

RM3100driver

1.0

Generated by Doxygen 1.8.6

Mon Sep 8 2014 18:23:51

Contents

1	Main Page	1
2	License	3
3	Microchip License	7
4	Module Index	9
4.1	Modules	9
5	Data Structure Index	11
5.1	Data Structures	11
6	File Index	13
6.1	File List	13
7	Module Documentation	15
7.1	Main	15
7.2	RM3100 Driver	16
7.2.1	Detailed Description	18
7.2.2	Macro Definition Documentation	18
7.2.2.1	ALARM_BIT	18
7.2.2.2	BIST_MASK	18
7.2.2.3	BIST_REG	18
7.2.2.4	BP_00	18
7.2.2.5	BP_01	18
7.2.2.6	BP_10	18
7.2.2.7	BP_11	18
7.2.2.8	BW_00	18
7.2.2.9	BW_01	18
7.2.2.10	BW_10	18
7.2.2.11	BW_11	19
7.2.2.12	CCX_LSB_REG	19
7.2.2.13	CCX_MSB_REG	19
7.2.2.14	CCY_LSB_REG	19

7.2.2.15	CCY_MSB_REG	19
7.2.2.16	CCZ_LSB_REG	19
7.2.2.17	CCZ_MSB_REG	19
7.2.2.18	CM_START	19
7.2.2.19	CMM_ALL_AXIS_ON	19
7.2.2.20	CMM_OFF	19
7.2.2.21	CMM_REG	19
7.2.2.22	CMM_TMRC_REG	19
7.2.2.23	CMM_UPDATERATE_0_075	19
7.2.2.24	CMM_UPDATERATE_0_15	19
7.2.2.25	CMM_UPDATERATE_0_3	19
7.2.2.26	CMM_UPDATERATE_0_6	19
7.2.2.27	CMM_UPDATERATE_150	19
7.2.2.28	CMM_UPDATERATE_18	19
7.2.2.29	CMM_UPDATERATE_1_2	19
7.2.2.30	CMM_UPDATERATE_2_3	19
7.2.2.31	CMM_UPDATERATE_300	19
7.2.2.32	CMM_UPDATERATE_37	19
7.2.2.33	CMM_UPDATERATE_4_5	19
7.2.2.34	CMM_UPDATERATE_600	19
7.2.2.35	CMM_UPDATERATE_75	19
7.2.2.36	CMM_UPDATERATE_9	19
7.2.2.37	CMM_X_AXIS	20
7.2.2.38	CMM_Y_AXIS	20
7.2.2.39	CMM_Z_AXIS	20
7.2.2.40	DRDY_WHEN_ALARM	20
7.2.2.41	DRDY_WHEN_ALARM_AND_ALL_AXIS	20
7.2.2.42	DRDY_WHEN_ALL_AXIS_MEASURED	20
7.2.2.43	DRDY_WHEN_ANY_AXIS_MEASURED	20
7.2.2.44	HSHAKE_REG	20
7.2.2.45	LDM_BIT	20
7.2.2.46	MX	20
7.2.2.47	MY	20
7.2.2.48	MZ	20
7.2.2.49	POLL_REG	20
7.2.2.50	REVID_REG	20
7.2.2.51	RM3100_ADDRESS_00	20
7.2.2.52	RM3100_ADDRESS_01	20
7.2.2.53	RM3100_ADDRESS_10	20
7.2.2.54	RM3100_ADDRESS_11	20

7.2.2.55	SM_ALL_AXIS	20
7.2.2.56	STATUS_MASK	20
7.2.2.57	STATUS_REG	20
7.2.2.58	STE_OFF	20
7.2.2.59	STE_ON	20
7.2.3	Function Documentation	20
7.2.3.1	continuousModeConfig	20
7.2.3.2	getDataReadyStatus	21
7.2.3.3	getRM3100CycleCount	21
7.2.3.4	getRM3100Gain	21
7.2.3.5	getRM3100MaxDataRate	21
7.2.3.6	getRM3100revision	22
7.2.3.7	getRM3100SampleRate	23
7.2.3.8	getRM3100Status	23
7.2.3.9	ReadRM3100Raw	23
7.2.3.10	requestSingleMeasurement	23
7.2.3.11	RM3100_init_CMM_Operation	24
7.2.3.12	RM3100_init_SM_Operation	25
7.2.3.13	setCMMdatarate	25
7.2.3.14	setCycleCount	25
7.2.4	Variable Documentation	25
7.2.4.1	cycle_count	25
7.2.4.2	gain	25
7.2.4.3	max_data_rate	25
7.2.4.4	rm	25
7.2.4.5	sample_rate	26
7.2.4.6	x	26
7.2.4.7	y	26
7.2.4.8	z	26
7.3	I2C generic Driver	27
7.3.1	Detailed Description	27
7.3.2	Enumeration Type Documentation	27
7.3.2.1	i2cmode	27
7.3.3	Function Documentation	27
7.3.3.1	i2c_init	27
7.3.3.2	i2c_read	28
7.3.3.3	i2c_write	28
7.4	UART Communications	29

8.1	config Struct Reference	31
8.1.1	Detailed Description	31
8.2	sensor_xyz Struct Reference	31
8.2.1	Detailed Description	31
9	File Documentation	33
9.1	documentation.h File Reference	33
9.2	hardware.c File Reference	33
9.2.1	Function Documentation	33
9.2.1.1	BoardInit	33
9.3	hardware.h File Reference	33
9.3.1	Macro Definition Documentation	34
9.3.1.1	BRG	34
9.3.1.2	BYTEPTR	34
9.3.1.3	CT_TICKS_SINCE	34
9.3.1.4	EXTERNAL_2_INT_VECTOR	34
9.3.1.5	FOSC	34
9.3.1.6	GetInstructionClock	34
9.3.1.7	GetPeripheralClock	34
9.3.1.8	MPU_I2C	34
9.3.1.9	MS_TO_CORE_TICKS	34
9.3.1.10	ONE_SECOND	34
9.3.1.11	PBCLK	34
9.3.1.12	SYS_FREQ	34
9.3.1.13	TIMER_1_INT_VECTOR	34
9.3.1.14	UARTBAUDRATE	35
9.3.1.15	uS_TO_CORE_TICKS	35
9.4	i2c.c File Reference	35
9.4.1	Detailed Description	35
9.5	i2c.h File Reference	35
9.5.1	Detailed Description	36
9.6	main.c File Reference	36
9.6.1	Macro Definition Documentation	36
9.6.1.1	MAG_EXT_CAL	36
9.6.1.2	PI	36
9.6.2	Function Documentation	36
9.6.2.1	__ISR	36
9.6.2.2	__ISR	37
9.6.2.3	get_time	37
9.6.2.4	main	37

9.6.2.5	reset_timer	37
9.6.3	Variable Documentation	37
9.6.3.1	mag_cal_matrix	37
9.6.3.2	mag_offsets	37
9.6.3.3	new_data	37
9.6.3.4	T1overflow	37
9.7	rm3100.c File Reference	37
9.7.1	Detailed Description	38
9.8	rm3100.h File Reference	38
9.8.1	Detailed Description	40
9.9	uart.c File Reference	40
9.9.1	Function Documentation	41
9.9.1.1	GetMenuChoice	41
9.9.1.2	SendDataBuffer	41
9.10	uart.h File Reference	41
9.10.1	Macro Definition Documentation	41
9.10.1.1	UART_MODULE_ID	41
9.10.2	Function Documentation	41
9.10.2.1	GetMenuChoice	41
9.10.2.2	SendDataBuffer	41

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 GNU 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:

that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

1. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change. b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License. c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.) These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

1. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or, b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or, c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.) The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components

(compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

1. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
2. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
3. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
4. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

1. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
2. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

1. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

1. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
2. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Chapter 3

Microchip License

Microchip Software License Agreement

The software supplied herewith by Microchip Technology Incorporated (the "Company") for its PIC32MX Microcontroller is intended and supplied to you, the Company's customer, for use solely and exclusively on Microchip Microcontroller products. The software is owned by the Company and/or its supplier, and is protected under applicable copyright laws. All rights are reserved. Any use in violation of the foregoing restrictions may subject the user to criminal sanctions under applicable laws, as well as to civil liability for the breach of the terms and conditions of this license.

THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

Main	15
RM3100 Driver	16
I2C generic Driver	27
UART Communications	29

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

config	31
sensor_xyz	31

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

documentation.h	33
hardware.c	33
hardware.h	33
i2c.c	
An I2C lib for i2c operations	35
i2c.h	
An I2C lib for i2c operations	35
main.c	36
rm3100.c	
An I2C-based driver for PNI RM3100 Evaluation Board	37
rm3100.h	
An I2C-based driver for PNI RM3100 Evaluation Board	38
uart.c	40
uart.h	41

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 measurement 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)
*Initializes the Ev. Board to do Single Measurements (On Request). In main cycle is necessary to use **requestSingleMeasurement()** (p. 23) and monitoring 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)
Initializes the Ev. Board in Continuous Measurements Mode. Must monitor 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 desired data rate 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	<i>CMM</i>	configuration BYTE
----	------------	--------------------

Returns

0 if successful, 1 otherwise.

7.2.3.2 BOOL getDataReadyStatus (void)

Get data ready Status.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

1 if data ready, 0 otherwise.

7.2.3.3 unsigned int getRM3100CycleCount (void)

request cycle count value

Parameters

in	<i>none</i>	
----	-------------	--

Returns

actual cycle count value

7.2.3.4 float getRM3100Gain (void)

request current gain value - Only depends off cycle count value

Parameters

in	<i>none</i>	
----	-------------	--

Returns

calculated gain (sensitivity)

7.2.3.5 float getRM3100MaxDataRate (void)

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

Parameters

in	<i>none</i>	
----	-------------	--

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)

Initializes the Ev. Board in Continuous Measurements Mode. Must monitor 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)

Initializes the Ev. Board to do Single Measurements (On Request). in main cycle is necessary to use **requestSingleMeasurement()** (p. 23) and monitoring 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 recommends 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	<i>I2C_MODULE</i>	i2cnum (use I2C1 otherwise is necessary to modify the functions)
in	<i>i2cmode</i>	mode (MASTER or SLAVE)
in	<i>BYTE</i>	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	<i>slave_addr</i>	- slave address
in	<i>reg_addr</i>	- register address
in	<i>length</i>	- number of bytes to read
in	<i>*data</i>	- 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	<i>slave_addr</i>	- slave address
in	<i>reg_addr</i>	- register address
in	<i>length</i>	- number of bytes to write
in	<i>*data</i>	- 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)
*Initializes the Ev. Board to do Single Measurements (On Request). in main cycle is necessary to use **requestSingleMeasurement()** (p. 23) and monitoring 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)
Initializes the Ev. Board in Continuous Measurements Mode. Must monitoring 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)
Initializes the Ev. Board in Continuous Measurements Mode. Must monitoring 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)
*Initializes the Ev. Board to do Single Measurements (On Request). in main cycle is necessary to use **requestSingleMeasurement()** (p. 23) and monitoring 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*)