PmodNAV Library Reference Manual

Revision 1: May 29, 2017

DIGILENT®

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Contents

Overview

Digilent provides a MPIDE driver library for the PmodNAV. This document provides an overview of the driver library and describes the functions that make up its chipkit programming interface. The library is called Nav.

The Nav module is a navigation tool based on the LSM9DS1 - 3D accelerometer, 3D gyroscope, 3D magnetometer and LPS25HB - MEMS pressure sensor: 260-1260 hPa absolute digital output barometer, so please refer to LSM9DS1 and LPS25HB datasheets for more details. This library intends to cover most of the functions defined in the two datasheets.

The PmodNAV is a serial device that is accessed via an SPI interface. It is, a read/write device, being able to send and receive data to/from microcontroller board. Read and write operations can be performed in a multi-byte mode.

The device allows configuration, acceleration, gyroscope, magnetic field, pressure and temperature readings, as well as interrupts related capabilities.

Library Operation

Library Interface

The header file Nav.h defines the interface for the NAV driver. The library is accessed via the methods and constants defined for the Nav object class. This header file must be included in the sketch. Also the DSPI header must be included in the sketch. DSPI is the Digilent SPI library, as explained in the SPI interface section.

```
#include <DSPI.h>
#include <Nav.h>
```

Module Initialization

The NAV module has an initialization sequence that should be performed first. Before making calls to any other library functions, the begin(…) function must be called, then Init() function. The first function initializes the PIC32 resources used by the library and the Init() function initializes the three instruments from LSM9DS1 chip as well as the digital barometer found in the LPS25HB chip. The function initializes instruments with default values of the range, sets the output data rate and enables the outputs for all instruments' axes.

To release the PIC32 resources and disable the instruments, use the end() function.

Basic functionality implemented with Init() and GetData() functions

In order to ensure an easy usage of the library, the user is given the possibility to access all the data read from the four sensors by calling only two functions.

The Init() function will initialize each instrument with the basic functionality settings:

- ➢ Accelerometer/Gyroscope instrument:
    - powered on in the ACL+GYRO mode,
    - ODR 10Hz,
    - range +-2g and +-245dps respectively,

        - Outputs enabled for all three axes.
- ➢ Magnetometer instrument:
    - powered on,
    - outputs enabled for all three axes,
    - ODR set to 10Hz,
    - +-4Gauss scale,
    - continuous conversion mode
- Altimeter is activated with 7Hz ODR value

All the instruments allow the address incrementing reading functionality, which can be enabled before starting communication with the device, or at any point in the project. Please refer to datasheet for details on how to enable this functionality.
GetData() function calls the read functions for each device and updates the structure fields specific to each instrument. It also updates the hPa and TempC global variables with the pressure reading expressed in hPa units and temperature expressed in degrees Celsius.

The member structure fields acclData.X, Y and Z, gyroData.X, Y and Z, magData.X, Y and Z are updated each time this function is called, being transmitted as parameters of the function.
They are declared in the library header file Nav.h:

```
struct ACCL_T {
              float X; // Acceleration in G's in X direction
              float Y; // Acceleration in G's in Y direction
              float Z; // Acceleration in G's in Z direction
};

struct GYRO_T {
              float X; // DPS in X direction
              float Y; // DPS in Y direction
              float Z; // DPS in Z direction
};

struct MAGNO_T {
              float X; // Gauss in X direction
              float Y; // Gauss in Y direction
              float Z; // Gauss in Z direction
struct ACCL_T acclData;
struct GYRO_T gyroData;
struct MAGNO_T magData;
float hPa;
float tempC;
```

Unexperienced users have easy access to the data provided by this function by simply accessing the members of few structures from the Nav library. Please refer to the NavdemoSimple.pde sketch for more details on how to use these functions and variables.

Compute Altitude using Reference Pressure

The pressure at any level in the atmosphere may be interpreted as the total weight of the air above a unit area at any elevation. What this implies is that atmospheric pressure decreases with increasing height.
The devices uses the pressure sensor information to calculate altitude.
The calculation of altitude is based on the following formula:

*Altitude_ft = (1-pow(*Pressure_mb/1013.25,0.190284))*145366.45*

The downside if that this formula uses the presumption that the air pressure at the sea level has a default constant value of 1013.25hPa, which is obviously not accurate in real life, under different weather situations.
The ComputePref function provides a way to inform the device of the pressure at the sea level (corrected value). Thus, the computed altitude will be compensated with this factor too.
The function that computes the Reference Pressure (ComputePref) performs a pressure reading, then computes the Reference pressure using the altitude parameter. It stores the value for the reference pressure (in Pref variable), to be later used when computing altitude.
It needs to be called once for the correct operation of the altitude function, all the following pressure readings being affected by it. If the ComputePref function is not called by the user, the Pref will have the default sea level value of 1013.25hPa.
This function is used to "calibrate" your altimeter according to the existing conditions in the current moment. For example, when you start your trip (or in any other moment) you provide to the instrument the current altitude, allowing it to compute the sea level air pressure and thus to provide correct further altitude information.

SPI interface

PmodNAV is accessed through a SPI interface. The library allows the use of any SPI hardware interface available on the board. At the Nav library level, the access to SPI is done using the Digilent DSPI library.
For each SPI interface available on the board, there is one DSPI object created and used (for example DSPI0 and DSPI1 for chipKIT™ Pro MX4 board). The correspondence between the number of DSPI object and the SPI interface on the board is explained in the board's reference manual.
When the begin() function is called you can specify which DSPI object will be used (therefore which SPI interface will be used).
In order to increase portability to the rest of the Digilent microcontroller boards, the begin function has been optimized to accept parameters for each of the instruments' chip selects, allowing the user to use the module in the wanted conditions.
When SSPinAG parameter of the begin() function is transmitted with -1 value, the SS for AG instrument will be selected as the default SS pin for prtSpi interface.
Also, when the slave select pins for MAG and ALT instruments are provided with -1 value, they are computed relative to the slave select of the AG instrument, assuming they belong to a Pmod connector.

A detailed description of the connections needed to be performed between module and master board is provided in the **NavdemoSimple.pde** example.
Each SPI transfer operation is performed by first transmitting a byte containing the register address where data will be written to/read from.
The two modules (LSM9DS1 and LPS25HB) allow multi-byte transfer in both reading and writing operations, however there are some particularities that need to be taken into account when communicating with the devices:

- For Accelerometer/Gyroscope instruments, having the same CS pin, a reading from device operation is indicated by transmitting a byte with bit 7(MSb) set to “1”, then the register address to read from or write to data. The address increment for multi byte reading/writing can be enabled by setting IF_ADD_INC bit in CTRL_REG8 register of the device.
- For Magnetometer and Altimeter instruments, besides the R/$\overline{W}$ bit of the first byte, there is the M$\overline{S}$ bit (bit 6 of the first byte transmitted in each SPI sequence) which indicates whether the address increments or not in multiple byte reading/writing.

Nav Library Functions

Basic Device Control and Initialization Functions

Nav()

Parameters:
none

This is the default constructor: initializes the global variables to default values.

void begin(uint8_t prtSpi, uint8_t SSPinAG, uint8_t SSPinMAG, uint8_t SSPinALT, int8_t DRDYPinMAG, int8_t INTPin)

Parameters:
uint8_t prtSpi - the number of the DSPI object to be used to access the desired SPI interface
uint8_t SSPinAG - SS pin for A/G instrument, specific to each master board
uint8_t SSPinMAG - SS pin for MAG instrument, specific to each master board
uint8_t SSPinALT - SS pin for ALT instrument, specific to each master board
uint8_t DRDYPinMAG - pin number for data ready signal coming from magnetometer
uint8_t INTPin - pin number for INT signal coming from either one of the three instruments

Initializes the SPI interface and defines the SS pins for each instrument, along with DRDY and INT pin.
When SSPinAG parameter is transmitted with -1 value, the SS for AG instrument will be used as the default SS pin for prtSpi interface.
When SSPinMAG is -1, then the SS for MAG instrument will have the value of the SS for AG instrument incremented by 6, according to the position in the Pmod connector.
When SSPinALT is -1, then the SS for ALT instrument will have the value of the SS for AG instrument incremented by 7, according to the position in the Pmod connector.
When DRDY is -1, then the Data ready pin for the MAG instrument will have the value of the SS for AG instrument incremented by 5, according to the position in the Pmod connector.
When INT is -1, then the INT pin coming from either one of the instruments will have the value of the SS for AG instrument incremented by 4, according to the position in the Pmod connector.

void begin()

Initializes the SPI interface and defines the SS pins for each instrument, along with DRDY and INT pin.
It uses the most common settings: DSPI0 interface and all pins situated in the Pmod connector.

void end(void)

Parameters:
none

Performs the power off sequence, floats all pins and releases the PIC32 resources used by the Nav interface.

void GetData()

Parameters:
none

This function calls the read functions of all the instruments and updates the global variables and structure members with new data. It reads the acceleration, degrees/second, magnetic field, pressure, temperature and stores them in the corresponding variables:
- acclData for all three axes of the Accelerometer instrument
- gyroData for all three axes of the Gyroscope instrument
- magData for all three axes of the Magnetometer instrument
- hPa pressure in hPa for the Barometer/Altimeter instrument
- tempC temperature in degrees Celsius read from Altimeter instrument

void Init()

Parameters:
none

This function initializes all the instruments by calling the specific Init functions for each of them. Default operating mode for Accelerometer and Gyroscope is ACL+GYRO.

void InitAG(bool fInit, uint8_t bModeSel)

Parameters:
fInit - enables the instrument for initialization or disables it
fInit = true - the intended operation is to enable the Accelerometer/Gyroscope and setup
fInit = false - the intended operation is to disable the Accelerometer/Gyroscope instrument

bModeSel - work mode for the two instruments, Accelerometer and Gyroscope: ACL only or ACL+GYRO. It can have one of the values:

MODE_INST_A	0
MODE_INST_AG	1

Initializes the accelerometer only or both the accelerometer and gyroscope instruments with the following settings:

- for ACL mode:
  - Enable all three axes in CTRL_REG5_XL register,
  - set 10 Hz ODR in CTRL_REG6_XL register
- for ACL+GYRO:
  - set 10Hz ODR in CTRL_REG1_G register and CTRL_REG6_XL, thus enabling the Gyroscope functionality as well;
  - enable the output of all three axes.

void InitMAG(bool fInit)

Parameters:

fInit - enables the instrument for initialization or disables it

fInit = true - the intended operation is to enable the Magnetometer and setup

fInit = false - the intended operation is to disable the Magnetometer instrument

Initializes the magnetometer instrument with the following settings:

- set medium performance mode
- 10H ODR, in register CTRL_REG1_M
- disable I2C and enable SPI read and write operations,
- set the operating mode to continuous in CTRL_REG3_M register.

void InitALT(bool fInit)

Parameters:

fInit - enables the instrument for initialization or disables it

fInit = true - the intended operation is to enable the Altimeter and setup

fInit = false - the intended operation is to disable the Altimeter instrument

Initializes the barometer/altimeter instrument with the following settings:

- set active mode and 7Hz ODR rate, in register CTRL_REG1,
- block data update active.

void GetDeviceID ()

Parameters:
none

The function gets the device id for all the instruments, updates the DEV_ID structure members.

```
struct DEV_ID {
        uint8_t ag; // device ID for AG instrument
        uint8_t mag; // device ID for MAG instrument
        uint8_t alt; // device ID for ALT instrument
}
```

SPI Specific Functions

void WriteRegister(int8_t ssPin, uint8_t bAddr, uint8_t bCntBytes, uint8_t *pData)

Parameters:

ssPin	- instrument Chip Select to be used: Accelerometer/Gyroscope, Magnetometer or Altimeter
bAddr	- register address to write to
bCntBytes	- number of bytes to be written
*pData	- pointer to the 16 bit data array to be written

Sends bCntBytes bytes to SPI, to be written in register having consecutive addresses, starting from bAddr.

void ReadRegister(int8_t ssPin, uint8_t bAddr, uint8_t bCntBytes, uint8_t *pData)

Parameters:

ssPin	- instrument Chip Select to be used: Accelerometer/Gyroscope, Magnetometer or Altimeter
bAddr	- register address to read from
bCntBytes	- number of bytes to be read
*pData	- pointer to the 16 bit data array to be read

Reads bCntBytes bytes from device via SPI, from register having consecutive addresses, starting from bAddr.

Accelerometer Specific Functions

void ReadAccel(int16_t &AclX, int16_t &AclY, int16_t &AclZ)

Parameters:

&AclX	- the output parameter that will receive acceleration on X axis - 16 bits value
&AclY	- the output parameter that will receive acceleration on Y axis - 16 bits value
&AclZ	- the output parameter that will receive acceleration on Z axis - 16 bits value

This function provides the 3 "raw" 16-bit values read from the accelerometer.

- It reads simultaneously the acceleration on three axes in a buffer of 6 bytes using the ReadRegister function.
- For each of the three axes, it combines the two bytes in order to get a 16-bit value.

void ReadAccelG(float &AclXg, float &AclYg, float &AclZg)

Parameters:

&AclXg	- the output parameter that will receive acceleration on X axis (in "g")
&AclYg	- the output parameter that will receive acceleration on Y axis (in "g")
&AclZg	- the output parameter that will receive acceleration on Z axis (in "g")

This function is the main function used for accelerometer values reading, providing the 3 current accelerometer values in “g”. For each of the three values, converts the 16-bit value to the value expressed in “g”, considering the currently selected g range.

void SetRangeXL(uint8_t bRangeXL)

Parameters:

bRangeXL - the parameter specifying the g range. Can be one of the parameters from the following list:

0	PAR_XL_2G	Parameter g range: +/- 2g
1	PAR_XL_4G	Parameter g range: +/- 4g
2	PAR_XL_8G	Parameter g range: +/- 8g
3	PAR_XL_16G	Parameter g range: +/- 16g

The function sets the appropriate g range bits in the CTRL_REG6_XL register of the accelerometer.

Uint8_t GetRangeXL()

Parameters:

none

Return Value:

uint8_t - returns the previously selected range from CTRL_REG6_XL register

The return value is one of:

0	PAR_XL_2G	Parameter g range: +/- 2g
1	PAR_XL_4G	Parameter g range: +/- 4g

2	PAR_XL_8G	Parameter g range: +/- 8g
3	PAR_XL_16G	Parameter g range: +/- 16g

The function reads the g range bits in the CTRL_REG6_XL register and computes the range to be provided to the user. The argument values are between 0 and 3. If the value is not a valid one, the default value for range is set.

Uint8_t DataAvailableXL()

Parameters:
none
Return Value:
uint8_t - returns the data available status in STATUS_REG register, for Accelerometer.

The function reads the STATUS_REG register and returns the data available bit in it (1 or 0).

Void ConfigIntXL(uint8_t bIntGen, bool aoi, bool latch)

Parameters:
bIntGen - The parameter indicating the interrupt generator. Can be one of the parameters from the following list

MSK_XLIE_XL	1<<0
MSK_XHIE_XL	1<<1
MSK_YLIE_XL	1<<2
MSK_YHIE_XL	1<<3
MSK_ZLIE_XL	1<<4
MSK_ZHIE_XL	1<<5
MSK_GEN_6D	1<<6

aoi - parameter indicating whether the interrupt generators are or-ed or and-ed together

aoi	0 – the interrupts are or-ed together
aoi	1 – the interrupts are and-ed together

latch - parameter that sets or not the latch interrupt request

latch	0 – interrupt requests are not latched
latch	1 – interrupts requests are latched

The function configures the interrupt register INT_GEN_CFG_XL setting the interrupt generator. Sets the interrupt events to or-ed or and-ed. Sets the interrupt event to be latched or not.

Void SetIntThresholdXL(float thValX, float thValY, float thValZ, uint8_t intDuration, bool wait)

Parameters:
thValX - Parameter containing the threshold value for X axis
thValY - Parameter containing the threshold value for Y axis
thValZ - Parameter containing the threshold value for Z axis
intDuration - parameter indicating the duration of the enter/exit interrupt
wait - parameter enabling or disabling the wait time before exiting the interrupt routine.

The function sets the interrupt threshold for each axis and also the duration of the enter/exit interrupt. Enables or disables the wait on duration before exiting interrupt.

Gyroscope Specific Functions

void ReadGyro(int16_t &GX, int16_t &GY, int16_t &GZ)

Parameters:

&GX	- the output parameter that will receive gyro value on X axis - 16 bits value
&GY	- the output parameter that will receive gyro value on Y axis - 16 bits value
&GZ	- the output parameter that will receive gyro value on Z axis - 16 bits value

This function provides the 3 "raw" 16-bit values read from the Gyroscope.

- It reads simultaneously the gyroscope value on three axes in a buffer of 6 bytes using the ReadRegister() function
- For each of the three axes, combines the two bytes in order to get a 16-bit value

void ReadGyroDps(int16_t &GXdps, int16_t &GYdps, int16_t &GZdps)

Parameters:

&GXdps	- the output parameter that will receive gyro value on X axis (in "dps")
&GYdps	- the output parameter that will receive gyro value on Y axis (in "dps")
&GZdps	- the output parameter that will receive gyro value on Z axis (in "dps")

This function is the main function used for gyro values reading, providing the 3 current gyro values in “dps”.
For each of the three values, converts the 16-bit raw value to the value expressed in “dps”, considering the currently selected dps range.

void SetRangeG(uint8_t bRangeG)

Parameters:

bRangeG - the parameter specifying the dps range. Can be one of the parameters from the following list:

0	PAR_G_245DPS	Parameter dps range: +/- 245dps
1	PAR_G_500DPS	Parameter dps range: +/- 500dps
3	PAR_G_2000DPS	Parameter dps range: +/- 2000dps

The function sets the appropriate dps range bits in the CTRL_REG1_G register.

Float GetRangeG()

Return Value:

float - returns the previously selected range from CTRL_REG1_G register

The return value is one of:

0	PAR_G_245DPS	Parameter dps range: +/- 245dps
1	PAR_G_500DPS	Parameter dps range: +/- 500dps
3	PAR_G_2000DPS	Parameter dps range: +/- 2000dps

The function reads the g range bits in the CTRL_REG1_G register and computes the range to be provided to the user. If the value is not a valid one, the default value for range is set.

Uint8_t DataAvailableG()

Return Value:

Uint8_t - returns the data available status in STATUS_REG register, for Gyroscope.

The function reads the STATUS_REG register and returns the data available bit in it (1 or 0).

ConfigIntG(uint8_t bIntGen, bool aoi, bool latch)

Parameters:

bIntGen - The parameter indicating the interrupt generator. Can be one of the parameters from the following list

MSK_XLIE_G	1<<0
MSK_XHIE_G	1<<1
MSK_YLIE_G	1<<2
MSK_YHIE_G	1<<3
MSK_ZLIE_G	1<<4
MSK_ZHIE_G	1<<5

aoi - parameter indicating whether the interrupt generators are or-ed or and-ed together

aoi	0 – the interrupts are or-ed together
aoi	1 – the interrupts are and-ed together

latch - parameter that sets or not the latch interrupt request

latch	0 – interrupt requests are not latched
latch	1 – interrupts requests are latched

The function configures the interrupt register INT_GEN_CFG_G setting the interrupt generator. Sets the interrupt events to or-ed or and-ed. Sets the interrupt event to be latched or not.

SetIntThresholdG(float thVal, uint16_t axis, bool drCntMode, uint8_t intDuration, bool wait)

Parameters:

thVal - Parameter containing the threshold value for one of the axes

axis - parameter indicating the axis for which the threshold is set. It can be one of the following values:

X_AXIS	0
Y_AXIS	1
Z_AXIS	2

drCntMode - counter mode for interrupt
intDuration - parameter indicating the duration of the enter/exit interrupt
wait - parameter enabling or disabling the wait time before exiting the interrupt routine.

The function sets the interrupt threshold for the selected or all axes, the counter mode for interrupt and also the duration of the enter/exit interrupt. Enables or disables the wait on duration before exiting interrupt.

Shared functions between Accelerometer and Gyroscope

Uint8_t GetIntSrcXLG(uint8_t bInstMode)

Parameters:

bInstMode - parameter selecting between the two instruments, Accelerometer and Gyroscope:

MODE_INST_A	0 - Accelerometer mode
MODE_INST_AG	1 - Gyroscope

Return Value:

Uint8_t - returns the content of the INT_GEN_SRC_XL or INT_GEN_SRC_G, depending of bInstMode parameter.

The function returns the source of interrupt for either Accelerometer or Gyroscope instruments.

Void ConfigInt(uint8_t bIntPin, uint8_t bIntGenMask, uint8_t bActiveType, uint8_t bOutputType)

Parameters:

bIntPin - The parameter indicating the INT pin number, INT1 or INT2. Can be one of the parameters from the following list

INT_PIN_1	0
INT_PIN_2	1

bIntGenMask - the events that trigger the interrupt. Can be one or more (OR-ed) parameters from the following list

MSK_INT1_IG_G	1<<7	
MSK_INT_IG_XL	1<<6	
MSK_INT_FSS5	1<<5	
MSK_INT_OVR	1<<4	
MSK_INT_FTH	1<<3	
MSK_INT_Boot	1<<2	
MSK_INT_DRDY_G	1<<1	
MSK_INT_DRDY_XL	1<<0	Overrun
MSK_INT2_INACT	1<<7	
MSK_INT2_FSS5	1<<5	
MSK_INT2_OVR	1<<4	
MSK_INT2_FTH	1<<3	

MSK_INT2_DRDY_TEMP	1<<2
MSK_INT2_DRDY_G	1<<1
MSK_INT2_DRDY_XL	1<<0

bActiveType – The parameter indicating the interrupt pin is active high or low. Can be one of the parameters from the following list

PAR_INT_ACTIVEHIGH	0
PAR_INT_ACTIVELOW	1

bOutputType – The parameter indicating the interrupt pin is set pushpull or opendrain. Can be one of the parameters from the following list

PAR_INT_PUSHPULL	0
PAR_INT_OPENDRAIN	1

The function sets the interrupt generator on one of the INT1 or INT2 pins. Also sets the active type of the interrupt and the output type.

ConfigIntForExternalIntUse(uint8_t bIntPin, uint8_t bParExtIntNo, uint8_t bIntGenMask,

void (*pfIntHandler)(), uint8_t bActiveType, uint8_t bOutputType)

Parameters:

bIntPin - The parameter indicating the INT pin number, INT1 or INT2. Can be one of the parameters from the following list

INT_PIN_1	0
INT_PIN_2	1

bParExtIntNo - The parameter indicating the external interrupt number. Can be one of the parameters from the following list

PAR_EXT_INT0	0
PAR_EXT_INT1	1
PAR_EXT_INT2	2
PAR_EXT_INT3	3
PAR_EXT_INT4	4

bIntGenMask - the events that trigger the interrupt. Can be one or more (OR-ed) parameters from the following list

MSK_INT1_IG_G	1<<7	
MSK_INT_IG_XL	1<<6	
MSK_INT_FSS5	1<<5	
MSK_INT_OVR	1<<4	
MSK_INT_FTH	1<<3	
MSK_INT_Boot	1<<2	
MSK_INT_DRDY_G	1<<1	
MSK_INT_DRDY_XL	1<<0	Overrun
MSK_INT2_INACT	1<<7	
MSK_INT2_FSS5	1<<5	
MSK_INT2_OVR	1<<4	
MSK_INT2_FTH	1<<3	

MSK_INT2_DRDY_TEMP	1<<2
MSK_INT2_DRDY_G	1<<1
MSK_INT2_DRDY_XL	1<<0

void (*pfIntHandler)() - pointer to a function that will serve as interrupt handler.
bActiveType – The parameter indicating the interrupt pin is active high or low. Can be one of the parameters from the following list

PAR_INT_ACTIVEHIGH	0
PAR_INT_ACTIVELOW	1

bOutputType – The parameter indicating the interrupt pin is active high or low. Can be one of the parameters from the following list

PAR_INT_PUSHPULL	0
PAR_INT_OPENDRAIN	1

The function configures the interrupt for each of the INT pins of the XL/Gyro instruments. It sets the interrupt generator, the type active high or low and the output type pushpull or open drain. It also attaches the interrupt handler to one of the microcontroller external interrupt pins.

Magnetometer Specific Functions

ReadMag(int16_t &MagX, int16_t &MagY, int16_t &MagZ)

Parameters:
&MagX - the output parameter that will receive magnetometer value on X axis - 16 bits value
&MagY - the output parameter that will receive magnetometer value on Y axis - 16 bits value
&MagZ - the output parameter that will receive magnetometer value on Z axis - 16 bits value

This function provides the 3 "raw" 16-bit values read from the magnetometer.
- It reads simultaneously the magnetic field value on three axes in a buffer of 6 bytes using the ReadRegister function
- For each of the three axes, combines the two bytes in order to get a 16-bit value

ReadMagGauss(float &MagXGauss, float &MagYGauss, float &MagZGauss)

Parameters:
&MagXGauss - the output parameter that will receive magnetic value on X axis (in "Gauss")
&MagYGauss - the output parameter that will receive magnetic value on Y axis (in "Gauss")
&MagZGauss - the output parameter that will receive magnetic value on Z axis (in "Gauss")

This function is the main function used for magnetic field values reading, providing the 3 current magnetometer values in “Gauss”.
For each of the three values, converts the 16-bit raw value to the value expressed in “Gauss”, considering the currently selected Gauss range.

void SetRangeMAG(uint8_t bRangeMAG)

Parameters:

bRangeMAG - the parameter specifying the g range. Can be one of the parameters from the following list:

0	PAR_MAG_4GAUSS	Parameter g range: +/- 4g
1	PAR_MAG_8GAUSS	Parameter g range: +/- 8g
2	PAR_MAG_12GAUSS	Parameter g range: +/- 12g
3	PAR_MAG_16GAUSS	Parameter g range: +/- 16g

The function sets the appropriate gauss range bits in the CTRL_REG2_M register.

Uint8_t GetRangeMAG()

Return value:

Uint8_t - returns the previously set range value

The return value is one of:

0	PAR_MAG_4GAUSS	Parameter g range: +/- 4g
1	PAR_MAG_8GAUSS	Parameter g range: +/- 8g
2	PAR_MAG_12GAUSS	Parameter g range: +/- 12g
3	PAR_MAG_16GAUSS	Parameter g range: +/- 16g

The function reads the gauss range bits in the CTRL_REG2_M register and computes the range to be provided to the user. If value is outside this range, the default value is set.

Uint8_t DataAvailableMAG(uint8_t axis)

Parameters:

axis - parameter indicating the axis for which the data availability is checked. It can be one of the values:

X_AXIS	0
Y_AXIS	1
Z_AXIS	2

Return Value:

uint8_t - returns the data available status in STATUS_REG_M register, for the selected axis.

The function reads the STATUS_REG_M register and returns the data available bit in it (1 or 0), for each of the axes.

void ConfigIntMAG(uint8_t bIntGen, uint8_t bActiveType, bool latch)

Parameters:

bIntGen - interrupt generator sources. It can be one of the following:

MSK_ZIEN_MAG	1<<5
MSK_YIEN_MAG	1<<6
MSK_XIEN_MAG	1<<7

bActiveType - interrupt active low or high parameter:

PAR_INT_ACTIVEHIGH	0

PAR_INT_ACTIVELOW 1

latch - parameter indicating the interrupt event is latched or not

The function configures the interrupts for magnetometer instrument. It sets the interrupt generator for the three axes. It sets the active level, low or high for the interrupt event. It enables/disables the interrupt latching.

Void SetIntThresholdM(float thVal)

Parameters:

thVal - the threshold value set to all axes.

The function sets the interrupt threshold for the Magnetometer instrument.

Uint8_t GetIntSrcMAG()

Return value:

Uint8_t - returns the interrupt sources for magnetometer instrument, after reading INT_SRC_M register.

The function returns the content of the interrupt source register for Magnetometer instrument.

POLAR_T ConvMagToPolar(float mXGauss, float mYGauss, float mZGauss)

Parameters:

mXGauss - the magnetic field value on X axis
mYGauss - the magnetic field value on Y axis
mZGauss - the magnetic field value on Z axis

Return Value:

POLAR_T - returns the POLAR_T structure members' values – this is a structure which keeps the magnetic field declination and total magnetic field value fields.

```
struct POLAR_T {
float R; // the total magnetic field computed from the three axes components;
float D; // the angle between Z axis and the resultant of X and Y
}
```

The function computes the R and D, polar coordinates of the magnetic field. Updates the POLAR_T structure members D and R with the calculated values - degrees of declination for D, to further help indicate North, in compass functioning.

Altimeter Specific Functions

Void ComputePref(float altitudeMeters)

Parameters:

altitudeMeters - the parameter used to calibrate the altitude computing, is considered known for the wanted location

This function provides the reference pressure computed from a known altitude (at the given location). If you don't know the altitude of your location, you can use: http://www.altitude-maps.com/ to get the altitude for your location. For more details about the reference pressure please see Compute Altitude from Pressure section of this document.

Int32_t ReadPressure()

Return value:
int32_t - returns the raw measured pressure value later used for converting it in hPa.

This function provides the 3 "raw" 8-bit values read from the Altimeter instrument.
- It reads the pressure value from the low, middle and high registers using the ReadRegister function
- Combines the three registers to obtain a 24-bit value for the pressure. The value of the 3 bytes is right justified within the returned 32 bit value.

float ReadPressurehPa()

Return value:
float - returns the value of measured pressure in hPa

This function provides the pressure in hPa. It is obtained from the read raw values, divided by 4096, according to the technical note that refers to this module: **TN1228: How to interpret pressure and temperature readings in the LPS25HB pressure sensor.**

Float ConvPresToAltM(float hPa)

Parameters:
hPa - parameter representing the value of pressure in hPa
Return value:
float - it returns the current altitude based on the measured pressure and the previously computed reference pressure in meters.

This function converts the provided pressure to altitude(in meters) using the previously computed Pref as reference pressure.
The function calls ConvPresToAltF in order to obtain the altitude in feet, then performs the feet-to-meters conversion.
Altitude_m = Altitude_ft *0.3048;

Float ConvPresToAltF(float hPa)

Parameters:
hPa - parameter representing the value of pressure in hPa
Return value:
float - it returns the current altitude based on the measured pressure and the previously computed reference pressure in feet.

This function converts the provided pressure to altitude (in feet) using the previously computed Pref as reference pressure.
The calculation of altitude is based on the following formula:
*Altitude_ft = (1-pow(*Pressure_mb/1013.25,0.190284))*145366.45*

Float ReadTempC()

Return value:
Float - the function returns the value of the read temperature in degrees Celsius

Reads and computes the temperature in Celsius degrees.
The formula used is taken from the same technical note as for pressure.
*T = 42.5 + TempRaw *1/480,* where 480 LSB/°C represents the sensitivity of the temperature.

float ConvTempCToTempF(float tempC)

Parameters:
tempC - parameter representing the value of temperature expressed in degrees Celsius
Return value:
float - returns the value of the temperature in degrees Fahrenheit.

This function performs the conversion from Celsius to Fahrenheit degrees and returns the value of temperature in F.

Uint8_t DataAvailableALT()

Return value:
Uint8_t - returns the data available status in STATUS_REG register, for altimeter

The function reads the STATUS_REG register and returns the data available bit in it (1 or 0).

Uint8_t TempAvailableALT()

Return value:
Uint8_t - returns the temperature available status in STATUS_REG register, for altimeter

The function reads the STATUS_REG register and returns the temperature available bit in it (1 or 0).

void ConfigIntALT(uint8_t bIntGen, uint8_t bActiveType, uint8_t bOutputType, uint8_t dataSignalVal, bool intEnable, bool latch, uint8_t intLevel)

Parameters:
bIntGen - interrupt generator sources. It can be one of the following:

MSK_INT_F_EMPTY	1<<3

MSK_INT_F_FTH	1<<2
MSK_INT_F_OVR	1<<1
MSK_INT_DRDY	1<<0

bActiveType - interrupt active low or high parameter:

PAR_INT_ACTIVEHIGH	0
PAR_INT_ACTIVELOW	1

dataSignalVal - INT_S bits value, representing the interrupt configurations, data signal, pressure high, low, or high and low it can be one of the following:

MSK_INT_P_HIGH	0x01
MSK_INT_P_LOW	0x02
MSK_INT_P_LOW_HIGH	0x03

intEnable - enable interrupts parameter

bOutputType - output type parameter, one of the following:

PAR_INT_OPENDRAIN	1
PAR_INT_PUSHPULL	0

latch - parameter indicating the interrupt event is latched or not

intLevel - set the level active interrupt for differential pressure value, on high or low

MSK_INT_LEVEL_HIGH	1<<0
MSK_INT_LEVEL_LOW	1<<1

The function configures the interrupts for Altimeter instrument. It sets the interrupt generator for the three axes. It sets the active level, low or high for the interrupt event, interrupt configurations, output type. It enables/disables the interrupt latching.

void SetIntThresholdALT(float thVal)

Parameters:

thVal - the interrupt threshold parameter for alt instrument

The function sets the interrupt threshold for the Altimeter instrument.

Uint8_t GetIntSrcALT()

Return value:

Uint8_t - parameter storing the value of INT_SOURCE register

The function gets the interrupt sources by reading the INT_SOURCE register.

Common Functions

void SetODR(uint8_t bInstMode, uint8_t odrVal)

Parameters:

bInstMode - parameter representing the instrument to be affected

odrVal - the parameter specifying the ODR value for each instrument. Can be one of the parameters from the following list:

----for Accelerometer

0	ODR_XL_PWR_DWN	Parameter ODR value: power down the device
1	ODR_XL_10_HZ	Parameter ODR value: 10 Hz
2	ODR_XL_50_HZ	Parameter ODR value: 50 Hz
3	ODR_XL_119_HZ	Parameter ODR value: 119 Hz
4	ODR_XL_238_HZ	Parameter ODR value: 238 Hz
5	ODR_XL_476_HZ	Parameter ODR value: 476 Hz
6	ODR_XL_952_HZ	Parameter ODR value: 952 Hz
7	ODR_XL_NA	Parameter ODR value: not defined

----for Gyroscope

0	ODR_G_PWR_DWN	Parameter ODR value: power down the device
1	ODR_G_14_9HZ	Parameter ODR value: 14.9 Hz
2	ODR_G_59_5HZ	Parameter ODR value: 59.5 Hz
3	ODR_G_119HZ	Parameter ODR value: 119 Hz
4	ODR_G_238HZ	Parameter ODR value: 238 Hz
5	ODR_G_476HZ	Parameter ODR value: 476 Hz
6	ODR_G_952HZ	Parameter ODR value: 952 Hz
7	ODR_G_NA	Parameter ODR value: not defined

----for Magnetometer

0	ODR_M_0_625HZ	Parameter ODR value: 0.625 HZ
1	ODR_M_1_25HZ	Parameter ODR value: 1.25 Hz
2	ODR_M_2_5HZ	Parameter ODR value: 2.5 Hz
3	ODR_M_5HZ	Parameter ODR value: 5 Hz
4	ODR_M_10HZ	Parameter ODR value: 10 Hz
5	ODR_M_20HZ	Parameter ODR value: 20 Hz
6	ODR_M_40HZ	Parameter ODR value: 40 Hz
7	ODR_M_80HZ	Parameter ODR value: 80 HZ

The function sets the ODR value for the selected instrument, in the corresponding register. The argument values are between 0 and 7.

Uint8_t GetODRRaw(uint8_t bInstMode)

Parameters:

bInstMode	- parameter representing the instrument to be affected

Return value:

Uint8_t	- returns the ODR raw value for the selected instrument, expressed in hexa

It can be one of the values:

---- for Accelerometer

0	ODR_XL_PWR_DWN	Parameter ODR value: power down the device
1	ODR_XL_10_HZ	Parameter ODR value: 10 Hz
2	ODR_XL_50_HZ	Parameter ODR value: 50 Hz
3	ODR_XL_119_HZ	Parameter ODR value: 119 Hz
4	ODR_XL_238_HZ	Parameter ODR value: 238 Hz
5	ODR_XL_476_HZ	Parameter ODR value: 476 Hz

6	ODR_XL_952_HZ	Parameter ODR value: 952 Hz
7	ODR_XL_NA	Parameter ODR value: not defined
----for Gyroscope		
0	ODR_G_PWR_DWN	Parameter ODR value: power down the device
1	ODR_G_14_9HZ	Parameter ODR value: 14.9 Hz
2	ODR_G_59_5HZ	Parameter ODR value: 59.5 Hz
3	ODR_G_119HZ	Parameter ODR value: 119 Hz
4	ODR_G_238HZ	Parameter ODR value: 238 Hz
5	ODR_G_476HZ	Parameter ODR value: 476 Hz
6	ODR_G_952HZ	Parameter ODR value: 952 Hz
7	ODR_G_NA	Parameter ODR value: not defined
----for Magnetometer		
0	ODR_M_0_625HZ	Parameter ODR value: 0.625 HZ
1	ODR_M_1_25HZ	Parameter ODR value: 1.25 Hz
2	ODR_M_2_5HZ	Parameter ODR value: 2.5 Hz
3	ODR_M_5HZ	Parameter ODR value: 5 Hz
4	ODR_M_10HZ	Parameter ODR value: 10 Hz
5	ODR_M_20HZ	Parameter ODR value: 20 Hz
6	ODR_M_40HZ	Parameter ODR value: 40 Hz
7	ODR_M_80HZ	Parameter ODR value: 80 HZ

The function sets the ODR raw value for the selected instrument.

float GetODR(uint8_t bInstMode)

Parameters:

bInstMode - parameter representing the instrument to be affected

Return value:

float - returns the ODR value of the selected instrument, as a numeric value corresponding to the bits reading from the corresponding register.

The function returns the ODR value of the selected instrument, as a numeric value corresponding to the bits from each instrument ODR register.

FIFO Specific Functions

Void FIFOEnable(int8_t ssPin, bool fEnable)

Parameters:

ssPin - parameter for selecting one of the instruments: A/G or ALT

fEnable – the parameter used to enable or disable the FIFO

The function enables or disables FIFO by writing FIFO_EN bit in CTRL_REG9 register for Accelerometer/Gyroscope, or CTRL_REG2 for Altimeter instrument.

void SetFIFO(int8_t ssPin, uint8_t parFIFOMode, uint8_t FIFOThreshold))

Parameters:

ssPin - parameter for selecting one of the instruments

parFIFOMode - the parameter specifying the FIFO mode for each instrument. Can be one of the parameters from the following list:

----for Accelerometer and Gyroscope instruments, when working together

0	FIFO_MODE_XL_G_BYPASS	Parameter FIFO mode value: bypass
1	FIFO_MODE_XL_G_FIFO	Parameter FIFO mode value: FIFO normal
3	FIFO_MODE_XL_G_CONTINUOUS_FIFO	Parameter FIFO mode value: continuous to FIFO
4	FIFO_MODE_XL_G_BYPASS_CONTINUOUS	Parameter FIFO mode value: bypass to continuous
6	FIFO_MODE_XL_G_CONTINUOUS	Parameter FIFO mode value: continuous

----for Altimeter instrument

0	FIFO_MODE_ALT_BYPASS	Parameter FIFO mode value: bypass
1	FIFO_MODE_ALT_FIFO	Parameter FIFO mode value: FIFO normal
2	FIFO_MODE_ALT_STREAM	Parameter FIFO mode value: stream mode
3	FIFO_MODE_ALT_STREAM_TO_FIFO	Parameter FIFO mode value: stream to fifo
4	FIFO_MODE_ALT_BYPASS_TO_STREAM	Parameter FIFO mode value: bypass to stream
6	FIFO_MODE_ALT_MEAN	Parameter FIFO mode value: mean mode
7	FIFO_MODE_ALT_BYPASS_TO_FIFO	Parameter FIFO mode value: bypass to fifo mode

FIFOThreshold - FIFO threshold level setting.

Can be one of the parameters from the following list:

----for Accelerometer and Gyroscope instruments, when working together. Any value from 0-0x1F is acceptable

0	Parameter FIFO mode mean value: 0 samples
.	
.	
.	
32	Parameter FIFO mode mean value: 32 samples

----for Altimeter instrument

0	FIFO_MODE_MEAN_ALT_2SAMPLES	Parameter FIFO mean mode value: 2 samples
1	FIFO_MODE_MEAN_ALT_4SAMPLES	Parameter FIFO mean mode value: 4 samples
2	FIFO_MODE_MEAN_ALT_8SAMPLES	Parameter FIFO mean mode value: 8 samples
3	FIFO_MODE_MEAN_ALT_16SAMPLES	Parameter FIFO mean mode value: 16 samples
4	FIFO_MODE_MEAN_ALT_32SAMPLES	Parameter FIFO mean mode value: 32 samples

The function sets the FIFO control mode and threshold in FIFO_CTRL register for the Accel/Gyro instruments and for altimeter instrument**.** The magnetometer instrument does not contain FIFO.

Uint8_t GetFIFOMode(int8_t ssPin)

Parameters:

ssPin - instrument to select, using SS pin

Return Value:

Uint8_t - returns the FIFO mode bits from FIFO_CTRL register

Can be one of the parameters from the following list:

----for Accelerometer and Gyroscope instruments, when working together

0	FIFO_MODE_XL_G_BYPASS	Parameter FIFO mode value: bypass
1	FIFO_MODE_XL_G_FIFO	Parameter FIFO mode value: FIFO normal
3	FIFO_MODE_XL_G_CONTINUOUS_FIFO	Parameter FIFO mode value: continuous to FIFO
4	FIFO_MODE_XL_G_BYPASS_CONTINUOUS	Parameter FIFO mode value: bypass to continuous
6	FIFO_MODE_XL_G_CONTINUOUS	Parameter FIFO mode value: continuous

----for Altimeter instrument

0	FIFO_MODE_ALT_BYPASS	Parameter FIFO mode value: bypass
1	FIFO_MODE_ALT_FIFO	Parameter FIFO mode value: FIFO normal
2	FIFO_MODE_ALT_STREAM	Parameter FIFO mode value: stream mode
3	FIFO_MODE_ALT_STREAM_TO_FIFO	Parameter FIFO mode value: stream to fifo
4	FIFO_MODE_ALT_BYPASS_TO_STREAM	Parameter FIFO mode value: bypass to stream
6	FIFO_MODE_ALT_MEAN	Parameter FIFO mode value: mean mode
7	FIFO_MODE_ALT_BYPASS_TO_FIFO	Parameter FIFO mode value: bypass to fifo mode

The function reads the FIFO mode bits in the FIFO_CTRL register for Accelerometer/Gyroscope and Altimeter instruments and returns their value.

Uint8_t GetFIFOThs(int8_t ssPin)

Parameters:

ssPin - parameter for selecting one of the instruments using corresponding SS pin

Return Value:

Uint8_t – specifies the FIFO mean number of samples for each instrument. Can be one of the parameters from the following list:

----for Accelerometer and Gyro instruments, when working together

0	Parameter FIFO mode mean value: 0 samples
.	
.	
.	
32	Parameter FIFO mode mean value: 32 samples

----for Altimeter instrument

0	FIFO_MODE_MEAN_ALT_2SAMPLES	Parameter FIFO mean mode value: 2 samples
1	FIFO_MODE_MEAN_ALT_4SAMPLES	Parameter FIFO mean mode value: 4 samples
2	FIFO_MODE_MEAN_ALT_8SAMPLES	Parameter FIFO mean mode value: 8 samples

3	FIFO_MODE_MEAN_ALT_16SAMPLES	Parameter FIFO mean mode value: 16 samples
4	FIFO_MODE_MEAN_ALT_32SAMPLES	Parameter FIFO mean mode value: 32 samples

The function sets the FIFO mean mode bits in FIFO_CTRL register for the Accel/Gyro instruments and for altimeter instrument. The magnetometer instrument does not contain FIFO.

Uint8_t GetFIFOStatus(int8_t ssPin)

Parameters:
ssPin - parameter for selecting one of the instruments using corresponding SS pin
Return Value:
uint8_t - returns the FIFO status FIFO_STATUS or FIFO_SRC register, depending on the selected instrument.

The function reads the FIFO_SRC register for Accelerometer/Gyroscope and FIFO_STATUS register for Altimeter instrument and returns their value.