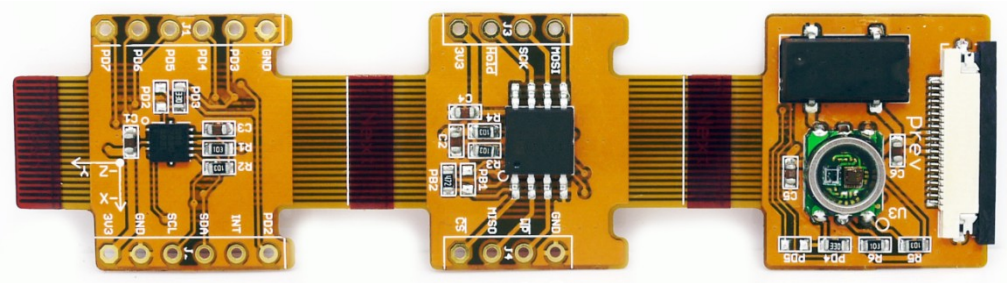
SEEEDUINO FILM MOTION FRAME



User Manual v0.9c

Altitude | Pressure | Temperature | Motion Sensing | Data Logging

Nov 30, 2010

seeed studio
Boost ideas, Extend your reach

WWW.SEEEDSTUDIO.COM

LIFE SUPPORT DEVICE POLICY

Seeeduino Film Motion Frame and other products of **Seeed Studio Inc** are not designed, intended or authorized for use in life support devices or systems. Life support devices or systems include, but are not limited to, surgical implants, medical systems, and other safety-critical systems in which failure of Seeed Studio Inc products could cause personal injury or loss of life. Should buyer purchase and use Seeed Studio Inc products in such an unauthorized and unintended manner, Buyer agrees to indemnify and hold harmless Seeed Studio Inc, its officers, employees, suppliers, affiliates, and distributors from any and all claims arising from such use, even if such claim alleges that **Seeed Studio Inc** was negligent in the design or manufacture of its product.

Nov 30, 2010

seeed studio
Boost ideas, Extend your reach

WWW.SEEEDSTUDIO.COM

OVERVIEW

Seeeduino Film Motion Frame is a motion sensing extension board designed for ***Seeeduino Film***. Its flexible, ultra-slim and small form factor is suitable for building wearable devices. It consists of a Barometer (**HP03M**), a 3-axis Accelerometer Sensor (**MMA7660FC)** and a 32 Mb serial Flash (**W25X32**). This provide sensing and logging of altitude, temperature, air pressure and motion for Seeeduino Film. FILM and other extension FRAMEs could be simply connected via the 20 pin universal bus like a chain. Its native 2.54mm pitch pins enable quick prototyping as well.

FEATURES

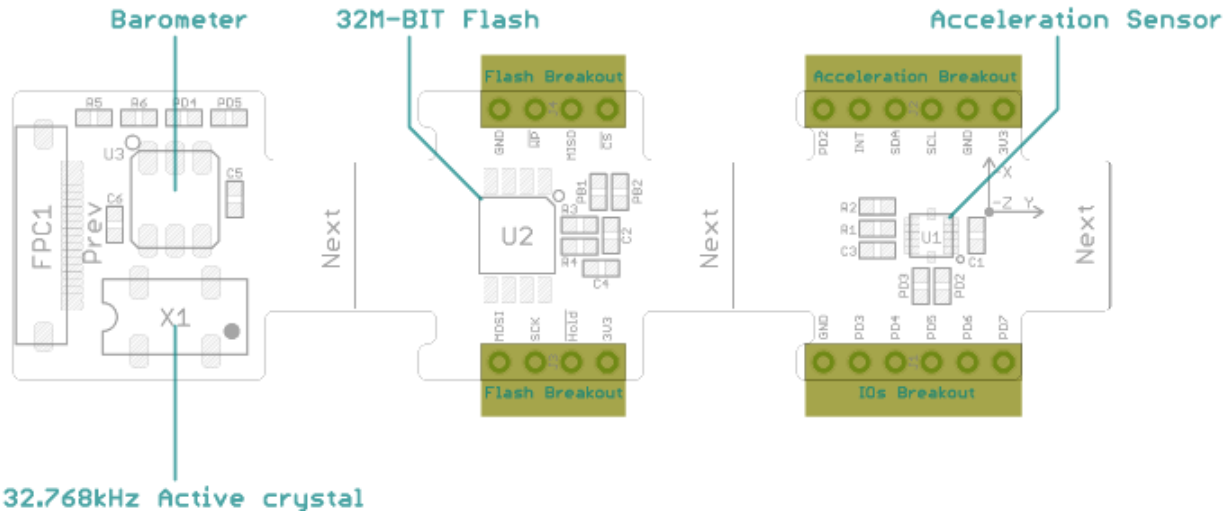
- Flexible, Ultra small / Slim form factor
- Seeeduino Film compatible
- I²C **Barometer**
- I²C **3-Axis Accelerometer**
- SPI **32M-BIT Serial Flash**
- 0.1” pitch pad breakout
- 20 pin daisy-chain flex bus
- Transform by cutting and chaining
- Reinforced to increase endurance

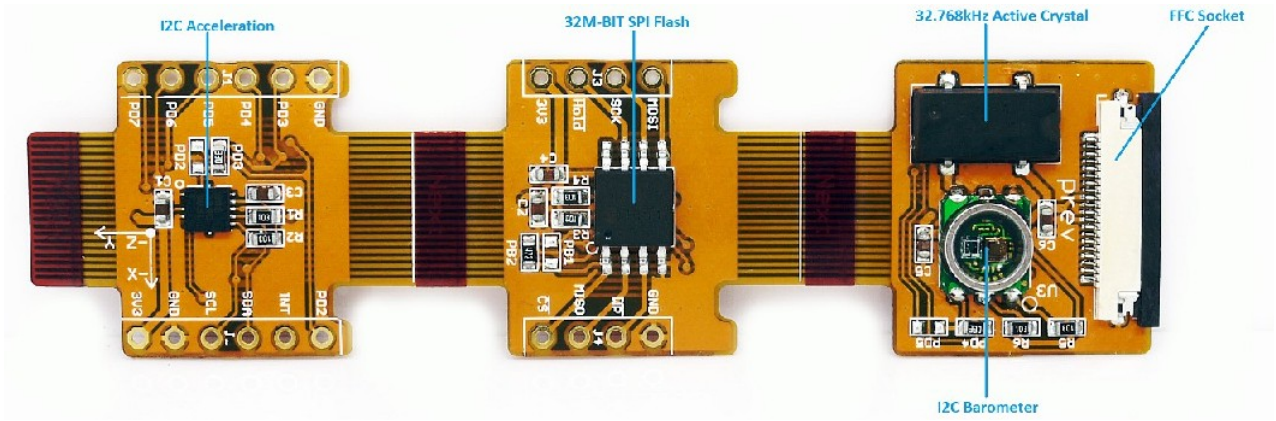
APPLICATION IDEAS

- Weather Watching
- Environment Sensing
- Sports and Gyms
- Hiking / Climbing
- Hobby Aviation
- Security
- Data logging

1.Block Diagram

The following block diagram presents the arrangement of various components of Motion Frame. Each module is connected by a 20 pin flex bus. Motion Frame is attached to Seeeduino Film with the help of a FPC socket.





Caution

1. The direction of accelerometer is completely reversed in the board.
2. The FFC socket and golden finger are easily lost or broken; pay more attention while handling, especially do not bend or move too many times.

2.BAROMETER

The **Hope RF** – **HP03M** is used for pressure, altitude and temperature sensing. It consist of a piezo-resistive pressure sensor connected to an ADC. It uses an I²C bus to communicate with the Seeeduino Film(ATMega168). HP03M internally integrates an I²C ADC and an I²C EEPROM. The EEPROM maintains 11 coefficient data stored during manufacturing. These coefficient data should be used for calibration / compensation of the measured values to find the real pressure and real temperature. The I²C ADC maintains the measured atmospheric pressure and temperature. The I²C EEPROM operation is compatible with 24C02.

HP03M's SCL, SDA, XCLR are connected to Seeeduino Film's PC5, PC4 and PD4 & PD5 as follows:

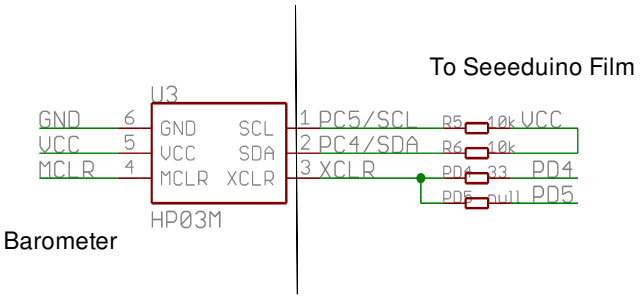


Fig: Barometer – Seeeduino Film Port Connection

XCLR:

- XCLR is connected to PD4 and PD5 of Seeeduino Film (PD4 default).
- XCLR is used to initialize the ADC / EEPROM operation.
- Coefficient data should be read only after pulling XCLR to LOW.
- Similarly XCLR should be pulled HIGH before start of an AD conversion cycle.

Chip Address:

- ADC and EEPROM share a common I²C bus.
- I²C ADC chip address is set to 0xEE.
- I²C EEPROM chip address is set to 0xEF.

Application Programming Interface (API):

HP03M library is provided in a Seeeduino (Arduino) compatible format. Copy the contents of the library folder to Arduino library folder to get started.

- The HP03M library provides two high level interfaces **`init()`** and **`read()`**.
- There exist a pre-instantiated object **`Hp03m`** which is used to access the above two interfaces.
- The result of **`read()`** is available in **`Temperature`**, **`Pressure`** and **`Altitude`** attributes(variables) provided by HP03M library. The raw uncompensated pressure and temperature values are available in **D1** and **D2** variables.

`init():`

The init() interface initializes the I²C communication.

- Configures Seeeduino PD4 as output
- XCLR (i.e PD4) is pulled LOW to read calibration data.
- Reads calibration data.

`read():`

- XCLR (i.e PD4) is pulled HIGH to read ADC data.
- The raw temperature and pressure are read from ADC.
- XCLR (i.e PD4) is reset to LOW (to reset the sensor)
- The real temperature and pressure are calculated by compensating with the coefficient data read during **`init()`** phase.
- Altitude is calculated from the measured pressure.
- The results are stored in **`Temperature, Pressure and Altitude`** variables.

The following two functions are used by the above ***read()*** function internally:

`realTemperaturePressureCalculate():`

- Calculate real temperature and pressure from raw value available in Hp03m.D1 & Hp03m.D2

`altitudeCalculate():`

- Calculate altitude from real atmospheric atmospheric pressure

`Temperature:`

- The real temperature in °C is available as *float*.

`Pressure`

- The real air pressure in hP is available as *float.*

Altitude

- The calculated altitude in 0.1m is available as *long integer*.

The following simple example demonstrates the use HP03M library

```
#include <Wire.h>
#include <HP03M.h>

void setup()
{
    Serial.begin(38400);
    Hp03m.init();
}

void loop()
{
    Hp03m.read(); // Read temperature, pressure and altitude
    Serial.print("Temperature=");
    Serial.println(Hp03m.Temperature);

    Serial.print("Pressure=");
    Serial.println(Hp03m.Pressure);

    Serial.print("Altitude=");
    Serial.println(Hp03m.Altitude);
    delay(1000);
}
```

Algorithms and Timing diagrams:

The HP03M library provides a very simple interface to users. All the low level functions are carried out by the library. The following section describes how Seeeduino interacts with HP03M. The overall operation can be summarized as below : -

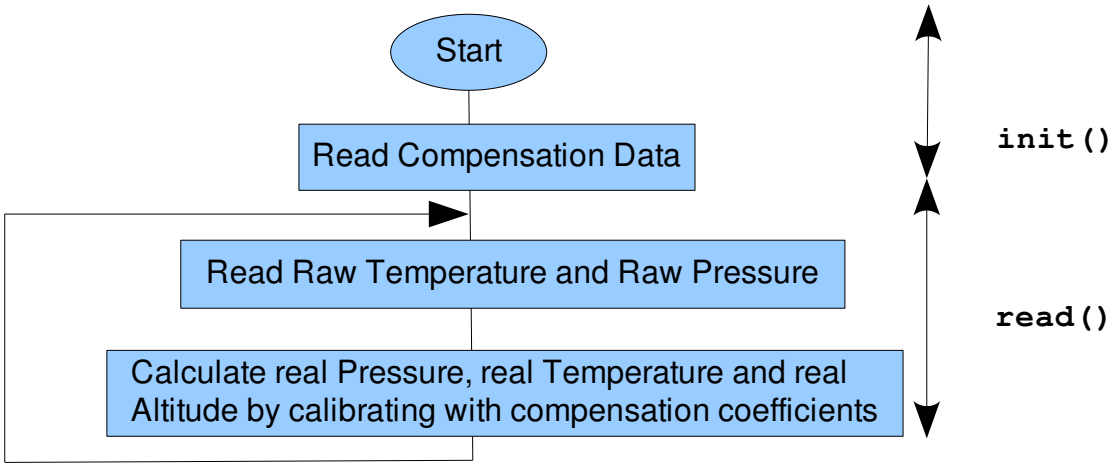


Fig: Barometer Read Process

Algorithms:

Read Compensation Data:

The operation of HP03M EEPROM is compatible to 24C02. The following operation are carried out to read the calibration data.

1. Start I²C transmission.
2. Send Slave Device Address 0x10.
3. Read 18 bytes of data.
4. Fill the compensation coefficient buffer(C1, C2, C3, C4, C5, C5, C7, AA, BB, CC and DD) with the read data.
5. Stop I²C transmission.

Read Raw Pressure:

1. Pull XCLR HIGH.
2. Start I²C transmission.
3. Send command 0xFF.
4. Send command 0xF0.
5. Read 2 bytes of data.
6. Fill the raw Pressure buffer(D1) with read data.
7. Stop I²C transmission.
8. Pull XCLR LOW.

Read Raw Temperature:

1. Pull XCLR HIGH.
2. Start I²C transmission.
3. Send command 0xFF.
4. Send command 0xE8.
5. Read 2 bytes of data.
6. Fill the raw Temperature buffer(D2) with read data.
7. Stop I²C transmission.
8. Pull XCLR LOW.

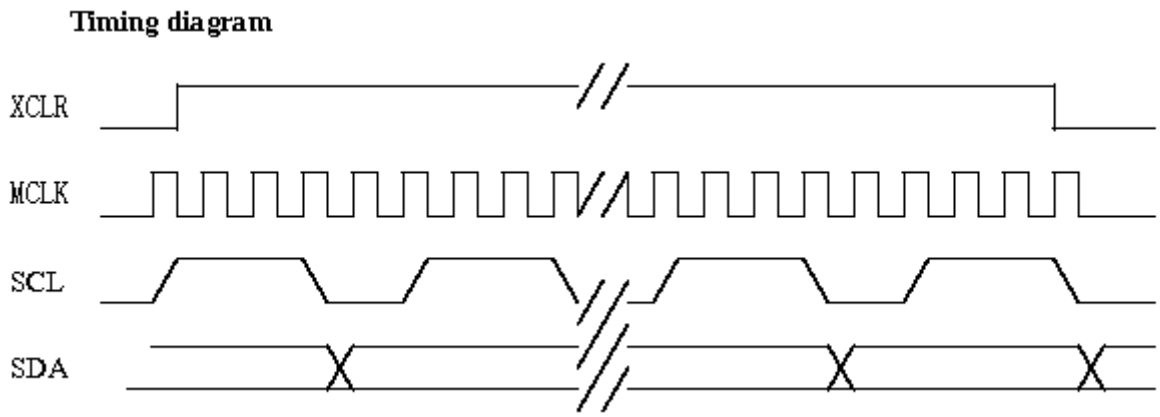


Fig: Typical Timing diagram of Read Temperature and Pressure operation

Calculate Real Temperature and Real Pressure:
Real Temperature and Pressure are calculated by using the below expressions provided by the data sheet:

if D2>=C5 dUT= D2-C5 - ((D2-C5)/2^7) * ((D2-C5)/2^7) * A / 2^C
if D2 < C5 dUT= D2-C5 - ((D2-C5)/2^7) * ((D2-C5)/2^7) * B / 2^C

OFF=(C2+(C4-1024)*dUT/2^14)*4
SENS = C1+ C3*dUT/2^10
X= SENS * (D1-7168)/2^14 - OFF

Real Pressure

P=X*10/2^5+C7

Real Temperature

T = 250 + dUT * C6 / 2 ^ 16-dUT/2^D

Calculate the Altitude:
The altitude is calculated by using a look-up table. Please refer the source code for exact calculation.

Application Ideas:
1.**Predict Rain**: A sudden fall in air pressure might bring rainfall. Similarly, raise in atmospheric pressure indicates clean sky.
2.Energy Saver: The temperature sensor along with a RTC can be used to control Air conditioners(AC) on time per day. No sophisticated relay circuit needed. An IR LED connected to Seeeduino film can be used to control the AC as a remote.

3. Three-Axis Accelerometer

Motion Frame is equipped with **Freescale MMA7660FC** 3-Axis accelerometer. MMA7660FC uses a capacitive type MEMS sensor and provides an I²C output. The device is capable of detecting acceleration in X, Y and Z direction. It also supports tilt orientation detection and gesture detection like shake detection and tap detection. The direction of accelerometer is **reverse mounted** on Motion Frame.

MMA7660FC's SCL, SDA are connected to Seeeduino Film's PC5 and PC4 respectively:

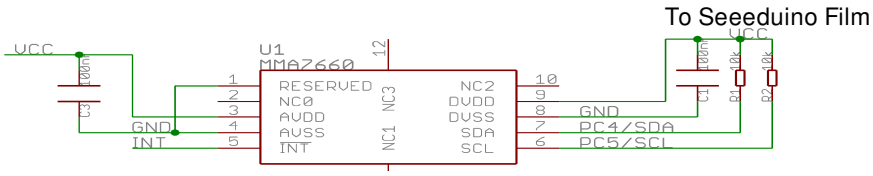


Fig: Accelerometer – Seeeduino Film Port connection

Chip Address:

- I²C Slave address of **MMA7660FC** is set to **0x4C**.

Application Programming Interface (API):

- The **MMA7660FC** library provides two high level interfaces **init()** and **accelarationRead()**.
- The result of the **accelarationRead()** is available in **accelarationData[3]** array in X, Y and Z order.

`init()`

- The init takes care of initialization of **MMA7660FC** chip.
- It begins the I²C communication with 0x4C as slave address.
- It sets the mode of MMA7660FC operation to Active mode.

`accelarationRead()`

- X, Y and Z values from acceleration is read by this function.
- The result of this operation is stored in **accelarationData**[] array.
- X, Y and Z values are provided in 6 bit 2's compliment signed byte format in the allowable range of +31 to -32.

The following simple example demonstrates the use **MMA7660FC** library. The access of functions and data variable of the library should be made through pre-instantiated object **Mma7660fc.**

```
#include <Wire.h>
#include <MMA7660FC.h>

void setup()
{
    Serial.begin(38400);
    Mma7660fc.init();  //Initialize Mma7660fc

}

void loop()
{
    char Acc_x=0;
    char Acc_y=0;
    char Acc_z=0;

    Mma7660fc.accelarationRead();// Read accelaration X, Y and Z

    Acc_x = ((char)(Mma7660fc.accelarationData[0]<<2))/4 ;
    Serial.print("Acc_x=");
    Serial.println(Acc_x,DEC);

    Acc_y = ((char)(Mma7660fc.accelarationData[1]<<2))/4 ;
    Serial.print("Acc_y=");
    Serial.println(Acc_y,DEC);

    Acc_z = ((char)(Mma7660fc.accelarationData[2]<<2))/4 ;
    Serial.print("Acc_z=");
    Serial.println(Acc_z,DEC);
    delay(1000);
}
```

Please refer the data sheet for the complete operation and full list of features provided by 3-Axis accelerometer. Features like tilt detection, shake detection are not provided by the library. This can be implemented by the user.

Application Ideas:

- **Orientation detection :** Accelerometer can be used whether the device is oriented in portrait or landscape direction.
- **Acceleration control and Measurement:** The 3 Axis acceleration provided by the module can be used to control the acceleration of a toy airplane or toy helicopter. It can also be used as an acceleration measuring device when connected to a suitable display device.

4.SPI Serial Flash Memory

Motion Frame is equipped with **Winbond W25X32** SPI Flash memory. It has a capacity of 32Mbit (4 Mega byte). Seeeduino Film PB3, PB4 and PB5 are connected to DI, DO and CLK pins of W25X32 flash memory.

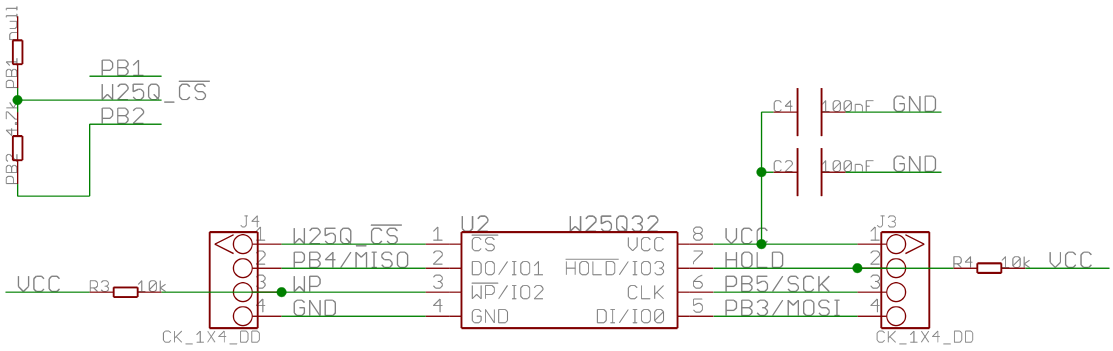


Fig : Connection between Seeeduino Film and SPI Flash Memory

The chip uses 256 bytes per programmable page. It supports a uniform 4K-byte sectors / 64K-byte blocks. W25X32 is capable of supporting data transfer up to 150bits / seconds. It also supports sector(4KB) erase and block(64KB) erase commands.

Application Programming Interface (API):
The following public interfaces are provided by **W25Xnn** library.This library uses **Spi** library available in Arduino Playground. There exist a pre-instantiated object W25xnn for accessing these methods. Please refer the source code for know how these functions are implemented.

`init():`
Initialize the W25X32 flash memory device by setting chip select pin.

`deviceInfoGet():`
Read and print manufacturer's and device's IDs.

`idleStatusWait():`
Waits for idle status of the chip. This is used before any new operation on the chip.

`statusRegisterRead():`
Reads the status register of the chip.

`jedec():`
Reads the JEDEC information of W25X32.

powerUp():
Sends power up command to W25X32.

powerDown():
Sends power down command to W25X32.

writeEnable():
Sends write enable command to W25X32.

writeDisable():
Sends write disable command to W25X32.

dataRead():
Reads *len* bytes of data from an *address* and stores it in a buffer pointed by *pHead* pointer

pageProgram():
Writes *len* bytes of data to an *address* from a buffer pointed by *pHead* pointer.

sectorErase():
Erases a sector of memory pointed by address *sector.*

blockErase():
Erases a block of memory pointed by address *block*.

chipErase():
Erases the complete chip.

The following example demonstrate the use of **W25Xnn** library.

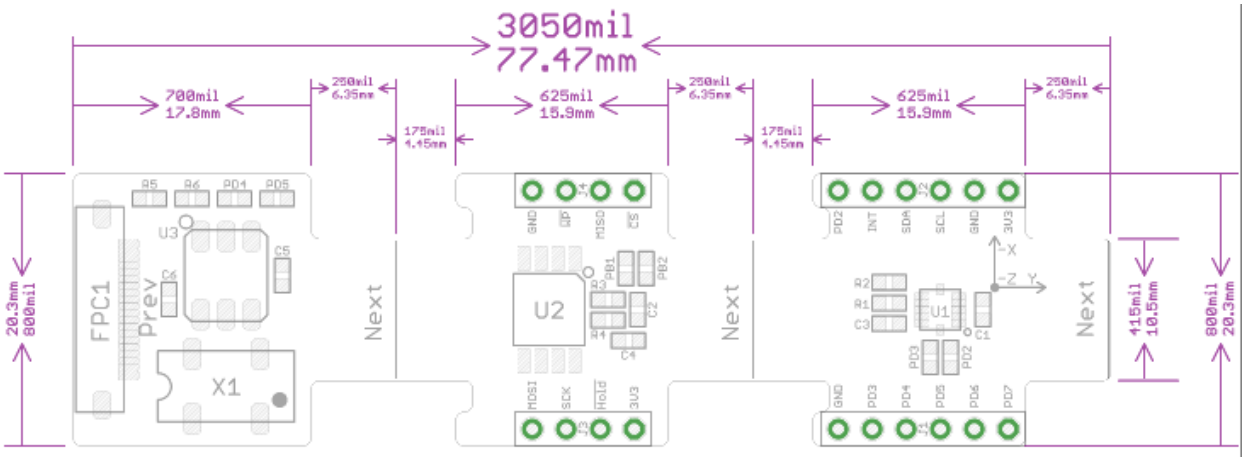
```
#include <Spi.h>
#include <W25Xnn.h>

void setup()
{
  unsigned char data[]="W25X32 32M bit flash";
  // Data to be written to the SPI flash
  Serial.begin(38400);
  W25xnn.init(); //initialize W25x32 serial Flash memory

  W25xnn.idleStatusWait();
  W25xnn.sectorErase(0x000000);   //erase first sector
  //W25xnn.blockErase(0x000000);  //erase first block
  //W25xnn.chipErase();
  delay(300);                     //sector erase time 150~300ms
  W25xnn.idleStatusWait();

  W25xnn.pageProgram(0x000000,data,20);
  Serial.println("<W25X32 32M bit flash> write to address 0x000000 to
0x000014");
```

```
  delay(10);

  /* Serial.println("erasing");
   W25xnn.chipErase();//chip erase time 40s~80s
   for(unsigned char i=0; i<80;i++)
   {
   Serial.println(i,DEC);
   delay(1000);
   }
   W25xnn.idleStatusWait();
   Serial.println("erased");
   */

  W25xnn.deviceInfoGet();
}

void loop()
{
  unsigned char temp_data[32];
  unsigned char i;
  W25xnn.dataRead(0x000000,temp_data,20);
  Serial.println("read data from address 0x000000 to 0x000014");
  for(i=0;i<20;i++)
  {
    Serial.print(temp_data[i]);
  }
  Serial.println();
  delay(1000);
}
```

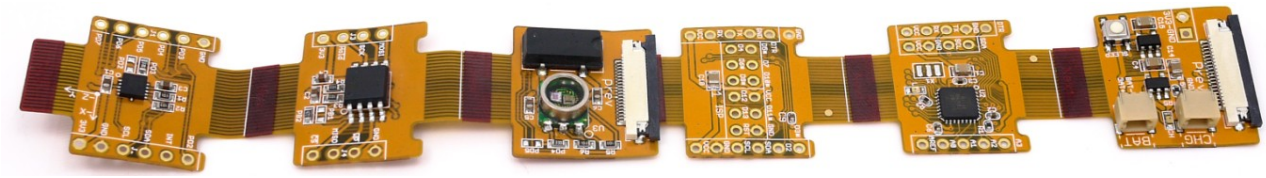
Please refer to W25X32 data-sheet for complete information of operations supported by the serial flash memory.

5.Mechanic Dimensions

The outline dimension is 77.5mm x 20.3mm x 3.45mm as below:



6.Application



Seeeduino Film Motion Frame can be used to build wearable devices. You can even put on a toy airplane or helicopter. It can store status information like altitude, temperature or acceleration to the Serial Flash. These information can be retrieved later by connecting Seeeduino Film to a PC. The barometer can be used to predict weather. 4 MB space available in serial flash memory can be used to build a data logger.

7.SPECIFICATION

KEY SPECIFICATION	Minimum	Normal	Maximum
Operating Voltage	3.0V	3.3V	3.6V
Operating Current	-	-	30mA
Power-down Current	30μA		65μA
Operating Temperature	-20℃	25℃	60℃
Storage Temperature	-30℃	25℃	70℃
Communication Protocol	I²C and SPI		

BAROMETER	Minimum	Normal	Maximum
Operating Voltage	2.2V	3.3V	3.6V
Active RTC Crystal	1.5V	3.3V	5.5V
Communication Protocol	I²C		
I²C Speed	100kHz ~ 400 kHz		
Absolute Pressure Accuracy	±1.5hPa @0~50℃, 750~1100hPa		
Absolute Pressure Accuracy	±3hPa @-20~60℃, 750~1100hPa		
Temperature Accuracy	±1℃ @ 0~50℃		
Temperature Accuracy	±2℃ @ -20~60℃		

32M-BIT FLASH	Minimum	Normal	Maximum
Operating Voltage	2.7V	3.3V	3.6V
Communication Protocol	SPI		
SPI Speed	Up to 75MHz		
Erase/Write cycles	Up to 100000 cycles		
Sector Erase/Write	4K Bytes		
Page Program	Up to 256 Bytes within 2ms		
Data Retention	20 year		

I²C 3-AXIS ACCELEROMETER	Minimum	Normal	Maximum
Operating Voltage	2.4v	3.3v	3.6V
Communication Protocol	I²C		
I²C Speed	100kHz~400kHz		
Maximum Acceleration (all axes, 100 μs)	10000 g		

8.SUPPORT

Please refer to product page for latest documents and development resources, any product related issue could be inquired via info@seeedstudio.com

9.REVISION HISTORY

Revision	Descriptions	Editor	Release date
v0.9c	Rewrite. Added detailed description.	Visweswara R	30th November 2010
v0.9b	Initial public release	Lafier	5th October 2010
v0.9b-1	Change the parameters of barometer	Lafier	26nd October 2010

10.LICENSE :