1300 Henley Court
Pullman, WA 99163
509.334.6306
www.digilentinc.com

PmodCMPS™ Library Reference Manual

Revised June 9, 2015

Overview

The PmodCMPS library provides an interface to the HMC5883L 3-axis digital compass. The library initializes the compass, reads real time data, and returns a magnetic declination adjusted degree along with the associated direction. The PmodCMPS must be flat on a horizontal surface to work with this library. Further functionality would require an accelerometer to report the orientation, which is out of the scope of this library.

1 Library Operation

1.1 Library Interface

The header file CMPS.h defines all the used register addresses and initialization bytes. The file holds one class. The CMPS class is the main interface with the HMC5883L. To instantiate a CMPS object, include the CMPS library and instantiate a CMPS object.

1.2 CMPS Initialization

The CMPS module is initialized by calling the function init(). This function starts the I2c communication as well as sets the compass to the following settings:

- Sets the measurement configuration to 15 Hz output
- Sets the gain to ± 1.3 Ga
- Set the mode to continuous

After the compass is initialized, the device will immediately start outputting real time data.

2 Used Registers and Their Functions

These are the main registers used in the CMPS libraries. A more in-depth view of these register functions can be found in the HMC5883L datasheet.

Address name	Address	Function
regA	0x70	Normal measurement configuration (Default)
regB	0x20	gain(Default)
RegMode	0x00	Mode:0x00 is continuous 0x01 is single measurement
writeReg	0x1E	The 7-bit address of the PmodCMPS will be 0x1E

3 CMPS Class

3.1 Public Functions

CMPS()

- Parameters: None
- Return Value: None

Constructor for class CMPS.

init()

- Parameters: None
- Return Value: None

This function starts the I2C communication and sets the compass to the default settings.

float get_adjusted_degree(float offset)

- Parameters: float offset
  - the Magnetic declination
- Return Value: float degree
  - the degree including Magnetic declination used to find direction

This function calls getX , getZ, getY, getDegree with x and y and adjusts for the magnetic declination as well as calls reSetRegister to set the register back to the X MSB register.

char * getDirection(float degree)

- Parameters: float degree
  - the magnetic declination adjusted degree
- Return Value: char* direction
  - the direction of the compass

This function calls findDirection with degree and returns the direction.

3.2 Private Functions

int getX()

- Parameters: None
- Return Value: int x
  - The value of compass in the X direction found by using getData()

This function calls getData with the x_A and x_B registers and return the X value for compass.

int getY()

- Parameters: None
- Return Value: Int y
    - The value of compass in the Y direction found by using getData()

This function calls getData with the y_A and y_B registers and return the Y value for compass.

int getZ()

- Parameters: None
- Return Value: int z
    - The value of compass in the Z direction found by using getData()

This function calls getData with the z_A and z_B registers and return the Z value for compass.

reSetRegister()

- Parameters: None
- Return Value: None

This function call sets the register back to the x MSB register.

float getDegree(int x, int y)

- Parameters: Int x - x axis, int y - y axis
- Return Value: float degree
    - degree of where the compass is pointing

This function converts the x and y axis info into degree format.

char* findDirection()

- Parameters: float degree
    - adjusted degree
- Return Value: char* direction
    - string for the different directions

This function converts the degree value into a direction and returns value.

int twosToBin(int input)

- Parameters: input
    - a 16-bit, two's complement value to be converted to a binary number

- Return Value: return
    - Returns a 16 bit unsigned integer with the positive value of the negative twos compliment

This function converts a negative twos compliment value and performs a bitwise flip and subtracts one to return a positive int value. **This does not return a negative number**.

int getData(int reg1, int reg2)

- Parameters: reg1
    - The first register to read from. The high data value which contains the 8 MSBs
- Parameters: reg2
    - The second register to read from. The low data value which contains the 8 LSB's
- Return Value: None

This function reads data from a register couple and does the masking and shifting to create an int value. This function is used often throughout the program. Changing it might cause many issues.