Accelerometer Sensor Module

Setup

1. Connect one end of the cable into either Molex connectors on the sensor
2. Connect the other end of the cable to the Arduino board:
   - RED: 5V
   - WHITE: I2C SDA (pin A4 on Uno; pin 20 on Mega)
   - BLACK: GND
   - GREY: I2C SCL (pin A5 on Uno; pin 21 on Mega)
3. Set the DIP switch on the sensor to set the sensor address (check back of sensor for possible addresses)

Example Sketch

```
// OSEPP Accelerometer Sensor Example Sketch
// by OSEPP <http://www.osepp.com>

// This sketch demonstrates interactions with the Accelerometer Sensor

#include <Wire.h>

// Possible sensor addresses (suffix correspond to DIP switch positions)
#define SENSOR_ADDR_OFF  (0x1D)
#define SENSOR_ADDR_ON   (0x53)

// Set the sensor address here
const uint8_t sensorAddr = SENSOR_ADDR_OFF;

// Sensor register addresses (gotten from datasheet)
#define REG_DEVID_ADDR        (0x00)
#define REG_THRESH_TAP_ADDR   (0x1d)
#define REG_TAP_DUR_ADDR      (0x21)
#define REG_TAP_LATENCY_ADDR  (0x22)
#define REG_TAP_WINDOW_ADDR   (0x23)
#define REG_BW_RATE_ADDR      (0x2c)
#define REG_PWR_CTL_ADDR      (0x2d)
#define REG_INT_ENABLE_ADDR   (0x2e)
#define REG_DATA_FORMAT_ADDR  (0x31)
#define REG_DATAX0_ADDR       (0x32)
#define REG_DATAX1_ADDR       (0x33)
#define REG_DATAY0_ADDR       (0x34)
#define REG_DATAY1_ADDR       (0x35)
#define REG_DATAZ0_ADDR       (0x36)
#define REG_DATAZ1_ADDR       (0x37)
#define REG_FIFO_CTL_ADDR     (0x38)

// One-time setup
void setup()
{
   // Start the serial port for output
   Serial.begin(9600);

   // Join the I2C bus as master
   Wire.begin();

   // Set 25 Hz output data rate and 25 Hz bandwidth and disable low power mode
   WriteByte(sensorAddr, REG_BW_RATE_ADDR, 0x08);

   // Disable auto sleep
   WriteByte(sensorAddr, REG_PWR_CTL_ADDR, 0x08);

   // Disable interrupts (the pins are not brought out anyway)
   WriteByte(sensorAddr, REG_INT_ENABLE_ADDR, 0x0);
}

// Main program loop
```

```
void loop()
{
   uint8_t devId;
   uint8_t x_msb;   // X-axis most significant byte
   uint8_t x_lsb;   // X-axis least significant byte
   uint8_t y_msb;   // Y-axis most significant byte
   uint8_t y_lsb;   // Y-axis least significant byte
   uint8_t z_msb;   // Z-axis most significant byte
   uint8_t z_lsb;   // Z-axis least significant byte
   uint16_t x;
   uint16_t y;
   uint16_t z;

   // Read the device ID just to make sure we are talking to the correct sensor
   if (ReadByte(sensorAddr, 0x0, &devId) != 0)
   {
      Serial.println("Cannot read device ID from sensor");
   }
   else if (devId != 0xE5)
   {
      Serial.print("Wrong/invalid device ID ");
      Serial.print(devId);
      Serial.println(" (expected 0xE5)");
   }
   else
   {
      // Read the output
      if ((ReadByte(sensorAddr, REG_DATAX1_ADDR, &x_msb) == 0) &&
          (ReadByte(sensorAddr, REG_DATAX0_ADDR, &x_lsb) == 0) &&
          (ReadByte(sensorAddr, REG_DATAY1_ADDR, &y_msb) == 0) &&
          (ReadByte(sensorAddr, REG_DATAY0_ADDR, &y_lsb) == 0) &&
          (ReadByte(sensorAddr, REG_DATAZ1_ADDR, &z_msb) == 0) &&
          (ReadByte(sensorAddr, REG_DATAZ0_ADDR, &z_lsb) == 0))
      {
         x = (x_msb << 8) | x_lsb;
         y = (y_msb << 8) | y_lsb;
         z = (z_msb << 8) | z_lsb;

         // Perform 2's complement
         int16_t real_x = ~(x - 1);
         int16_t real_y = ~(y - 1);
         int16_t real_z = ~(z - 1);

         Serial.print("X: ");
         Serial.println(real_x);
         Serial.print("Y: ");
         Serial.println(real_y);
         Serial.print("Z: ");
         Serial.println(real_z);
      }
      else
      {
         Serial.println("Failed to read from sensor");
      }
   }

   // Run again in 1 s (1000 ms)
```

```
   delay(1000);
}

// Read a byte on the i2c interface
int ReadByte(uint8_t addr, uint8_t reg, uint8_t *data)
{
   // Do an i2c write to set the register that we want to read from
   Wire.beginTransmission(addr);
   Wire.write(reg);
   Wire.endTransmission();

   // Read a byte from the device
   Wire.requestFrom(addr, (uint8_t)1);
   if (Wire.available())
   {
      *data = Wire.read();
   }
   else
   {
      // Read nothing back
      return -1;
   }

   return 0;
}

// Write a byte on the i2c interface
void WriteByte(uint8_t addr, uint8_t reg, byte data)
{
   // Begin the write sequence
   Wire.beginTransmission(addr);

   // First byte is to set the register pointer
   Wire.write(reg);

   // Write the data byte
   Wire.write(data);

   // End the write sequence; bytes are actually transmitted now
   Wire.endTransmission();
}
```

Industry:

Submit

- 

-
-
-
-

- Products
- Where to Buy
- Videos
- Learning Centre
- About Us
- Contact Us