Arduino Accelerometer Driver

Generated by Doxygen 1.8.9.1

Wed Aug 19 2015 01:07:02

Contents

1 Deprecated List

Member AccelerometerMMA8451::routeInterruptToInt1 (Interrupt interrupt)

Parameters

interrupt	The interrupt flag.

Member AccelerometerMMA8451::routeInterruptToInt2 (Interrupt interrupt)

Parameters

interrupt	The interrupt flag.

2 Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

3 Class Index

3.1 Class List

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

4 File Index

4.1 File List

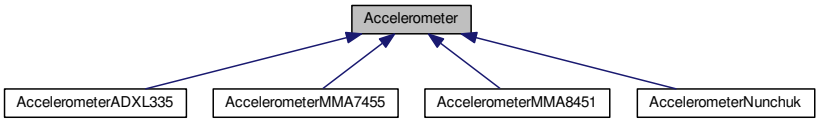
Here is a list of all files with brief descriptions:

5 Class Documentation

5.1 Accelerometer Class Reference

`#include <Accelerometer.h>`

Inheritance diagram for Accelerometer:



Public Member Functions

- virtual float readXg ()=0
- virtual float readYg ()=0
- virtual float readZg ()=0

5.1.1 Detailed Description

Arduino - Accelerometer driver.

Accelerometer.h

The header file for the accelerometer driver

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 14 of file Accelerometer.h.

5.1.2 Member Function Documentation

5.1.2.1 virtual float Accelerometer::readXg ( ) [pure virtual]

Reads the x axis from the accelerometer device.

The x result.

Implemented in AccelerometerMMA8451, AccelerometerMMA7455, AccelerometerADXL335, and Accelerometer↩ Nunchuk.

5.1.2.2 virtual float Accelerometer::readYg ( ) [pure virtual]

Reads the y axis from the accelerometer device.

The y result.

Implemented in AccelerometerMMA8451, AccelerometerMMA7455, AccelerometerADXL335, and Accelerometer↩ Nunchuk.

5.1.2.3 virtual float Accelerometer::readZg ( ) [pure virtual]

Reads the z axis from the accelerometer device.

The z result.

Implemented in AccelerometerMMA8451, AccelerometerMMA7455, AccelerometerADXL335, and AccelerometerNunchuk.

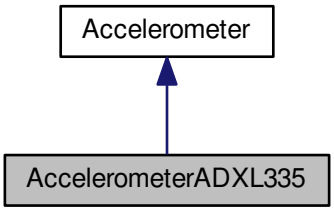
The documentation for this class was generated from the following file:

- Accelerometer.h

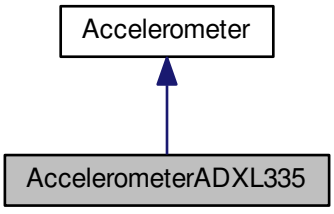
5.2 AccelerometerADXL335 Class Reference

`#include <AccelerometerADXL335.h>`

Inheritance diagram for AccelerometerADXL335:



Collaboration diagram for AccelerometerADXL335:



Public Member Functions

- AccelerometerADXL335 (int xPin, int yPin, int zPin)
- virtual float readXg ()
- virtual float readYg ()
- virtual float readZg ()

Private Member Functions

- float readPin (int pin)

Private Attributes

- int xPin
- int yPin
- int zPin
- float arduinoPowerSupply
- float sensorPowerSupply
- float zeroGBias

5.2.1 Detailed Description

Arduino - Accelerometer driver.

AccelerometerADXL335.h

The implementation of the ADXL335 accelerometer.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 16 of file AccelerometerADXL335.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 AccelerometerADXL335::AccelerometerADXL335 (int *xPin*, int *yPin*, int *zPin*)

Public constructor.

Parameters

xPin	
yPin	
zPin	

Definition at line 17 of file AccelerometerADXL335.cpp.

5.2.3 Member Function Documentation

5.2.3.1 float AccelerometerADXL335::readPin (int *pin*) [private]

Reads an analog value fro pin.

Parameters

pin	

Returns

Definition at line 26 of file AccelerometerADXL335.cpp.

5.2.3.2 float AccelerometerADXL335::readXg ( ) [virtual]

Reads the x axis from the accelerometer device.

The x result.

Implements Accelerometer.

Definition at line 31 of file AccelerometerADXL335.cpp.

5.2.3.3 float AccelerometerADXL335::readYg ( ) [virtual]

Reads the y axis from the accelerometer device.

The y result.

Implements Accelerometer.

Definition at line 35 of file AccelerometerADXL335.cpp.

5.2.3.4 float AccelerometerADXL335::readZg ( ) [virtual]

Reads the z axis from the accelerometer device.

The z result.

Implements Accelerometer.

Definition at line 39 of file AccelerometerADXL335.cpp.

5.2.4 Member Data Documentation

5.2.4.1 float AccelerometerADXL335::arduinoPowerSupply [private]

The arduino power supply voltage.

Definition at line 36 of file AccelerometerADXL335.h.

5.2.4.2 float AccelerometerADXL335::sensorPowerSupply [private]

The sensor power supply voltage.

Definition at line 41 of file AccelerometerADXL335.h.

5.2.4.3 int AccelerometerADXL335::xPin [private]

The x pin.

Definition at line 21 of file AccelerometerADXL335.h.

5.2.4.4 int AccelerometerADXL335::yPin [private]

The y pin.

Definition at line 26 of file AccelerometerADXL335.h.

5.2.4.5 float AccelerometerADXL335::zeroGBias [private]

The zero g bias output is also ratiometric, thus the zero g output is nominally equal to VS/2 at all supply voltages.

Definition at line 47 of file AccelerometerADXL335.h.

5.2.4.6 int AccelerometerADXL335::zPin [private]

The z pin.

Definition at line 31 of file AccelerometerADXL335.h.

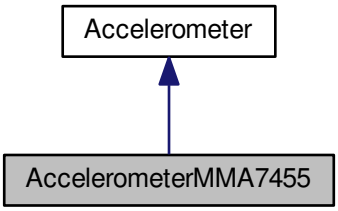
The documentation for this class was generated from the following files:

- AccelerometerADXL335.h
- AccelerometerADXL335.cpp

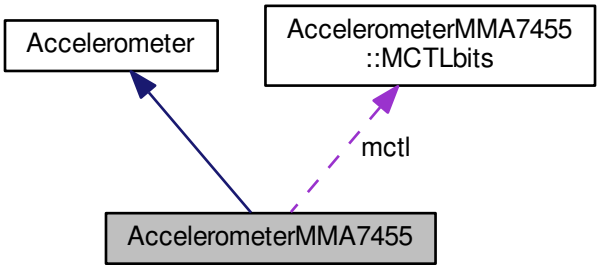
5.3 AccelerometerMMA7455 Class Reference

`#include <AccelerometerMMA7455.h>`

Inheritance diagram for AccelerometerMMA7455:



Collaboration diagram for AccelerometerMMA7455:



Classes

- union CTL1bits
- union CTL2bits
- union DETSRCbits
- union I2CADbits
- union INTRSTbits
- union MCTLbits
- union STATUSbits

Public Types

- enum Location {
  XOUTL = 0x00, XOUTH = 0x01, YOUTL = 0x02, YOUTH = 0x03,
  ZOUTL = 0x04, ZOUTH = 0x05, XOUT8 = 0x06, YOUT8 = 0x07,
  ZOUT8 = 0x08, STATUS = 0x09, DETSRC = 0x0a, TOUT = 0x0b,
  RESERVED1 = 0x0c, I2CAD = 0x0d, USRINF = 0x0e, WHOAMI = 0x0f,
  XOFFL = 0x10, XOFFH = 0x11, YOFFL = 0x12, YOFFH = 0x13,
  ZOFFL = 0x14, ZOFFH = 0x15, MCTL = 0x16, INTRST = 0x17,
  CTL1 = 0x18, CTL2 = 0x19, LDTH = 0x1a, PDTH = 0x1b,
  PD = 0x1c, LT = 0x1d, TW = 0x1e, RESERVED2 = 0x1f }
- enum Mask {
  STATUS_DRDY = 0x01, STATUS_DOVR = 0x02, STATUS_PERR = 0x04, DETSRC_INT1 = 0x01,
  DETSRC_INT2 = 0x02, DETSRC_PDZ = 0x04, DETSRC_PDY = 0x08, DETSRC_PDX = 0x10,
  DETSRC_LDZ = 0x20, DETSRC_LDY = 0x40, DETSRC_LDX = 0x80, I2CAD_DVAD = 0x7f,
  I2CAD_I2CDIS = 0x80, MCTL_MODE = 0x03, MCTL_GLVL = 0x0c, MCTL_STON = 0x10,
  MCTL_SPI3W = 0x20, MCTL_DRPD = 0x40, INTRST_CLR_INT1 = 0x01, INTRST_CLR_INT2 = 0x02,
  CTL1_INTPIN = 0x01, CTL1_INTREG = 0x06, CTL1_XDA = 0x08, CTL1_YDA = 0x10,
  CTL1_ZDA = 0x20, CTL1_THOPT = 0x40, CTL1_DFBW = 0x80, CTL2_LDPL = 0x01,
  CTL2_PDPL = 0x02, CTL2_DRVO = 0x04 }
- enum DeviceMode { STANDBY_MODE = 0x00, MEASUREMENT_MODE = 0x01, LEVEL_DETECTION_↩
  MODE = 0x02, PULSE_DETECTION_MODE = 0x03 }
- enum DynamicRange { DR_8G = 0x00, DR_2G = 0x01, DR_4G = 0x02 }
- enum DigitalFilterBandWidth { DFBW_62_5_HZ = 0x00, DFBW_125_HZ = 0x01 }
- enum InterruptConfiguration { INT1_LEVEL_INT2_PULSE = 0x00, INT1_PULSE_INT2_LEVEL = 0x01, IN↩
  T1_S_PULSE_INT2_D_PULSE = 0x02 }
- enum Axis { AXIS_X = 0x00, AXIS_Y = 0x01, AXIS_Z = 0x02 }
- enum DetectionCondition { MOTION_DETECTION = 0x00, FREEFALL_DETECTION = 0x01 }

Public Member Functions

- AccelerometerMMA7455 ()
- virtual float readXg ()
- virtual float readYg ()
- virtual float readZg ()
- void readXYZ (unsigned char buf[6])
- bool isDataReady ()
- void setDeviceMode (DeviceMode mode)
- void calibrate0gOffset (unsigned char samples)
- void standby ()
- void measurementMode ()
- void levelDetectionMode ()
- void pulseDetectionMode ()
- void setUse8bit (bool use)
- void setDetectionCondition (DetectionCondition condition)
- void setDynamicRange (DynamicRange range)
- void enableInterrupt (Axis axis)
- void disableInterrupt (Axis axis)
- void setInterruptConfiguration (InterruptConfiguration configuration)
- void clearInterruptLatch ()
- void writeRegister (Location location, unsigned char v)
- unsigned char readRegister (Location location)
- void writeRegisterBlock (unsigned char to, unsigned char ∗buf, unsigned char len)
- void fillRegisterBlock (unsigned char to, unsigned char b, unsigned char len)
- void readRegisterBlock (unsigned char from, unsigned char ∗buf, unsigned char len)
- float convertToG (unsigned char ∗buf, bool is8bit)
- void configureRegisterBits (Location location, Mask mask, unsigned char v)

Protected Attributes

- int int1Pin
- int int2Pin
- unsigned char address
- MCTLbits mctl
- bool use8bit

5.3.1 Detailed Description

Arduino - Accelerometer driver.

AccelerometerMMA7455.h

The implementation of the MMA7455 accelerometer.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 20 of file AccelerometerMMA7455.h.

5.3.2 Member Enumeration Documentation

5.3.2.1 enum AccelerometerMMA7455::Axis

Axis list.

Enumerator

AXIS_X

AXIS_Y

AXIS_Z

Definition at line 279 of file AccelerometerMMA7455.h.

5.3.2.2 enum AccelerometerMMA7455::DetectionCondition

Control 2 (Read/Write): Motion Detection (OR condition) or Freefall Detection (AND condition)

```
PDPL
0: Pulse detection polarity is positive and detecting condition is OR 3 axes.
1: Pulse detection polarity is negative and detecting condition is AND 3 axes.
```

Enumerator

MOTION_DETECTION

FREEFALL_DETECTION

Definition at line 295 of file AccelerometerMMA7455.h.

5.3.2.3 enum AccelerometerMMA7455::DeviceMode

Device mode.

```
MODE[1:0]     Function
00            Standby Mode
01            Measurement Mode
10            Level Detection Mode
11            Pulse Detection Mode
```

Enumerator

STANDBY_MODE

MEASUREMENT_MODE

LEVEL_DETECTION_MODE

PULSE_DETECTION_MODE

Definition at line 234 of file AccelerometerMMA7455.h.

5.3.2.4 enum AccelerometerMMA7455::DigitalFilterBandWidth

DigitalFilterBandWidth.

Enumerator

DFBW_62_5_HZ

DFBW_125_HZ

Definition at line 261 of file AccelerometerMMA7455.h.

5.3.2.5 enum AccelerometerMMA7455::DynamicRange

Configuring the g-Select for 8-bit output using Register $16 with GLVL[1:0] bits.

```
GLVL[1:0]     g Range
     00       ±8g
     01       ±2g
     10       ±4g
```

Enumerator

DR_8G

DR_2G

DR_4G

Definition at line 252 of file AccelerometerMMA7455.h.

5.3.2.6 enum AccelerometerMMA7455::InterruptConfiguration

Configuring the Interrupt settings using Register $18 with INTREG[1:0] bits.

Enumerator

INT1_LEVEL_INT2_PULSE

INT1_PULSE_INT2_LEVEL

INT1_S_PULSE_INT2_D_PULSE

Definition at line 270 of file AccelerometerMMA7455.h.

5.3.2.7 enum AccelerometerMMA7455::Location

Internal registers.

Enumerator

XOUTL

XOUTH

YOUTL

YOUTH
ZOUTL
ZOUTH
XOUT8
YOUT8
ZOUT8
STATUS
DETSRC
TOUT
RESERVED1
I2CAD
USRINF
WHOAMI
XOFFL
XOFFH
YOFFL
YOFFH
ZOFFL
ZOFFH
MCTL
INTRST
CTL1
CTL2
LDTH
PDTH
PD
LT
TW
RESERVED2

Definition at line 144 of file AccelerometerMMA7455.h.

5.3.2.8 enum AccelerometerMMA7455::Mask

Some useful masks.

Enumerator

STATUS_DRDY
STATUS_DOVR
STATUS_PERR
DETSRC_INT1
DETSRC_INT2
DETSRC_PDZ
DETSRC_PDY
DETSRC_PDX
DETSRC_LDZ
DETSRC_LDY

DETSRC_LDX
I2CAD_DVAD
I2CAD_I2CDIS
MCTL_MODE
MCTL_GLVL
MCTL_STON
MCTL_SPI3W
MCTL_DRPD
INTRST_CLR_INT1
INTRST_CLR_INT2
CTL1_INTPIN
CTL1_INTREG
CTL1_XDA
CTL1_YDA
CTL1_ZDA
CTL1_THOPT
CTL1_DFBW
CTL2_LDPL
CTL2_PDPL
CTL2_DRVO

Definition at line 183 of file AccelerometerMMA7455.h.

5.3.3 Constructor & Destructor Documentation

5.3.3.1 AccelerometerMMA7455::AccelerometerMMA7455 ( )

Public constructor.

Definition at line 16 of file AccelerometerMMA7455.cpp.

5.3.4 Member Function Documentation

5.3.4.1 void AccelerometerMMA7455::calibrate0gOffset (unsigned char *samples*)

The offset can be calibrated by storing the offset values in the designated offset drift registers $10 to $15 in the accelerometer.

These values will be stored here until the part loses power. It is a very simple to store these values written to the registers in the memory of a microcontroller, if used in conjunction with the sensor. This will provide automatic calibration of the sensor each time the sensor is turned back on.

In order to calibrate the MMA745xL 0g offset, the predetermined digital offset values should be subtracted from the reading of the actual digital sensing values. The following procedure is a recommendation for how this can be accomplished:

1. After power up, set up the "Mode Control Register" (Register $16) to be in "measurement mode" by writing $05 into Register $16. Then read the X, Y and Z offset values from the Registers $00-$08. The first 6 registers of the 9 are 10-bit XYZ output values: LSB, first; MSB, second. Please verify with the data sheet for detailed register information.
2. In this step, the offset compensation is calculated to shift the offset to zero.

Definition at line 39 of file AccelerometerMMA7455.cpp.

5.3.4.2 void AccelerometerMMA7455::clearInterruptLatch ( ) [inline]

After interrupt has triggered due to a detection, the interrupt pin (INT1 or INT2) need to be cleared by writing a logic 1.

Then the interrupt pin should be enabled to trigger the next detection by setting it to a logic 0.

Definition at line 505 of file AccelerometerMMA7455.h.

5.3.4.3 void AccelerometerMMA7455::configureRegisterBits (Location *location*, Mask *mask*, unsigned char *v*)

Configures the register.

Basically it reads the register from the device. Applies the given mask on such register and makes an OR bitwise operation whit the v value.

(the v value will be masked to only use the bits of the corresponding mask).

Parameters

reg	
mask	
v	

Definition at line 173 of file AccelerometerMMA7455.cpp.

5.3.4.4 float AccelerometerMMA7455::convertToG (unsigned char * *buf*, bool *is8bit*)

Converts an array of chars into a float type.

Parameters

buf	1 (8-bit) our 2 (10-bit) bytes to be converted.
is8bit	boolean indication if the date is 8 bit only

Definition at line 152 of file AccelerometerMMA7455.cpp.

5.3.4.5 void AccelerometerMMA7455::disableInterrupt (Axis *axis*)

Enables interrupt.

Parameters

axis	The axis to enable.

Definition at line 138 of file AccelerometerMMA7455.cpp.

5.3.4.6 void AccelerometerMMA7455::enableInterrupt (Axis *axis*)

Disable interrupt.

Parameters

axis	The axis to disable.

Definition at line 128 of file AccelerometerMMA7455.cpp.

5.3.4.7 void AccelerometerMMA7455::fillRegisterBlock (unsigned char *to*, unsigned char *b*, unsigned char *len*)

Fills a block of data into the device starting at the 'to' register.

Parameters

to	The address to write.
b	The byte data to be filled into register.
len	The number of bytes to write.

Definition at line 200 of file AccelerometerMMA7455.cpp.

5.3.4.8 bool AccelerometerMMA7455::isDataReady ( )

Return true if the data is ready to be read.

Returns

Definition at line 78 of file AccelerometerMMA7455.cpp.

5.3.4.9 void AccelerometerMMA7455::levelDetectionMode ( ) [inline]

Put sensor into Level Detection Mode.

The user can access XYZ measurements and can use the level interrupt only. The level detection mechanism has no timers associated with it. Once a set acceleration level is reached the interrupt pin will go high and remain high until the interrupt pin is cleared (See Assigning, Clearing & Detecting Interrupts).

Definition at line 423 of file AccelerometerMMA7455.h.

5.3.4.10 void AccelerometerMMA7455::measurementMode ( ) [inline]

Put sensor into Measurement Mode.

Measurement Mode

The device can read XYZ measurements in this mode. The pulse and threshold interrupts are not active. During measurement mode, continuous measurements on all three axes enabled. The g-range for 2g, 4g, or 8g are selectable with 8-bit data and the g-range of 8g is selectable with 10-bit data. The sample rate during measurement mode is 125 Hz with 62.5 BW filter selected.

The sample rate is 250 Hz with the 125 Hz filter selected. Therefore, when a conversion is complete (signaled by the DRDY flag), the next measurement will be ready. When measurements on all three axes are completed, a logic high level is output to the DRDY pin, indicating "measurement data is ready." The DRDY status can be monitored by the DRDY bit in Status Register (Address: $09). The DRDY pin is kept high until one of the three Output Value Registers are read. If the next measurement data is written before the previous data is read, the DOVR bit in the Status Register will be set. Also note that in measurement mode, level detection mode and pulse detection mode are not available.

By default all three axes are enabled. X and/or Y and/or Z can be disabled. There is a choice between detecting an absolute signal or a positive or negative only signal on the enabled axes. There is also a choice between doing a detection for motion where X or Y or Z $>$ Threshold vs. doing a detection for freefall where X & Y & Z $<$ Threshold.

Definition at line 410 of file AccelerometerMMA7455.h.

5.3.4.11 void AccelerometerMMA7455::pulseDetectionMode ( ) [inline]

Put sensor into Pulse Detection Mode.

The user can access XYZ measurements and can use the level interrupt only. The level detection mechanism has no timers associated with it. Once a set acceleration level is reached the interrupt pin will go high and remain high until the interrupt pin is cleared (See Assigning, Clearing & Detecting Interrupts).

Definition at line 436 of file AccelerometerMMA7455.h.

5.3.4.12 unsigned char AccelerometerMMA7455::readRegister (Location *location*)

Reads the sensor register.

Returns

The current register value.

Definition at line 185 of file AccelerometerMMA7455.cpp.

5.3.4.13 void AccelerometerMMA7455::readRegisterBlock (unsigned char *from*, unsigned char * *buf*, unsigned char *len*)

Reads a block of data from the device starting at the 'from' register.

Parameters

from	The address to read.
buf	The buffer to be used.
len	The number of bytes to read.

Definition at line 209 of file AccelerometerMMA7455.cpp.

5.3.4.14 float AccelerometerMMA7455::readXg ( ) [virtual]

Reads the x axis from the accelerometer device.

The x result.

Implements Accelerometer.

Definition at line 84 of file AccelerometerMMA7455.cpp.

5.3.4.15 void AccelerometerMMA7455::readXYZ (unsigned char *buf[6]*)

Definition at line 108 of file AccelerometerMMA7455.cpp.

5.3.4.16 float AccelerometerMMA7455::readYg ( ) [virtual]

Reads the y axis from the accelerometer device.

The y result.

Implements Accelerometer.

Definition at line 92 of file AccelerometerMMA7455.cpp.

5.3.4.17 float AccelerometerMMA7455::readZg ( ) [virtual]

Reads the z axis from the accelerometer device.

The z result.

Implements Accelerometer.

Definition at line 100 of file AccelerometerMMA7455.cpp.

5.3.4.18 void AccelerometerMMA7455::setDetectionCondition (DetectionCondition *condition*)

Sets the detection condition.

Control 2 (Read/Write): Motion Detection (OR condition) or Freefall Detection (AND condition)

```
PDPL
0: Pulse detection polarity is positive and detecting condition is OR 3 axes.
1: Pulse detection polarity is negative and detecting condition is AND 3 axes.
```

Parameters

condition	The detection condition.

Definition at line 114 of file AccelerometerMMA7455.cpp.

5.3.4.19 void AccelerometerMMA7455::setDeviceMode (DeviceMode *mode*)

Device mode.

Activate or deactivate the device.

Parameters

mode	A possible device mode.

Definition at line 34 of file AccelerometerMMA7455.cpp.

5.3.4.20 void AccelerometerMMA7455::setDynamicRange (DynamicRange *range*)

Set sensor to work in Ng range.

```
00: 8g is selected for measurement range.
10: 4g is selected for measurement range.
01: 2g is selected for measurement range

GLVL [1:0]   g-Range Sensitivity
     00      8g      16 LSB/g
     01      2g      64 LSB/g
     10      4g      32 LSB/g
```

Definition at line 118 of file AccelerometerMMA7455.cpp.

5.3.4.21 void AccelerometerMMA7455::setInterruptConfiguration (InterruptConfiguration *configuration*)

Sets the interrupt configuration;.

Parameters

configuration	The interrupt configuration.

Definition at line 148 of file AccelerometerMMA7455.cpp.

5.3.4.22 void AccelerometerMMA7455::setUse8bit (bool *use*)

Uses 8 bit measurement.

Definition at line 25 of file AccelerometerMMA7455.cpp.

5.3.4.23 void AccelerometerMMA7455::standby ( ) [inline]

Put sensor into Standby Mode.

Definition at line 377 of file AccelerometerMMA7455.h.

5.3.4.24 void AccelerometerMMA7455::writeRegister (Location *location,* unsigned char *v*)

Writes into the sensor register.

Parameters

reg	The new register.

Definition at line 181 of file AccelerometerMMA7455.cpp.

5.3.4.25 void AccelerometerMMA7455::writeRegisterBlock (unsigned char *to*, unsigned char * *buf*, unsigned char *len*)

Writes a block of data into the device starting at the 'to' register.

Parameters

to	The address to write.
buf	The buffer to be used.
len	The number of bytes to write.

Definition at line 191 of file AccelerometerMMA7455.cpp.

5.3.5 Member Data Documentation

5.3.5.1 unsigned char AccelerometerMMA7455::address [protected]

The device address.

Definition at line 590 of file AccelerometerMMA7455.h.

5.3.5.2 int AccelerometerMMA7455::int1Pin [protected]

The interruption 1 pin.

Definition at line 580 of file AccelerometerMMA7455.h.

5.3.5.3 int AccelerometerMMA7455::int2Pin [protected]

The interruption 2 pin.

Definition at line 585 of file AccelerometerMMA7455.h.

5.3.5.4 MCTLbits AccelerometerMMA7455::mctl [protected]

The current device mode control.

Definition at line 595 of file AccelerometerMMA7455.h.

5.3.5.5 bool AccelerometerMMA7455::use8bit [protected]

use 8bit mode

Definition at line 600 of file AccelerometerMMA7455.h.

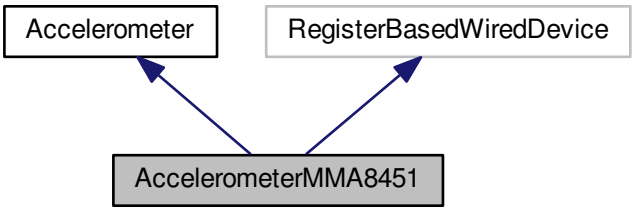
The documentation for this class was generated from the following files:

- AccelerometerMMA7455.h
- AccelerometerMMA7455.cpp

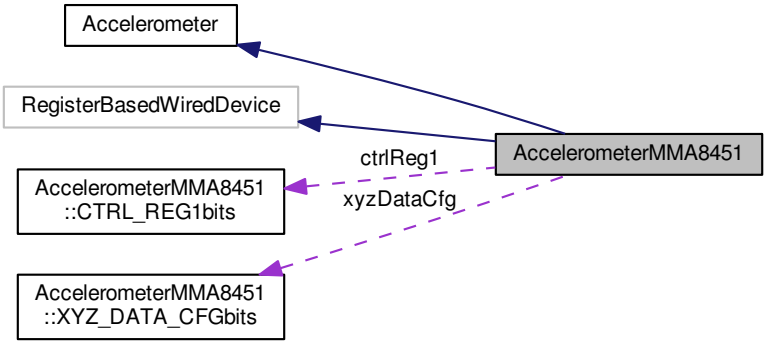
5.4 AccelerometerMMA8451 Class Reference

```
#include <AccelerometerMMA8451.h>
```

Inheritance diagram for AccelerometerMMA8451:



Collaboration diagram for AccelerometerMMA8451:



Classes

- union CTRL_REG1bits
- union CTRL_REG2bits
- union CTRL_REG3bits
- union CTRL_REG4bits
- union CTRL_REG5bits
- union F_SETUPbits
- union F_STATUSbits
- union FF_MT_CFGbits
- union FF_MT_SRCbits
- union FF_MT_THSbits
- union HP_FILTER_CUTOFFbits
- union INT_SOURCEbits
- union P_L_THS_REGbits
- union PL_BF_ZCOMPbits
- union PL_CFGbits
- union PL_STATUSbits
- union PULSE_CFGbits

- union PULSE_SRCbits
- union STATUSbits
- union SYSMODbits
- union TRANSIENT_CFGbits
- union TRANSIENT_SRCbits
- union TRANSIENT_THSbits
- union TRIG_CFGbits
- union XYZ_DATA_CFGbits

Public Types

- enum Location {
  STATUS = 0x00, F_STATUS = 0x00, OUT_X_MSB = 0x01, OUT_X_LSB = 0x02,
  OUT_Y_MSB = 0x03, OUT_Y_LSB = 0x04, OUT_Z_MSB = 0x05, OUT_Z_LSB = 0x06,
  F_SETUP = 0x09, TRIG_CFG = 0x0a, SYSMOD = 0x0b, INT_SOURCE = 0x0c,
  WHO_AM_I = 0x0d, XYZ_DATA_CFG = 0x0e, HP_FILTER_CUTOFF = 0x0f, PL_STATUS = 0x10,
  PL_CFG = 0x11, PL_COUNT = 0x12, PL_BF_ZCOMP = 0x13, P_L_THS_REG = 0x14,
  FF_MT_CFG = 0x15, FF_MT_SRC = 0x16, FF_MT_THS = 0x17, FF_MT_COUNT = 0x18,
  TRANSIENT_CFG = 0x1d, TRANSIENT_SRC = 0x1e, TRANSIENT_THS = 0x1f, TRANSIENT_COUNT = 0x20,
  PULSE_CFG = 0x21, PULSE_SRC = 0x22, PULSE_THSX = 0x23, PULSE_THSY = 0x24,
  PULSE_THSZ = 0x25, PULSE_TMLT = 0x26, PULSE_LTCY = 0x27, PULSE_WIND = 0x28,
  ASLP_COUNT = 0x29, CTRL_REG1 = 0x2a, CTRL_REG2 = 0x2b, CTRL_REG3 = 0x2c,
  CTRL_REG4 = 0x2d, CTRL_REG5 = 0x2e, OFF_X = 0x2f, OFF_Y = 0x30,
  OFF_Z = 0x31 }
- enum Mask {
  STATUS_XDR = 0x01, STATUS_YDR = 0x02, STATUS_ZDR = 0x04, STATUS_ZYXDR = 0x08,
  STATUS_XOW = 0x10, STATUS_YOW = 0x20, STATUS_ZOW = 0x40, STATUS_ZYXOW = 0x80,
  F_STATUS_F_CNT = 0x3f, F_STATUS_F_WMRK_FLAG = 0x40, F_STATUS_F_OVF = 0x80, F_SETUP↩
  _F_WMRK = 0x3f,
  F_SETUP_F_MODE = 0xc0, TRIG_CFG_TRIG_FF_MT = 0x04, TRIG_CFG_TRIG_PULSE = 0x08, TRIG↩
  _CFG_TRIG_LNDPRT = 0x10,
  TRIG_CFG_TRIG_TRANS = 0x20, SYSMOD_SYSMOD = 0x03, SYSMOD_FGT = 0x7c, SYSMOD_FGERR = 0x80,
  INT_SOURCE_SRC_DRDY = 0x01, INT_SOURCE_SRC_FF_MT = 0x04, INT_SOURCE_SRC_PULSE = 0x08, INT_SOURCE_SRC_LNDPRT = 0x10,
  INT_SOURCE_SRC_TRANS = 0x20, INT_SOURCE_SRC_FIFO = 0x40, INT_SOURCE_SRC_ASLP = 0x80, XYZ_DATA_CFG_FS = 0x03,
  XYZ_DATA_CFG_HPF_OUT = 0x10, HP_FILTER_CUTOFF_SEL = 0x03, HP_FILTER_PULSE_LPF_EN = 0x10, HP_FILTER_PULSE_HPF_BYP = 0x20,
  PL_STATUS_BAFRO = 0x01, PL_STATUS_LAPO = 0x06, PL_STATUS_LO = 0x40, PL_STATUS_NEWLP = 0x80,
  PL_CFG_PL_EN = 0x40, PL_CFG_DBCNTM = 0x80, PL_BF_ZCOMP_ZLOCK = 0x07, PL_BF_ZCOMP_↩
  BKFR = 0xc0,
  P_L_THS_REG_HYS = 0x07, P_L_THS_REG_P_L_THS = 0xf8, FF_MT_CFG_XEFE = 0x08, FF_MT_C↩
  FG_YEFE = 0x10,
  FF_MT_CFG_ZEFE = 0x20, FF_MT_CFG_OAE = 0x40, FF_MT_CFG_ELE = 0x80, FF_MT_SRC_XHP = 0x01,
  FF_MT_SRC_XHE = 0x02, FF_MT_SRC_YHP = 0x04, FF_MT_SRC_YHE = 0x08, FF_MT_SRC_ZHP = 0x10,
  FF_MT_SRC_ZHE = 0x20, FF_MT_SRC_EA = 0x80, FF_MT_THS_THS = 0x7f, FF_MT_THS_DBCNTM = 0x80,
  TRANSIENT_CFG_HPF_BYP = 0x01, TRANSIENT_CFG_XTEFE = 0x02, TRANSIENT_CFG_YTEFE = 0x04, TRANSIENT_CFG_ZTEFE = 0x08,
  TRANSIENT_CFG_ELE = 0x10, TRANSIENT_SRC_X_TRANS_POL = 0x01, TRANSIENT_SRC_XTRAN↩
  SE = 0x02, TRANSIENT_SRC_Y_TRANS_POL = 0x04,
  TRANSIENT_SRC_YTRANSE = 0x08, TRANSIENT_SRC_Z_TRANS_POL = 0x10, TRANSIENT_SRC_Z↩

TRANSE = 0x20, TRANSIENT_SRC_EA = 0x40,
TRANSIENT_THS_THS = 0x7f, TRANSIENT_THS_DBCNTM = 0x80, PULSE_CFG_XSPEFE = 0x01, PU↩
LSE_CFG_XDPEFE = 0x02,
PULSE_CFG_YSPEFE = 0x04, PULSE_CFG_YDPEFE = 0x08, PULSE_CFG_ZSPEFE = 0x10, PULSE_↩
CFG_ZDPEFE = 0x20,
PULSE_CFG_ELE = 0x40, PULSE_CFG_DPA = 0x80, PULSE_SRC_POLX = 0x01, PULSE_SRC_POLY =
0x02,
PULSE_SRC_POLZ = 0x04, PULSE_SRC_DPE = 0x08, PULSE_SRC_AXX = 0x10, PULSE_SRC_AXY =
0x20,
PULSE_SRC_AXZ = 0x40, PULSE_SRC_EA = 0x80, CTRL_REG1_ACTIVE = 0x01, CTRL_REG1_F_RE↩
AD = 0x02,
CTRL_REG1_LNOISE = 0x04, CTRL_REG1_DR = 0x38, CTRL_REG1_ASLP_RATE = 0xc0, CTRL_RE↩
G2_MODS = 0x03,
CTRL_REG2_SLPE = 0x04, CTRL_REG2_SMODS = 0x18, CTRL_REG2_RST = 0x40, CTRL_REG2_ST =
0x80,
CTRL_REG3_PP_OD = 0x01, CTRL_REG3_IPOL = 0x02, CTRL_REG3_WAKE_FF_MT = 0x08, CTRL_↩
REG3_WAKE_PULSE = 0x10,
CTRL_REG3_WAKE_LNDPRT = 0x20, CTRL_REG3_WAKE_TRANS = 0x40, CTRL_REG3_FIFO_GATE =
0x80, CTRL_REG4_INT_EN_DRDY = 0x01,
CTRL_REG4_INT_EN_FF_MT = 0x04, CTRL_REG4_INT_EN_PULSE = 0x08, CTRL_REG4_INT_EN_L↩
NDPR = 0x10, CTRL_REG4_INT_EN_TRANS = 0x20,
CTRL_REG4_INT_EN_FIFO = 0x40, CTRL_REG4_INT_EN_ASLP = 0x80, CTRL_REG5_INT_CFG_DRDY
= 0x01, CTRL_REG5_INT_CFG_FF_MT = 0x04,
CTRL_REG5_INT_CFG_PULSE = 0x08, CTRL_REG5_INT_CFG_LNDPRT = 0x10, CTRL_REG5_INT_C↩
FG_TRANS = 0x20, CTRL_REG5_INT_CFG_FIFO = 0x40,
CTRL_REG5_INT_CFG_ASLP = 0x80 }

- enum DeviceActivation { DEACTIVATE = 0x00, ACTIVATE = 0x01 }
- enum DynamicRange { DR_2G = 0x00, DR_4G = 0x01, DR_8G = 0x02 }
- enum OutputDataRate {
  ODR_800HZ_1_25_MS = 0x00, ODR_400HZ_2_5_MS = 0x01, ODR_200HZ_5_MS = 0x02, ODR_100H↩
  Z_10_MS = 0x03,
  ODR_50HZ_20_MS = 0x04, ODR_12_5HZ_80_MS = 0x05, ODR_6_25HZ_1_160_MS = 0x06, ODR_1_↩
  563HZ_1_640_MS = 0x07 }
- enum OversamplingMode { NORMAL_MODS = 0x00, LOW_NOISE_LOW_POWER_MODS = 0x01, HI_R↩
  ESOLUTION_MODS = 0x02, LOW_POWER_MODS = 0x03 }
- enum HighPassFilterCutoffFrequency { HP_FILTER_CUTOFF_0 = 0x00, HP_FILTER_CUTOFF_1 = 0x01,
  HP_FILTER_CUTOFF_2 = 0x02, HP_FILTER_CUTOFF_3 = 0x03 }
- enum FifoBufferOverflowMode { FIFO_DISABLED = 0x00, FIFO_CIRCULAR_BUFFER = 0x01, FIFO_ST↩
  OP_WHEN_OVERFLOWED = 0x02, FIFO_TRIGGER = 0x03 }
- enum ReadMode { FAST_READ = 0x01, NORMAL_READ = 0x00 }
- enum HysteresisAngle {
  HYS_0 = 0x00, HYS_4 = 0x01, HYS_7 = 0x02, HYS_11 = 0x03,
  HYS_14 = 0x04, HYS_17 = 0x05, HYS_21 = 0x06, HYS_24 = 0x07 }
- enum Interrupt {
  INT_DRDY = 0x01, INT_FF_MT = 0x04, INT_PULSE = 0x08, INT_LNDPRT = 0x10,
  INT_TRANS = 0x20, INT_FIFO = 0x40, INT_ASLP = 0x80, INT_ALL = 0xff }
- enum InterruptPolarity { ACTIVE_LOW = 0x00, ACTIVE_HIGH = 0x01 }
- enum PushPullOpenDrain { PUSH_PULL = 0x00, OPEN_DRAIN = 0x01 }
- enum InternalErrors { BUS_ERROR_READ = 0x00 }

Public Member Functions

- AccelerometerMMA8451 (bool sa0)
- virtual float readXg ()
- virtual float readYg ()
- virtual float readZg ()
- void readXYZ (unsigned char buf[6])

- bool isDataReady ()
- void deviceActivation (DeviceActivation activation)
- void standby ()
- void activate ()
- void setDynamicRange (DynamicRange range)
- void setOutputDataRate (OutputDataRate rate)
- void setPortraitLandscapeDetection (bool enable)
- void setBackFrontTrip (BackFrontTrip trip)
- void setZLockThresholdAngle (ZLockThresholdAngle angle)
- void setPortraitLandscapeThresholdAngle (PortraitLandscapeThresholdAngle angle)
- void setHysteresisAngle (HysteresisAngle angle)
- void enableInterrupt (Interrupt interrupt)
- void enableInterrupt (Interrupt interrupt, unsigned char routePin)
- void disableInterrupt (Interrupt interrupt)
- void routeInterruptToInt1 (Interrupt interrupt)
- void routeInterruptToInt2 (Interrupt interrupt)
- void routeInterrupt (Interrupt interrupt, unsigned char routePin)
- void setInterruptPolarity (InterruptPolarity polarity)
- void setAslpOutputDataRate (AslpOutputDataRate rate)
- void setReadMode (ReadMode mode)
- void setOversamplingMode (OversamplingMode mode)
- void setHighPassFilterCutoffFrequency (HighPassFilterCutoffFrequency frequency)
- void highPassFilteredData (bool filtered)
- void setFifoBufferOverflowMode (FifoBufferOverflowMode mode)
- void setFifoWatermark (unsigned char watermark)
- bool getFifoGateError ()
- unsigned char getFifoFgt ()
- unsigned char getSysmod ()
- void setPushPullOpenDrain (PushPullOpenDrain ppod)
- float convertToG (unsigned char *buf, bool fastRead)
- unsigned char gerLastError ()

Protected Attributes

- int int1Pin
- int int2Pin
- unsigned char lastError
- XYZ_DATA_CFGbits xyzDataCfg
- CTRL_REG1bits ctrlReg1

5.4.1 Detailed Description

Arduino - Accelerometer driver.

AccelerometerMMA8451.h

The implementation of the MMA8451 accelerometer.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 18 of file AccelerometerMMA8451.h.

5.4.2 Member Enumeration Documentation

5.4.2.1 enum AccelerometerMMA8451::DeviceActivation

Enable/Disable device.

Enumerator

DEACTIVATE

ACTIVATE

Definition at line 651 of file AccelerometerMMA8451.h.

5.4.2.2 enum AccelerometerMMA8451::DynamicRange

Dynamic range.

Table 3.Full Scale Selection

```
FS1 FS0  g Range
0    0   +/-2g
0    1   +/-4g
1    0   +/-8g
1    1   -
```

Enumerator

DR_2G

DR_4G

DR_8G

Definition at line 668 of file AccelerometerMMA8451.h.

5.4.2.3 enum AccelerometerMMA8451::FifoBufferOverflowMode

FIFO buffer overflow mode.

FIFO buffer overflow mode. Default value: 0. 00: FIFO is disabled. 01: FIFO contains the most recent samples when overflowed (circular buffer). Oldest sample is discarded to be replaced by new sample. 10: FIFO stops accepting new samples when overflowed. 11: Trigger mode. The FIFO will be in a circular mode up to the number of samples in the watermark. The FIFO will be in a circular mode until the trigger event occurs after that the FIFO will continue to accept samples for 32-WMRK samples and then stop receiving further samples. This allows data to be collected both before and after the trigger event and it is definable by the watermark setting. The FIFO is flushed whenever the FIFO is disabled, during an automatic ODR change (Auto-WAKE/SLEEP), or transitioning from STANDBY mode to ACTIVE mode. Disabling the FIFO (F_MODE = 00) resets the F_OVF, F_WMRK_FLAG, F_CNT to zero. A FIFO overflow event (i.e., F_CNT = 32) will assert the F_OVF flag and a FIFO sample count equal to the sample count watermark (i.e., F_WMRK) asserts the F_WMRK_FLAG event flag.

Enumerator

FIFO_DISABLED

FIFO_CIRCULAR_BUFFER

FIFO_STOP_WHEN_OVERFLOWED

FIFO_TRIGGER

Definition at line 772 of file AccelerometerMMA8451.h.

5.4.2.4 enum AccelerometerMMA8451::HighPassFilterCutoffFrequency

See Table 8.

HP_FILTER_CUTOFF Setting Options.

Enumerator

HP_FILTER_CUTOFF_0

HP_FILTER_CUTOFF_1

HP_FILTER_CUTOFF_2

HP_FILTER_CUTOFF_3

Definition at line 741 of file AccelerometerMMA8451.h.

5.4.2.5 enum AccelerometerMMA8451::HysteresisAngle

Trigger bits.

```
   INT_SOURCE   Description

   Trig_TRANS   Transient interrupt trigger bit. Default value: 0
   Trig_LNDPRT  Landscape/Portrait Orientation interrupt trigger
                bit. Default value: 0
   Trig_PULSE   Pulse interrupt trigger bit. Default value: 0
   Trig_FF_MT   Freefall/Motion trigger bit. Default value: 0


 */
enum InterruptTriggerBits {
    TRIG_TRANS = 0x20,
    TTRIG_LNDPRT = 0x10,
    TRIG_PULSE = 0x08,
    TTRIG_FF_MT = 0x04
};



/**
   Back/Front Trip Angle Threshold. Default: 01 >= +/-75°.
   Step size is 5°.
   Range: +/-(65° to 80°).
 */
enum BackFrontTrip {
    BKFR_80 = 0x00,
    BKFR_75 = 0x01,
    BKFR_70 = 0x02,
    BKFR_65 = 0x03
};



/**
   Z-Lock Angle Threshold. Range is from 14° to 43°.
   Step size is 4°.
   Default value: 100 >= 29°.
   Maximum value: 111 >= 43°.
 */
enum ZLockThresholdAngle {
    ZLOCK_14 = 0x00,
    ZLOCK_18 = 0x01,
    ZLOCK_21 = 0x02,
```

```
    ZLOCK_25 = 0x03,
    ZLOCK_29 = 0x04,
    ZLOCK_33 = 0x05,
    ZLOCK_37 = 0x06,
    ZLOCK_42 = 0x07
};


enum PortraitLandscapeThresholdAngle {
    P_L_THS_15 = 0x07,
    P_L_THS_20 = 0x09,
    P_L_THS_30 = 0x0c,
    P_L_THS_35 = 0x0d,
    P_L_THS_40 = 0x0f,
    P_L_THS_45 = 0x10,
    P_L_THS_55 = 0x13,
    P_L_THS_60 = 0x14,
    P_L_THS_70 = 0x17,
    P_L_THS_75 = 0x19
};


/**
   Trip Angles with Hysteresis for 45° Angle


   Hysteresis       Hysteresis      Landscape to Portrait   Portrait to Landscape
   Register Value   +/- Angle Range   Trip Angle              Trip Angle
   0                +/-0              45°                     45°
   1                +/-4              49°                     41°
   2                +/-7              52°                     38°
   3                +/-11             56°                     34°
   4                +/-14             59°                     31°
   5                +/-17             62°                     28°
   6                +/-21             66°                     24°
   7                +/-24             69°                     21°
```

Enumerator

HYS_0

HYS_4

HYS_7

HYS_11

HYS_14

HYS_17

HYS_21

HYS_24

Definition at line 867 of file AccelerometerMMA8451.h.

5.4.2.6 enum AccelerometerMMA8451::InternalErrors

Internal errors.

Enumerator

BUS_ERROR_READ

Definition at line 967 of file AccelerometerMMA8451.h.

5.4.2.7 enum AccelerometerMMA8451::Interrupt

INT_EN_ASLP 0: Auto-SLEEP/WAKE interrupt disabled; 1: Auto-SLEEP/WAKE interrupt enabled.

INT_EN_FIFO 0: FIFO interrupt disabled; 1: FIFO interrupt enabled.

INT_EN_TRANS 0: Transient interrupt disabled; 1: Transient interrupt enabled.

INT_EN_LNDPRT 0: Orientation (Landscape/Portrait) interrupt disabled. 1: Orientation (Landscape/Portrait) interrupt enabled.

INT_EN_PULSE 0: Pulse Detection interrupt disabled; 1: Pulse Detection interrupt enabled

INT_EN_FF_MT 0: Freefall/Motion interrupt disabled; 1: Freefall/Motion interrupt enabled

INT_EN_DRDY 0: Data Ready interrupt disabled; 1: Data Ready interrupt enabled

INT_CFG_ASLP 0: Interrupt is routed to INT2 pin; 1: Interrupt is routed to INT1 pin

INT_CFG_FIFO 0: Interrupt is routed to INT2 pin; 1: Interrupt is routed to INT1 pin

INT_CFG_TRANS 0: Interrupt is routed to INT2 pin; 1: Interrupt is routed to INT1 pin

INT_CFG_LNDPRT 0: Interrupt is routed to INT2 pin; 1: Interrupt is routed to INT1 pin

INT_CFG_PULSE 0: Interrupt is routed to INT2 pin; 1: Interrupt is routed to INT1 pin

INT_CFG_FF_MT 0: Interrupt is routed to INT2 pin; 1: Interrupt is routed to INT1 pin

INT_CFG_DRDY 0: Interrupt is routed to INT2 pin; 1: Interrupt is routed to INT1 pin

Enumerator

INT_DRDY

INT_FF_MT

INT_PULSE

INT_LNDPRT

INT_TRANS

INT_FIFO

INT_ASLP

INT_ALL

Definition at line 935 of file AccelerometerMMA8451.h.

5.4.2.8 enum AccelerometerMMA8451::InterruptPolarity

Interrupt polarity.

Enumerator

ACTIVE_LOW

ACTIVE_HIGH

Definition at line 949 of file AccelerometerMMA8451.h.

5.4.2.9 enum AccelerometerMMA8451::Location

Internal registers.

Enumerator

STATUS

F_STATUS

OUT_X_MSB

OUT_X_LSB
OUT_Y_MSB
OUT_Y_LSB
OUT_Z_MSB
OUT_Z_LSB
F_SETUP
TRIG_CFG
SYSMOD
INT_SOURCE
WHO_AM_I
XYZ_DATA_CFG
HP_FILTER_CUTOFF
PL_STATUS
PL_CFG
PL_COUNT
PL_BF_ZCOMP
P_L_THS_REG
FF_MT_CFG
FF_MT_SRC
FF_MT_THS
FF_MT_COUNT
TRANSIENT_CFG
TRANSIENT_SRC
TRANSIENT_THS
TRANSIENT_COUNT
PULSE_CFG
PULSE_SRC
PULSE_THSX
PULSE_THSY
PULSE_THSZ
PULSE_TMLT
PULSE_LTCY
PULSE_WIND
ASLP_COUNT
CTRL_REG1
CTRL_REG2
CTRL_REG3
CTRL_REG4
CTRL_REG5
OFF_X
OFF_Y
OFF_Z

Definition at line 453 of file AccelerometerMMA8451.h.

5.4.2.10 enum AccelerometerMMA8451::Mask

Some useful masks.

Enumerator

STATUS_XDR
STATUS_YDR
STATUS_ZDR
STATUS_ZYXDR
STATUS_XOW
STATUS_YOW
STATUS_ZOW
STATUS_ZYXOW
F_STATUS_F_CNT
F_STATUS_F_WMRK_FLAG
F_STATUS_F_OVF
F_SETUP_F_WMRK
F_SETUP_F_MODE
TRIG_CFG_TRIG_FF_MT
TRIG_CFG_TRIG_PULSE
TRIG_CFG_TRIG_LNDPRT
TRIG_CFG_TRIG_TRANS
SYSMOD_SYSMOD
SYSMOD_FGT
SYSMOD_FGERR
INT_SOURCE_SRC_DRDY
INT_SOURCE_SRC_FF_MT
INT_SOURCE_SRC_PULSE
INT_SOURCE_SRC_LNDPRT
INT_SOURCE_SRC_TRANS
INT_SOURCE_SRC_FIFO
INT_SOURCE_SRC_ASLP
XYZ_DATA_CFG_FS
XYZ_DATA_CFG_HPF_OUT
HP_FILTER_CUTOFF_SEL
HP_FILTER_PULSE_LPF_EN
HP_FILTER_PULSE_HPF_BYP
PL_STATUS_BAFRO
PL_STATUS_LAPO
PL_STATUS_LO
PL_STATUS_NEWLP
PL_CFG_PL_EN
PL_CFG_DBCNTM
PL_BF_ZCOMP_ZLOCK
PL_BF_ZCOMP_BKFR
P_L_THS_REG_HYS

CTRL_REG1_ACTIVE
CTRL_REG1_F_READ
CTRL_REG1_LNOISE
CTRL_REG1_DR
CTRL_REG1_ASLP_RATE
CTRL_REG2_MODS
CTRL_REG2_SLPE
CTRL_REG2_SMODS
CTRL_REG2_RST
CTRL_REG2_ST
CTRL_REG3_PP_OD
CTRL_REG3_IPOL
CTRL_REG3_WAKE_FF_MT
CTRL_REG3_WAKE_PULSE
CTRL_REG3_WAKE_LNDPRT
CTRL_REG3_WAKE_TRANS
CTRL_REG3_FIFO_GATE
CTRL_REG4_INT_EN_DRDY
CTRL_REG4_INT_EN_FF_MT
CTRL_REG4_INT_EN_PULSE
CTRL_REG4_INT_EN_LNDPR
CTRL_REG4_INT_EN_TRANS
CTRL_REG4_INT_EN_FIFO
CTRL_REG4_INT_EN_ASLP
CTRL_REG5_INT_CFG_DRDY
CTRL_REG5_INT_CFG_FF_MT
CTRL_REG5_INT_CFG_PULSE
CTRL_REG5_INT_CFG_LNDPRT
CTRL_REG5_INT_CFG_TRANS
CTRL_REG5_INT_CFG_FIFO
CTRL_REG5_INT_CFG_ASLP

Definition at line 504 of file AccelerometerMMA8451.h.

5.4.2.11 enum AccelerometerMMA8451::OutputDataRate

Table 5.

Output Data Rates

DR2	DR1	DR0	Output	Data Rate (ODR)	Time Between Data Samples
0	0	0	800 Hz	1.25 ms	
0	0	1	400 Hz	2.5 ms	
0	1	0	200 Hz	5 ms	
0	1	1	100 Hz	10 ms	
1	0	0	50 Hz	20 ms	
1	0	1	12.5 Hz	80 ms	
1	1	0	6.25 Hz	160 ms	
1	1	1	1.563 Hz	640 ms	

Enumerator

ODR_800HZ_1_25_MS

ODR_400HZ_2_5_MS

ODR_200HZ_5_MS

ODR_100HZ_10_MS

ODR_50HZ_20_MS

ODR_12_5HZ_80_MS

ODR_6_25HZ_1_160_MS

ODR_1_563HZ_1_640_MS

Definition at line 689 of file AccelerometerMMA8451.h.

5.4.2.12 enum AccelerometerMMA8451::OversamplingMode

It is important to note that when the device is Auto-SLEEP mode, the system ODR and the data rate for all the system functional blocks are overridden by the data rate set by the ASLP_RATE field.

```
   ASLP_RATE1   ASLP_RATE0  Frequency (Hz)
   0            0           50
   0            1           12.5
   1            0           6.25
   1            1           1.56
 */
enum AslpOutputDataRate {
    ASLP_50HZ = 0x00,
    ASLP_12_5HZ = 0x01,
    ASLP_6_25HZ = 0x02,
    ASLP_1_56HZ = 0x03
};

/**
   Oversampling Mode.

   MODS1    MODS0
   0        0       Normal
   0        1       Low noise, low power
   1        0       High resolution
   1        1       Low power
```

Enumerator

NORMAL_MODS

LOW_NOISE_LOW_POWER_MODS

HI_RESOLUTION_MODS

LOW_POWER_MODS

Definition at line 731 of file AccelerometerMMA8451.h.

5.4.2.13 enum AccelerometerMMA8451::PushPullOpenDrain

Push-Pull/Open Drain selection on interrupt pad.

Default value: 0. 0: Push-Pull; 1: Open Drain

Enumerator

PUSH_PULL

OPEN_DRAIN

Definition at line 959 of file AccelerometerMMA8451.h.

5.4.2.14 enum AccelerometerMMA8451::ReadMode

Fast Read mode: Data format limited to single Byte Default value: 0.

(0: Normal mode 1: Fast Read Mode)

Enumerator

FAST_READ

NORMAL_READ

Definition at line 784 of file AccelerometerMMA8451.h.

5.4.3 Constructor & Destructor Documentation

5.4.3.1 AccelerometerMMA8451::AccelerometerMMA8451 (bool *sa0*)

Public constructor.

Parameters

sa0	The LSBit of the address.

Definition at line 16 of file AccelerometerMMA8451.cpp.

5.4.4 Member Function Documentation

5.4.4.1 void AccelerometerMMA8451::activate ( ) [inline]

Put sensor into Active Mode.

Read current value of System Control 1 Register. Put sensor into Active Mode by setting the Active bit Return with previous value of System Control 1 Register.

Definition at line 1056 of file AccelerometerMMA8451.h.

5.4.4.2 float AccelerometerMMA8451::convertToG (unsigned char * *buf*, bool *fastRead*)

Converts an array of chars into a float type.

Parameters

buf	1 (8-bit) our 2 (14-bit) bytes to be converted.
fastRead	boolean indication if the buffer has a fast read value.

Definition at line 176 of file AccelerometerMMA8451.cpp.

5.4.4.3 void AccelerometerMMA8451::deviceActivation (DeviceActivation *activation*)

Device activation.

Activate or deactivate the device.

Parameters

activation	

Definition at line 21 of file AccelerometerMMA8451.cpp.

5.4.4.4 void AccelerometerMMA8451::disableInterrupt (Interrupt *interrupt*)

Disable some interrupt.

Parameters

interrupt	The interrupt flag.

Definition at line 104 of file AccelerometerMMA8451.cpp.

5.4.4.5 void AccelerometerMMA8451::enableInterrupt (Interrupt *interrupt*)

Enables some interrupt.

NOTE: The interrupt will be routed to the default pin.

Parameters

interrupt	The interrupt flag.

Definition at line 95 of file AccelerometerMMA8451.cpp.

5.4.4.6 void AccelerometerMMA8451::enableInterrupt (Interrupt *interrupt*, unsigned char *routePin*)

Enables some interrupt, and route to the given pin.

Parameters

interrupt	The interrupt flag.
routePin	The pin where the interrupt will be routed. should be 1 or 2

Definition at line 99 of file AccelerometerMMA8451.cpp.

5.4.4.7 unsigned char AccelerometerMMA8451::gerLastError ( )

Gets the last error happened.

Returns

The last error, 0 means no error.

Definition at line 194 of file AccelerometerMMA8451.cpp.

5.4.4.8 unsigned char AccelerometerMMA8451::getFifoFgt ( )

Gets the FIFO Fgt.

Definition at line 164 of file AccelerometerMMA8451.cpp.

5.4.4.9 bool AccelerometerMMA8451::getFifoGateError ( )

Gets the FIFO Gate Error.

Definition at line 158 of file AccelerometerMMA8451.cpp.

5.4.4.10 unsigned char AccelerometerMMA8451::getSysmod ( )

Gets the FIFO Sysmode.

Definition at line 170 of file AccelerometerMMA8451.cpp.

5.4.4.11 void AccelerometerMMA8451::highPassFilteredData (bool *filtered*)

Registers 0x01 through 0x06 are used to read the X, Y, Z data.

The device can be configured to produce high-pass filtered data or low-pass filtered data by setting or clearing the HPF_Out bit in the XYZ_Data_Cfg Register 0x0E. The following code example shows how to set the HPF_Out bit.

Definition at line 141 of file AccelerometerMMA8451.cpp.

5.4.4.12 bool AccelerometerMMA8451::isDataReady ( )

Return true if the data is ready to be read.

Returns

Definition at line 25 of file AccelerometerMMA8451.cpp.

5.4.4.13 float AccelerometerMMA8451::readXg ( ) [virtual]

Reads the x axis from the accelerometer device.

The x result.

Implements Accelerometer.

Definition at line 36 of file AccelerometerMMA8451.cpp.

5.4.4.14 void AccelerometerMMA8451::readXYZ (unsigned char *buf[6]*)

The MMA8451Q has 14-bit XYZ data.

The MMA8452Q has 12-bit XYZ data and the MMA8453 has 10-bit data. This section is an overview of how to manipulate the data to continuously burst out 14-bit data in different data formats from the MCU. The examples will be shown for the 14-bit data but the reader can understand what changes would be made for the 12-bit data or the 10-bit data. The driver code has all the functions for all data formats available. The event flag can be monitored by reading the STATUS register (0x00). This can be done by using either a polling or interrupt technique, which is discussed later in Section 9.0 of this document. It is not absolutely necessary to read the STATUS register to clear it. Reading the data clears the STATUS register. Table 9. 0x00 STATUS: Data Status Registers (Read Only) Bit 7 Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 Bit 0 ZYXOW ZOW YOW XOW ZYXDR ZDR YDR XDR The ZYXDR flag is set whenever there is new data available in any axis. The following code example monitors this flag and, upon the detection of new data, reads the 14/12/10-bit XYZ data into an array (value[]) in RAM with a single, multi-byte I2C access. These values are then copied into 16-bit variables prior to further processing.

Definition at line 60 of file AccelerometerMMA8451.cpp.

5.4.4.15 float AccelerometerMMA8451::readYg ( ) [virtual]

Reads the y axis from the accelerometer device.

The y result.

Implements Accelerometer.

Definition at line 44 of file AccelerometerMMA8451.cpp.

5.4.4.16 float AccelerometerMMA8451::readZg ( ) [virtual]

Reads the z axis from the accelerometer device.

The z result.

Implements Accelerometer.

Definition at line 52 of file AccelerometerMMA8451.cpp.

5.4.4.17 void AccelerometerMMA8451::routeInterrupt (Interrupt *interrupt,* unsigned char *routePin*)

Interrupt is routed to the given pin;.

Parameters

interrupt	The interrupt flag.
routePin	The pin where the interrupt will be routed. should be 1 or 2

Definition at line 116 of file AccelerometerMMA8451.cpp.

5.4.4.18 void AccelerometerMMA8451::routeInterruptToInt1 (Interrupt *interrupt*)

Interrupt is routed to INT1 pin;.

Deprecated

Parameters

interrupt	The interrupt flag.

Definition at line 108 of file AccelerometerMMA8451.cpp.

5.4.4.19 void AccelerometerMMA8451::routeInterruptToInt2 (Interrupt *interrupt*)

Interrupt is routed to INT2 pin;.

Deprecated

Parameters

interrupt	The interrupt flag.

Definition at line 112 of file AccelerometerMMA8451.cpp.

5.4.4.20 void AccelerometerMMA8451::setAslpOutputDataRate (AslpOutputDataRate *rate*)

Sets the sllep output data rate.

It is important to note that when the device is Auto-SLEEP mode, the system ODR and the data rate for all the system functional blocks are overridden by the data rate set by the ASLP_RATE field.

Parameters

rate	The Output Data Rate.

Definition at line 124 of file AccelerometerMMA8451.cpp.

5.4.4.21 void AccelerometerMMA8451::setBackFrontTrip (BackFrontTrip *trip*)

Sets the Back/Front Trip Angle Threshold.

Default: 01 >= +/-75º. Step size is 5º. Range: +/-(65º to 80º).

Parameters

trip	The trip.

Definition at line 79 of file AccelerometerMMA8451.cpp.

5.4.4.22 void AccelerometerMMA8451::setDynamicRange (DynamicRange *range*)

Set sensor to work in Ng range by changing the FS1 and the FS0.

There are 3 different dynamic ranges that can be set (2g, 4g, 8g). The dynamic range is changeable only in the Standby Mode. The dynamic range is controlled by setting the FS0 and FS1 bits in register 0x0E. The device changes from Standby to Active Mode via bit 0 in register 0x2A.

Table 3. Full Scale Selection

```
FS1  FS0 g Range
0    0   +/-2g
```

```
0     1    +/-4g
1     0    +/-8g
1     1    -
```

Definition at line 66 of file AccelerometerMMA8451.cpp.

5.4.4.23 void AccelerometerMMA8451::setFifoBufferOverflowMode (FifoBufferOverflowMode *mode*)

Sets the fifo overflow mode.

NOTE: The FIFO mode can be changed while in the active state. The mode must first be disabled F_MODE = 00 then the mode can be switched between Fill mode, Circular mode and Trigger mode.

Parameters

mode	The overflow mode.

Definition at line 149 of file AccelerometerMMA8451.cpp.

5.4.4.24 void AccelerometerMMA8451::setFifoWatermark (unsigned char *watermark*)

Sets the fifo watermark.

FIFO Event Sample Count Watermark. Default value: 00_0000. These bits set the number of FIFO samples required to trigger a watermark interrupt. A FIFO watermark event flag is raised when FIFO sample count F_CN↩ T[5:0] >= F_WMRK[5:0] watermark.

Setting the F_WMRK[5:0] to 00_0000 will disable the FIFO watermark event flag generation. Also used to set the number of pre-trigger samples in Trigger mode.

Parameters

watermark	The fifo watermark count.

Definition at line 154 of file AccelerometerMMA8451.cpp.

5.4.4.25 void AccelerometerMMA8451::setHighPassFilterCutoffFrequency (HighPassFilterCutoffFrequency *frequency*)

Sets the High-Pass Filter Cutoff Frequency.

The HP_FILTER_CUTOFF register (at 0x0F) sets the high-pass cutoff frequency, Fc, for the data. The output of this filter is provided in the output data registers (0x01 to 0x06). Note that the high-pass filtered output data is available for the MMA8451Q and the MMA8452Q only. The MMA8453Q has the internal high-pass filter for the embedded functions but does not have access to the output data. The available cutoff frequencies change depending upon the set Output Data Rate.

Parameters

hpf	The High-Pass Filter Cutoff Frequency.

Definition at line 137 of file AccelerometerMMA8451.cpp.

5.4.4.26 void AccelerometerMMA8451::setHysteresisAngle (HysteresisAngle *angle*)

Sets the Hysteresis Angle.

This angle is added to the threshold angle for a smoother transition from Portrait to Landscape and Landscape to Portrait. This angle ranges from 0º to +/-24º. The default is 100 (+/-14º).

Parameters

angle	The angle.

Definition at line 91 of file AccelerometerMMA8451.cpp.

5.4.4.27 void AccelerometerMMA8451::setInterruptPolarity (InterruptPolarity *polarity*)

Sets the interrupt polarity.

Parameters

polarity	The polarity of the interrupt.

Definition at line 120 of file AccelerometerMMA8451.cpp.

5.4.4.28 void AccelerometerMMA8451::setOutputDataRate (OutputDataRate *rate*)

Sets the Output Data Rate.

The active mode Output Data Rate (ODR) and Sleep Mode Data Rate are programmable via other control bits in the CTRL_REG1 register, seen in Table 4. Unless the sleep mode is enabled the active mode data rate is the data rate that will always be enabled. Table 5 shows how the DR2:DR0 bits affect the ODR. These are the active mode data rates available. The default data rate is DR = 000, 800 Hz.

Parameters

rate	The Output Data Rate.

Definition at line 71 of file AccelerometerMMA8451.cpp.

5.4.4.29 void AccelerometerMMA8451::setOversamplingMode (OversamplingMode *mode*)

Sets the oversampling mode.

There are four different oversampling modes. There is a normal mode, a low noise + power mode, a high-resolution mode and a low-power mode. The difference between these are the amount of averaging of the sampled data, which is done internal to the device. The following chart shows the amount of averaging at each data rate, which is the OSRatio (oversampling ratio). There is a trade-off between the oversampling and the current consumption at each ODR value.

Parameters

om	

Definition at line 133 of file AccelerometerMMA8451.cpp.

5.4.4.30 void AccelerometerMMA8451::setPortraitLandscapeDetection (bool *enable*)

Portrait/Landscape Detection Enable.

Default value: 0 0: Portrait/Landscape Detection is Disabled. 1: Portrait/Landscape Detection is Enabled.

Parameters

enable	The enable flag.

Definition at line 75 of file AccelerometerMMA8451.cpp.

5.4.4.31 void AccelerometerMMA8451::setPortraitLandscapeThresholdAngle (PortraitLandscapeThresholdAngle *angle*)

Sets Portrait/Landscape trip threshold angle from 15º to 75º.

See Table 31 for the values with the corresponding approximate threshold angle. Default value: 1_0000 (45º).

Parameters

angle	The angle.

Definition at line 87 of file AccelerometerMMA8451.cpp.

5.4.4.32 void AccelerometerMMA8451::setPushPullOpenDrain (PushPullOpenDrain *ppod*)

Sets the selection on interrupt pad.

Parameters

ppod	PushPullOpenDrain

Definition at line 190 of file AccelerometerMMA8451.cpp.

5.4.4.33 void AccelerometerMMA8451::setReadMode (ReadMode *mode*)

Set the read mode.

F_READ bit selects between normal and Fast Read mode. When selected, the auto increment counter will skip over the LSB data bytes. Data read from the FIFO will skip over the LSB data, reducing the acquisition time. Note F_READ can only be changed when FMODE = 00. The F_READ bit applies for both the output registers and the FIFO.

Parameters

mode	The read mode.

Definition at line 128 of file AccelerometerMMA8451.cpp.

5.4.4.34 void AccelerometerMMA8451::setZLockThresholdAngle (ZLockThresholdAngle *angle*)

Sets the Z-Lock Angle Threshold.

Range is from 14º to 43º. Step size is 4º. Default value: 100 >= 29º. Maximum value: 111 >= 43º.

Parameters

angle	The angle.

Definition at line 83 of file AccelerometerMMA8451.cpp.

5.4.4.35 void AccelerometerMMA8451::standby ( ) [inline]

Put sensor into Standby Mode.

Read current value of System Control 1 Register. Put sensor into Standby Mode by clearing the Active bit Return with previous value of System Control 1 Register.

Definition at line 1045 of file AccelerometerMMA8451.h.

5.4.5 Member Data Documentation

5.4.5.1 CTRL_REG1bits AccelerometerMMA8451::ctrlReg1 [protected]

Holds the system Control 1 Register.

It is important to hold this on the object to avoid unnecessary read operations on the device.

Definition at line 1367 of file AccelerometerMMA8451.h.

5.4.5.2 int AccelerometerMMA8451::int1Pin [protected]

The interruption 1 pin.

Definition at line 1341 of file AccelerometerMMA8451.h.

5.4.5.3 int AccelerometerMMA8451::int2Pin [protected]

The interruption 2 pin.

Definition at line 1346 of file AccelerometerMMA8451.h.

5.4.5.4 unsigned char AccelerometerMMA8451::lastError [protected]

Last error.

Definition at line 1351 of file AccelerometerMMA8451.h.

5.4.5.5 XYZ_DATA_CFGbits AccelerometerMMA8451::xyzDataCfg [protected]

Holds the current Dynamic Range Settings.

It is important to hold this on the object to avoid unnecessary read operations on the device.

Definition at line 1359 of file AccelerometerMMA8451.h.

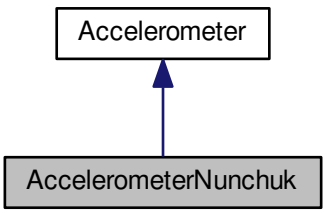
The documentation for this class was generated from the following files:

- AccelerometerMMA8451.h
- AccelerometerMMA8451.cpp

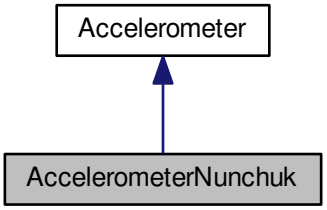
5.5 AccelerometerNunchuk Class Reference

`#include <AccelerometerNunchuk.h>`

Inheritance diagram for AccelerometerNunchuk:



Collaboration diagram for AccelerometerNunchuk:



Public Types

- enum Axis { AXIS_X = 0x00, AXIS_Y = 0x01, AXIS_Z = 0x02 }

Public Member Functions

- AccelerometerNunchuk ()
- float readZg (bool updateFrame)
- virtual float readZg ()
- float readXg (bool updateFrame)
- virtual float readXg ()
- float readYg (bool updateFrame)
- virtual float readYg ()
- bool readZButton (bool updateFrame)
- bool readZButton ()
- bool readCButton (bool updateFrame)
- bool readCButton ()
- unsigned char readXJoystick (bool updateFrame)
- unsigned char readXJoystick ()
- unsigned char readYJoystick (bool updateFrame)
- unsigned char readYJoystick ()
- float convertToG (unsigned int i)
- void readFrame ()
- unsigned char * getFrame ()
- void begin ()
- unsigned int readAcceleration (Axis axis, bool updateFrame)

Protected Member Functions

- unsigned char decode (unsigned char b)

Protected Attributes

- unsigned char address
- unsigned char initializationSequence [2]
- unsigned char frame [6]

5.5.1 Detailed Description

Definition at line 22 of file AccelerometerNunchuk.h.

5.5.2 Member Enumeration Documentation

5.5.2.1 enum AccelerometerNunchuk::Axis

Enumerator

AXIS_X

AXIS_Y

AXIS_Z

Definition at line 25 of file AccelerometerNunchuk.h.

5.5.3 Constructor & Destructor Documentation

5.5.3.1 AccelerometerNunchuk::AccelerometerNunchuk ( )

Public constructor.

Parameters

sa0	The LSBit of the address.

Definition at line 16 of file AccelerometerNunchuk.cpp.

5.5.4 Member Function Documentation

5.5.4.1 void AccelerometerNunchuk::begin ( )

Initializes the device.

Definition at line 63 of file AccelerometerNunchuk.cpp.

5.5.4.2 float AccelerometerNunchuk::convertToG (unsigned int *i*)

Converts to G;.

Parameters

i	The integer to be converted.

Definition at line 98 of file AccelerometerNunchuk.cpp.

5.5.4.3 unsigned char AccelerometerNunchuk::decode (unsigned char *b*) [protected]

Decodes the data from device.

Definition at line 94 of file AccelerometerNunchuk.cpp.

5.5.4.4 unsigned char* AccelerometerNunchuk::getFrame ( ) [inline]

Gets the frame pointer.

Returns

Definition at line 175 of file AccelerometerNunchuk.h.

5.5.4.5 unsigned int AccelerometerNunchuk::readAcceleration (Axis *axis,* bool *updateFrame*)

Reads the acceleration of the given axis.