# RM3100driver

1.0

Generated by Doxygen 1.8.6

Mon Sep 8 2014 18:23:51

# Contents

# Chapter 1

# Main Page

## Project RM3100

### Author

Miguel Rasteiro
MEE-ESTG-IPLEIRIA
E-Mail: `2130904@my.ipleiria.pt`

### Version

1.0

This report documents a simple driver developed for the RM3100, from PNIcorp. This was developed in the scope of "HERMES - Sistemas de Interatividade entre Consumidores e Conteudos Digitais (Co-promocao 34149)" project.

# Chapter 2

# License

## GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the G-NU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law:

# Chapter 3

# Microchip License

**Microchip Software License Agreement**

# Chapter 4

# Module Index

## 4.1 Modules

Here is a list of all modules:

Chapter 5

# Data Structure Index

## 5.1 Data Structures

Here are the data structures with brief descriptions:

# Chapter 6

# File Index

## 6.1 File List

Here is a list of all files with brief descriptions:

# Chapter 7

# Module Documentation

## 7.1 Main

Entry point and main control.

Entry point and main control.

## 7.2 RM3100 Driver

Basic driver for RM3100.

### Files

- file **rm3100.c**

  *An I2C-based driver for PNI RM3100 Evaluation Board.*
- file **rm3100.h**

  *An I2C-based driver for PNI RM3100 Evaluation Board.*

### Data Structures

- struct **config**
- struct **sensor_xyz**

### Macros

- #define **POLL_REG** 0x00 /∗∗ POLLS A SINGLE MEASUREMENT ∗/
- #define **CMM_REG** 0x01 /∗∗ CONTINUOUS MEASUREMENT MODE REGISTER ∗/
- #define **CCX_MSB_REG** 0x04 /∗∗ CYCLE COUNT REGISTERS ∗/
- #define **CCX_LSB_REG** 0x05
- #define **CCY_MSB_REG** 0x06
- #define **CCY_LSB_REG** 0x07
- #define **CCZ_MSB_REG** 0x08
- #define **CCZ_LSB_REG** 0x09
- #define **CMM_TMRC_REG** 0x0B /∗∗ CONTINUOUS MEASUREMENT MODE DATA RATE∗/
- #define **MX** 0x24 /∗∗ MEASUREMENT RESULTS REGISTERS ∗/
- #define **MY** 0x27
- #define **MZ** 0x2A
- #define **BIST_REG** 0x33 /∗∗ SELF TEST REGISTER ∗/
- #define **STATUS_REG** 0x34 /∗∗ DATA READY REGISTER ∗/
- #define **HSHAKE_REG** 0x35
- #define **REVID_REG** 0x36 /∗∗ EV BOARD REVISION REGISTER ∗/
- #define **RM3100_ADDRESS_00** 0x20 /∗∗ address SA0 & SA1 pins low (GND) ∗/
- #define **RM3100_ADDRESS_11** 0x23 /∗∗ address SA0 & SA1 pins high (VCC) ∗/
- #define **RM3100_ADDRESS_01** 0x21 /∗∗ address SA0 high (VCC) & SA1 pins low (GND) ∗/
- #define **RM3100_ADDRESS_10** 0x22 /∗∗ address SA0 low (GND) & SA1 pinshigh (VCC) ∗/
- #define **CMM_OFF** 0x00

  *Configurations for continuous measurement mode.*
- #define **CM_START** 0x01
- #define **ALARM_BIT** 0x02
- #define **DRDY_WHEN_ALARM_AND_ALL_AXIS** 0x00
- #define **DRDY_WHEN_ANY_AXIS_MEASURED** 0x04
- #define **DRDY_WHEN_ALL_AXIS_MEASURED** 0x08
- #define **DRDY_WHEN_ALARM** 0x0C
- #define **CMM_ALL_AXIS_ON** 0x70
- #define **CMM_X_AXIS** 0x10
- #define **CMM_Y_AXIS** 0x20
- #define **CMM_Z_AXIS** 0x40
- #define **LDM_BIT** 0x80
- #define **CMM_UPDATERATE_600** 0x92

*Possible data rate choices.*

- #define **CMM_UPDATERATE_300** 0x93
- #define **CMM_UPDATERATE_150** 0x94
- #define **CMM_UPDATERATE_75** 0x95
- #define **CMM_UPDATERATE_37** 0x96
- #define **CMM_UPDATERATE_18** 0x97
- #define **CMM_UPDATERATE_9** 0x98
- #define **CMM_UPDATERATE_4_5** 0x99
- #define **CMM_UPDATERATE_2_3** 0x9A
- #define **CMM_UPDATERATE_1_2** 0x9B
- #define **CMM_UPDATERATE_0_6** 0x9C
- #define **CMM_UPDATERATE_0_3** 0x9D
- #define **CMM_UPDATERATE_0_15** 0x9E
- #define **CMM_UPDATERATE_0_075** 0x9F
- #define **STE_ON** 0x80

  *Configurations for Self test.*
- #define **STE_OFF** 0x00
- #define **BW_00** 0x00
- #define **BW_01** 0x04
- #define **BW_10** 0x08
- #define **BW_11** 0x0C
- #define **BP_00** 0x00
- #define **BP_01** 0x01
- #define **BP_10** 0x02
- #define **BP_11** 0x03
- #define **SM_ALL_AXIS** 0x70 /** Single measument mode */
- #define **STATUS_MASK** 0x80 /** To get status of data ready */
- #define **BIST_MASK** 0x70 /** To get status of the Ev Board */

## Functions

- void **RM3100_init_SM_Operation** (void)

  *Inicializes the Ev. Board to do Single Measurents (On Request). in main cycle is necessary to use **requestSingle-Measurement()** (p. 23) and monotoring DRDY, or STATUS register to know when data is ready to be read. Sets Cycle Count to 200 cycles/s.*
- void **RM3100_init_CMM_Operation** (void)

  *Inicializes the Ev. Board in Continuous Measurents Mode. Must monotoring DRDY, or STATUS register to know when data is ready to be read. Sets Cycle Count to 200 cycles/s. Sets data rate to 300 readings/s.*
- float **getRM3100Gain** (void)

  *request current gain value - Only depends off cycle count value*
- float **getRM3100SampleRate** (void)

  *request current sample rate*
- float **getRM3100MaxDataRate** (void)

  *request max data rate possible - Only depends off cycle count value*
- unsigned int **getRM3100CycleCount** (void)

  *request cycle count value*
- BOOL **setCycleCount** (unsigned int value)

  *Sets cycle count and updates gain and max_data_rate values.*
- BOOL **setCMMdatarate** (BYTE conf)

  *Sets data rate in Continuous Measurement Mode. Fails if desire datarate is higher than the max data rate recommended by PNI.*
- BOOL **continuousModeConfig** (BYTE conf)

  *Continuous Measurement Mode (CMM) Register Configuration.*

- BOOL **requestSingleMeasurement** (void)

  *Request Single Measurement to PNI ASIC.*
- BOOL **getDataReadyStatus** (void)

  *Get data ready Status.*
- BOOL **getRM3100Status** (void)

  *Self test to Ev Board.*
- BYTE **getRM3100revision** (void)

  *Request RM3100 Ev. Board Revision.*
- **sensor_xyz ReadRM3100Raw** (void)

  *Read the raw magnetic values of all 3 axis.*

## Variables

- unsigned int **cycle_count**
- float **sample_rate**
- float **max_data_rate**
- float **gain**
- struct **config rm**
- long **x**
- long **y**
- long **z**

  *x-axis data.*

### 7.2.1 Detailed Description

Basic driver for RM3100. Hardware drivers to communicate with sensors via I2C.

Driver developed in HERMES project

Driver developed in HERMES project - ESTG IPLEIRIA by Miguel Rasteiro `2130904@my.ipleiria.pt`

### 7.2.2 Macro Definition Documentation

#### 7.2.2.1 #define ALARM_BIT 0x02

#### 7.2.2.2 #define BIST_MASK 0x70 /∗∗ To get status of the Ev Board ∗/

#### 7.2.2.3 #define BIST_REG 0x33 /∗∗ SELF TEST REGISTER ∗/

#### 7.2.2.4 #define BP_00 0x00

#### 7.2.2.5 #define BP_01 0x01

#### 7.2.2.6 #define BP_10 0x02

#### 7.2.2.7 #define BP_11 0x03

#### 7.2.2.8 #define BW_00 0x00

#### 7.2.2.9 #define BW_01 0x04

#### 7.2.2.10 #define BW_10 0x08

#### 7.2.2.11 #define BW_11 0x0C

#### 7.2.2.12 #define CCX_LSB_REG 0x05

#### 7.2.2.13 #define CCX_MSB_REG 0x04 /∗∗ CYCLE COUNT REGISTERS ∗/

#### 7.2.2.14 #define CCY_LSB_REG 0x07

#### 7.2.2.15 #define CCY_MSB_REG 0x06

#### 7.2.2.16 #define CCZ_LSB_REG 0x09

#### 7.2.2.17 #define CCZ_MSB_REG 0x08

#### 7.2.2.18 #define CM_START 0x01

#### 7.2.2.19 #define CMM_ALL_AXIS_ON 0x70

#### 7.2.2.20 #define CMM_OFF 0x00

Configurations for continuous measurement mode.

#### 7.2.2.21 #define CMM_REG 0x01 /∗∗ CONTINUOUS MEASUREMENT MODE REGISTER ∗/

#### 7.2.2.22 #define CMM_TMRC_REG 0x0B /∗∗ CONTINUOUS MEASUREMENT MODE DATA RATE∗/

#### 7.2.2.23 #define CMM_UPDATERATE_0_075 0x9F

#### 7.2.2.24 #define CMM_UPDATERATE_0_15 0x9E

#### 7.2.2.25 #define CMM_UPDATERATE_0_3 0x9D

#### 7.2.2.26 #define CMM_UPDATERATE_0_6 0x9C

#### 7.2.2.27 #define CMM_UPDATERATE_150 0x94

#### 7.2.2.28 #define CMM_UPDATERATE_18 0x97

#### 7.2.2.29 #define CMM_UPDATERATE_1_2 0x9B

#### 7.2.2.30 #define CMM_UPDATERATE_2_3 0x9A

#### 7.2.2.31 #define CMM_UPDATERATE_300 0x93

#### 7.2.2.32 #define CMM_UPDATERATE_37 0x96

#### 7.2.2.33 #define CMM_UPDATERATE_4_5 0x99

#### 7.2.2.34 #define CMM_UPDATERATE_600 0x92

Possible data rate choices.

#### 7.2.2.35 #define CMM_UPDATERATE_75 0x95

#### 7.2.2.36 #define CMM_UPDATERATE_9 0x98

#### 7.2.2.37 #define CMM_X_AXIS 0x10

#### 7.2.2.38 #define CMM_Y_AXIS 0x20

#### 7.2.2.39 #define CMM_Z_AXIS 0x40

#### 7.2.2.40 #define DRDY_WHEN_ALARM 0x0C

#### 7.2.2.41 #define DRDY_WHEN_ALARM_AND_ALL_AXIS 0x00

#### 7.2.2.42 #define DRDY_WHEN_ALL_AXIS_MEASURED 0x08

#### 7.2.2.43 #define DRDY_WHEN_ANY_AXIS_MEASURED 0x04

#### 7.2.2.44 #define HSHAKE_REG 0x35

#### 7.2.2.45 #define LDM_BIT 0x80

#### 7.2.2.46 #define MX 0x24 /∗∗ MEASUREMENT RESULTS REGISTERS ∗/

#### 7.2.2.47 #define MY 0x27

#### 7.2.2.48 #define MZ 0x2A

#### 7.2.2.49 #define POLL_REG 0x00 /∗∗ POLLS A SINGLE MEASUREMENT ∗/

Registers of the Ev Board ASIC by PNI

#### 7.2.2.50 #define REVID_REG 0x36 /∗∗ EV BOARD REVISION REGISTER ∗/

#### 7.2.2.51 #define RM3100_ADDRESS_00 0x20 /∗∗ address SA0 & SA1 pins low (GND) ∗/

#### 7.2.2.52 #define RM3100_ADDRESS_01 0x21 /∗∗ address SA0 high (VCC) & SA1 pins low (GND) ∗/

#### 7.2.2.53 #define RM3100_ADDRESS_10 0x22 /∗∗ address SA0 low (GND) & SA1 pinshigh (VCC) ∗/

#### 7.2.2.54 #define RM3100_ADDRESS_11 0x23 /∗∗ address SA0 & SA1 pins high (VCC) ∗/

#### 7.2.2.55 #define SM_ALL_AXIS 0x70 /∗∗ Single measument mode ∗/

#### 7.2.2.56 #define STATUS_MASK 0x80 /∗∗ To get status of data ready ∗/

#### 7.2.2.57 #define STATUS_REG 0x34 /∗∗ DATA READY REGISTER ∗/

#### 7.2.2.58 #define STE_OFF 0x00

#### 7.2.2.59 #define STE_ON 0x80

Configurations for Self test.

### 7.2.3 Function Documentation

#### 7.2.3.1 BOOL continuousModeConfig ( BYTE *conf* )

Continuous Measurement Mode (CMM) Register Configuration.

**Parameters**

| | | |
|---|---|---|
| in | *CMM* | configuration BYTE |

**Returns**

0 if successful, 1 otherwise.

#### 7.2.3.2 BOOL getDataReadyStatus ( void )

Get data ready Status.

**Parameters**

| | | |
|---|---|---|
| in | *none* | |

**Returns**

1 if data ready, 0 otherwise.

#### 7.2.3.3 unsigned int getRM3100CycleCount ( void )

request cycle count value

**Parameters**

| | | |
|---|---|---|
| in | *none* | |

**Returns**

actual cycle count value

#### 7.2.3.4 float getRM3100Gain ( void )

request current gain value - Only depends off cycle count value

**Parameters**

| | | |
|---|---|---|
| in | *none* | |

**Returns**

calculated gain (sensibility)

#### 7.2.3.5 float getRM3100MaxDataRate ( void )

request max data rate possible - Only depends off cycle count value

**Parameters**

| | | |
|---|---|---|
| in | *none* | |

**Returns**

max data rate

#### 7.2.3.6 BYTE getRM3100revision ( void )

Request RM3100 Ev. Board Revision.

**Parameters**

| | | |
|---|---|---|
| in | *none* | |

**Returns**

Revision value.

#### 7.2.3.7 float getRM3100SampleRate ( void )

request current sample rate

**Parameters**

| | | |
|---|---|---|
| in | *none* | |

**Returns**

actual sample rate

#### 7.2.3.8 BOOL getRM3100Status ( void )

Self test to Ev Board.

**Parameters**

| | | |
|---|---|---|
| in | *none* | |

**Returns**

1 if all axis OK, 0 otherwise.

#### 7.2.3.9 sensor_xyz ReadRM3100Raw ( void )

Read the raw magnetic values of all 3 axis.

**Parameters**

| | | |
|---|---|---|
| in | *none* | |

**Returns**

x,y,z 32 bits raw_data of **sensor_xyz** (p. 31) type

#### 7.2.3.10 BOOL requestSingleMeasurement ( void )

Request Single Measurement to PNI ASIC.

**Parameters**

| | | |
|---|---|---|
| in | *none* | |

**Returns**

0 if successful, 1 otherwise.

#### 7.2.3.11 void RM3100_init_CMM_Operation ( void )

Inicializes the Ev. Board in Continuous Measurents Mode. Must monotoring DRDY, or STATUS register to know when data is ready to be read. Sets Cycle Count to 200 cycles/s. Sets data rate to 300 readings/s.

**Parameters**

| | | |
|---|---|---|
| `in` | *none* | |

**Returns**

none

#### 7.2.3.12 void RM3100_init_SM_Operation ( void )

Inicializes the Ev. Board to do Single Measurents (On Request). in main cycle is necessary to use **requestSingle-Measurement()** (p. 23) and monotoring DRDY, or STATUS register to know when data is ready to be read. Sets Cycle Count to 200 cycles/s.

**Parameters**

| | | |
|---|---|---|
| `in` | *none* | |

**Returns**

none

#### 7.2.3.13 BOOL setCMMdatarate ( BYTE *conf* )

Sets data rate in Continuous Measurement Mode. Fails if desire datarate is higher than the max data rate recommended by PNI.

**Parameters**

| | | |
|---|---|---|
| `in` | *Data* | rate configuration BYTE |

**Returns**

0 if successful, 1 otherwise.

#### 7.2.3.14 BOOL setCycleCount ( unsigned int *value* )

Sets cycle count and updates gain and max_data_rate values.

**Parameters**

| | | |
|---|---|---|
| `in` | *desire* | value PNI recomends values between 30 and 400 Hz |

**Returns**

0 if successful, 1 otherwise.

### 7.2.4 Variable Documentation

#### 7.2.4.1 unsigned int cycle_count

#### 7.2.4.2 float gain

#### 7.2.4.3 float max_data_rate

#### 7.2.4.4 struct config rm

**Initial value:**

```
= {
    .cycle_count   = 200,
    .sample_rate   = 37,
    .max_data_rate = 440,
    .gain          = 75
}
```

#### 7.2.4.5 float sample_rate

#### 7.2.4.6 long x

#### 7.2.4.7 long y

#### 7.2.4.8 long z

x-axis data.

## 7.3 I2C generic Driver

Communication with RM3100.

### Files

- file **i2c.c**

    *An I2C lib for i2c operations.*

- file **i2c.h**

    *An I2C lib for i2c operations.*

### Enumerations

- enum **i2cmode** { **MASTER**, **SLAVE** }

### Functions

- void **i2c_init** (I2C_MODULE i2cnum, **i2cmode** mode, BYTE address)

    *Initialize i2c module: I2C clock is BRG If mode parameter is SLAVE, uses address to set slave address for the module Enable module.*

- int **i2c_write** (unsigned char slave_addr, unsigned char reg_addr, unsigned char length, unsigned char const ∗data)

    *Write to device using generic i2c protocol.*

- int **i2c_read** (unsigned char slave_addr, unsigned char reg_addr, unsigned char length, unsigned char ∗data)

    *Write to device using generic i2c protocol.*

### 7.3.1 Detailed Description

Communication with RM3100. My I2C lib.

#### Default I2C configurations

| I2C Configuration | values |
|---|---|
| speed | 400 kHz (FastMode) |
| PIC mode | MASTER |
| PIC I2Cmodule | I2C1 |

Driver developed in HERMES project

### 7.3.2 Enumeration Type Documentation

#### 7.3.2.1 enum i2cmode

**Enumerator**

***MASTER***

***SLAVE***

### 7.3.3 Function Documentation

#### 7.3.3.1 void i2c_init ( I2C_MODULE *i2cnum,* i2cmode *mode,* BYTE *address* )

Initialize i2c module: I2C clock is BRG If mode parameter is SLAVE, uses address to set slave address for the module Enable module.

**Parameters**

| | | |
|---|---|---|
| `in` | *I2C_MODULE* | i2cnum (use I2C1 otherwise is necessary to modify the functions) |
| `in` | *i2cmode* | mode (MASTER or SLAVE) |
| `in` | *BYTE* | address for SLAVE mode |

**Returns**

none

#### 7.3.3.2 int i2c_read ( unsigned char *slave_addr,* unsigned char *reg_addr,* unsigned char *length,* unsigned char ∗ *data* )

Write to device using generic i2c protocol.

**Parameters**

| | | |
|---|---|---|
| `in` | *slave_addr* | - slave address |
| `in` | *reg_addr* | - register address |
| `in` | *length* | - number of bytes to read |
| `in` | ∗*data* | - pointer to where register data is to be transfered |

**Returns**

0 if sucessfull, 1 otherwise

#### 7.3.3.3 int i2c_write ( unsigned char *slave_addr,* unsigned char *reg_addr,* unsigned char *length,* unsigned char const ∗ *data* )

Write to device using generic i2c protocol.

**Parameters**

| | | |
|---|---|---|
| `in` | *slave_addr* | - slave address |
| `in` | *reg_addr* | - register address |
| `in` | *length* | - number of bytes to write |
| `in` | ∗*data* | - pointer for data to write |

**Returns**

0 if sucessfull, 1 otherwise

## 7.4 UART Communications

Sends data out.

Sends data out.

#### Default UART configurations

| UART Configuration | values |
|---|---|
| baudrate | 230400 |
| number of bits | 8 |
| parity | NO |
| stop bits | 1 |
| PIC UARTmodule | UART1 |

# Chapter 8

# Data Structure Documentation

## 8.1 config Struct Reference

### Data Fields

- unsigned int **cycle_count**
- float **sample_rate**
- float **max_data_rate**
- float **gain**

### 8.1.1 Detailed Description

Information of the ASIC configurations

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

- **rm3100.c**

## 8.2 sensor_xyz Struct Reference

```
#include <rm3100.h>
```

### Data Fields

- long **x**
- long **y**
- long **z**

  *x-axis data.*

### 8.2.1 Detailed Description

Saves Raw data from sensors.

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

- **rm3100.h**

# Chapter 9

# File Documentation

## 9.1 documentation.h File Reference

## 9.2 hardware.c File Reference

```
#include <stdio.h>
#include "hardware.h"
#include "uart.h"
#include "i2c.h"
```

### Functions

- void **BoardInit** (void)

### 9.2.1 Function Documentation

#### 9.2.1.1 void BoardInit ( void )

Configuration Bit settings SYSCLK = 80 MHz (8MHz Crystal / FPLLIDIV $*$ FPLLMUL / FPLLODIV) PBCLK = 80 MHz (SYSCLK / FPBDIV) Primary Osc w/PLL (XT+,HS+,EC+PLL) WDT OFF Other options are don't care PIN SETUP ///

SPI 1 SETUP ////

UART 1 SETUP ////

TIMER 1 SETUP /////

I2C 1 SETUP ////

EXTERNAL INTERRUPTIONS SETUP ////

## 9.3 hardware.h File Reference

### Macros

- #define **SYS_FREQ** (80000000L)
- #define **FOSC** (**SYS_FREQ**)
- #define **GetPeripheralClock**() (**SYS_FREQ**/(1 $<<$ OSCCONbits.PBDIV))

- #define **GetInstructionClock**() (**SYS_FREQ**)
- #define **PBCLK** **SYS_FREQ**/4
- #define **BRG** (400000)
- #define **UARTBAUDRATE** (230400)
- #define **ONE_SECOND** (**FOSC**/2)
- #define **MS_TO_CORE_TICKS**(x) ((UINT64)(x)∗**ONE_SECOND**/1000)
- #define **uS_TO_CORE_TICKS**(x) ((UINT64)(x)∗**ONE_SECOND**/1000000)
- #define **CT_TICKS_SINCE**(tick) (ReadCoreTimer() - (tick))
- #define **TIMER_1_INT_VECTOR** (4)
- #define **EXTERNAL_2_INT_VECTOR** (11)
- #define **MPU_I2C** (I2C1)
- #define **BYTEPTR**(x) ((UINT8∗)&(x))

## 9.3.1 Macro Definition Documentation

#### 9.3.1.1 #define BRG (400000)

I2C frequency FastMode = 400kHz

#### 9.3.1.2 #define BYTEPTR( *x* ) ((UINT8∗)&(x))

#### 9.3.1.3 #define CT_TICKS_SINCE( *tick* ) (ReadCoreTimer() - (tick))

#### 9.3.1.4 #define EXTERNAL_2_INT_VECTOR (11)

Interruption Vector For external interrupt 2

#### 9.3.1.5 #define FOSC (SYS_FREQ)

CPU clock frequency

#### 9.3.1.6 #define GetInstructionClock( ) (SYS_FREQ)

Instructions frequency

#### 9.3.1.7 #define GetPeripheralClock( ) (SYS_FREQ/(1 << OSCCONbits.PBDIV))

#### 9.3.1.8 #define MPU_I2C (I2C1)

#### 9.3.1.9 #define MS_TO_CORE_TICKS( *x* ) ((UINT64)(x)∗ONE_SECOND/1000)

#### 9.3.1.10 #define ONE_SECOND (FOSC/2)

#### 9.3.1.11 #define PBCLK SYS_FREQ/4

#### 9.3.1.12 #define SYS_FREQ (80000000L)

CPU clock frequency

#### 9.3.1.13 #define TIMER_1_INT_VECTOR (4)

Interruption Vector For timer1

#### 9.3.1.14 #define UARTBAUDRATE (230400)

#### 9.3.1.15 #define uS_TO_CORE_TICKS( *x* ) ((UINT64)(x)∗ONE_SECOND/1000000)

## 9.4 i2c.c File Reference

An I2C lib for i2c operations.

```
#include "i2c.h"
#include "hardware.h"
#include <peripheral/i2c.h>
```

### Functions

- void **i2c_init** (I2C_MODULE i2cnum, **i2cmode** mode, BYTE address)

    *Initialize i2c module: I2C clock is BRG If mode parameter is SLAVE, uses address to set slave address for the module Enable module.*
- int **i2c_write** (unsigned char slave_addr, unsigned char reg_addr, unsigned char length, unsigned char const ∗data)

    *Write to device using generic i2c protocol.*
- int **i2c_read** (unsigned char slave_addr, unsigned char reg_addr, unsigned char length, unsigned char ∗data)

    *Write to device using generic i2c protocol.*

### 9.4.1 Detailed Description

An I2C lib for i2c operations. Change History:

| VERSION | DATE | AUTHORS | DESCRIPTION |
|---|---|---|---|
| 1.0 | 25/8/2014 | MR | First Release |

## 9.5 i2c.h File Reference

An I2C lib for i2c operations.

```
#include <plib.h>
#include <peripheral/i2c.h>
```

### Enumerations

- enum **i2cmode** { **MASTER**, **SLAVE** }

### Functions

- void **i2c_init** (I2C_MODULE i2cnum, **i2cmode** mode, BYTE address)

    *Initialize i2c module: I2C clock is BRG If mode parameter is SLAVE, uses address to set slave address for the module Enable module.*
- int **i2c_write** (unsigned char slave_addr, unsigned char reg_addr, unsigned char length, unsigned char const ∗data)

    *Write to device using generic i2c protocol.*
- int **i2c_read** (unsigned char slave_addr, unsigned char reg_addr, unsigned char length, unsigned char ∗data)

    *Write to device using generic i2c protocol.*

### 9.5.1 Detailed Description

An I2C lib for i2c operations. Change History:

| VERSION | DATE | AUTHORS | DESCRIPTION |
|---|---|---|---|
| 1.0 | 25/8/2014 | MR | First Release |

## 9.6 main.c File Reference

```
#include <p32xxxx.h>
#include <plib.h>
#include <math.h>
#include "uart.h"
#include "hardware.h"
#include "rm3100.h"
#include "i2c.h"
```

#### Macros

- #define **PI** 3.14159265358979
- #define **MAG_EXT_CAL** 0

#### Functions

- float **get_time** (void)
- void **reset_timer** (void)
- void **__ISR** (**TIMER_1_INT_VECTOR**, ipl1)
- void **__ISR** (**EXTERNAL_2_INT_VECTOR**, ipl2)
- int **main** (void)

#### Variables

- UINT8 **T1overflow** = 1
- BYTE **new_data** =0
- float **mag_offsets** [3] = {0, 0, 0}
- float **mag_cal_matrix** [3][3]

### 9.6.1 Macro Definition Documentation

#### 9.6.1.1 #define MAG_EXT_CAL 0

1 - Using Calibrated Matrix; 0 - Default

#### 9.6.1.2 #define PI 3.14159265358979

### 9.6.2 Function Documentation

#### 9.6.2.1 void __ISR ( TIMER_1_INT_VECTOR , ipl1 )

Timer 1 ISR Interrupt Priority Level = 2 Vector 4

#### 9.6.2.2 void __ISR ( EXTERNAL_2_INT_VECTOR , ipl2 )

#### 9.6.2.3 float get_time ( void )

Temporization functions

#### 9.6.2.4 int main ( void )

#### 9.6.2.5 void reset_timer ( void )

### 9.6.3 Variable Documentation

#### 9.6.3.1 float mag_cal_matrix[3][3]

**Initial value:**

```
= {{1, 0, 0 },
                                        {0, 1, 0 },
                                        {0, 0, 1 }}
```

#### 9.6.3.2 float mag_offsets[3] = {0, 0, 0}

#### 9.6.3.3 BYTE new_data =0

Identity matrix if not using external calibration data.

#### 9.6.3.4 UINT8 T1overflow = 1

## 9.7 rm3100.c File Reference

An I2C-based driver for PNI RM3100 Evaluation Board.

```
#include "i2c.h"
#include "rm3100.h"
#include <plib.h>
```

### Data Structures

- struct **config**

### Functions

- void **RM3100_init_SM_Operation** (void)

  *Inicializes the Ev. Board to do Single Measurents (On Request). in main cycle is necessary to use* ***requestSingleMeasurement()*** (p. *23) and monotoring DRDY, or STATUS register to know when data is ready to be read. Sets Cycle Count to 200 cycles/s.*
- void **RM3100_init_CMM_Operation** (void)

  *Inicializes the Ev. Board in Continuous Measurents Mode. Must monotoring DRDY, or STATUS register to know when data is ready to be read. Sets Cycle Count to 200 cycles/s. Sets data rate to 300 readings/s.*
- float **getRM3100Gain** (void)

  *request current gain value - Only depends off cycle count value*
- float **getRM3100SampleRate** (void)

  *request current sample rate*
- float **getRM3100MaxDataRate** (void)

  *request max data rate possible - Only depends off cycle count value*
- unsigned int **getRM3100CycleCount** (void)

  *request cycle count value*
- BOOL **setCycleCount** (unsigned int value)

  *Sets cycle count and updates gain and max_data_rate values.*
- BOOL **setCMMdatarate** (BYTE conf)

  *Sets data rate in Continuous Measurement Mode. Fails if desire datarate is higher than the max data rate recommended by PNI.*
- BOOL **continuousModeConfig** (BYTE conf)

  *Continuous Measurement Mode (CMM) Register Configuration.*
- BOOL **requestSingleMeasurement** (void)

  *Request Single Measurement to PNI ASIC.*
- BOOL **getDataReadyStatus** (void)

  *Get data ready Status.*
- BOOL **getRM3100Status** (void)

  *Self test to Ev Board.*
- BYTE **getRM3100revision** (void)

  *Request RM3100 Ev. Board Revision.*
- **sensor_xyz ReadRM3100Raw** (void)

  *Read the raw magnetic values of all 3 axis.*

#### Variables

- struct **config rm**

### 9.7.1 Detailed Description

An I2C-based driver for PNI RM3100 Evaluation Board. Change History:

| VERSION | DATE | AUTHORS | DESCRIPTION |
|---|---|---|---|
| 1.0 | 8/9/2014 | MR | First Release |

## 9.8 rm3100.h File Reference

An I2C-based driver for PNI RM3100 Evaluation Board.

```
#include <plib.h>
```

#### Data Structures

- struct **sensor_xyz**

#### Macros

- #define **POLL_REG** 0x00 /∗∗ POLLS A SINGLE MEASUREMENT ∗/
- #define **CMM_REG** 0x01 /∗∗ CONTINUOUS MEASUREMENT MODE REGISTER ∗/
- #define **CCX_MSB_REG** 0x04 /∗∗ CYCLE COUNT REGISTERS ∗/
- #define **CCX_LSB_REG** 0x05

- #define **CCY_MSB_REG** 0x06
- #define **CCY_LSB_REG** 0x07
- #define **CCZ_MSB_REG** 0x08
- #define **CCZ_LSB_REG** 0x09
- #define **CMM_TMRC_REG** 0x0B /** CONTINUOUS MEASUREMENT MODE DATA RATE*/
- #define **MX** 0x24 /** MEASUREMENT RESULTS REGISTERS */
- #define **MY** 0x27
- #define **MZ** 0x2A
- #define **BIST_REG** 0x33 /** SELF TEST REGISTER */
- #define **STATUS_REG** 0x34 /** DATA READY REGISTER */
- #define **HSHAKE_REG** 0x35
- #define **REVID_REG** 0x36 /** EV BOARD REVISION REGISTER */
- #define **RM3100_ADDRESS_00** 0x20 /** address SA0 & SA1 pins low (GND) */
- #define **RM3100_ADDRESS_11** 0x23 /** address SA0 & SA1 pins high (VCC) */
- #define **RM3100_ADDRESS_01** 0x21 /** address SA0 high (VCC) & SA1 pins low (GND) */
- #define **RM3100_ADDRESS_10** 0x22 /** address SA0 low (GND) & SA1 pinshigh (VCC) */
- #define **CMM_OFF** 0x00

  *Configurations for continuous measurement mode.*
- #define **CM_START** 0x01
- #define **ALARM_BIT** 0x02
- #define **DRDY_WHEN_ALARM_AND_ALL_AXIS** 0x00
- #define **DRDY_WHEN_ANY_AXIS_MEASURED** 0x04
- #define **DRDY_WHEN_ALL_AXIS_MEASURED** 0x08
- #define **DRDY_WHEN_ALARM** 0x0C
- #define **CMM_ALL_AXIS_ON** 0x70
- #define **CMM_X_AXIS** 0x10
- #define **CMM_Y_AXIS** 0x20
- #define **CMM_Z_AXIS** 0x40
- #define **LDM_BIT** 0x80
- #define **CMM_UPDATERATE_600** 0x92

  *Possible data rate choices.*
- #define **CMM_UPDATERATE_300** 0x93
- #define **CMM_UPDATERATE_150** 0x94
- #define **CMM_UPDATERATE_75** 0x95
- #define **CMM_UPDATERATE_37** 0x96
- #define **CMM_UPDATERATE_18** 0x97
- #define **CMM_UPDATERATE_9** 0x98
- #define **CMM_UPDATERATE_4_5** 0x99
- #define **CMM_UPDATERATE_2_3** 0x9A
- #define **CMM_UPDATERATE_1_2** 0x9B
- #define **CMM_UPDATERATE_0_6** 0x9C
- #define **CMM_UPDATERATE_0_3** 0x9D
- #define **CMM_UPDATERATE_0_15** 0x9E
- #define **CMM_UPDATERATE_0_075** 0x9F
- #define **STE_ON** 0x80

  *Configurations for Self test.*
- #define **STE_OFF** 0x00
- #define **BW_00** 0x00
- #define **BW_01** 0x04
- #define **BW_10** 0x08
- #define **BW_11** 0x0C
- #define **BP_00** 0x00
- #define **BP_01** 0x01
- #define **BP_10** 0x02
- #define **BP_11** 0x03
- #define **SM_ALL_AXIS** 0x70 /** Single measument mode */
- #define **STATUS_MASK** 0x80 /** To get status of data ready */
- #define **BIST_MASK** 0x70 /** To get status of the Ev Board */

### Functions

- **sensor_xyz ReadRM3100Raw** (void)

  *Read the raw magnetic values of all 3 axis.*
- void **RM3100_init_CMM_Operation** (void)

  *Inicializes the Ev. Board in Continuous Measurents Mode. Must monotoring DRDY, or STATUS register to know when data is ready to be read. Sets Cycle Count to 200 cycles/s. Sets data rate to 300 readings/s.*
- void **RM3100_init_SM_Operation** (void)

  *Inicializes the Ev. Board to do Single Measurents (On Request). in main cycle is necessary to use* ***requestSingle-Measurement()*** *(*p. *23) and monotoring DRDY, or STATUS register to know when data is ready to be read. Sets Cycle Count to 200 cycles/s.*
- BOOL **setCycleCount** (unsigned int value)

  *Sets cycle count and updates gain and max_data_rate values.*
- BOOL **setCMMdatarate** (BYTE conf)

  *Sets data rate in Continuous Measurement Mode. Fails if desire datarate is higher than the max data rate recommended by PNI.*
- BOOL **continuousModeConfig** (BYTE conf)

  *Continuous Measurement Mode (CMM) Register Configuration.*
- BOOL **requestSingleMeasurement** (void)

  *Request Single Measurement to PNI ASIC.*
- BOOL **getDataReadyStatus** (void)

  *Get data ready Status.*
- BYTE **getRM3100revision** (void)

  *Request RM3100 Ev. Board Revision.*
- BOOL **getRM3100Status** (void)

  *Self test to Ev Board.*
- float **getRM3100Gain** (void)

  *request current gain value - Only depends off cycle count value*
- float **getRM3100SampleRate** (void)

  *request current sample rate*
- float **getRM3100MaxDataRate** (void)

  *request max data rate possible - Only depends off cycle count value*
- unsigned int **getRM3100CycleCount** (void)

  *request cycle count value*

### 9.8.1 Detailed Description

An I2C-based driver for PNI RM3100 Evaluation Board. Change History:

| VERSION | DATE | AUTHORS | DESCRIPTION |
|---|---|---|---|
| 1.0 | 8/9/2014 | MR | First Release |

## 9.9 uart.c File Reference

```
#include "uart.h"
```

### Functions

- void **SendDataBuffer** (const char ∗buffer, UINT32 size)
- UINT32 **GetMenuChoice** (void)

### 9.9.1 Function Documentation

#### 9.9.1.1 UINT32 GetMenuChoice ( void )

#### 9.9.1.2 void SendDataBuffer ( const char ∗ *buffer,* UINT32 *size* )

## 9.10 uart.h File Reference

```
#include <plib.h>
```

### Macros

- #define **UART_MODULE_ID** UART1

### Functions

- void **SendDataBuffer** (const char ∗buffer, UINT32 size)
- UINT32 **GetMenuChoice** (void)

### 9.10.1 Macro Definition Documentation

#### 9.10.1.1 #define UART_MODULE_ID UART1

### 9.10.2 Function Documentation

#### 9.10.2.1 UINT32 GetMenuChoice ( void )

#### 9.10.2.2 void SendDataBuffer ( const char ∗ *buffer,* UINT32 *size* )