thread.

Parameters

id	Pointer to integer containing thread ID

Returns

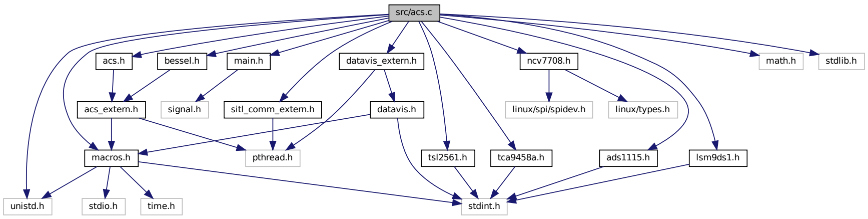
NULL

5.27 src/acs.c File Reference

Attitude Control System related functions.

```
#include <macros.h>
#include <acs.h>
#include <main.h>
#include <bessel.h>
#include <sitl_comm_extern.h>
#include <datavis_extern.h>
#include <ads1115.h>
#include <lsm9ds1.h>
#include <ncv7708.h>
#include <tsl2561.h>
#include <tca9458a.h>
#include <math.h>
#include <stdlib.h>
#include <unistd.h>
```

Include dependency graph for acs.c:



Macros

- #define RST "\x1B[0m"

  This is color indicator for printf statements in ACS, for use in debug only.".
- #define BLK "\x1B[30m"

  black
- #define RED "\x1B[31m"

  red
- #define GRN "\x1B[32m"

  green
- #define YLW "\x1B[33m"

  yellow
- #define BLU "\x1B[34m"

  blue
- #define MGT "\x1B[35m"

  magenta
- #define CYN "\x1B[36m"

  cyan
- #define LGY "\x1B[37m"

  light gray
- #define DGY "\x1B[90m"

  dark gray
- #define LRD "\x1B[91m"

light red

- #define LGR "\x1B[92m"

  light green
- #define LYW "\x1B[93m"

  light yellow
- #define LBU "\x1B[94m"

  light blue
- #define LMT "\x1B[95m"

  light magenta
- #define LCY "\x1B[96m"

  light cyan
- #define WHT "\x1B[97m"

  white
- #define **M_PI** 3.1415

Functions

- DECLARE_VECTOR (g_readB, unsigned short)

  Declares vector to store magnetic field reading from serial.
- DECLARE_BUFFER (g_W, float)

  Creates buffer for $\vec{\omega}$.
- DECLARE_BUFFER (g_B, double)

  Creates buffer for $\vec{B}$.
- DECLARE_BUFFER (g_Bt, double)

  Creates buffer for $\dot{\vec{B}}$.
- DECLARE_VECTOR (g_L_target, float)

  Creates vector for target angular momentum.
- DECLARE_VECTOR (g_W_target, float)

  Creates vector for target angular speed.
- DECLARE_BUFFER (g_S, float)

  Creates buffer for sun vector.
- static void detumbleAction ()

  This function executes the detumble algorithm.
- static void sunpointAction ()

  This function executes the sunpointing algorithm.
- 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 ω *using* $\dot{\vec{B}}$ *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.
- void $*$ acs_thread (void $*$id)

  Attitude Control System Thread.
- void insertionSort (int a1[ ], int a2[ ])

  Sorts the first array and reorders the second array according to the first array.
- int acs_init (void)

  Initializes the devices required to run the attitude control system.
- void acs_destroy (void)

  Powers down ACS devices and closes relevant file descriptors.

Variables

- pthread_cond_t data_available

  Condition variable to synchronize ACS and Serial thread in SITL.
- pthread_mutex_t data_check

  Mutex for locking on data_available.
- 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).
- unsigned long long g_t_acs

  Current timestamp after readSensors() in ACS thread, used to keep track of time taken by ACS loop.

5.27.1 Detailed Description

Attitude Control System related functions.

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.2

Date

2020-07-01

Copyright

Copyright (c) 2020

5.27.2 Macro Definition Documentation

5.27.2.1 RST

```
#define RST "\x1B[0m"
```

This is color indicator for printf statements in ACS, for use in debug only.".

reset to default

5.27.3 Function Documentation

5.27.3.1 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.

5.27.3.2 acs_thread()

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

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

id	Thread ID passed as a pointer to an integer.

Returns

NULL

5.27.3.3 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 $\hat{B} \times L_0 \hat{-} L$, 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 $\vec{\tau} = \vec{\mu} \times \vec{B}$, where $\vec{mu}$ is calculated by multiplying the firing direction vector with the dipole moment of the magnetorquers (0.21 A ·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.

5.27.3.4 getOmega()

```
void getOmega (
             void  )
```

Calculates ω 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.

5.27.3.5 getSVec()

```
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.

Approximate definition of Pi in case M_PI is not included from math.h

5.27.3.6 HBRIDGE_DISABLE()

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

Disables magnetorquer in the axis indicated by the input.

Parameters

num	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.

5.27.3.7 hbridge_enable()

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

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

Parameters

x	Fires in the +X or -X direction depending on the input being +1 or -1, and does nothing if x = 0
y	Fires in the +Y or -Y direction depending on the input being +1 or -1, and does nothing if y = 0
z	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.

5.27.3.8 insertionSort()

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

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

Parameters

a1	Pointer to integer array to sort.
a2	Pointer to integer array to reorder.

5.27.3.9 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.

5.27.3.10 sunpointAction()

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

This function executes the sunpointing algorithm.

The sunpointing algoritm 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.

5.27.4 Variable Documentation

5.27.4.1 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).

5.27.4.2 MOI

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

Initial value:

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

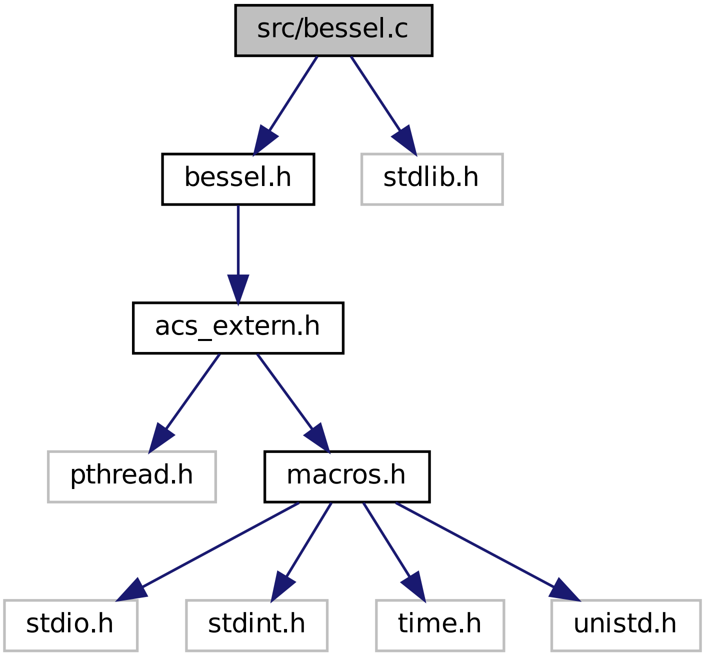
Moment of inertia of the satellite (SI).

5.28 src/bessel.c File Reference

Bessel filter implementation for Attitude Control System.

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

Variables

- float bessel_coeff [SH_BUFFER_SIZE]

  Coefficients for the Bessel filter, calculated using calculateBessel().

5.28.1 Detailed Description

Bessel filter implementation for Attitude Control System.

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.2

Date

2020-03-19

Copyright

Copyright (c) 2020

5.28.2 Function Documentation

5.28.2.1 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

arr	Stores the filter coefficients
size	Size of the filter coefficients array
order	Order of the Bessel filter
freq_cutoff	Cut-off frequency of the Bessel filter

5.28.2.2 dfilterBessel()

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

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

Parameters

arr	Input array
index	Index of current value in the array

Returns

double Filtered value

5.28.2.3 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	Input

Returns

float Factorial of input

5.28.2.4 ffilterBessel()

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

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

Parameters

arr	Input array
index	Index of current value in the array

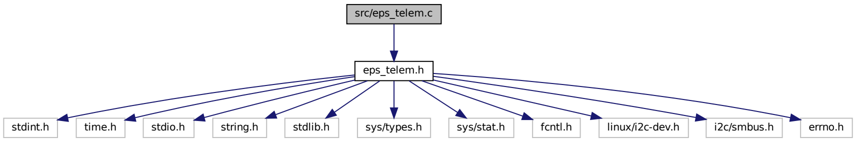
Returns

double Filtered value

5.29 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)

5.29.1 Detailed Description

GomSpace P31u I2C interface function declarations.

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.1

Date

2020-03-19

Copyright

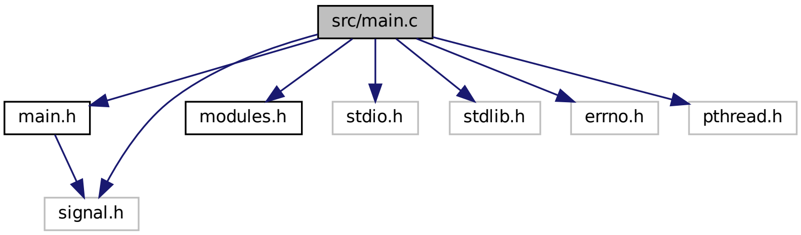
Copyright (c) 2020

5.30 src/main.c File Reference

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

```
#include <main.h>
#include <modules.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <pthread.h>
#include <signal.h>
```

Include dependency graph for main.c:



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.
- int **bootCount** ()

Variables

- int sys_boot_count = -1

  System variable containing the current boot count of the system. This variable is provided to all modules by main.
- volatile sig_atomic_t done = 0

  Control variable for thread loops.
- __thread int sys_status

  Thread-local system status variable (similar to errno).

5.30.1 Detailed Description

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

Author

Sunip K. Mukherjee (`sunipkmukherjee@gmail.com`)

Version

0.2

Date

2020-03-19

Copyright

Copyright (c) 2020

5.30.2 Function Documentation

5.30.2.1 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

sig	Receives the signal as input.

5.30.2.2 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

5.30.2.3 sherror()

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

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

Parameters

msg	Input message to print along with error description

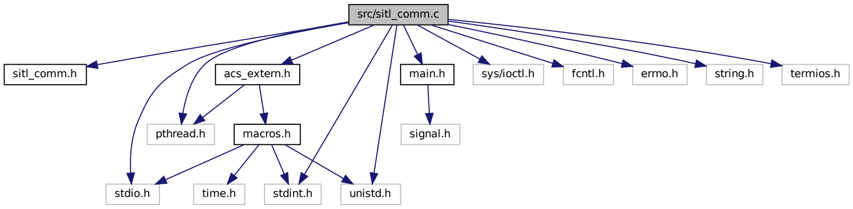
5.31 src/sitl_comm.c File Reference

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

```
#include <sitl_comm.h>
#include <acs_extern.h>
#include <main.h>
#include <stdio.h>
#include <stdint.h>
#include <pthread.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
```

```
#include <string.h>
#include <termios.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.

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.
- unsigned long long t_comm = 0

  SITL communication time.
- unsigned long long **comm_time**

5.31.1 Detailed Description

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

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.2

Date

2020-03-19

Copyright

Copyright (c) 2020

5.31.2 Function Documentation

5.31.2.1 set_blocking()

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

Set the serial device as blocking or non-blocking.

Parameters

fd	Serial device file descriptor
should_block	0 for non-blocking, 1 for blocking mode operation

5.31.2.2 set_interface_attribs()

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

Set speed and parity attributes for the serial device.

Parameters

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

Returns

0 on success, -1 on error

5.31.2.3 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

5.31.2.4 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

id	Pointer to an int that specifies thread ID

Returns

NULL

5.32 src/uhf.c File Reference

UHF interface code.

5.32.1 Detailed Description

UHF interface code.

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.1

Date

2020-03-19

Copyright

Copyright (c) 2020

5.33 src/xband.c File Reference

X-Band Radio interface code.

5.33.1 Detailed Description

X-Band Radio interface code.

Author

Sunip K. Mukherjee (sunipkmukherjee@gmail.com)

Version

0.1

Date

2020-03-19

Copyright

Copyright (c) 2020

Index