3-Axis Magnetometer - LIS3MDL - Trēo™ Module

Module Features

- STMicro LIS3MDL
- RoHS Compliant
- Software Library
- NightShade Trēo™ Compatible
- Breakout Headers

LIS3MDL Features

(from STMicro)

- ±4/±8/±12/±16 gauss selectable magnetic full scales
- 16-bit data output
- Continuous and single-conversion modes
- Interrupt generator
- Self-test
- Power-down & low-power modes

Applications

- Compasses
- Magnetometers

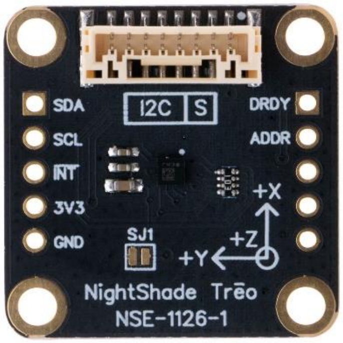
Trēo™ Compatibility

Electrical

Communication	I2C
Max Current, 3.3V	1mA
Max Current, 5V	0mA

Mechanical

- 25mm x 25mm Outline
- 20mm x 20mm Hole Pattern
- M2.5 Mounting Holes



Description

The LIS3MDL Trēo™ Module is a 3-Axis Magnetometer module that that features STMicro's LIS3MDL 3-Axis Magnetometer. Its full-scale measurement range can be set to ±4, ±8, ±12, or ±16 gauss with samples rates up to 1000Hz. This module is a part of the NightShade Treo system, patent pending.

Table of Contents

1 Summary

The LIS3MDL is a 3-axis magnetometer. It is first initialized with the begin() method. Then data can be measured with the acquireMagData() method and retrieved with the axis specific methods. (e.g. readX(), readY(), readZ(), readTemp(), etc.) The measurement parameters can be varied using the other methods available in this library.

2 What is Trēo™?

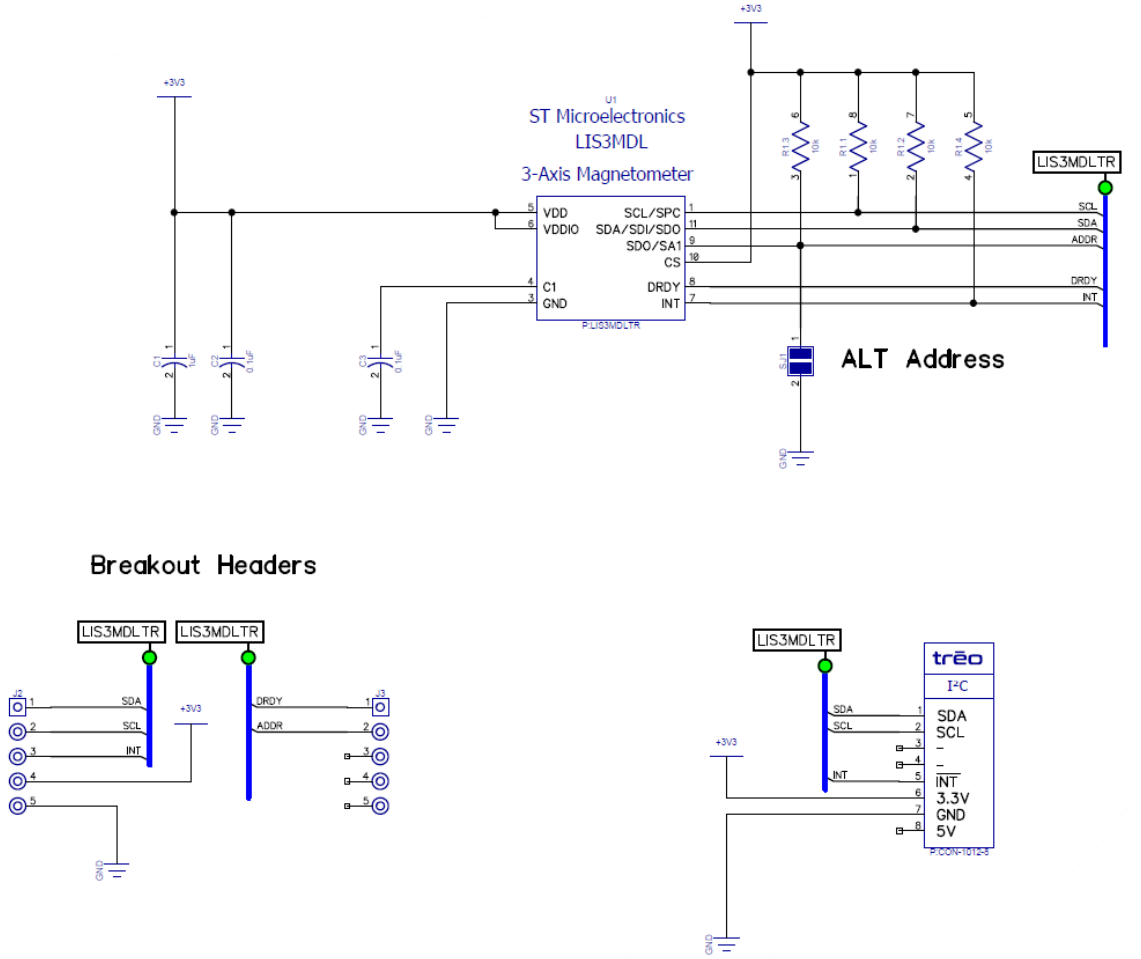
NightShade Trēo is a system of electronic modules that have standardized mechanical, electrical, and software interfaces. It provides you with a way to quickly develop electronic systems around microprocessor development boards. The grid attachment system, common connector/cabling, and extensive cross-platform software library allow you more time to focus on your application. Trēo is supported with detailed documentation and CAD models for each device.

Learn more about Trēo here.

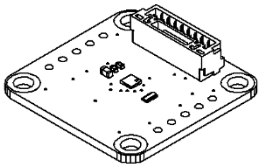
3 Electrical Characteristics

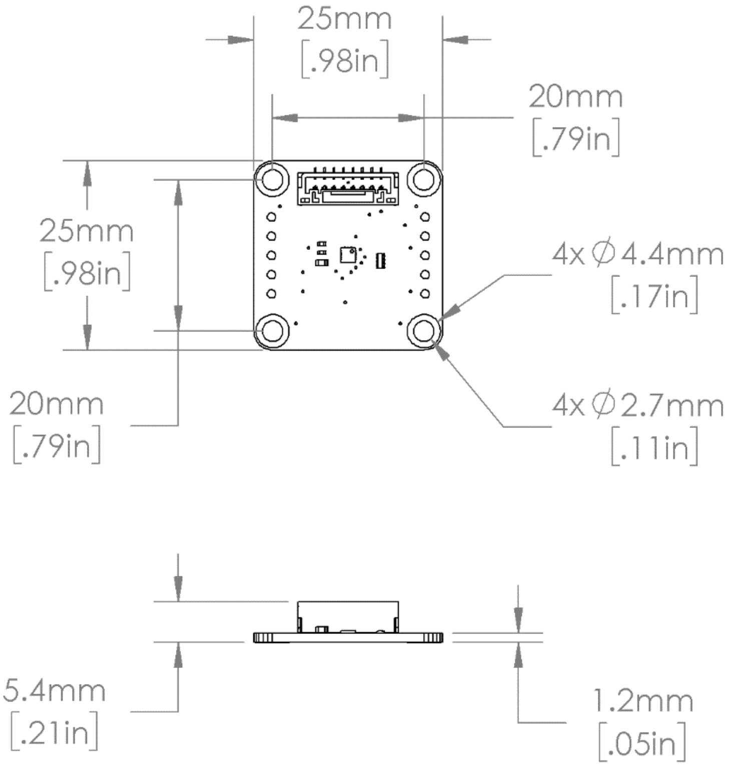
	Minimum	**Nominal**	**Maximum**
Voltages			
$V_{i/o}$ (SDA, SCL, INT)	-0.3V	-	3.6V
$V_{3.3V}$	3.1V	3.3V	3.5V
Measurement			
Bandwidth	Single sample	-	1000Hz
Range	-16 gauss	-	+16 gauss
Precision	584 µgauss/LSB	-	146 µgauss/LSB
Error	-	-	0.12%FS + 4.1mG
I2C Slave Address			
SJ1 Open (Default)		0x1E	
SJ1 Closed (Soldered)		0x1C	
Operating Temperature	-25°C	-	+85°C

4 Electrical Schematic



5 Mechanical Outline





6 Example Arduino Program

```
/**********************************************************
  LIS3MDL_Magnetometer - NightShade_Treo by NightShade Electronics

  This sketch demonstrates the functionality of the
  NightShade Trēo LIS3MDL magnetometer module. (NSE-1126-1)
  It prints the magnetometer data, a calculated heading,
  and temperature from the sensor and prints in out as
  Serial at 115200 baudrate.

  Created by Aaron D. Liebold
  on February 15, 2021

  Links:
  NightShade Trēo System: https://nightshade.net/treo
  Product Page: https://nightshade.net/product/treo-3-axis-magnetometer-lis3mdl/

  Distributed under the MIT license
  Copyright (C) 2021 NightShade Electronics
  https://opensource.org/licenses/MIT
**********************************************************/

// Include NightShade Treo Library
#include <NightShade_Treo.h>
#include <math.h>

// Declare Objects
NightShade_Treo_LIS3MDL sensor(1);

void setup() {
  sensor.begin();
  Serial.begin(115200);
}

void loop() {
  sensor.acquireMagData();
  float X = (float) sensor.readX() / sensor.oneGaussValue(); // X value in Gauss
  float Y = (float) sensor.readY() / sensor.oneGaussValue(); // Y value in Gauss
  float Z = (float) sensor.readZ() / sensor.oneGaussValue(); // Z value in Gauss
  float temp = (float) sensor.readTemp() / 10; // Retrieve temperature in deg C

  // Calculate Azimuth at X+ (2-axis)
  float azimuth =  atan2(Y, X) * 180 / M_PI;
  if (azimuth < 0) azimuth += 360.0; // Keep azimuth positive (0 - 360deg)

  Serial.print("Az: ");
  Serial.print(azimuth);
  Serial.print("deg \t");
  Serial.print(X, 2);
  Serial.print("G\t");
  Serial.print(Y, 2);
  Serial.print("G\t");
  Serial.print(Z, 2);
  Serial.print("G\t");
  Serial.print(temp, 2);
  Serial.print("C\t\n");

  delay(500);
}
```

7 Library Overview (C++ & Python)

C++ Class

NightShade_Treo_LIS3MDL <classObject>();

Python Module

<classObject> = NightShade_Treo.LIS3MDL()

7.1 Constructors

NightShade_Treo_LIS3MDL(int port, uint8_t slaveAddress, uint32_t clockSpeed)

Creates a LIS3MDL object.

Arguments:

port	Integer of the I2C port used (e.g. 0 = "/dev/i2c_0")
slaveAddress	7-bit slave address
clockSpeed	Desired clock speed for the bus

Returns:

Nothing

NightShade_Treo_LIS3MDL(int port)

Creates a LIS3MDL object assuming the default slave address and clock speed.

Arguments:

port	Integer of the I2C port used. (e.g. 0 = "/dev/i2c_0")

Returns:

Nothing

7.2 Methods

begin()

Initializes the LIS3MDL. (80Hz, Ultra-high-performance mode, ±4G range, single-conversion, temperature measurement enabled)

Arguments:

None

Returns:

Error	0 = Success

setOutputDataRate(int setting)

Set the output data rate (ODR).

Arguments:

setting	0: 0.625Hz
	1: 1.25Hz
	2: 2.5Hz
	3: 5Hz
	4: 10Hz
	5: 20Hz
	6: 40Hz
	7: 80Hz
	8: Max ODR (limited by operating mode)

Returns:

Error	0 = Success

setOperatingMode(int xyMode, int zMode)

Sets the operating mode for the X/Y axes and the Z axis.

Arguments:

xyMode	0: Low-power mode	(Max ODR = 1000Hz)
	1: Medium-power mode	(Max ODR = 560Hz)
	2: High-power mode	(Max ODR = 300Hz)
	4: Ultra-high-power mode	(Max ODR = 165Hz)
zMode	0: Low-power mode	(Max ODR = 1000Hz)
	1: Medium-power mode	(Max ODR = 560Hz)
	2: High-power mode	(Max ODR = 300Hz)
	4: Ultra-high-power mode	(Max ODR = 165Hz)

Returns:

Error	0 = Success

setMeasurementMode(int setting)

Sets the operational mode of the LIS3MDL.

Arguments:

setting	0: Continuous-conversion mode
	1: Single-conversion mode (**Must be used with ODR 0.625-80Hz**)
	3: Power-down mode
	4: Power-down mode (duplicated mode)

Returns:

Error	0 = Success

enableTemperature(int enable)

Enables temperature measurement.

Arguments:

enable	true/false

Returns:

Error	0 = Success

setFullScaleRange(int setting)

Sets the full-scale range (FSR) of the LIS3MDL.

Arguments:

setting	0: ±4 gauss 1: ±8 gauss 2: ±12 gauss 3: ±16 gauss

Returns:

Error	0 = Success

enableInterrupt(int enableIntX, int enableIntY, int enableIntZ)

Enables an axis to generate an interrupt. These interrupts will set the interrupts flags and it will set the external interrupt pin.

Arguments:

enableIntX	true/false
enableIntY	true/false
enableIntZ	true/false

Returns:

Error	0 = Success

setInterruptThreshold(int threshold)

Sets the axis threshold to generate an interrupt.

Arguments:

threshold	Threshold value (0 – 32767)

Returns:

Error	0 = Success

readInterruptFlags()

Reads the state of the interrupt flag register. Reading this register clears the flags.

Arguments:
None

Returns:

INT_SRC (uint8_t)	B7: X-axis exceeds positive threshold
	B6: Y-axis exceeds positive threshold
	B5: Z-axis exceeds positive threshold
	B4: X-axis exceeds negative threshold
	B3: Y-axis exceeds negative threshold
	B2: Z-axis exceeds negative threshold
	B1: Internal measurement range overflowed on magnetic value
	B0: An interrupt has occurred (logical OR of INT flags)

acquireMagData()

Reads data from sensor and stores it in a local software buffer.

Arguments:
None

Returns:

Error	0 = Success

readX()

Returns the X-axis value from the local software buffer.

Arguments:
None

Returns:
X-axis value

readY()

Returns the Y-axis value from the local software buffer.

Arguments:
None

Returns:
Y-axis value

readZ()

Returns the Z-axis value from the local software buffer.

Arguments:
None

Returns:
Z-axis value

readTemp()

Returns the temperature data from the local software buffer. Temperature measurement must be enabled for temperature data to be collected.

Arguments:
None

Returns:

Temperature value	(0.125°C/LSB)

deviceId()

Returns the device ID code.

Arguments:
None

Returns:

Device ID	(uint8_t)

enableSelfTest(int enable)

Enables the LIS3MDL self-test mode.

Arguments:

enable	true/false

Returns:

Error	0 = Success

oneGaussValue()

Returns the LSB/gauss value based on the current FSR setting.

Arguments:
None

Returns:

Value of 1 gauss	(LSB/gauss)

dataReady()

Indicates if new data is ready.

Arguments:

None

Returns:

Data-ready flag	true/false

rebootMemory()

Restarts the LIS3MDL’s memory engine.

Arguments:

None

Returns:

Error	0 = Success

restart()

Restarts the LIS3MDL device.

Arguments:

None

Returns:

Error	0 = Success