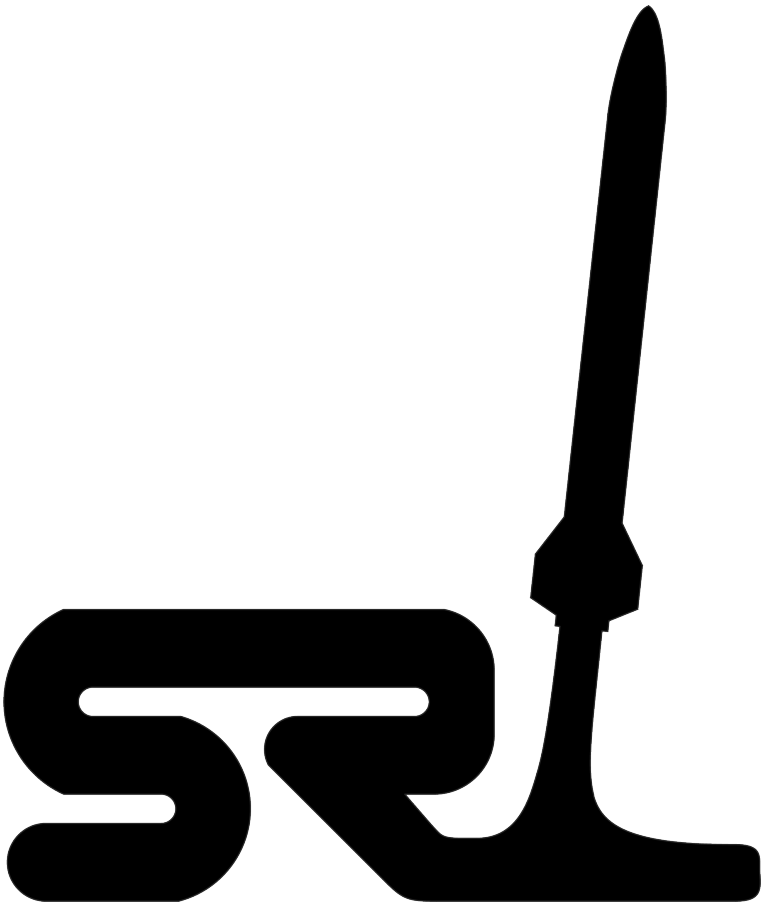


University of Colorado Boulder - Sounding Rocket Laboratory
Avionics Documentation - Rev. A

Jason Popich
Lyon Foster
Chava Friedman
Giselle Koo
Carter Mak

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

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 files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 I2C Namespace Reference

Functions

- bool write_reg (uint8_t i2c, uint8_t addr, uint8_t val)
- bool read_regs (uint8_t i2c, uint8_t addr, uint8_t *data, uint8_t num)
- bool read_regs (uint8_t i2c, uint8_t *data, uint8_t num)

4.1.1 Detailed Description

In this namespace the I2C drivers are declared and then defined for use with the I2C protocol In order to achieve I2C communication, the Arduino Wire library is used to simplify the complexity of the functions.

THESE FUNCTIONS ONLY WORK ON I2C BUS 0 (WIP to work on all buses, simple fix but need to find a way of making it dynamic)

4.1.2 Function Documentation

4.1.2.1 read_regs() [1/2]

```
bool I2C::read_regs (
            uint8_t i2c,
            uint8_t * data,
            uint8_t num )
```

I2C - Second read from registry function that takes in the I2C device address, a data buffer to write to, and the amount of bytes to read. The difference with the first read_regs function is that this function does not request data from a specific register on the device

Parameters

i2c	I2C Device Address
data	The data buffer you will place incoming data into for processing
num	The number of bytes you are grabbing from the device.

Definition at line 94 of file Namespaces.cpp.

4.1.2.2 read_regs() [2/2]

```
bool I2C::read_regs (
            uint8_t i2c,
            uint8_t addr,
            uint8_t * data,
            uint8_t num )
```

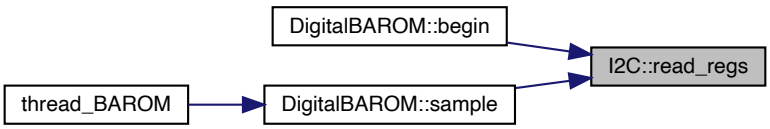
I2C - First read from registry function that takes in the I2C device address, Device Registry Address to read from, a data buffer to write to, and the amount of bytes to read. The difference with the second read_regs function is that this function requests data from a specific register on the device

Parameters

i2c	I2C Device Address
addr	I2C Device Registry Address you are attempting to read from
data	The data buffer you will place incoming data into for processing
num	The number of bytes you are grabbing from the device.

Definition at line 72 of file Namespaces.cpp.

Here is the caller graph for this function:



4.1.2.3 write_reg()

```
bool I2C::write_reg (
            uint8_t i2c,
```

```
uint8_t addr,
uint8_t val )
```

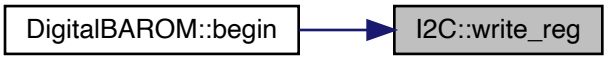
I2C - Write to registry function that takes in the I2C device address, Device Registry Address to write to, and the data to write

Parameters

i2c	I2C Device Address
addr	I2C Device Registry Address you are attempting to write to
val	The data you are trying to write

Definition at line 63 of file Namespaces.cpp.

Here is the caller graph for this function:



4.2 INITS Namespace Reference

Variables

- int speakerPin = 2
- int highG_xPin = 33
- int highG_yPin = 34
- int highG_zPin = 35
- DigitalIMU IMU = DigitalIMU(55,0x28)
- DigitalBAROM BAROM
- HIGHG_ACCEL HIGHG = HIGHG_ACCEL(highG_xPin,highG_yPin,highG_zPin,true)
- BeepyBOI berp = BeepyBOI(speakerPin)
- DLLflash ∗ flash = nullptr
- IMUdata imu_data
- BAROMdata barom_data
- ACCELdata accel_data

4.2.1 Detailed Description

All pointers and objects that are required in the main file are put within the INITS namespace to avoid global variables and their implications

4.2.2 Variable Documentation

4.2.2.1 accel_data

```
ACCELdata INITS::accel_data
```

The struct ACCELdata object, or Instance, that holds all the HIGHG Accelerometer data for processing and transmission

Definition at line 42 of file Namespaces.cpp.

4.2.2.2 BAROM

```
DigitalBAROM INITS::BAROM
```

The DigitalBAROM class object, that will be initialized for the MPL3115A2 Barometer

Definition at line 34 of file Namespaces.cpp.

4.2.2.3 barom_data

```
BAROMdata INITS::barom_data
```

The struct BAROMdata object, or Instance, that holds all BAROM data for processing and transmission

Definition at line 41 of file Namespaces.cpp.

4.2.2.4 berp

```
BeepyBOI INITS::berp = BeepyBOI(speakerPin)
```

The BeepyBOI class object, that will be initialized for the Piezo Buzzer

Definition at line 36 of file Namespaces.cpp.

4.2.2.5 flash

```
DLLflash * INITS::flash = nullptr
```

The DLLflash pointer that will point to the DLLflash class instance

Definition at line 38 of file Namespaces.cpp.

4.2.2.6 HIGHG

```
HIGHG_ACCEL INITS::HIGHG = HIGHG_ACCEL(highG_xPin,highG_yPin,highG_zPin,true)
```

The HIGHG_ACCEL class object, that will be initialized for the ADXL377 High-G Accelerometer

Definition at line 35 of file Namespaces.cpp.

4.2.2.7 highG_xPin

```
int INITS::highG_xPin = 33
```

The High-G Accelerometer X Pin Assignment

Definition at line 29 of file Namespaces.cpp.

4.2.2.8 highG_yPin

```
int INITS::highG_yPin = 34
```

The High-G Accelerometer Y Pin Assignment

Definition at line 30 of file Namespaces.cpp.

4.2.2.9 highG_zPin

```
int INITS::highG_zPin = 35
```

The High-G Accelerometer Z Pin Assignment

Definition at line 31 of file Namespaces.cpp.

4.2.2.10 IMU

```
DigitalIMU INITS::IMU = DigitalIMU(55,0x28)
```

The DigitalIMU class object, that will be initialized for the BNO055 IMU

Definition at line 33 of file Namespaces.cpp.

4.2.2.11 imu_data

```
IMUdata INITS::imu_data
```

The struct IMUData object, or Instance, that holds all IMU data for processing and transmission

Definition at line 40 of file Namespaces.cpp.

4.2.2.12 speakerPin

```
int INITS::speakerPin = 2
```

The Piezo Buzzer pin

Definition at line 28 of file Namespaces.cpp.

4.3 PROTOTHREADING Namespace Reference

Variables

- int interval_IMU = 45
- int interval_BAROM = 2000
- int interval_ACCEL = 50
- ThreadController thread_control = ThreadController()
- Thread * ThreadIMU = new Thread()
- Thread * ThreadBAROM = new Thread()
- Thread * ThreadACCEL = new Thread()

4.3.1 Detailed Description

The protothreading system is implemented by the ArduinoThread library

The Teensy 3.6 / 4.0 micro-controllers are one core, one thread therefore true asynchronous operation cannot be accomplished This is unfortunate because optimally all sampling of the data should be done at the same time Therefore in order to work around this limitation, a form of threading was introduced that approaches asynchronous operation without actually achieving it, aka protothreading

The way it works is essentially by having a(n) overall controller (i.e. ThreadController class) that manages the timing of all the functions that you want to run, that timing keeps everything running in an orderly fashion. Whenever a function has reached the time it needs to be called again the ThreadController will call the function and interrupt whatever is currently running in order to keep order in the system.

The pro about this is that it also allows us to deal with different intervals that sensors or components require Such as one component needing to be called more frequently than the other... Every process interval is based off of the datasheet and its recommendations on sampling time

4.3.2 Variable Documentation

4.3.2.1 interval_ACCEL

```
int PROTOTHREADING::interval_ACCEL = 50
```

The inverval at which the High-G Accelerometer will refresh

Definition at line 50 of file Namespaces.cpp.

4.3.2.2 interval_BAROM

```
int PROTOTHREADING::interval_BAROM = 2000
```

The interval at which the Barometer will refresh

Definition at line 49 of file Namespaces.cpp.

4.3.2.3 interval_IMU

```
int PROTOTHREADING::interval_IMU = 45
```

The interval at which the IMU will refresh

Definition at line 48 of file Namespaces.cpp.

4.3.2.4 thread_control

```
ThreadController PROTOTHREADING::thread_control = ThreadController()
```

thread_control is the overarching ThreadController that handles all the timing and calling of threads

Definition at line 52 of file Namespaces.cpp.

4.3.2.5 ThreadACCEL

```
Thread * PROTOTHREADING::ThreadACCEL = new Thread()
```

The pointer that will point to the instance of the Thread for the High-G Accelerometer

Definition at line 56 of file Namespaces.cpp.

4.3.2.6 ThreadBAROM

```
Thread * PROTOTHREADING::ThreadBAROM = new Thread()
```

The pointer that will point to the instance of the Thread for the Barometeer

Definition at line 55 of file Namespaces.cpp.

4.3.2.7 ThreadIMU

```
Thread * PROTOTHREADING::ThreadIMU = new Thread()
```

The pointer that will point to the instance of the Thread for IMU

Definition at line 54 of file Namespaces.cpp.

Chapter 5

Class Documentation

5.1 ACCELdata Struct Reference

HIGH-G Accelerometer Struct.

`#include <yonics.hpp>`

Public Attributes

- float x
- float y
- float z
- uint32_t t

5.1.1 Detailed Description

HIGH-G Accelerometer Struct.

This structs holds the ADXL377 sample at a point in time to be stored and processed.

Definition at line 31 of file yonics.hpp.

5.1.2 Member Data Documentation

5.1.2.1 t

`uint32_t ACCELdata::t`

Time

Definition at line 42 of file yonics.hpp.

5.1.2.2 x

```
float ACCELdata::x
```

Acceleration in X axis

Definition at line 33 of file yonics.hpp.

5.1.2.3 y

```
float ACCELdata::y
```

Acceleration in Y axis

Definition at line 36 of file yonics.hpp.

5.1.2.4 z

```
float ACCELdata::z
```

Acceleration in Z axis

Definition at line 39 of file yonics.hpp.

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

- src/src/yonics.hpp

5.2 BAROMdata Struct Reference

Barometer Struct.

```
#include <yonics.hpp>
```

Public Attributes

- float pressure = 0
- float altitude = 0
- float temperature = 0
- uint32_t t = 0

5.2.1 Detailed Description

Barometer Struct.

This structs holds the MPL3115A2 sample at a point in time to be stored and processed.

Definition at line 79 of file yonics.hpp.

5.2.2 Member Data Documentation

5.2.2.1 altitude

```
float BAROMdata::altitude = 0
```

MPL3115A2 Altitude

Definition at line 84 of file yonics.hpp.

5.2.2.2 pressure

```
float BAROMdata::pressure = 0
```

MPL3115A2 Barometric Pressure

Definition at line 81 of file yonics.hpp.

5.2.2.3 t

```
uint32_t BAROMdata::t = 0
```

Time

Definition at line 90 of file yonics.hpp.

5.2.2.4 temperature

```
float BAROMdata::temperature = 0
```

MPL3115A2 temperature in C

Definition at line 87 of file yonics.hpp.

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

- src/src/yonics.hpp

5.3 BeepyBOI Class Reference

Piezo Buzzer Class.

`#include <yonics.hpp>`

Public Member Functions

- BeepyBOI ()
- BeepyBOI (int pin)
- void hello ()
- void error ()
- void countdown (int s)
- void lowBeep ()
- void midBeep ()
- void hiBeep ()
- void bombBeep ()

5.3.1 Detailed Description

Piezo Buzzer Class.

This class handles all the Piezo Buzzer interactions. The Buzzer will sound certain noises to indicate errors, startup, etc...

Definition at line 185 of file yonics.hpp.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 BeepyBOI() [1/2]

```
BeepyBOI::BeepyBOI ( )
```

BeepyBOI Default Constructor

Definition at line 7 of file BeepyBOI.cpp.

5.3.2.2 BeepyBOI() [2/2]

```
BeepyBOI::BeepyBOI (
            int pin )
```

BeepyBOI Constructor

Definition at line 11 of file BeepyBOI.cpp.

5.3.3 Member Function Documentation

5.3.3.1 bombBeep()

```
void BeepyBOI::bombBeep ( )
```

Definition at line 47 of file BeepyBOI.cpp.

Here is the caller graph for this function:

5.3.3.2 countdown()

```
void BeepyBOI::countdown (
            int s )
```

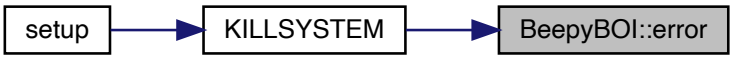
Definition at line 25 of file BeepyBOI.cpp.

5.3.3.3 error()

```
void BeepyBOI::error ( )
```

Definition at line 20 of file BeepyBOI.cpp.

Here is the caller graph for this function:

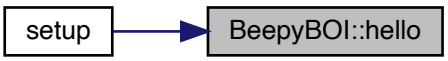


5.3.3.4 hello()

```
void BeepyBOI::hello ( )
```

Definition at line 15 of file BeepyBOI.cpp.

Here is the caller graph for this function:



5.3.3.5 hiBeep()

```
void BeepyBOI::hiBeep ( )
```

Definition at line 42 of file BeepyBOI.cpp.

5.3.3.6 lowBeep()

```
void BeepyBOI::lowBeep ( )
```

Definition at line 32 of file BeepyBOI.cpp.

5.3.3.7 midBeep()

```
void BeepyBOI::midBeep ( )
```

Definition at line 37 of file BeepyBOI.cpp.

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

- src/src/yonics.hpp
- src/src/BeepyBOI.cpp

5.4 DigitalBAROM Class Reference

MPL3115A2 Barometer Class.

```
#include <yonics.hpp>
```

Public Member Functions

- DigitalBAROM ()
- bool begin ()
- bool sample (BAROMdata *data)

  DigitalBAROM Sample Function.

5.4.1 Detailed Description

MPL3115A2 Barometer Class.

Class to manage the MPL3115A2 I2C breakout board

Definition at line 162 of file yonics.hpp.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 DigitalBAROM()

```
DigitalBAROM::DigitalBAROM ( )
```

DigitalBAROM Default Constructor

Definition at line 7 of file DigitalBAROM.cpp.

5.4.3 Member Function Documentation

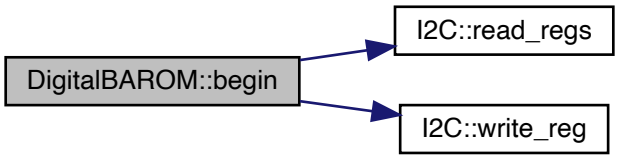
5.4.3.1 begin()

```
bool DigitalBAROM::begin ( )
```

Function that initializes the MPL3115A2 Barometer

Definition at line 9 of file DigitalBAROM.cpp.

Here is the call graph for this function:



5.4.3.2 sample()

```
bool DigitalBAROM::sample (
            BAROMdata * data )
```

DigitalBAROM Sample Function.

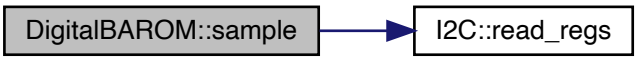
A function that will called from the threadBAROM to sample from the MPL3115A2 Barometric Sensor

Parameters

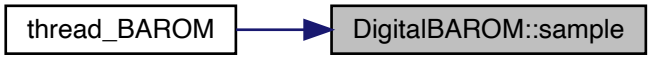
data	The pointer for the BAROMdata struct

Definition at line 32 of file DigitalBAROM.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- src/src/yonics.hpp
- src/src/DigitalBAROM.cpp

5.5 DigitalIMU Class Reference

BNO055 IMU Class.

`#include <yonics.hpp>`

Public Member Functions

- DigitalIMU ()
- DigitalIMU (int32_t sensorID, uint8_t address)
- bool begin ()
- void sample (IMUdata *data)

  DigitalIMU Sample Function.

5.5.1 Detailed Description

BNO055 IMU Class.

Class to manage the Adafruit BNO055 Absolute Orientation IMU Fusion breakout board KEEP IN MIND NO DIF↩ FERENCE BETWEEN RAW AND FUSED DATA FROM BNO055...

Definition at line 132 of file yonics.hpp.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 DigitalIMU() [1/2]

```
DigitalIMU::DigitalIMU ( )
```

DigitalIMU Default Constructor

Definition at line 7 of file DigitalIMU.cpp.

5.5.2.2 DigitalIMU() [2/2]

```
DigitalIMU::DigitalIMU (
            int32_t sensorID,
            uint8_t address )
```

DigitalIMU Constructor with arguments for sensorID and address per the library

Definition at line 11 of file DigitalIMU.cpp.

5.5.3 Member Function Documentation

5.5.3.1 begin()

```
bool DigitalIMU::begin ( )
```

Function that initializes the BNO055 IMU

Definition at line 15 of file DigitalIMU.cpp.

5.5.3.2 sample()

```
void DigitalIMU::sample (
            IMUdata * data )
```

DigitalIMU Sample Function.

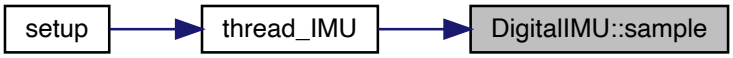
A function that will called from the threadIMU to sample from the BNO055 IMU

Parameters

data	The pointer for the IMUdata struct

Definition at line 25 of file DigitalIMU.cpp.

Here is the caller graph for this function:



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

- src/src/yonics.hpp
- src/src/DigitalIMU.cpp

5.6 DLLflash Class Reference

`#include <DLLflash.hpp>`

Public Member Functions

- DLLflash ()
- DLLflash (int)
- ~DLLflash ()
- template<class T >
  void addType (T *, char *)
- bool writeSample (char *)
- void setToWrite ()
- void setToRead ()

5.6.1 Detailed Description

Definition at line 57 of file DLLflash.hpp.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 DLLflash() [1/2]

`DLLflash::DLLflash ( )`

5.6.2.2 DLLflash() [2/2]

```
DLLflash::DLLflash (
            int flashpin )
```

Definition at line 105 of file DLLflash.cpp.

5.6.2.3 ~DLLflash()

```
DLLflash::~DLLflash ( )
```

Definition at line 124 of file DLLflash.cpp.

5.6.3 Member Function Documentation

5.6.3.1 addType()

```
template<class T >
void DLLflash::addType (
            T * data,
            char * id )
```

Definition at line 133 of file DLLflash.cpp.

5.6.3.2 setToRead()

```
void DLLflash::setToRead ( )
```

5.6.3.3 setToWrite()

```
void DLLflash::setToWrite ( )
```

5.6.3.4 writeSample()

```
bool DLLflash::writeSample (
            char *  )
```

Definition at line 142 of file DLLflash.cpp.

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

- src/src/DLLflash.hpp
- src/src/DLLflash.cpp

5.7 DLLtype Class Reference

```
#include <DLLflash.hpp>
```

Public Member Functions

- DLLtype (void *, int, char *)
- ~DLLtype ()
- bool setType (void *, int)
- bool writeSample (uint32_t, SPIFlash *)
- bool readSample ()
- char * getID ()

5.7.1 Detailed Description

Definition at line 18 of file DLLflash.hpp.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 DLLtype()

```
DLLtype::DLLtype (
            void * dataPtr,
            int dataSize,
            char * id )
```

Definition at line 8 of file DLLflash.cpp.

5.7.2.2 ~DLLtype()

```
DLLtype::~DLLtype ( )
```

Definition at line 23 of file DLLflash.cpp.

5.7.3 Member Function Documentation

5.7.3.1 getID()

```
char * DLLtype::getID ( )
```

Definition at line 101 of file DLLflash.cpp.

5.7.3.2 readSample()

```
bool DLLtype::readSample ( )
```

5.7.3.3 setType()

```
bool DLLtype::setType (
            void * dataPtr,
            int dataSize )
```

Definition at line 57 of file DLLflash.cpp.

5.7.3.4 writeSample()

```
bool DLLtype::writeSample (
            uint32_t next,
            SPIFlash * flash )
```

Definition at line 68 of file DLLflash.cpp.

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

- src/src/DLLflash.hpp
- src/src/DLLflash.cpp

5.8 HIGHG_ACCEL Class Reference

ADXL377 High-G Accelerometer Class.

`#include <yonics.hpp>`

Public Member Functions

- HIGHG_ACCEL ()
- HIGHG_ACCEL (int xPin, int yPin, int zPin)
- HIGHG_ACCEL (int xPin, int yPin, int zPin, bool highBitDepth)
- void sample (ACCELdata *data)

  HIGHG_ACCEL Sample Function.

5.8.1 Detailed Description

ADXL377 High-G Accelerometer Class.

Class to manage the Adafruit ADXL377 High-G Accelerometer breakout board

Definition at line 97 of file yonics.hpp.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 HIGHG_ACCEL() [1/3]

```
HIGHG_ACCEL::HIGHG_ACCEL ( )
```

HIGHG_ACCEL Default Constructor

Definition at line 7 of file AnalogIMU.cpp.

5.8.2.2 HIGHG_ACCEL() [2/3]

```
HIGHG_ACCEL::HIGHG_ACCEL (
            int xPin,
            int yPin,
            int zPin )
```

HIGHG_ACCEL with arguments for pin assignments of the HIGHG_ACCEL

Definition at line 16 of file AnalogIMU.cpp.

5.8.2.3 HIGHG_ACCEL() [3/3]

```
HIGHG_ACCEL::HIGHG_ACCEL (
            int xPin,
            int yPin,
            int zPin,
            bool highBitDepth )
```

HIGHG_ACCEL with arguments for pin assignments of the HIGHG_ACCEL and bitDepth for analog input

Definition at line 29 of file AnalogIMU.cpp.

5.8.3 Member Function Documentation

5.8.3.1 sample()

```
void HIGHG_ACCEL::sample (
            ACCELdata * data )
```

HIGHG_ACCEL Sample Function.

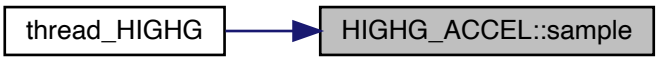
A function that will called from the threadACCEL to sample from the ADXL377 High G Accelerometer

Parameters

data	The pointer for the ACCELdata struct

Definition at line 50 of file AnalogIMU.cpp.

Here is the caller graph for this function:



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

- src/src/yonics.hpp
- src/src/AnalogIMU.cpp

5.9 IMUdata Struct Reference

IMU Struct.

`#include <yonics.hpp>`

Public Attributes

- double orient_euler [3] = {0,0,0}
- double gyro_fused [3] = {0,0,0}
- double accel_fused [3] = {0,0,0}
- double accel_raw [3] = {0,0,0}
- double gyro_raw [3] = {0,0,0}
- double magnetometer [3] = {0,0,0}
- double orient_quat [4] = {0,0,0,0}
- uint32_t t = 0

5.9.1 Detailed Description

IMU Struct.

This structs holds the BNO055 sample at a point in time to be stored and processed.

Definition at line 49 of file yonics.hpp.

5.9.2 Member Data Documentation

5.9.2.1 accel_fused

```
double IMUdata::accel_fused[3] = {0,0,0}
```

Fused Accel Data {x,y,z}

Definition at line 57 of file yonics.hpp.

5.9.2.2 accel_raw

```
double IMUdata::accel_raw[3] = {0,0,0}
```

Raw Accel Data {x,y,z}

Definition at line 60 of file yonics.hpp.

5.9.2.3 gyro_fused

```
double IMUdata::gyro_fused[3] = {0,0,0}
```

Fused Gyro Data {x,y,z}

Definition at line 54 of file yonics.hpp.

5.9.2.4 gyro_raw

```
double IMUdata::gyro_raw[3] = {0,0,0}
```

Raw Gyro Data {x,y,z}

Definition at line 63 of file yonics.hpp.

5.9.2.5 magnetometer

```
double IMUdata::magnetometer[3] = {0,0,0}
```

Magnetometer Data {x,y,z}

Definition at line 66 of file yonics.hpp.

5.9.2.6 orient_euler

```
double IMUdata::orient_euler[3] = {0,0,0}
```

Orientation in Euler {x,y,z}

Definition at line 51 of file yonics.hpp.

5.9.2.7 orient_quat

```
double IMUdata::orient_quat[4] = {0,0,0,0}
```

Orientation in Quaternions {w,x,y,z}

Definition at line 69 of file yonics.hpp.

5.9.2.8 t

```
uint32_t IMUdata::t = 0
```

Time

Definition at line 72 of file yonics.hpp.

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

- src/src/yonics.hpp

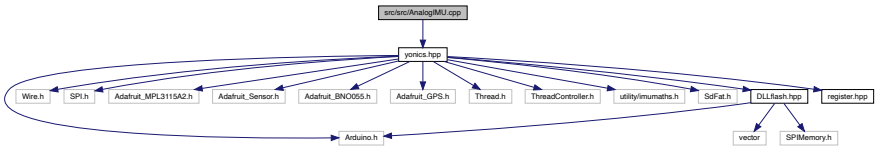
Chapter 6

File Documentation

6.1 src/src/AnalogIMU.cpp File Reference

The main source file for the HIGHG_ACCEL class.

`#include "yonics.hpp"`
Include dependency graph for AnalogIMU.cpp:



6.1.1 Detailed Description

The main source file for the HIGHG_ACCEL class.

Definition in file AnalogIMU.cpp.

6.2 AnalogIMU.cpp

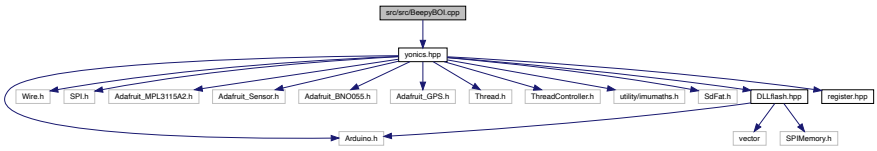
```
00001
00005 #include "yonics.hpp"
00006
00007 HIGHG_ACCEL::HIGHG_ACCEL() {
00008     xPin = 0;
00009     yPin = 1;
00010     zPin = 2;
00011     bitDepth = 10;
00012
00013     init();
00014 }
00015
00016 HIGHG_ACCEL::HIGHG_ACCEL(int xPin, int yPin, int zPin) {
00017     // Define the x y z pin assignments and set the class members
00018     this->xPin = xPin;
00019     this->yPin = yPin;
00020     this->zPin = zPin;
00021
00022     // Define the bit Depth at 10
```

```
00023     bitDepth = 10;
00024
00025     // Set offset and ratio based on bitDepth
00026     init();
00027 }
00028
00029 HIGHG_ACCEL::HIGHG_ACCEL(int xPin, int yPin, int zPin, bool highBitDepth) {
00030     // Define the x y z pin assignments and set the class members
00031     this->xPin = xPin;
00032     this->yPin = yPin;
00033     this->zPin = zPin;
00034
00035     // Define the bit depth at 16
00036     bitDepth = 16;
00037
00038     // If highBitDepth set the analog read resolution to 16 (MAX)
00039     if (highBitDepth) {analogReadRes(16);}
00040
00041     // Set offset and ratio based on bitDepth
00042     init();
00043 }
00044
00045 void HIGHG_ACCEL::init() {
00046     offset = 0.5*(2^bitDepth);
00047     ratio = (float)offset/(float)(2*maxG);
00048 }
00049
00050 void HIGHG_ACCEL::sample(ACCELdata* data) {
00051     data->t = millis();
00052     data->x = formatVal(analogRead(xPin));
00053     data->y = formatVal(analogRead(yPin));
00054     data->z = formatVal(analogRead(zPin));
00055
00056
00057     /*Serial.print(formatVal(analogRead(xPin)));
00058     Serial.print(" ");
00059     Serial.print(formatVal(analogRead(yPin)));
00060     Serial.print(" ");
00061     Serial.print(formatVal(analogRead(zPin)));
00062     Serial.println(" ");*/
00063 }
00064
00065 float HIGHG_ACCEL::formatVal(int rawVal) {
00066     rawVal = rawVal - offset;
00067     return (float)rawVal*ratio;
00068 }
```

6.3 src/src/BeepyBOI.cpp File Reference

The main source file for the BeepyBOI class.

`#include "yonics.hpp"`
Include dependency graph for BeepyBOI.cpp:



6.3.1 Detailed Description

The main source file for the BeepyBOI class.

Definition in file BeepyBOI.cpp.

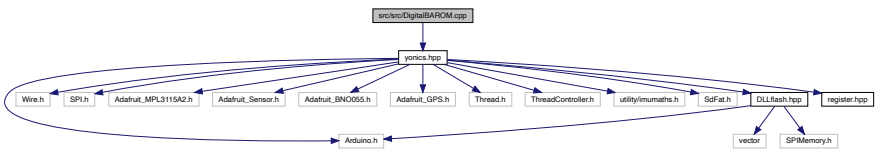
6.4 BeepyBOI.cpp

```
00001
00005 #include "yonics.hpp"
00006
00007 BeepyBOI::BeepyBOI() {
00008     pin = 2;
00009 }
00010
00011 BeepyBOI::BeepyBOI(int pin) {
00012     this->pin = pin;
00013 }
00014
00015 void BeepyBOI::hello() {
00016     tone(pin,midTone,200); // hehe concert A
00017     delay(1000);
00018 }
00019
00020 void BeepyBOI::error() {
00021     tone(pin,errTone,4000);
00022     delay(5000);
00023 }
00024
00025 void BeepyBOI::countdown(int s) {
00026     for(int i=0;i<s;i++) {
00027         tone(pin,midTone,20);
00028         delay(1000);
00029     }
00030 }
00031
00032 void BeepyBOI::lowBeep() {
00033     tone(pin,lowTone,500);
00034     delay(500);
00035 }
00036
00037 void BeepyBOI::midBeep() {
00038     tone(pin,midTone,500);
00039     delay(500);
00040 }
00041
00042 void BeepyBOI::hiBeep() {
00043     tone(pin,hiTone,500);
00044     delay(500);
00045 }
00046
00047 void BeepyBOI::bombBeep() {
00048     for(int i=0;i<25;i++) {
00049         tone(pin,hiTone,25);
00050         delay(50);
00051     }
00052     delay(250);
00053     tone(pin,2*hiTone,250);
00054     delay(250);
00055 }
```

6.5 src/src/DigitalBAROM.cpp File Reference

The main source file for the DigitalBAROM class.

`#include "yonics.hpp"`
Include dependency graph for DigitalBAROM.cpp:



6.5.1 Detailed Description

The main source file for the DigitalBAROM class.

Definition in file DigitalBAROM.cpp.

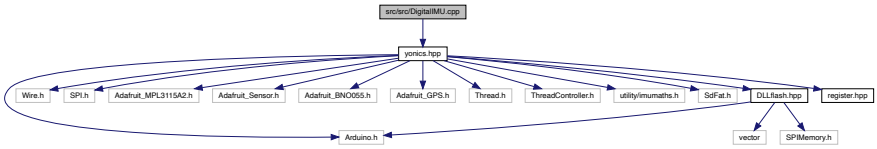
6.6 DigitalBAROM.cpp

```
00001
00005 #include "yonics.hpp"
00006
00007 DigitalBAROM::DigitalBAROM() {}
00008
00009 bool DigitalBAROM::begin()
00010 {
00011     const uint8_t i2c_addr = MPL3115_I2C_ADDR;
00012     uint8_t b;
00013
00014     // Make sure we are talking to the correct device
00015     if (!I2C::read_regs(i2c_addr, MPL3115_WHO_AM_I, &b, 1)) return false;
00016     if (b != 0xC4) return false;
00017
00018     // place into standby mode
00019     if (!I2C::write_reg(i2c_addr, MPL3115_CTRL_REG1, 0)) return false;
00020
00021     // switch to 34ms
00022     if (!I2C::write_reg(i2c_addr, MPL3115_CTRL_REG1, 0x98)) return false;
00023
00024     // switch to active, set altimeter mode, set polling mode
00025     if (!I2C::write_reg(i2c_addr, MPL3115_CTRL_REG1, 0xB9)) return false;
00026
00027     // enable events
00028     if (!I2C::write_reg(i2c_addr, MPL3115_PT_DATA_CFG, 0x07)) return false;
00029     return true;
00030 }
00031
00032 bool DigitalBAROM::sample(BAROMdata* data) {
00033
00034     static elapsedMicros usec_since;
00035     static int32_t usec_history = 980000;
00036     const uint8_t i2c_addr = MPL3115_I2C_ADDR;
00037     uint8_t buf[6];
00038
00039     // KEEPS TIME BECAUSE MPL3115A2 times out after 512ms
00040     int32_t usec = usec_since;
00041     if (usec + 500 < usec_history) return false;
00042
00043     // GET THE DATA FROM THE STATUS REGISTER
00044     // The Status register lets us know if there is data to be read.
00045     if (!I2C::read_regs(i2c_addr, MPL3115_STATUS, buf, 1))
00046     {
00047         return false; // If we fail to read that register, return false
00048     }
00049     if (buf[0] == 0)
00050     {
00051         return false; // If no data to read, return false
00052     }
00053
00054     // Grab all the data that is ready to be read from the MPL3115A2 and stick it into the buffer
00055     if (!I2C::read_regs(i2c_addr, buf, 6))
00056     {
00057         return false; // If that fails for some reason, return false
00058     }
00059
00060     // Updating time.
00061     usec_since -= usec;
00062     int diff = (usec - usec_history) » 3;
00063     if (diff < -1000) diff = -1000;
00064     else if (diff > 1000) diff = 1000;
00065     usec_history += diff;
00066
00067     // Get altitude from buffer and stick into altitude in struct
00068     // Bit shifting according to the MPL3115A2 Datasheet
00069     int32_t a = ((uint32_t)buf[1] « 12) | ((uint16_t)buf[2] « 4) | (buf[3] » 4);
00070     if (a & 0x00080000) a |= 0xFFF00000;
00071     data->altitude = a;
00072
00073     // Get temperature from buffer and stick in temperature in struct
00074     data->temperature = (int16_t)((buf[4] « 8) | buf[5]);
00075
00076     // Serial.print(data->altitude);
00077     // Serial.print(" ");
00078     // Serial.println(data->temperature);
00079
00080     data->t = millis(); // Place the time data into the struct
00081
00082     return true;
00083 }
```

6.7 src/src/DigitalIMU.cpp File Reference

The main source file for the DigitalIMU class.

`#include "yonics.hpp"`
Include dependency graph for DigitalIMU.cpp:



6.7.1 Detailed Description

The main source file for the DigitalIMU class.

Definition in file DigitalIMU.cpp.

6.8 DigitalIMU.cpp

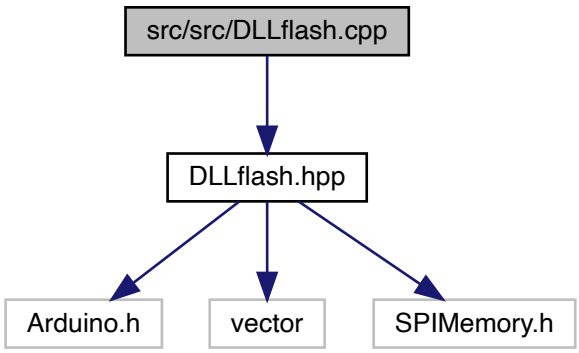
```
00001
00005 #include "yonics.hpp"
00006
00007 DigitalIMU::DigitalIMU() {
00008     board = Adafruit_BNO055(55, 0x28);
00009 }
00010
00011 DigitalIMU::DigitalIMU(int32_t sensorID, uint8_t address) {
00012     board = Adafruit_BNO055(sensorID, address);
00013 }
00014
00015 bool DigitalIMU::begin() {
00016     if (board.begin()) {
00017         board.enableAutoRange(true); // Hopefully this will enable more than +/-4g???
00018         return true;
00019     }
00020     else {
00021         return false;
00022     }
00023 }
00024
00025 void DigitalIMU::sample(IMUdata* data) {
00026     // Get data and store it to the imu_data struct
00027
00028     // Processed acceleration data
00029     board.getEvent(&event,Adafruit_BNO055::VECTOR_LINEARACCEL);
00030     data->accel_fused[0] = event.acceleration.x;
00031     data->accel_fused[1] = event.acceleration.y;
00032     data->accel_fused[2] = event.acceleration.z;
00033
00034     // Processed gyro data
00035     board.getEvent(&event,Adafruit_BNO055::VECTOR_GYROSCOPE);
00036     data->gyro_fused[0] = event.gyro.x;
00037     data->gyro_fused[1] = event.gyro.y;
00038     data->gyro_fused[2] = event.gyro.z;
00039     // Serial.print("Gyro X: ");
00040     // Serial.print(event.gyro.x);
00041     // Serial.print(" Gyro Y: ");
00042     // Serial.print(event.gyro.y);
00043     // Serial.print(" Gyro Z: ");
00044     // Serial.println(event.gyro.z);
00045
00046     // Processed euler orientation vectors
00047     board.getEvent(&event,Adafruit_BNO055::VECTOR_EULER);
00048     data->orient_euler[0] = event.orientation.x;
00049     data->orient_euler[1] = event.orientation.y;
```

```
00050     data->orient_euler[2] = event.orientation.z;
00051
00052     //
00053     quat = board.getQuat();
00054     data->orient_quat[0] = quat.w();
00055     data->orient_quat[1] = quat.x();
00056     data->orient_quat[2] = quat.y();
00057     data->orient_quat[3] = quat.z();
00058
00059     /*Serial.printf("w acceleration: %.5f",quat.w());
00060     Serial.printf("x acceleration: %.5f",quat.x());
00061     Serial.printf("y acceleration: %.5f",quat.y());
00062     Serial.printf("z acceleration: %.5f",quat.z());*/
00063
00064     accel = board.getVector(Adafruit_BNO055::VECTOR_ACCELEROMETER);
00065     data->accel_raw[0] = accel.x();
00066     data->accel_raw[1] = accel.y();
00067     data->accel_raw[2] = accel.z();
00068
00069     /*Serial.printf("x acceleration: %.5f",accel.x());
00070     Serial.printf("y acceleration: %.5f",accel.y());
00071     Serial.printf("z acceleration: %.5f",accel.z());*/
00072
00073     board.getEvent(&event,Adafruit_BNO055::VECTOR_MAGNETOMETER);
00074     data->magnetometer[0] = event.magnetic.x;
00075     data->magnetometer[1] = event.magnetic.y;
00076     data->magnetometer[2] = event.magnetic.z;
00077
00078     /*Serial.printf("x Magnetometer: %.5f",event.magnetic.x);
00079     Serial.printf("y Magnetometer: %.5f",event.magnetic.y);
00080     Serial.printf("z Magnetometer: %.5f\n",event.magnetic.z);*/
00081
00082     data->t = millis();
00083 }
```

6.9 src/src/DLLflash.cpp File Reference

The main source file for the flash chip classes.

`#include "DLLflash.hpp"`
Include dependency graph for DLLflash.cpp:



6.9.1 Detailed Description

The main source file for the flash chip classes.

Definition in file DLLflash.cpp.

6.10 DLLflash.cpp

```
00001
00006 #include "DLLflash.hpp"
00007
00008 DLLtype::DLLtype(void* dataPtr,int dataSize,char* id) {
00009     /*
00010     Parameterized constructor
00011     */
00012
00013     // Set ID
00014     strcpy(this->id,id);
00015
00016     // Set values
00017     refData = dataPtr;
00018     this->dataSize = dataSize;
00019
00020     addrSize = sizeof(uint32_t);
00021 }
00022
00023 DLLtype:: DLLtype() {
00024     free(nextBuffer);
00025     free(currBuffer);
00026 }
00027
00028 bool DLLtype::init() {
00029     /*
00030     Allocates memory for the data buffers using the dataSize member. Fails if the buffers have already
      been allocated.
00031     */
00032
00033     // If data size hasn't been initialized, return false
00034     if ( dataSize) {return false;}
00035
00036     // If data buffers have already been allocated, return false
00037     if (nextBuffer || currBuffer) {return false;}
00038
00039     // Check that dataSize is populated with something reasonable
00040     if (dataSize>1000) {return false;}
00041
00042     // Allocate buffers
00043     nextBuffer = malloc(dataSize);
00044     currBuffer = malloc(dataSize);
00045
00046     return true;
00047 }
00048
00049 void DLLtype::bufferFirstSample() {
00050     memcpy(nextBuffer,refData,dataSize);
00051 }
00052
00053 bool DLLtype::buffer2flash(uint32_t writeAddr,SPIFlash* flash) {
00054
00055 }
00056
00057 bool DLLtype::setType(void* dataPtr,int dataSize) {
00058     // Check whether type is already set
00059     if (dataSize) {return false;}
00060
00061     // Set values
00062     refData = dataPtr;
00063     this->dataSize = dataSize;
00064
00065     return true;
00066 }
00067
00068 bool DLLtype::writeSample(uint32_t next,SPIFlash* flash) {
00069     /*
00070     TODO: The rest of this function
00071     - Error checking
00072     - Correctly assign the head/tail addresses
00073     - Copy over the correct data
00074
00075     */
00076
00077     this->next = next;
00078     uint32_t writeAddr = curr;
00079
00080     // Write next address
00081     flash->writeULong(writeAddr,next);
00082     writeAddr+=addrSize;
00083
00084     // Write prev address
00085     flash->writeULong(writeAddr,prev);
00086     writeAddr+=addrSize;
00087
00088     for (int i=0;i<dataSize;i++) {
```

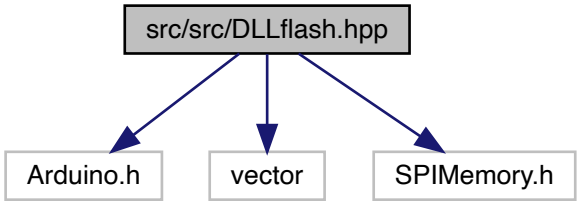
```
00089         flash->writeByte(writeAddr+=1,*(uint8_t*)(currBuffer+i));
00090     }
00091
00092     // Copy next to curr
00093     memcpy(currBuffer,nextBuffer,dataSize);
00094
00095     // Copy data from reference pointer to next
00096     memcpy(nextBuffer,refData,dataSize);
00097
00098     return true;
00099 }
00100
00101 char* DLLtype::getID() {
00102     return id;
00103 }
00104
00105 DLLflash::DLLflash(int flashpin)
00106 {
00107     // Tell SPIFlash class to yeet the flash chip into action
00108     flash = new SPIFlash(flashpin);
00109
00110     // Initialize flash chip
00111     if(!flash->begin())
00112     {
00113         // If the flash chip doesn't successfully initialize
00114         while(true)
00115         {
00116             // Absolutely spam over Serial, and also don't do anything
00117             Serial.println("Failed to initialize the Flash Chip");
00118         }
00119     }
00120     // Store the size of the flash chip to flashSize
00121     flashSize = flash->getCapacity();
00122 }
00123
00124 DLLflash:: DLLflash() {
00125     // Free memory
00126     for (std::vector<DLLtype*>::iterator it = types.begin();it!=types.end();it++) {
00127         delete *it;
00128         *it = nullptr;
00129     }
00130 }
00131
00132 template <class T>
00133 void DLLflash::addType(T* data,char* id) {
00134
00135     int dataSize = sizeof(*data);
00136     void* dataPtr = (void*) data;
00137
00138     DLLtype* newType = new DLLtype(dataPtr,dataSize,id);
00139     types.push_back(newType);
00140 }
00141
00142 bool DLLflash::writeSample(char*)
00143 {
00144     // Loop over vector of data types until finding the one with the correct ID
00145     // for () {}
00146 }
```

6.11 src/src/DLLflash.hpp File Reference

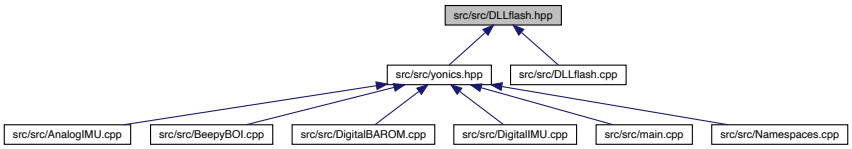
The header file for the flash chip classes.

```
#include <Arduino.h>
#include <vector>
#include <SPIMemory.h>
```

Include dependency graph for DLLflash.hpp:



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



Classes

- class DLLtype
- class DLLflash

6.11.1 Detailed Description

The header file for the flash chip classes.

Definition in file DLLflash.hpp.

6.12 DLLflash.hpp

```
00001
00006 #include <Arduino.h>
00007 #include<vector>
00008 // #include "StructDefs.hpp"
00009 // #include "SPIMemory.h"
00010 #include <SPIMemory.h>
00011
00012 /*
00013 class DLLtype
00014 -------------
00015
00016 Class definition to store, manage, and operate a single data type being stored to flash memory
00017 */
00018 class DLLtype {
00019     private:
00020         void* nextBuffer = NULL; // Buffer of next sample
00021         void* currBuffer = NULL; // Buffer of current sample
```

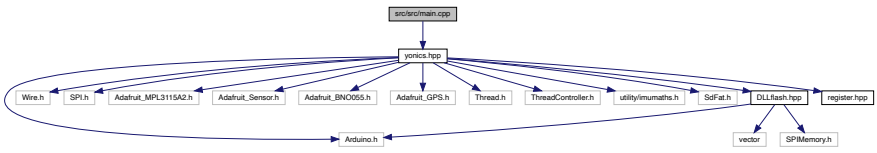
```
00022         void* refData = NULL; // Pointer to struct containing data
00023
00024         uint32_t head = 0; // Flash chip addr of head
00025         uint32_t tail = 0; // Flash chip addr of tail
00026
00027         uint32_t prev = 0; // Address on flash chip of prev sample
00028         uint32_t curr = 0; // Address on flash chip of curr sample
00029         uint32_t next = 0; // Address on flash chip allocated for next sample
00030         int nSamplesWritten = 0; // Number of samples stored to flash chip
00031
00032
00033         int dataSize = 0; // Size of one sample (i.e. size of refData)
00034         int addrSize; // Size of one flash chip address
00035         char id[3];  // Three-character identifier (e.g. IMU)
00036
00037         bool init();
00038         void bufferFirstSample();
00039         bool buffer2flash(uint32_t,SPIFlash*);
00040     public:
00041         DLLtype(void*,int,char*);
00042          DLLtype();
00043         bool setType(void*,int);
00044         bool writeSample(uint32_t,SPIFlash*);
00045         bool readSample();
00046         char* getID();
00047
00048
00049 };
00050
00051 /*
00052 class DLLflash
00053 --------------
00054
00055 Class to manage and operate all flash memory interactions
00056 */
00057 class DLLflash {
00058     private:
00059         uint32_t addr_next_available = 0; // Next available address
00060         std::vector <DLLtype*> types; // Vector of instantiated types
00061         SPIFlash* flash = NULL; // Pointer to SPIflash object (flash chip)
00062         uint32_t flashSize = 0;
00063
00064         bool READ_WRITE = true; // true if in reading mode, false if in writing mode
00065     public:
00066         DLLflash();
00067         DLLflash(int);
00068          DLLflash();
00069         template <class T>
00070         void addType(T*,char*);
00071         bool writeSample(char*);
00072         void setToWrite();
00073         void setToRead();
00074 };
```

6.13 src/src/main.cpp File Reference

The main source file for the CUSRL_Avionics Code Base.

```
#include "yonics.hpp"
```

Include dependency graph for main.cpp:



Functions

- void thread_IMU ()

- void thread_BAROM ()
- void thread_HIGHG ()
- void KILLSYSTEM ()
- void setup ()
- void loop ()

6.13.1 Detailed Description

The main source file for the CUSRL_Avionics Code Base.

This source file initializes all the threads and defines them. Furthermore, the void setup() and void loop() functions are defined here also.

Definition in file main.cpp.

6.13.2 Function Documentation

6.13.2.1 KILLSYSTEM()

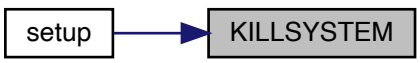
```
void KILLSYSTEM ( )
```

Continous loud obnoxious beeping to alert that the system is kill

Definition at line 36 of file main.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



6.13.2.2 loop()

```
void loop ( )
```

Loop the ThreadController

Definition at line 87 of file main.cpp.

6.13.2.3 setup()

```
void setup ( )
```

Wait 2.5 seconds before starting everything up

Start serial comms

Hello beep

Set the ThreadIMU looping function for the ThreadController

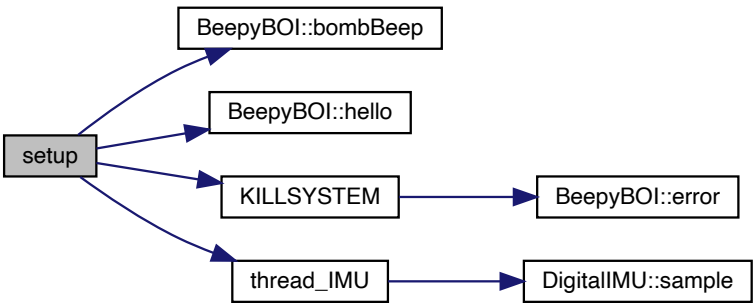
Set the ThreadIMU refresh rate (Interval)

Add the ThreadIMU to the ThreadController for continous processing

Beep the piezo again

Definition at line 44 of file main.cpp.

Here is the call graph for this function:



6.13.2.4 thread_BAROM()

```
void thread_BAROM ( )
```

Sample barometer by calling the BAROM Sample function

Definition at line 28 of file main.cpp.

Here is the call graph for this function:

6.13.2.5 thread_HIGHG()

```
void thread_HIGHG ( )
```

Sample high-g accelerometer by calling the HIGHG Sample function

Definition at line 32 of file main.cpp.

Here is the call graph for this function:

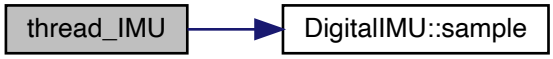
6.13.2.6 thread_IMU()

```
void thread_IMU ( )
```

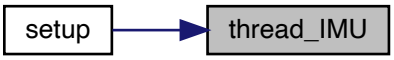
Sample the IMU by calling the IMU Sample function

Definition at line 24 of file main.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.14 main.cpp

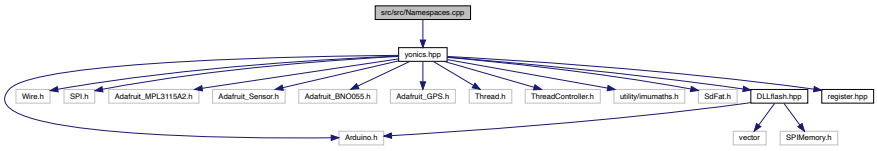
```
00001
00010 /*****************************************************************/
00011 /*                                                               */
00012 /*                                                               */
00013 /*                    CU-SRL AVIONICS TEAM                       */
00014 /*                    MAIN FLIGHT COMPUTER                       */
00015 /*                                                               */
00016 /*                       Jason Popich                            */
00017 /*                       Lyon Foster                             */
00018 /*                       Carter Mak                              */
00019 /*                       Giselle Koo                             */
00020 /*                                                               */
00021 /*****************************************************************/
00022 #include "yonics.hpp"
00023
00024 void thread_IMU() {
00025     INITS::IMU.sample(&INITS::imu_data);
00026 }
00027
00028 void thread_BAROM() {
00029     INITS::BAROM.sample(&INITS::barom_data);
00030 }
00031
00032 void thread_HIGHG() {
00033     INITS::HIGHG.sample(&INITS::accel_data);
00034 }
00035
00036 void KILLSYSTEM() {
00037     /* TO RESET THE SYSTEM, POWER MUST BE REMOVED AND THEN REAPPLIED */
00038     while(true) {
00039         delay(500);
```

```
00040         INITS::berp.error();
00041     }
00042 }
00043
00044 void setup() {
00045
00046     delay(2500);
00048     Serial.begin(115200);
00050     INITS::berp.hello();
00052     // INITS::flash = new DLLflash(10); /*! STILL VERY WIP FLASH INITIALIZATION */
00053
00054     /* Initialize BNO055 IMU sensor */
00055     if (!INITS::IMU.begin()) {
00056         KILLSYSTEM();
00057     }
00058
00059     /* Initialize MPL3115A2 sensor */
00060     // if (!INITS::BAROM.begin()) {
00061     //     KILLSYSTEM();
00062     // }
00063
00064     /* Configure IMU thread */
00065     PROTOTHREADING::ThreadIMU->onRun(thread_IMU);
00066     PROTOTHREADING::ThreadIMU->setInterval(PROTOTHREADING::interval_IMU);
00068     /* Configure Barometer thread */
00069     // PROTOTHREADING::ThreadBAROM->onRun(thread_BAROM);                          /*! Set the
      ThreadBAROM looping function for the ThreadController */
00070     // PROTOTHREADING::ThreadBAROM->setInterval(PROTOTHREADING::interval_BAROM); /*! Set the
      ThreadBAROM refresh rate (Interval) */
00071
00072     /* Configure Accelerometer thread */
00073     // PROTOTHREADING::ThreadACCEL->onRun(thread_HIGHG);                          /*! Set the
      ThreadACCEL looping function for the ThreadController */
00074     // PROTOTHREADING::ThreadACCEL->setInterval(PROTOTHREADING::interval_ACCEL); /*! Set the
      ThreadACCEL refresh rate (Interval) */
00075
00076     /* Add threads to ThreadController */
00077     PROTOTHREADING::thread_control.add(PROTOTHREADING::ThreadIMU);
00078     // PROTOTHREADING::thread_control.add(PROTOTHREADING::ThreadBAROM);          /*! Add the
      ThreadBAROM to the ThreadController for continous processing */
00079     // PROTOTHREADING::thread_control.add(PROTOTHREADING::ThreadACCEL);          /*! Add the
      ThreadACCEL to the ThreadController for continous processing */
00080
00082     INITS::berp.bombBeep();
00083
00084     // START LOOP AFTER THIS IN VOID() LOOP
00085 }
00086
00087 void loop() {
00089     PROTOTHREADING::thread_control.run();
00090 }
```

6.15 src/src/Namespaces.cpp File Reference

The namespace source file for the CUSRL_Avionics Code Base.

`#include "yonics.hpp"`
Include dependency graph for Namespaces.cpp:



Namespaces

- INITS
- PROTOTHREADING
- I2C

Functions

- bool I2C::write_reg (uint8_t i2c, uint8_t addr, uint8_t val)
- bool I2C::read_regs (uint8_t i2c, uint8_t addr, uint8_t *data, uint8_t num)
- bool I2C::read_regs (uint8_t i2c, uint8_t *data, uint8_t num)

Variables

- int INITS::speakerPin = 2
- int INITS::highG_xPin = 33
- int INITS::highG_yPin = 34
- int INITS::highG_zPin = 35
- DigitalIMU INITS::IMU = DigitalIMU(55,0x28)
- DigitalBAROM INITS::BAROM
- HIGHG_ACCEL INITS::HIGHG = HIGHG_ACCEL(highG_xPin,highG_yPin,highG_zPin,true)
- BeepyBOI INITS::berp = BeepyBOI(speakerPin)
- DLLflash * INITS::flash = nullptr
- IMUdata INITS::imu_data
- BAROMdata INITS::barom_data
- ACCELdata INITS::accel_data
- int PROTOTHREADING::interval_IMU = 45
- int PROTOTHREADING::interval_BAROM = 2000
- int PROTOTHREADING::interval_ACCEL = 50
- ThreadController PROTOTHREADING::thread_control = ThreadController()
- Thread * PROTOTHREADING::ThreadIMU = new Thread()
- Thread * PROTOTHREADING::ThreadBAROM = new Thread()
- Thread * PROTOTHREADING::ThreadACCEL = new Thread()

6.15.1 Detailed Description

The namespace source file for the CUSRL_Avionics Code Base.

This source file defines all the namespaces and their member variables

Definition in file Namespaces.cpp.

6.16 Namespaces.cpp

```
00001
00009 /*****************************************************************/
00010 /*                                                               */
00011 /*                                                               */
00012 /*                    CU-SRL AVIONICS TEAM                       */
00013 /*                   Namespace Definitions                       */
00014 /*                                                               */
00015 /*                       Jason Popich                            */
00016 /*                       Lyon Foster                             */
00017 /*                       Carter Mak                              */
00018 /*                       Giselle Koo                             */
00019 /*                                                               */
00020 /*****************************************************************/
00021
00022 #include "yonics.hpp"
00023
00024 // INITS Namespace Definitions
00025 namespace INITS
00026 {
00027     // ALL PIN ASSIGNMENTS ARE ON A BOARD BY BOARD CASE
00028     int speakerPin = 2;     // Board speakerPin definition
```

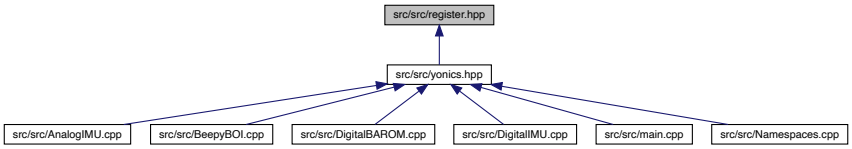
```
00029     int highG_xPin = 33;    // Board highG_xPin defintion
00030     int highG_yPin = 34;    // Board highG_yPin defintion
00031     int highG_zPin = 35;    // Board highG_zPin defintion
00032
00033     DigitalIMU IMU = DigitalIMU(55,0x28);                                    // Define the DigitalIMU
      object and set it equal to the DigitalIMU instance
00034     DigitalBAROM BAROM;                                                      // Create the DigitalBAROM
      class object
00035     HIGHG_ACCEL HIGHG = HIGHG_ACCEL(highG_xPin,highG_yPin,highG_zPin,true); // Define the AnalogIMU
      object and set it equal to the AnalogIMU instance
00036     BeepyBOI berp = BeepyBOI(speakerPin);                                    // Define the BeepyBOI
      object and set it equal to the BeepBOI instance
00037
00038     DLLflash* flash = nullptr;                                               // Define the DLLflash
      object and set it equal to nullptr b/c it doesn't point to anything yet
00039
00040     IMUdata imu_data;                                                        // Create the IMUData
      struct object
00041     BAROMdata barom_data;                                                    // Create the BAROMdata
      struct object
00042     ACCELdata accel_data;                                                    // Create the ACCELdata
      struct object
00043 };
00044
00045 // PROTOTHREADING Namespace Definitions
00046 namespace PROTOTHREADING
00047 {
00048     int interval_IMU = 45;      // Define the interval at which to sample the IMU in milliseconds
00049     int interval_BAROM = 2000;  // Define the interval at which to sample the BAROM in milliseconds
00050     int interval_ACCEL = 50;    // Define the interval at which to sample the High-G ACCEL in
      milliseconds
00051
00052     ThreadController thread_control = ThreadController(); // Create a new ThreadController instance
      for the ThreadController and set thread_control pointer equal to it
00053
00054     Thread* ThreadIMU = new Thread();       // Create a new Thread instance for ThreadIMU and set
      ThreadIMU pointer equal to it
00055     Thread* ThreadBAROM = new Thread();     // Create a new Thread instance for ThreadBAROM and set
      ThreadBAROM pointer equal to it
00056     Thread* ThreadACCEL = new Thread();     // Create a new Thread instance for ThreadACCEL and set
      ThreadACCEL pointer equal to it
00057 };
00058
00059 // I2C Namespace Definitions
00060 namespace I2C
00061 {
00062     // I2C - Write to Registry function
00063     bool write_reg(uint8_t i2c, uint8_t addr, uint8_t val)
00064     {
00065         Wire.beginTransmission(i2c);            // Signal beginning of I2C transmission at i2c device
00066         Wire.write(addr);                       // Place the register address we want to write to in
      the transmission buffer
00067         Wire.write(val);                        // Place the value we want written in the transmission
      buffer
00068         return Wire.endTransmission() == 0;     // Executes the buffer and signals end of MASTER
      transmission
00069     }
00070
00071     // I2C - First Read from Registry function
00072     bool read_regs(uint8_t i2c, uint8_t addr, uint8_t *data, uint8_t num)
00073     {
00074         Wire.beginTransmission(i2c);            // Signal beginning of I2C transmission at i2c device
00075         Wire.write(addr);                       // Place the register address we want to read from in
      the transmission buffer
00076         if (Wire.endTransmission(false) != 0)   // Execute the buffer and but DO NOT signal end of
      transmission because we want to read from the device
00077         {
00078             return false;                       // If failed, return false
00079         }
00080         Wire.requestFrom(i2c, num);             // Request, from the address we specified above, a
      certain amount of bytes
00081         if (Wire.available() != num)            // Make sure we have that amount of bytes available to
      read
00082         {
00083             return false;                       // If we dont have that amount of bytes, return false
00084         }
00085         while (num > 0)
00086         {
00087             *data++ = Wire.read();              // Read every byte and place into data buffer
00088             num--;                              // Reduce by one num of bytes left to read
00089         }
00090         return true;                            // Return Success after reading
00091     }
00092
00093     // I2C - Second Read from Registry function
00094     bool read_regs(uint8_t i2c, uint8_t *data, uint8_t num)
00095     {
```

```
00096         Wire.requestFrom(i2c, num);             // Request, from the i2c device, a certain amount of
      bytes
00097         if (Wire.available() != num)            // Make sure we have that amount of bytes available to
      read
00098         {
00099             return false;                       // If we dont have that amount of bytes, return false
00100         }
00101         while (num > 0)
00102         {
00103             *data++ = Wire.read();              // Read every byte and place into data buffer
00104             num--;                              // Reduce by one num of bytes left to read
00105         }
00106         return true;                            // Return Success after reading
00107     }
00108 };
```

6.17 src/src/register.hpp File Reference

The main register header file for the CUSRL_Avionics Code Base.

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



Macros

- #define BNO055_MAG_RADIUS_MSB 0x6A /∗! Magnetometer Radius ∗/
- #define BNO055_MAG_RADIUS_LSB 0x69 /∗! Magnetometer Radius ∗/
- #define BNO055_ACC_RADIUS_MSB 0x68 /∗! Accelerometer Radius ∗/
- #define BNO055_ACC_RADIUS_LSB 0x67 /∗! Accelerometer Radius ∗/
- #define BNO055_GYR_OFFSET_Z_MSB 0x66 /∗! Gyroscope Offset Z <15:8> ∗/
- #define BNO055_GYR_OFFSET_Z_LSB 0x65 /∗! Gyroscope Offset Z <7:0> ∗/
- #define BNO055_GYR_OFFSET_Y_MSB 0x64 /∗! Gyroscope Offset Y <15:8> ∗/
- #define BNO055_GYR_OFFSET_Y_LSB 0x63 /∗! Gyroscope Offset Y <7:0> ∗/
- #define BNO055_GYR_OFFSET_X_MSB 0x62 /∗! Gyroscope Offset X <15:8> ∗/
- #define BNO055_GYR_OFFSET_X_LSB 0x61 /∗! Gyroscope Offset X <7:0> ∗/
- #define BNO055_MAG_OFFSET_Z_MSB 0x60 /∗! Magnetometer Offset Z <15:8> ∗/
- #define BNO055_MAG_OFFSET_Z_LSB 0x5F /∗! Magnetometer Offset Z <7:0> ∗/
- #define BNO055_MAG_OFFSET_Y_MSB 0x5E /∗! Magnetometer Offset Y <15:8> ∗/
- #define BNO055_MAG_OFFSET_Y_LSB 0x5D /∗! Magnetometer Offset Y <7:0> ∗/
- #define BNO055_MAG_OFFSET_X_MSB 0x5C /∗! Magnetometer Offset X <15:8> ∗/
- #define BNO055_MAG_OFFSET_X_LSB 0x5B /∗! Magnetometer Offset X <7:0> ∗/
- #define BNO055_ACC_OFFSET_Z_MSB 0x5A /∗! Accelerometer Offset Z <15:8> ∗/
- #define BNO055_ACC_OFFSET_Z_LSB 0x59 /∗! Accelerometer Offset Z <7:0> ∗/
- #define BNO055_ACC_OFFSET_Y_MSB 0x58 /∗! Accelerometer Offset Y <15:8> ∗/
- #define BNO055_ACC_OFFSET_Y_LSB 0x57 /∗! Accelerometer Offset Y <7:0> ∗/
- #define BNO055_ACC_OFFSET_X_MSB 0x56 /∗! Accelerometer Offset X <15:8> ∗/
- #define BNO055_ACC_OFFSET_X_LSB 0x55 /∗! Accelerometer Offset X <7:0> ∗/
- #define BNO055_AXIS_MAP_SIGN 0x42
- #define BNO055_AXIS_MAP_CONFIG 0x41

- #define BNO055_TEMP_SOURCE 0x40
- #define BNO055_SYS_TRIGGER 0x3F
- #define BNO055_PWR_MODE 0x3E
- #define BNO055_OPR_MODE 0x3D
- #define BNO055_Reserved 0x3C
- #define BNO055_UNIT_SEL 0x3B
- #define BNO055_SYS_ERR 0x3A
- #define BNO055_SYS_STATUS 0x39
- #define BNO055_SYS_CLK_STA 0x38
- #define BNO055_INT_STA 0x37
- #define BNO055_ST_RESULT 0x36
- #define BNO055_CALIB_STAT 0x35
- #define BNO055_TEMP 0x34
- #define BNO055_GRV_Data_Z_M 0x33
- #define BNO055_GRV_Data_Z_L 0x32
- #define BNO055_GRV_Data_Y_M 0x31
- #define BNO055_GRV_Data_Y_L 0x30
- #define BNO055_GRV_Data_X_M 0x2F
- #define BNO055_GRV_Data_X_L 0x2E
- #define BNO055_LIA_Data_Z_MB 0x2D
- #define BNO055_LIA_Data_Z_LS 0x2C
- #define BNO055_LIA_Data_Y_MB 0x2B
- #define BNO055_LIA_Data_Y_LS 0x2A
- #define BNO055_LI A_Data_X_MB 0x29
- #define BNO055_LIA_Data_X_LS 0x28
- #define BNO055_QUA_Data_z_M 0x27
- #define BNO055_QUA_Data_z_LS 0x26
- #define BNO055_QUA_Data_y_M 0x25
- #define BNO055_QUA_Data_y_LS 0x24
- #define BNO055_QUA_Data_x_M 0x23
- #define BNO055_QUA_Data_x_LS 0x22
- #define BNO055_QUA_Data_w_M 0x21
- #define BNO055_QUA_Data_w_L 0x20
- #define BNO055_EUL_Pitch_MSB 0x1F
- #define BNO055_EUL_Pitch_LSB 0x1E
- #define BNO055_EUL_Roll_MSB 0x1D /∗! Roll Data <15:8> ∗/
- #define BNO055_EUL_Roll_LSB 0x1C /∗! Roll Data <7:0> ∗/
- #define BNO055_EUL_Heading_MSB 0x1B /∗! Heading Data <15:8> ∗/
- #define BNO055_EUL_Heading_LSB 0x1A /∗! Heading Data <7:0> ∗/
- #define BNO055_GYR_DATA_Z_MSB 0x19 /∗! Gyroscope Data Z <15:8> ∗/
- #define BNO055_GYR_DATA_Z_LSB 0x18 /∗! Gyroscope Data Z <7:0> ∗/
- #define BNO055_GYR_DATA_Y_MSB 0x17 /∗! Gyroscope Data Y <15:8> ∗/
- #define BNO055_GYR_DATA_Y_LSB 0x16 /∗! Gyroscope Data Y <7:0> ∗/
- #define BNO055_GYR_DATA_X_MSB 0x15 /∗! Gyroscope Data X <15:8> ∗/
- #define BNO055_GYR_DATA_X_LSB 0x14 /∗! Gyroscope Data X <7:0> ∗/
- #define BNO055_MAG_DATA_Z_MSB 0x13 /∗! Magnetometer Data Z <15:8> ∗/
- #define BNO055_MAG_DATA_Z_LSB 0x12 /∗! Magnetometer Data Z <7:0> ∗/
- #define BNO055_MAG_DATA_Y_MSB 0x11 /∗! Magnetometer Data Y <15:8> ∗/
- #define BNO055_MAG_DATA_Y_LSB 0x10 /∗! Magnetometer Data Y <7:0> ∗/
- #define BNO055_MAG_DATA_X_MSB 0x0F /∗! Magnetometer Data X <15:8> ∗/
- #define BNO055_MAG_DATA_X_LSB 0x0E /∗! Magnetometer Data X <7:0> ∗/
- #define BNO055_ACC_DATA_Z_MSB 0x0D /∗! Acceleration Data Z <15:8> ∗/
- #define BNO055_ACC_DATA_Z_LSB 0x0C /∗! Acceleration Data Z <7:0> ∗/
- #define BNO055_ACC_DATA_Y_MSB 0x0B /∗! Acceleration Data Y <15:8> ∗/
- #define BNO055_ACC_DATA_Y_LSB 0x0A /∗! Acceleration Data Y <7:0> ∗/

- #define BNO055_ACC_DATA_X_MSB 0x09 /*! Acceleration Data X <15:8> */
- #define BNO055_PWR_ACC_DATA_X_LSB 0x08 /*! Acceleration Data X <7:0> */
- #define BNO055_Page_ID 0x07 /*! Page ID */
- #define BNO055_BL_Rev_ID 0x06 /*! Boot loader Version */
- #define BNO055_SW_REV_ID_MSB 0x05 /*! SW Revision ID <15:8> */
- #define BNO055_SW_REV_ID_LSB 0x04 /*! SW Revision ID <7:0> */
- #define BNO055_GYR_ID 0x03 /*! GYRO chip ID */
- #define BNO055_MAG_ID 0x02 /*! MAG chip ID */
- #define BNO055_ACC_ID 0x01 /*! ACC chip ID */
- #define BNO055_CHIP_ID 0x00 /*! BNO055 CHIP ID */
- #define MPL3115_I2C_ADDR 0x60 /*! I2C address */
- #define MPL3115_STATUS 0x00 /*! Sensor Status Register */
- #define MPL3115_OUT_P_MSB 0x01 /*! Pressure Data Out MSB */
- #define MPL3115_OUT_P_CSB 0x02 /*! Pressure Data Out CSB */
- #define MPL3115_OUT_P_LSB 0x03 /*! Pressure Data Out LSB */
- #define MPL3115_OUT_T_MSB 0x04 /*! Temperature Data Out MSB */
- #define MPL3115_OUT_T_LSB 0x05 /*! Temperature Data Out LSB */
- #define MPL3115_DR_STATUS 0x06 /*! Sensor Status Register */
- #define MPL3115_OUT_P_DELTA MSB 0x07 /*! Pressure Data Out Delta MSB */
- #define MPL3115_OUT_P_DELTA_CSB 0x08 /*! Pressure Data Out Delta CSB */
- #define MPL3115_OUT_P_DELTA_LSB 0x09 /*! Pressure Data Out Delta LSB */
- #define MPL3115_OUT_T_DELTA_MSB 0x0A /*! Temperature Data Out Delta MSB */
- #define MPL3115_OUT_T_DELTA_LSB 0x0B /*! Temperature Data Out Delta LSB */
- #define MPL3115_WHO_AM_I 0x0C /*! Device Identification Register */
- #define MPL3115_F_STATUS 0x0D /*! FIFO Status Register */
- #define MPL3115_F_DATA 0X0E /*! FIFO 8-bit Data Access */
- #define MPL3115_F_SETUP 0x0F /*! FIFO Setup Register */
- #define MPL3115_TIME_DLY 0x10 /*! Time Delay */
- #define MPL3115_SYSMOD 0x11 /*! System Mode Register */
- #define MPL3115_INT_SOURCE 0x12 /*! Interrupt Source Register */
- #define MPL3115_PT_DATA_CFG 0x13 /*! PT Data Configuration Register */
- #define MPL3115_BAR_IN_MSB 0x14 /*! BAR Input in MSB */
- #define MPL3115_BAR_IN_LSB 0x15 /*! BAR Input in LSB */
- #define MPL3115_P_TGT_MSB 0x16 /*! Pressure Target MSB */
- #define MPL3115_P_TGT_LSB 0x17 /*! Pressure Target LSB */
- #define MPL3115_T_TGT 0x18 /*! Temperature Target */
- #define MPL3115_P_WND_MSB 0x19 /*! Pressure/Altitude Window MSB */
- #define MPL3115_P_WND_LSB 0x1A /*! Pressure/Altitude Window LSB */
- #define MPL3115_T_WND 0X1B /*! Temperature Window */
- #define MPL3115_P_MIN_MSB 0x1C /*! Minimum Pressure Data Out MSB */
- #define MPL3115_P_MIN_CSB 0x1D /*! Minimum Pressure Data Out CSB */
- #define MPL3115_P_MIN_LSB 0x1E /*! Minimum Pressure Data Out LSB */
- #define MPL3115_T_MIN_MSB 0x1F /*! Minimum Temperature Data Out MSB */
- #define MPL3115_T_MIN_LSB 0x20 /*! Minimum Temperature Data Out LSB */
- #define MPL3115_P_MAX_MSB 0x21 /*! Maximum Pressure Data Out MSB */
- #define MPL3115_P_MAX_CSB 0x22 /*! Maximum Pressure Data Out CSB */
- #define MPL3115_P_MAX_LSB 0x23 /*! Maximum Pressure Data Out LSB */
- #define MPL3115_T_MAX_MSB 0X24 /*! Maximum Temperature Data Out MSB */
- #define MPL3115_T_MAX_LSB 0x25 /*! Maximum Temperature Data Out LSB */
- #define MPL3115_CTRL_REG1 0x26 /*! Control Register 1 */
- #define MPL3115_CTRL_REG2 0x27 /*! Control Register 2 */
- #define MPL3115_CTRL_REG3 0x28 /*! Control Register 3 */
- #define MPL3115_CTRL_REG4 0x29 /*! Control Register 4 */
- #define MPL3115_CTRL_REG5 0x2A /*! Control Register 5 */
- #define MPL3115_OFF_P 0x2B /*! Pressure Data User Offset */
- #define MPL3115_OFF_T 0x2C /*! Temperature Data User Offset */
- #define MPL3115_OFF_H 0x2D /*! Altitude Data User Offset Register */

6.17.1 Detailed Description

The main register header file for the CUSRL_Avionics Code Base.

All registers are declared and defined here for use with I2C and/or any other comms interface

Definition in file register.hpp.

6.17.2 Macro Definition Documentation

6.17.2.1 BNO055_ACC_DATA_X_MSB

```
#define BNO055_ACC_DATA_X_MSB 0x09 /*!  Acceleration Data X <15:8> */
```

Definition at line 100 of file register.hpp.

6.17.2.2 BNO055_ACC_DATA_Y_LSB

```
#define BNO055_ACC_DATA_Y_LSB 0x0A /*!  Acceleration Data Y <7:0> */
```

Definition at line 99 of file register.hpp.

6.17.2.3 BNO055_ACC_DATA_Y_MSB

```
#define BNO055_ACC_DATA_Y_MSB 0x0B /*!  Acceleration Data Y <15:8> */
```

Definition at line 98 of file register.hpp.

6.17.2.4 BNO055_ACC_DATA_Z_LSB

```
#define BNO055_ACC_DATA_Z_LSB 0x0C /*!  Acceleration Data Z <7:0> */
```

Definition at line 97 of file register.hpp.

6.17.2.5 BNO055_ACC_DATA_Z_MSB

```
#define BNO055_ACC_DATA_Z_MSB 0x0D /*!  Acceleration Data Z <15:8> */
```

Definition at line 96 of file register.hpp.

6.17.2.6 BNO055_ACC_ID

```
#define BNO055_ACC_ID 0x01 /*!  ACC chip ID */
```

Definition at line 108 of file register.hpp.

6.17.2.7 BNO055_ACC_OFFSET_X_LSB

```
#define BNO055_ACC_OFFSET_X_LSB 0x55 /*!  Accelerometer Offset X <7:0> */
```

Definition at line 38 of file register.hpp.

6.17.2.8 BNO055_ACC_OFFSET_X_MSB

```
#define BNO055_ACC_OFFSET_X_MSB 0x56 /*!  Accelerometer Offset X <15:8> */
```

Definition at line 37 of file register.hpp.

6.17.2.9 BNO055_ACC_OFFSET_Y_LSB

```
#define BNO055_ACC_OFFSET_Y_LSB 0x57 /*!  Accelerometer Offset Y <7:0> */
```

Definition at line 36 of file register.hpp.

6.17.2.10 BNO055_ACC_OFFSET_Y_MSB

```
#define BNO055_ACC_OFFSET_Y_MSB 0x58 /*!  Accelerometer Offset Y <15:8> */
```

Definition at line 35 of file register.hpp.

6.17.2.11 BNO055_ACC_OFFSET_Z_LSB

```
#define BNO055_ACC_OFFSET_Z_LSB 0x59 /*!  Accelerometer Offset Z <7:0> */
```

Definition at line 34 of file register.hpp.

6.17.2.12 BNO055_ACC_OFFSET_Z_MSB

```
#define BNO055_ACC_OFFSET_Z_MSB 0x5A /*!  Accelerometer Offset Z <15:8> */
```

Definition at line 33 of file register.hpp.

6.17.2.13 BNO055_ACC_RADIUS_LSB

```
#define BNO055_ACC_RADIUS_LSB 0x67 /*!  Accelerometer Radius */
```

Definition at line 20 of file register.hpp.

6.17.2.14 BNO055_ACC_RADIUS_MSB

```
#define BNO055_ACC_RADIUS_MSB 0x68 /*!  Accelerometer Radius */
```

Definition at line 19 of file register.hpp.

6.17.2.15 BNO055_AXIS_MAP_CONFIG

```
#define BNO055_AXIS_MAP_CONFIG 0x41
```

Definition at line 43 of file register.hpp.

6.17.2.16 BNO055_AXIS_MAP_SIGN

```
#define BNO055_AXIS_MAP_SIGN 0x42
```

Definition at line 42 of file register.hpp.

6.17.2.17 BNO055_BL_Rev_ID

```
#define BNO055_BL_Rev_ID 0x06 /*!  Boot loader Version */
```

Definition at line 103 of file register.hpp.

6.17.2.18 BNO055_CALIB_STAT

```
#define BNO055_CALIB_STAT 0x35
```

Definition at line 55 of file register.hpp.

6.17.2.19 BNO055_CHIP_ID

```
#define BNO055_CHIP_ID 0x00 /*!  BNO055 CHIP ID */
```

Definition at line 109 of file register.hpp.

6.17.2.20 BNO055_EUL_Heading_LSB

```
#define BNO055_EUL_Heading_LSB 0x1A /*!  Heading Data <7:0> */
```

Definition at line 83 of file register.hpp.

6.17.2.21 BNO055_EUL_Heading_MSB

```
#define BNO055_EUL_Heading_MSB 0x1B /*!  Heading Data <15:8> */
```

Definition at line 82 of file register.hpp.

6.17.2.22 BNO055_EUL_Pitch_LSB

```
#define BNO055_EUL_Pitch_LSB 0x1E
```

Definition at line 78 of file register.hpp.

6.17.2.23 BNO055_EUL_Pitch_MSB

```
#define BNO055_EUL_Pitch_MSB 0x1F
```

Definition at line 77 of file register.hpp.

6.17.2.24 BNO055_EUL_Roll_LSB

```
#define BNO055_EUL_Roll_LSB 0x1C /*!  Roll Data <7:0> */
```

Definition at line 81 of file register.hpp.

6.17.2.25 BNO055_EUL_Roll_MSB

```
#define BNO055_EUL_Roll_MSB 0x1D /*!  Roll Data <15:8> */
```

Definition at line 80 of file register.hpp.

6.17.2.26 BNO055_GRV_Data_X_L

```
#define BNO055_GRV_Data_X_L 0x2E
```

Definition at line 62 of file register.hpp.

6.17.2.27 BNO055_GRV_Data_X_M

```
#define BNO055_GRV_Data_X_M 0x2F
```

Definition at line 61 of file register.hpp.

6.17.2.28 BNO055_GRV_Data_Y_L

```
#define BNO055_GRV_Data_Y_L 0x30
```

Definition at line 60 of file register.hpp.

6.17.2.29 BNO055_GRV_Data_Y_M

```
#define BNO055_GRV_Data_Y_M 0x31
```

Definition at line 59 of file register.hpp.

6.17.2.30 BNO055_GRV_Data_Z_L

```
#define BNO055_GRV_Data_Z_L 0x32
```

Definition at line 58 of file register.hpp.

6.17.2.31 BNO055_GRV_Data_Z_M

```
#define BNO055_GRV_Data_Z_M 0x33
```

Definition at line 57 of file register.hpp.

6.17.2.32 BNO055_GYR_DATA_X_LSB

```
#define BNO055_GYR_DATA_X_LSB 0x14 /*!  Gyroscope Data X <7:0> */
```

Definition at line 89 of file register.hpp.

6.17.2.33 BNO055_GYR_DATA_X_MSB

```
#define BNO055_GYR_DATA_X_MSB 0x15 /*!  Gyroscope Data X <15:8> */
```

Definition at line 88 of file register.hpp.

6.17.2.34 BNO055_GYR_DATA_Y_LSB

```
#define BNO055_GYR_DATA_Y_LSB 0x16 /*!  Gyroscope Data Y <7:0> */
```

Definition at line 87 of file register.hpp.

6.17.2.35 BNO055_GYR_DATA_Y_MSB

```
#define BNO055_GYR_DATA_Y_MSB 0x17 /*!  Gyroscope Data Y <15:8> */
```

Definition at line 86 of file register.hpp.

6.17.2.36 BNO055_GYR_DATA_Z_LSB

```
#define BNO055_GYR_DATA_Z_LSB 0x18 /*!  Gyroscope Data Z <7:0> */
```

Definition at line 85 of file register.hpp.

6.17.2.37 BNO055_GYR_DATA_Z_MSB

```
#define BNO055_GYR_DATA_Z_MSB 0x19 /*!  Gyroscope Data Z <15:8> */
```

Definition at line 84 of file register.hpp.

6.17.2.38 BNO055_GYR_ID

```
#define BNO055_GYR_ID 0x03 /*!  GYRO chip ID */
```

Definition at line 106 of file register.hpp.

6.17.2.39 BNO055_GYR_OFFSET_X_LSB

```
#define BNO055_GYR_OFFSET_X_LSB 0x61 /*!  Gyroscope Offset X <7:0> */
```

Definition at line 26 of file register.hpp.

6.17.2.40 BNO055_GYR_OFFSET_X_MSB

```
#define BNO055_GYR_OFFSET_X_MSB 0x62 /*!  Gyroscope Offset X <15:8> */
```

Definition at line 25 of file register.hpp.

6.17.2.41 BNO055_GYR_OFFSET_Y_LSB

```
#define BNO055_GYR_OFFSET_Y_LSB 0x63 /*!  Gyroscope Offset Y <7:0> */
```

Definition at line 24 of file register.hpp.

6.17.2.42 BNO055_GYR_OFFSET_Y_MSB

```
#define BNO055_GYR_OFFSET_Y_MSB 0x64 /*!  Gyroscope Offset Y <15:8> */
```

Definition at line 23 of file register.hpp.

6.17.2.43 BNO055_GYR_OFFSET_Z_LSB

```
#define BNO055_GYR_OFFSET_Z_LSB 0x65 /*!  Gyroscope Offset Z <7:0> */
```

Definition at line 22 of file register.hpp.

6.17.2.44 BNO055_GYR_OFFSET_Z_MSB

```
#define BNO055_GYR_OFFSET_Z_MSB 0x66 /*!  Gyroscope Offset Z <15:8> */
```

Definition at line 21 of file register.hpp.

6.17.2.45 BNO055_INT_STA

```
#define BNO055_INT_STA 0x37
```

Definition at line 53 of file register.hpp.

6.17.2.46 BNO055_LI

```
#define BNO055_LI A_Data_X_MB 0x29
```

Definition at line 67 of file register.hpp.

6.17.2.47 BNO055_LIA_Data_X_LS

```
#define BNO055_LIA_Data_X_LS 0x28
```

Definition at line 68 of file register.hpp.

6.17.2.48 BNO055_LIA_Data_Y_LS

```
#define BNO055_LIA_Data_Y_LS 0x2A
```

Definition at line 66 of file register.hpp.

6.17.2.49 BNO055_LIA_Data_Y_MB

```
#define BNO055_LIA_Data_Y_MB 0x2B
```

Definition at line 65 of file register.hpp.

6.17.2.50 BNO055_LIA_Data_Z_LS

```
#define BNO055_LIA_Data_Z_LS 0x2C
```

Definition at line 64 of file register.hpp.

6.17.2.51 BNO055_LIA_Data_Z_MB

```
#define BNO055_LIA_Data_Z_MB 0x2D
```

Definition at line 63 of file register.hpp.

6.17.2.52 BNO055_MAG_DATA_X_LSB

```
#define BNO055_MAG_DATA_X_LSB 0x0E /*!  Magnetometer Data X <7:0> */
```

Definition at line 95 of file register.hpp.

6.17.2.53 BNO055_MAG_DATA_X_MSB

```
#define BNO055_MAG_DATA_X_MSB 0x0F /*!  Magnetometer Data X <15:8> */
```

Definition at line 94 of file register.hpp.

6.17.2.54 BNO055_MAG_DATA_Y_LSB

```
#define BNO055_MAG_DATA_Y_LSB 0x10 /*!  Magnetometer Data Y <7:0> */
```

Definition at line 93 of file register.hpp.

6.17.2.55 BNO055_MAG_DATA_Y_MSB

```
#define BNO055_MAG_DATA_Y_MSB 0x11 /*!  Magnetometer Data Y <15:8> */
```

Definition at line 92 of file register.hpp.

6.17.2.56 BNO055_MAG_DATA_Z_LSB

```
#define BNO055_MAG_DATA_Z_LSB 0x12 /*!  Magnetometer Data Z <7:0> */
```

Definition at line 91 of file register.hpp.

6.17.2.57 BNO055_MAG_DATA_Z_MSB

```
#define BNO055_MAG_DATA_Z_MSB 0x13 /*!  Magnetometer Data Z <15:8> */
```

Definition at line 90 of file register.hpp.

6.17.2.58 BNO055_MAG_ID

```
#define BNO055_MAG_ID 0x02 /*!  MAG chip ID */
```

Definition at line 107 of file register.hpp.

6.17.2.59 BNO055_MAG_OFFSET_X_LSB

```
#define BNO055_MAG_OFFSET_X_LSB 0x5B /*!  Magnetometer Offset X <7:0> */
```

Definition at line 32 of file register.hpp.

6.17.2.60 BNO055_MAG_OFFSET_X_MSB

```
#define BNO055_MAG_OFFSET_X_MSB 0x5C /*!  Magnetometer Offset X <15:8> */
```

Definition at line 31 of file register.hpp.

6.17.2.61 BNO055_MAG_OFFSET_Y_LSB

```
#define BNO055_MAG_OFFSET_Y_LSB 0x5D /*!  Magnetometer Offset Y <7:0> */
```

Definition at line 30 of file register.hpp.

6.17.2.62 BNO055_MAG_OFFSET_Y_MSB

```
#define BNO055_MAG_OFFSET_Y_MSB 0x5E /*!  Magnetometer Offset Y <15:8> */
```

Definition at line 29 of file register.hpp.

6.17.2.63 BNO055_MAG_OFFSET_Z_LSB

```
#define BNO055_MAG_OFFSET_Z_LSB 0x5F /*!  Magnetometer Offset Z <7:0> */
```

Definition at line 28 of file register.hpp.

6.17.2.64 BNO055_MAG_OFFSET_Z_MSB

```
#define BNO055_MAG_OFFSET_Z_MSB 0x60 /*!  Magnetometer Offset Z <15:8> */
```

Definition at line 27 of file register.hpp.

6.17.2.65 BNO055_MAG_RADIUS_LSB

```
#define BNO055_MAG_RADIUS_LSB 0x69 /*!  Magnetometer Radius */
```

Definition at line 18 of file register.hpp.

6.17.2.66 BNO055_MAG_RADIUS_MSB

```
#define BNO055_MAG_RADIUS_MSB 0x6A /*!  Magnetometer Radius */
```

Definition at line 17 of file register.hpp.

6.17.2.67 BNO055_OPR_MODE

```
#define BNO055_OPR_MODE 0x3D
```

Definition at line 47 of file register.hpp.

6.17.2.68 BNO055_Page_ID

```
#define BNO055_Page_ID 0x07 /*!  Page ID */
```

Definition at line 102 of file register.hpp.

6.17.2.69 BNO055_PWR_ACC_DATA_X_LSB

```
#define BNO055_PWR_ACC_DATA_X_LSB 0x08 /*!  Acceleration Data X <7:0> */
```

Definition at line 101 of file register.hpp.

6.17.2.70 BNO055_PWR_MODE

```
#define BNO055_PWR_MODE 0x3E
```

Definition at line 46 of file register.hpp.

6.17.2.71 BNO055_QUA_Data_w_L

```
#define BNO055_QUA_Data_w_L 0x20
```

Definition at line 76 of file register.hpp.

6.17.2.72 BNO055_QUA_Data_w_M

```
#define BNO055_QUA_Data_w_M 0x21
```

Definition at line 75 of file register.hpp.

6.17.2.73 BNO055_QUA_Data_x_LS

```
#define BNO055_QUA_Data_x_LS 0x22
```

Definition at line 74 of file register.hpp.

6.17.2.74 BNO055_QUA_Data_x_M

```
#define BNO055_QUA_Data_x_M 0x23
```

Definition at line 73 of file register.hpp.

6.17.2.75 BNO055_QUA_Data_y_LS

```
#define BNO055_QUA_Data_y_LS 0x24
```

Definition at line 72 of file register.hpp.

6.17.2.76 BNO055_QUA_Data_y_M

```
#define BNO055_QUA_Data_y_M 0x25
```

Definition at line 71 of file register.hpp.

6.17.2.77 BNO055_QUA_Data_z_LS

```
#define BNO055_QUA_Data_z_LS 0x26
```

Definition at line 70 of file register.hpp.

6.17.2.78 BNO055_QUA_Data_z_M

```
#define BNO055_QUA_Data_z_M 0x27
```

Definition at line 69 of file register.hpp.

6.17.2.79 BNO055_Reserved

```
#define BNO055_Reserved 0x3C
```

Definition at line 48 of file register.hpp.

6.17.2.80 BNO055_ST_RESULT

```
#define BNO055_ST_RESULT 0x36
```

Definition at line 54 of file register.hpp.

6.17.2.81 BNO055_SW_REV_ID_LSB

```
#define BNO055_SW_REV_ID_LSB 0x04 /*!  SW Revision ID <7:0> */
```

Definition at line 105 of file register.hpp.

6.17.2.82 BNO055_SW_REV_ID_MSB

```
#define BNO055_SW_REV_ID_MSB 0x05 /*!  SW Revision ID <15:8> */
```

Definition at line 104 of file register.hpp.

6.17.2.83 BNO055_SYS_CLK_STA

```
#define BNO055_SYS_CLK_STA 0x38
```

Definition at line 52 of file register.hpp.

6.17.2.84 BNO055_SYS_ERR

```
#define BNO055_SYS_ERR 0x3A
```

Definition at line 50 of file register.hpp.

6.17.2.85 BNO055_SYS_STATUS

```
#define BNO055_SYS_STATUS 0x39
```

Definition at line 51 of file register.hpp.

6.17.2.86 BNO055_SYS_TRIGGER

```
#define BNO055_SYS_TRIGGER 0x3F
```

Definition at line 45 of file register.hpp.

6.17.2.87 BNO055_TEMP

```
#define BNO055_TEMP 0x34
```

Definition at line 56 of file register.hpp.

6.17.2.88 BNO055_TEMP_SOURCE

```
#define BNO055_TEMP_SOURCE 0x40
```

Definition at line 44 of file register.hpp.

6.17.2.89 BNO055_UNIT_SEL

```
#define BNO055_UNIT_SEL 0x3B
```

Definition at line 49 of file register.hpp.

6.17.2.90 MPL3115_BAR_IN_LSB

```
#define MPL3115_BAR_IN_LSB 0x15 /*!  BAR Input in LSB */
```

Definition at line 140 of file register.hpp.

6.17.2.91 MPL3115_BAR_IN_MSB

```
#define MPL3115_BAR_IN_MSB 0x14 /*!  BAR Input in MSB */
```

Definition at line 139 of file register.hpp.

6.17.2.92 MPL3115_CTRL_REG1

```
#define MPL3115_CTRL_REG1 0x26 /*!  Control Register 1 */
```

Definition at line 157 of file register.hpp.

6.17.2.93 MPL3115_CTRL_REG2

```
#define MPL3115_CTRL_REG2 0x27 /*!  Control Register 2 */
```

Definition at line 158 of file register.hpp.

6.17.2.94 MPL3115_CTRL_REG3

```
#define MPL3115_CTRL_REG3 0x28 /*!  Control Register 3 */
```

Definition at line 159 of file register.hpp.

6.17.2.95 MPL3115_CTRL_REG4

```
#define MPL3115_CTRL_REG4 0x29 /*!  Control Register 4 */
```

Definition at line 160 of file register.hpp.

6.17.2.96 MPL3115_CTRL_REG5

```
#define MPL3115_CTRL_REG5 0x2A /*!  Control Register 5 */
```

Definition at line 161 of file register.hpp.

6.17.2.97 MPL3115_DR_STATUS

```
#define MPL3115_DR_STATUS 0x06 /*!  Sensor Status Register */
```

Definition at line 125 of file register.hpp.

6.17.2.98 MPL3115_F_DATA

```
#define MPL3115_F_DATA 0X0E /*!  FIFO 8-bit Data Access */
```

Definition at line 133 of file register.hpp.

6.17.2.99 MPL3115_F_SETUP

```
#define MPL3115_F_SETUP 0x0F /*!  FIFO Setup Register */
```

Definition at line 134 of file register.hpp.

6.17.2.100 MPL3115_F_STATUS

```
#define MPL3115_F_STATUS 0x0D /*!  FIFO Status Register */
```

Definition at line 132 of file register.hpp.

6.17.2.101 MPL3115_I2C_ADDR

```
#define MPL3115_I2C_ADDR 0x60 /*!  I2C address */
```

Definition at line 118 of file register.hpp.

6.17.2.102 MPL3115_INT_SOURCE

```
#define MPL3115_INT_SOURCE 0x12 /*!  Interrupt Source Register */
```

Definition at line 137 of file register.hpp.

6.17.2.103 MPL3115_OFF_H

```
#define MPL3115_OFF_H 0x2D /*!  Altitude Data User Offset Register */
```

Definition at line 164 of file register.hpp.

6.17.2.104 MPL3115_OFF_P

```
#define MPL3115_OFF_P 0x2B /*!  Pressure Data User Offset */
```

Definition at line 162 of file register.hpp.

6.17.2.105 MPL3115_OFF_T

```
#define MPL3115_OFF_T 0x2C /*!  Temperature Data User Offset */
```

Definition at line 163 of file register.hpp.

6.17.2.106 MPL3115_OUT_P_CSB

```
#define MPL3115_OUT_P_CSB 0x02 /*!  Pressure Data Out CSB */
```

Definition at line 121 of file register.hpp.

6.17.2.107 MPL3115_OUT_P_DELTA

```
#define MPL3115_OUT_P_DELTA_MSB 0x07 /*!  Pressure Data Out Delta MSB */
```

Definition at line 126 of file register.hpp.

6.17.2.108 MPL3115_OUT_P_DELTA_CSB

```
#define MPL3115_OUT_P_DELTA_CSB 0x08 /*!  Pressure Data Out Delta CSB */
```

Definition at line 127 of file register.hpp.

6.17.2.109 MPL3115_OUT_P_DELTA_LSB

```
#define MPL3115_OUT_P_DELTA_LSB 0x09 /*!  Pressure Data Out Delta LSB */
```

Definition at line 128 of file register.hpp.

6.17.2.110 MPL3115_OUT_P_LSB

```
#define MPL3115_OUT_P_LSB 0x03 /*!  Pressure Data Out LSB */
```

Definition at line 122 of file register.hpp.

6.17.2.111 MPL3115_OUT_P_MSB

```
#define MPL3115_OUT_P_MSB 0x01 /*!  Pressure Data Out MSB */
```

Definition at line 120 of file register.hpp.

6.17.2.112 MPL3115_OUT_T_DELTA_LSB

```
#define MPL3115_OUT_T_DELTA_LSB 0x0B /*!  Temperature Data Out Delta LSB */
```

Definition at line 130 of file register.hpp.

6.17.2.113 MPL3115_OUT_T_DELTA_MSB

```
#define MPL3115_OUT_T_DELTA_MSB 0x0A /*!  Temperature Data Out Delta MSB */
```

Definition at line 129 of file register.hpp.

6.17.2.114 MPL3115_OUT_T_LSB

```
#define MPL3115_OUT_T_LSB 0x05 /*!  Temperature Data Out LSB */
```

Definition at line 124 of file register.hpp.

6.17.2.115 MPL3115_OUT_T_MSB

```
#define MPL3115_OUT_T_MSB 0x04 /*!  Temperature Data Out MSB */
```

Definition at line 123 of file register.hpp.

6.17.2.116 MPL3115_P_MAX_CSB

```
#define MPL3115_P_MAX_CSB 0x22 /*!  Maximum Pressure Data Out CSB */
```

Definition at line 153 of file register.hpp.

6.17.2.117 MPL3115_P_MAX_LSB

```
#define MPL3115_P_MAX_LSB 0x23 /*!  Maximum Pressure Data Out LSB */
```

Definition at line 154 of file register.hpp.

6.17.2.118 MPL3115_P_MAX_MSB

```
#define MPL3115_P_MAX_MSB 0x21 /*!  Maximum Pressure Data Out MSB */
```

Definition at line 152 of file register.hpp.

6.17.2.119 MPL3115_P_MIN_CSB

```
#define MPL3115_P_MIN_CSB 0x1D /*!  Minimum Pressure Data Out CSB */
```

Definition at line 148 of file register.hpp.

6.17.2.120 MPL3115_P_MIN_LSB

```
#define MPL3115_P_MIN_LSB 0x1E /*!  Minimum Pressure Data Out LSB */
```

Definition at line 149 of file register.hpp.

6.17.2.121 MPL3115_P_MIN_MSB

```
#define MPL3115_P_MIN_MSB 0x1C /*!  Minimum Pressure Data Out MSB */
```

Definition at line 147 of file register.hpp.

6.17.2.122 MPL3115_P_TGT_LSB

```
#define MPL3115_P_TGT_LSB 0x17 /*!  Pressure Target LSB */
```

Definition at line 142 of file register.hpp.

6.17.2.123 MPL3115_P_TGT_MSB

```
#define MPL3115_P_TGT_MSB 0x16 /*!  Pressure Target MSB */
```

Definition at line 141 of file register.hpp.

6.17.2.124 MPL3115_P_WND_LSB

```
#define MPL3115_P_WND_LSB 0x1A /*!  Pressure/Altitude Window LSB */
```

Definition at line 145 of file register.hpp.

6.17.2.125 MPL3115_P_WND_MSB

```
#define MPL3115_P_WND_MSB 0x19 /*!  Pressure/Altitude Window MSB */
```

Definition at line 144 of file register.hpp.

6.17.2.126 MPL3115_PT_DATA_CFG

```
#define MPL3115_PT_DATA_CFG 0x13 /*!  PT Data Configuration Register */
```

Definition at line 138 of file register.hpp.

6.17.2.127 MPL3115_STATUS

```
#define MPL3115_STATUS 0x00 /*!  Sensor Status Register */
```

Definition at line 119 of file register.hpp.

6.17.2.128 MPL3115_SYSMOD

```
#define MPL3115_SYSMOD 0x11 /*!  System Mode Register */
```

Definition at line 136 of file register.hpp.

6.17.2.129 MPL3115_T_MAX_LSB

```
#define MPL3115_T_MAX_LSB 0x25 /*!  Maximum Temperature Data Out LSB */
```

Definition at line 156 of file register.hpp.

6.17.2.130 MPL3115_T_MAX_MSB

```
#define MPL3115_T_MAX_MSB 0X24 /*!  Maximum Temperature Data Out MSB */
```

Definition at line 155 of file register.hpp.

6.17.2.131 MPL3115_T_MIN_LSB

```
#define MPL3115_T_MIN_LSB 0x20 /*!  Minimum Temperature Data Out LSB */
```

Definition at line 151 of file register.hpp.

6.17.2.132 MPL3115_T_MIN_MSB

```
#define MPL3115_T_MIN_MSB 0x1F /*!  Minimum Temperature Data Out MSB */
```

Definition at line 150 of file register.hpp.

6.17.2.133 MPL3115_T_TGT

```
#define MPL3115_T_TGT 0x18 /*!  Temperature Target */
```

Definition at line 143 of file register.hpp.

6.17.2.134 MPL3115_T_WND

```
#define MPL3115_T_WND 0X1B /*!  Temperature Window */
```

Definition at line 146 of file register.hpp.

6.17.2.135 MPL3115_TIME_DLY

```
#define MPL3115_TIME_DLY 0x10 /*!  Time Delay */
```

Definition at line 135 of file register.hpp.

6.17.2.136 MPL3115_WHO_AM_I

```
#define MPL3115_WHO_AM_I 0x0C /*!  Device Identification Register */
```

Definition at line 131 of file register.hpp.

6.18 register.hpp

```
00001
00009 #ifndef _REGISTER_HPP_
00010 #define _REGISTER_HPP_
00011
00012 //------------------------------------------------------
00013 //                  BNO055 Registers
00014 //------------------------------------------------------
00015
00016                                          // bit 7 || bit 6 || bit 5 || bit 4 || bit 3 || bit 2 || bit
       1 || bit 0
00017 #define BNO055_MAG_RADIUS_MSB        0x6A
00018 #define BNO055_MAG_RADIUS_LSB        0x69
00019 #define BNO055_ACC_RADIUS_MSB        0x68
00020 #define BNO055_ACC_RADIUS_LSB        0x67
00021 #define BNO055_GYR_OFFSET_Z_MSB      0x66
00022 #define BNO055_GYR_OFFSET_Z_LSB      0x65
00023 #define BNO055_GYR_OFFSET_Y_MSB      0x64
00024 #define BNO055_GYR_OFFSET_Y_LSB      0x63
00025 #define BNO055_GYR_OFFSET_X_MSB      0x62
00026 #define BNO055_GYR_OFFSET_X_LSB      0x61
00027 #define BNO055_MAG_OFFSET_Z_MSB      0x60
00028 #define BNO055_MAG_OFFSET_Z_LSB      0x5F
00029 #define BNO055_MAG_OFFSET_Y_MSB      0x5E
00030 #define BNO055_MAG_OFFSET_Y_LSB      0x5D
00031 #define BNO055_MAG_OFFSET_X_MSB      0x5C
00032 #define BNO055_MAG_OFFSET_X_LSB      0x5B
00033 #define BNO055_ACC_OFFSET_Z_MSB      0x5A
00034 #define BNO055_ACC_OFFSET_Z_LSB      0x59
00035 #define BNO055_ACC_OFFSET_Y_MSB      0x58
00036 #define BNO055_ACC_OFFSET_Y_LSB      0x57
00037 #define BNO055_ACC_OFFSET_X_MSB      0x56
00038 #define BNO055_ACC_OFFSET_X_LSB      0x55
00039 // WIP FINISH COMMENTS based on the BNO055 Datasheet
00040                                          // bit 7      ||      bit 6       ||      bit 5       ||
       bit 4       ||      bit 3       ||      bit 2       ||      bit 1       ||      bit 0
00041
00042 #define BNO055_AXIS_MAP_SIGN         0x42 //
                                  ||Remapped X-Ax Sign||Remapped Y-Ax Sign||Remapped Z-Ax Sign
00043 #define BNO055_AXIS_MAP_CONFIG       0x41 //                                    ||        Remapped Z axis
       value         ||         Remapped Y axis value        ||        Remapped X axis value
00044 #define BNO055_TEMP_SOURCE           0x40 //
                                                          ||          TEMP_Source <1:0>
00045 #define BNO055_SYS_TRIGGER           0x3F // CLK_SEL    ||     RST_INT      ||      RST_SYS     ||
                                                                    ||    Self_Test
00046 #define BNO055_PWR_MODE              0x3E //
00047 #define BNO055_OPR_MODE              0x3D //
00048 #define BNO055_Reserved              0x3C //
00049 #define BNO055_UNIT_SEL              0x3B //
00050 #define BNO055_SYS_ERR               0x3A //
00051 #define BNO055_SYS_STATUS            0x39 //
00052 #define BNO055_SYS_CLK_STA           0x38 //
00053 #define BNO055_INT_STA               0x37 //
00054 #define BNO055_ST_RESULT             0x36 //
00055 #define BNO055_CALIB_STAT            0x35 //
00056 #define BNO055_TEMP                  0x34 //
00057 #define BNO055_GRV_Data_Z_M          0x33 //
00058 #define BNO055_GRV_Data_Z_L          0x32 //
00059 #define BNO055_GRV_Data_Y_M          0x31 //
00060 #define BNO055_GRV_Data_Y_L          0x30 //
00061 #define BNO055_GRV_Data_X_M          0x2F //
00062 #define BNO055_GRV_Data_X_L          0x2E //
00063 #define BNO055_LIA_Data_Z_MB         0x2D //
00064 #define BNO055_LIA_Data_Z_LS         0x2C //
00065 #define BNO055_LIA_Data_Y_MB         0x2B //
00066 #define BNO055_LIA_Data_Y_LS         0x2A //
00067 #define BNO055_LI A_Data_X_MB        0x29 //
00068 #define BNO055_LIA_Data_X_LS         0x28 //
00069 #define BNO055_QUA_Data_z_M          0x27 //
00070 #define BNO055_QUA_Data_z_LS         0x26 //
00071 #define BNO055_QUA_Data_y_M          0x25 //
00072 #define BNO055_QUA_Data_y_LS         0x24 //
00073 #define BNO055_QUA_Data_x_M          0x23 //
00074 #define BNO055_QUA_Data_x_LS         0x22 //
00075 #define BNO055_QUA_Data_w_M          0x21 //
00076 #define BNO055_QUA_Data_w_L          0x20 //
00077 #define BNO055_EUL_Pitch_MSB         0x1F //
00078 #define BNO055_EUL_Pitch_LSB         0x1E //
00079
00080 #define BNO055_EUL_Roll_MSB          0x1D
00081 #define BNO055_EUL_Roll_LSB          0x1C
00082 #define BNO055_EUL_Heading_MSB       0x1B
00083 #define BNO055_EUL_Heading_LSB       0x1A
00084 #define BNO055_GYR_DATA_Z_MSB        0x19
00085 #define BNO055_GYR_DATA_Z_LSB        0x18
00086 #define BNO055_GYR_DATA_Y_MSB        0x17
```

```
00087 #define BNO055_GYR_DATA_Y_LSB        0x16
00088 #define BNO055_GYR_DATA_X_MSB        0x15
00089 #define BNO055_GYR_DATA_X_LSB        0x14
00090 #define BNO055_MAG_DATA_Z_MSB        0x13
00091 #define BNO055_MAG_DATA_Z_LSB        0x12
00092 #define BNO055_MAG_DATA_Y_MSB        0x11
00093 #define BNO055_MAG_DATA_Y_LSB        0x10
00094 #define BNO055_MAG_DATA_X_MSB        0x0F
00095 #define BNO055_MAG_DATA_X_LSB        0x0E
00096 #define BNO055_ACC_DATA_Z_MSB        0x0D
00097 #define BNO055_ACC_DATA_Z_LSB        0x0C
00098 #define BNO055_ACC_DATA_Y_MSB        0x0B
00099 #define BNO055_ACC_DATA_Y_LSB        0x0A
00100 #define BNO055_ACC_DATA_X_MSB        0x09
00101 #define BNO055_PWR_ACC_DATA_X_LSB    0x08
00102 #define BNO055_Page_ID               0x07
00103 #define BNO055_BL_Rev_ID             0x06
00104 #define BNO055_SW_REV_ID_MSB         0x05
00105 #define BNO055_SW_REV_ID_LSB         0x04
00106 #define BNO055_GYR_ID                0x03
00107 #define BNO055_MAG_ID                0x02
00108 #define BNO055_ACC_ID                0x01
00109 #define BNO055_CHIP_ID               0x00
00114 //-------------------------------------------------------
00115 //                  MPL3115A2 Registers
00116 //-------------------------------------------------------
00117
00118 #define MPL3115_I2C_ADDR             0x60
00119 #define MPL3115_STATUS               0x00
00120 #define MPL3115_OUT_P_MSB            0x01
00121 #define MPL3115_OUT_P_CSB            0x02
00122 #define MPL3115_OUT_P_LSB            0x03
00123 #define MPL3115_OUT_T_MSB            0x04
00124 #define MPL3115_OUT_T_LSB            0x05
00125 #define MPL3115_DR_STATUS            0x06
00126 #define MPL3115_OUT_P_DELTA_MSB      0x07
00127 #define MPL3115_OUT_P_DELTA_CSB      0x08
00128 #define MPL3115_OUT_P_DELTA_LSB      0x09
00129 #define MPL3115_OUT_T_DELTA_MSB      0x0A
00130 #define MPL3115_OUT_T_DELTA_LSB      0x0B
00131 #define MPL3115_WHO_AM_I             0x0C
00132 #define MPL3115_F_STATUS             0x0D
00133 #define MPL3115_F_DATA               0X0E
00134 #define MPL3115_F_SETUP              0x0F
00135 #define MPL3115_TIME_DLY             0x10
00136 #define MPL3115_SYSMOD               0x11
00137 #define MPL3115_INT_SOURCE           0x12
00138 #define MPL3115_PT_DATA_CFG          0x13
00139 #define MPL3115_BAR_IN_MSB           0x14
00140 #define MPL3115_BAR_IN_LSB           0x15
00141 #define MPL3115_P_TGT_MSB            0x16
00142 #define MPL3115_P_TGT_LSB            0x17
00143 #define MPL3115_T_TGT                0x18
00144 #define MPL3115_P_WND_MSB            0x19
00145 #define MPL3115_P_WND_LSB            0x1A
00146 #define MPL3115_T_WND                0X1B
00147 #define MPL3115_P_MIN_MSB            0x1C
00148 #define MPL3115_P_MIN_CSB            0x1D
00149 #define MPL3115_P_MIN_LSB            0x1E
00150 #define MPL3115_T_MIN_MSB            0x1F
00151 #define MPL3115_T_MIN_LSB            0x20
00152 #define MPL3115_P_MAX_MSB            0x21
00153 #define MPL3115_P_MAX_CSB            0x22
00154 #define MPL3115_P_MAX_LSB            0x23
00155 #define MPL3115_T_MAX_MSB            0X24
00156 #define MPL3115_T_MAX_LSB            0x25
00157 #define MPL3115_CTRL_REG1            0x26
00158 #define MPL3115_CTRL_REG2            0x27
00159 #define MPL3115_CTRL_REG3            0x28
00160 #define MPL3115_CTRL_REG4            0x29
00161 #define MPL3115_CTRL_REG5            0x2A
00162 #define MPL3115_OFF_P                0x2B
00163 #define MPL3115_OFF_T                0x2C
00164 #define MPL3115_OFF_H                0x2D
00166 #endif
```

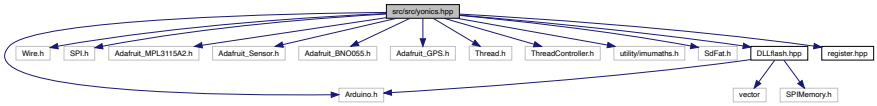
6.19 src/src/yonics.hpp File Reference

The main header file for the CUSRL_Avionics Code Base.

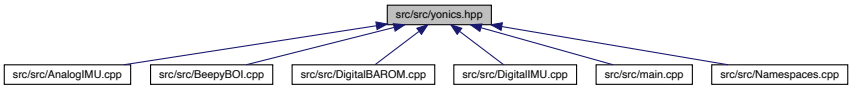
```
#include <Arduino.h>
#include <Wire.h>
```

```
#include <SPI.h>
#include <Adafruit_MPL3115A2.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <Adafruit_GPS.h>
#include "Thread.h"
#include <ThreadController.h>
#include <utility/imumaths.h>
#include <SdFat.h>
#include "DLLflash.hpp"
#include "register.hpp"
```

Include dependency graph for yonics.hpp:



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



Classes

- struct ACCELdata

  HIGH-G Accelerometer Struct.
- struct IMUdata

  IMU Struct.
- struct BAROMdata

  Barometer Struct.
- class HIGHG_ACCEL

  ADXL377 High-G Accelerometer Class.
- class DigitalIMU

  BNO055 IMU Class.
- class DigitalBAROM

  MPL3115A2 Barometer Class.
- class BeepyBOI

  Piezo Buzzer Class.

Namespaces

- I2C
- INITS
- PROTOTHREADING

Functions

- bool I2C::write_reg (uint8_t i2c, uint8_t addr, uint8_t val)
- bool I2C::read_regs (uint8_t i2c, uint8_t addr, uint8_t *data, uint8_t num)
- bool I2C::read_regs (uint8_t i2c, uint8_t *data, uint8_t num)

6.19.1 Detailed Description

The main header file for the CUSRL_Avionics Code Base.

Definition in file yonics.hpp.

6.20 yonics.hpp

```
00001
00005 #ifndef _YONICS_HPP_
00006 #define _YONICS_HPP_
00007
00008 // HEADER FILES
00009 /*
00010 *   All the header files that the project uses are linked here.
00011 *   Libraries, Sensors, Etc...
00012 */
00013 #include <Arduino.h>
00014 #include <Wire.h>
00015 #include <SPI.h>
00016 #include <Adafruit_MPL3115A2.h>
00017 #include <Adafruit_Sensor.h>
00018 #include <Adafruit_BNO055.h>
00019 #include <Adafruit_GPS.h>
00020 #include "Thread.h"
00021 #include <ThreadController.h>
00022 #include <utility/imumaths.h>
00023 #include <SdFat.h>
00024 #include "DLLflash.hpp"
00025 #include "register.hpp"
00026
00028
00031 struct ACCELdata {
00033     float x;
00034
00036     float y;
00037
00039     float z;
00040
00042     uint32_t t;
00043 };
00044
00046
00049 struct IMUdata {
00051     double orient_euler[3] = {0,0,0};
00052
00054     double gyro_fused[3] = {0,0,0};
00055
00057     double accel_fused[3] = {0,0,0};
00058
00060     double accel_raw[3] = {0,0,0};
00061
00063     double gyro_raw[3] = {0,0,0};
00064
00066     double magnetometer[3] = {0,0,0};
00067
00069     double orient_quat[4] = {0,0,0,0};
00070
00072     uint32_t t = 0;
00073 };
00074
00076
00079 struct BAROMdata {
00081     float pressure = 0;
00082
00084     float altitude = 0;
00085
00087     float temperature = 0;
00088
```

```
00090     uint32_t t = 0;
00091 };
00092
00094
00097 class HIGHG_ACCEL {
00098     private:
00099         int xPin, yPin, zPin;
00100         int bitDepth;
00101         int offset;
00102         float ratio;
00103         int maxG = 200;
00104
00105         void init();
00106         float formatVal(int rawVal);
00107
00108     public:
00109
00111         HIGHG_ACCEL();
00112
00114         HIGHG_ACCEL(int xPin, int yPin, int zPin);
00115
00117         HIGHG_ACCEL(int xPin, int yPin, int zPin, bool highBitDepth);
00118
00120
00124         void sample(ACCELdata* data);
00125 };
00126
00128
00132 class DigitalIMU {
00133     private:
00134         Adafruit_BNO055 board;
00135         sensors_event_t event;
00136         imu::Quaternion quat;
00137         imu::Vector<3> accel;
00138
00139     public:
00140
00142         DigitalIMU();
00143
00145         DigitalIMU(int32_t sensorID, uint8_t address);
00146
00148         bool begin();
00149
00151
00155         void sample(IMUdata* data);
00156 };
00157
00159
00162 class DigitalBAROM {
00163     private:
00164     public:
00165
00167         DigitalBAROM();
00168
00170         bool begin();
00171
00173
00177         bool sample(BAROMdata* data);
00178 };
00179
00181
00185 class BeepyBOI {
00186     private:
00187
00189         int pin;
00190
00191         // Some predefined tones for simplicity in the code...
00192         // Define all the tones here.
00193
00195         int errTone = 300;
00196
00198         int lowTone = 220;
00199
00201         int midTone = 440;
00202
00204         int  hiTone = 880;
00205
00206     public:
00207
00209         BeepyBOI();
00210
00212         BeepyBOI(int pin);
00213
00214
00215         void hello();
00216         void error();
00217         void countdown(int s);
```

```
00218         void lowBeep();
00219         void midBeep();
00220         void  hiBeep();
00221         void bombBeep();
00222 };
00223
00230 namespace I2C
00231 {
00238     extern bool write_reg(uint8_t i2c, uint8_t addr, uint8_t val);
00239
00248     extern bool read_regs(uint8_t i2c, uint8_t addr, uint8_t *data, uint8_t num);
00249
00257     extern bool read_regs(uint8_t i2c, uint8_t *data, uint8_t num);
00258 };
00259
00264 namespace INITS
00265 {
00266     //-----
00267     // PIN ASSIGNMENTS
00268     //-----
00270     extern int speakerPin;
00271
00273     extern int highG_xPin;
00274
00276     extern int highG_yPin;
00277
00279     extern int highG_zPin;
00280
00281     //-----
00282     // CLASS INITIALIZATIONS
00283     //-----
00284
00286     extern DigitalIMU IMU;
00287
00289     extern DigitalBAROM BAROM;
00290
00292     extern HIGHG_ACCEL HIGHG;
00293
00295     extern BeepyBOI berp;
00296
00297     //-----
00298     // POINTERS
00299     //-----
00300
00302     extern DLLflash* flash;
00303
00304     //-----
00305     // DATA STRUCTS
00306     //-----
00307
00309     extern IMUdata imu_data;
00310
00312     extern BAROMdata barom_data;
00313
00315     extern ACCELdata accel_data;
00316 };
00317
00336 namespace PROTOTHREADING
00337 {
00338     // PROTOTHREADING TIME INTERVALS
00339     /*
00340     *   Defining the time intervals (in milliseconds) at which to call the "threads"
00341     *
00342     *   Every important task (i.e. sampling, writing to flash, RF, etc...) has a "thread"
00343     *   and their intervals are defined here.
00344     */
00345
00347     extern int interval_IMU;
00348
00350     extern int interval_BAROM;
00351
00353     extern int interval_ACCEL;
00354
00355     // PROTOTHREADING Declaration and Definitions
00356     //
00357     //  All the required thread objects and pointers are declared and defined here
00358     //
00359
00361     extern ThreadController thread_control;
00362
00363     //-----
00364     // Every thread is a pointer that is pointing to an object, or instance, of the Thread class
      initiated dynamically in order to use them in any scope
00365     //-----
00366
00368     extern Thread* ThreadIMU;
00369
```

```
00371     extern Thread* ThreadBAROM;
00372
00374     extern Thread* ThreadACCEL;
00375 };
00376
00377 #endif
```