# Shield Code 2.0

Generated by Doxygen 1.11.0

**1 Class Index** 1

1.1 Class List . . . 1

**2 File Index** 3

2.1 File List . . . 3

**3 Class Documentation** 5

3.1 PDC Class Reference . . . 5

3.1.1 Constructor & Destructor Documentation . . . 5

3.1.1.1 PDC() [1/2] . . . 5

3.1.1.2 PDC() [2/2] . . . 6

3.1.2 Member Function Documentation . . . 6

3.1.2.1 flush() . . . 6

3.1.2.2 init() . . . 6

3.1.2.3 write() . . . 6

3.1.2.4 write_array() . . . 6

3.1.2.5 write_vec() . . . 7

3.2 Pip Class Reference . . . 7

**4 File Documentation** 9

4.1 include/imu.hpp File Reference . . . 9

4.1.1 Detailed Description . . . 9

4.1.2 Function Documentation . . . 9

4.1.2.1 initIMU() . . . 9

4.2 imu.hpp . . . 10

4.3 include/PDC.hpp File Reference . . . 10

4.3.1 Detailed Description . . . 10

4.4 PDC.hpp . . . 11

4.5 include/Pip.hpp File Reference . . . 11

4.5.1 Detailed Description . . . 12

4.6 Pip.hpp . . . 12

**Index** 13

# Chapter 1

# Class Index

## 1.1 Class List

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

PDC . . . . . 5
Pip . . . . . 7

# Chapter 2

# File Index

## 2.1 File List

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

include/imu.hpp
Header file for the IMU library for Dartmouth's 317 Lab . . . . . 9
include/PDC.hpp
Header file for the PDC library for Dartmouth's 317 Lab . . . . . 10
include/Pip.hpp
Header file for the Pip library for Dartmouth's 317 Lab . . . . . 11

# Chapter 3

# Class Documentation

## 3.1 PDC Class Reference

### Public Member Functions

- PDC ()

  *Default constructor for the PDC class.*
- PDC (RingBuffer *pRx_buffer, RingBuffer *pTx_buffer)

  *Constructor for the PDC class.*
- void init ()

  *Initializes the PDC.*
- void **send** (uint16_t imu_data, uint32_t imu_timestamp, uint16_t sweep_data, uint32_t sweep_timestamp, uint8_t buffer_data, uint32_t buffer_timestamp)

  *Sends shield data over UART using the PDC.*
- size_t write (const int uc_data)

  *Writes a single byte to the PDC buffer.*
- size_t write_array (uint8_t const *uc_data, int size)

  *Writes an array of bytes to the PDC buffer.*
- void flush ()

  *Flushes the PDC buffer.*
- void write_vec (Vector< uint8_t > data)

  *Writes a Vector of bytes to the PDC buffer.*
- void **timing_tester** ()

  *Tester function for timing PDC functions.*

### 3.1.1 Constructor & Destructor Documentation

#### 3.1.1.1 PDC() [1/2]

```
PDC::PDC ()  [inline]
```

Default constructor for the PDC class.

Shouldn't ever be used, but is here for completeness.

#### 3.1.1.2 PDC() [2/2]

```
PDC::PDC (
            RingBuffer * pRx_buffer,
            RingBuffer * pTx_buffer)  [inline]
```

Constructor for the PDC class.

Takes RingBuffer pointers as parameters, which are used to buffer data into the PDC.

### 3.1.2 Member Function Documentation

#### 3.1.2.1 flush()

```
void PDC::flush ()
```

Flushes the PDC buffer.

Used to send any remaining data in the buffer.

When sending larger array, most data usually goes to buffer before PDC is available to send it. So, have to flush before moving on.

#### 3.1.2.2 init()

```
void PDC::init ()
```

Initializes the PDC.

Sets up the PDC for UART communication. Enables transmit register, sets up ring buffer, and initializes data buffer.

#### 3.1.2.3 write()

```
size_t PDC::write (
            const int uc_data)
```

Writes a single byte to the PDC buffer.

Used inside send()

#### 3.1.2.4 write_array()

```
size_t PDC::write_array (
            uint8_t const * uc_data,
            int size)
```

Writes an array of bytes to the PDC buffer.

Deprecated, but may be useful for debugging.

#### 3.1.2.5 write_vec()

```
void PDC::write_vec (
            Vector< uint8_t > data)
```

Writes a Vector of bytes to the PDC buffer.

Used inside send()

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

- include/PDC.hpp
- src/PDC.cpp

## 3.2 Pip Class Reference

### Public Member Functions

- **Pip** ()

  *Default constructor for the Pip class.*
- **Pip** (int delay_us, uint16_t avg_num, uint16_t num_samples, uint16_t min, uint16_t max, uint8_t dac_↩ channel)

  *Constructor for the Pip class. All parameters modifiable.*
- void **sweep** ()

  *Sweeps the DAC output from min to max. Step length, delay, and number of samples are all set by the constructor.*

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

- include/Pip.hpp
- src/Pip.cpp

# Chapter 4

# File Documentation

## 4.1 include/imu.hpp File Reference

Header file for the IMU library for Dartmouth's 317 Lab.

```
#include <Arduino.h>
#include <Wire.h>
#include <LIS3MDL.h>
#include <LSM6.h>
```

### Functions

- void initIMU (LIS3MDL ∗compass, LSM6 ∗gyro)

  *Initializes the IMU.*
- void **sampleIMU** (LIS3MDL ∗compass, LSM6 ∗gyro, int ∗data)

  *Gets the IMU data.*

### 4.1.1 Detailed Description

Header file for the IMU library for Dartmouth's 317 Lab.

This library is used to query the Pololu IMU. This is where we get location, acceleration, and gyroscopic data. This is basically identical to the library used on the Uno based board, which was written by Max Roberts circa 2015, ed. Leah Ryu 2019.

Deprecated code was removed by Sean Wallace 2024-06-20.

Author: Max Roberts Date: 2015-06-22

### 4.1.2 Function Documentation

#### 4.1.2.1 initIMU()

```
void initIMU (
            LIS3MDL * compass,
            LSM6 * gyro)
```

Initializes the IMU.

Info about registers and settings can be found in the initIMU() function in imu.cpp.

## 4.2 imu.hpp

Go to the documentation of this file.

```
00001
00014 #ifndef IMU_HPP
00015 #define IMU_HPP
00016 #include <Arduino.h>
00017 #include <Wire.h>
00018 #include <LIS3MDL.h>
00019 #include <LSM6.h>
00025 void initIMU(LIS3MDL* compass, LSM6* gyro);
00029 void sampleIMU(LIS3MDL* compass, LSM6* gyro, int* data);
00030 #endif
```

## 4.3 include/PDC.hpp File Reference

Header file for the PDC library for Dartmouth's 317 Lab.

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

### Classes

- class PDC

### Macros

- #define **UART_BASE** 0x400E0800
- #define **UART_PERIPH_TPR_ADDR** (UART_BASE + 0x108)
- #define **UART_PERIPH_TCR_ADDR** (UART_BASE + 0x10C)
- #define **UART_PERIPH_TNPR_ADDR** (UART_BASE + 0x118)
- #define **UART_PERIPH_TNCR_ADDR** (UART_BASE + 0x11C)
- #define **UART_PERIPH_PTCR_ADDR** (UART_BASE + 0x120)
- #define **UART_PERIPH_PTSR_ADDR** (UART_BASE + 0x124)
- #define **TXTEN_MASK** (1<<8)
- #define **UART_ID** 8

### 4.3.1 Detailed Description

Header file for the PDC library for Dartmouth's 317 Lab.

This library is used to manage UART communication on the Arduino Due using the Peripheral DMA Controller (PDC). This allows for non-blocking communication, so that we can run other processes while sending out data.

Author: Sean Wallace Date: 2024-06-20

## 4.4 PDC.hpp

Go to the documentation of this file.

```
00001
00013 #ifndef PDC_HPP
00014 #define PDC_HPP
00015 #include <Arduino.h>
00016 #include <Vector.h>
00017
00018 // Defining relevant UART registers
00019 #define UART_BASE 0x400E0800
00020 #define UART_PERIPH_TPR_ADDR (UART_BASE + 0x108) // Transmit pointer register
00021 #define UART_PERIPH_TCR_ADDR (UART_BASE + 0x10C) // Transmit counter register
00022 #define UART_PERIPH_TNPR_ADDR (UART_BASE + 0x118) // Transmit next pointer register
00023 #define UART_PERIPH_TNCR_ADDR (UART_BASE + 0x11C) // Transmit next counter register
00024 #define UART_PERIPH_PTCR_ADDR (UART_BASE + 0x120) // Peripheral transfer control register
00025 #define UART_PERIPH_PTSR_ADDR (UART_BASE + 0x124) // Peripheral transfer status register
00026
00027 #define TXTEN_MASK (1«8) //mask used to enable UART transmitter
00028
00029 #define UART_ID    8 // UART Peripheral ID
00030
00031 class PDC {
00032 private:
00033     // pointers to UART registers
00034     volatile void* const p_UART_TPR;
00035     volatile uint32_t* const p_UART_TCR;
00036     volatile uint32_t* const p_UART_TNPR;
00037     volatile uint32_t* const p_UART_TNCR;
00038     volatile uint32_t* const p_UART_PTCR;
00039     volatile uint32_t* const p_UART_PTSR;
00040
00045     RingBuffer *_rx_buffer;
00046     RingBuffer *_tx_buffer;
00047
00056     void process_data(uint16_t imu_data, uint32_t imu_timestamp, uint16_t sweep_data, uint32_t
      sweep_timestamp, uint8_t buffer_data, uint32_t buffer_timestamp);
00057
00064     template <typename T>
00065     uint8_t convert_data(const T& value);
00066
00070     uint8_t serialized_storage[20];
00076     Vector<uint8_t> serialized_data;
00077
00078 public:
00084     PDC()
00085         : p_UART_TPR((void*)UART_PERIPH_TPR_ADDR),
00086           p_UART_TCR((uint32_t*)UART_PERIPH_TCR_ADDR),
00087           p_UART_TNPR((uint32_t*)UART_PERIPH_TNPR_ADDR),
00088           p_UART_TNCR((uint32_t*)UART_PERIPH_TNCR_ADDR),
00089           p_UART_PTCR((uint32_t*)UART_PERIPH_PTCR_ADDR),
00090           p_UART_PTSR((uint32_t*)UART_PERIPH_PTSR_ADDR)
00091     {}
00092
00098     PDC(RingBuffer *pRx_buffer, RingBuffer *pTx_buffer)
00099         : p_UART_TPR((void*)UART_PERIPH_TPR_ADDR),
00100           p_UART_TCR((uint32_t*)UART_PERIPH_TCR_ADDR),
00101           p_UART_TNPR((uint32_t*)UART_PERIPH_TNPR_ADDR),
00102           p_UART_TNCR((uint32_t*)UART_PERIPH_TNCR_ADDR),
00103           p_UART_PTCR((uint32_t*)UART_PERIPH_PTCR_ADDR),
00104           p_UART_PTSR((uint32_t*)UART_PERIPH_PTSR_ADDR)
00105     {
00106         _rx_buffer = pRx_buffer;
00107         _tx_buffer = pTx_buffer;
00108     }
00109
00115     void init();
00120     void send(uint16_t imu_data, uint32_t imu_timestamp, uint16_t sweep_data, uint32_t
      sweep_timestamp, uint8_t buffer_data,
00121     uint32_t buffer_timestamp);
00128     size_t write(const int uc_data);
00134     size_t write_array(uint8_t const *uc_data, int size);
00143     void flush();
00150     void write_vec(Vector<uint8_t> data);
00154     void timing_tester();
00155 };
00156
00157 #endif
```

## 4.5 include/Pip.hpp File Reference

Header file for the Pip library for Dartmouth's 317 Lab.

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

**Classes**

- class Pip

**Macros**

- #define **SWEEP_DEFAULT_DELAY** 1000
- #define **SWEEP_DEFAULT_AVG_NUM** 25
- #define **SWEEP_DEFAULT_SAMPLES** 28
- #define **SWEEP_DEFAULT_MIN** 0
- #define **SWEEP_DEFAULT_MAX** 4095
- #define **SWEEP_MAX_SAMPLES** 256
- #define **DEFAULT_DAC_CHANNEL** DAC1

### 4.5.1 Detailed Description

Header file for the Pip library for Dartmouth's 317 Lab.

This library manages the voltage sweep on the PIP sensors using the Arduino Due's onboard DAC. It is basically a vastly simplified version of the Pip and DAC libraries in the Isinglass shield code. written by Max Roberts in 2015. We are able to use this simplified version because of how nice the Due is :)

**Author**

Sean Wallace

**Date**

2024-06-20

## 4.6 Pip.hpp

Go to the documentation of this file.

```
00001
00013 #ifndef PIP_HPP
00014 #define PIP_HPP
00015 #include <Arduino.h>
00016 #include <Wire.h>
00017 //default sweep parameters. can be modified with the constructor.
00018 #define SWEEP_DEFAULT_DELAY 1000
00019 #define SWEEP_DEFAULT_AVG_NUM 25
00020 #define SWEEP_DEFAULT_SAMPLES 28
00021 #define SWEEP_DEFAULT_MIN 0
00022 #define SWEEP_DEFAULT_MAX 4095
00023 #define SWEEP_MAX_SAMPLES 256
00024 #define DEFAULT_DAC_CHANNEL DAC1
00025
00026 class Pip{
00027     private:
00028         int delay_us;
00029         uint16_t avg_num;
00030         uint16_t num_samples;
00031         uint16_t min;
00032         uint16_t max;
00033         uint16_t data[SWEEP_MAX_SAMPLES];
00034         uint8_t dac_channel;
00035     public:
00039         Pip();
00043         Pip(int delay_us, uint16_t avg_num, uint16_t num_samples, uint16_t min, uint16_t max, uint8_t
      dac_channel);
00044
00049         void sweep();
00050 };
00051 #endif
```

# Index

flush
    PDC, 6

imu.hpp
    initIMU, 9
include/imu.hpp, 9, 10
include/PDC.hpp, 10, 11
include/Pip.hpp, 11, 12
init
    PDC, 6
initIMU
    imu.hpp, 9

PDC, 5
    flush, 6
    init, 6
    PDC, 5
    write, 6
    write_array, 6
    write_vec, 6
Pip, 7

write
    PDC, 6
write_array
    PDC, 6
write_vec
    PDC, 6