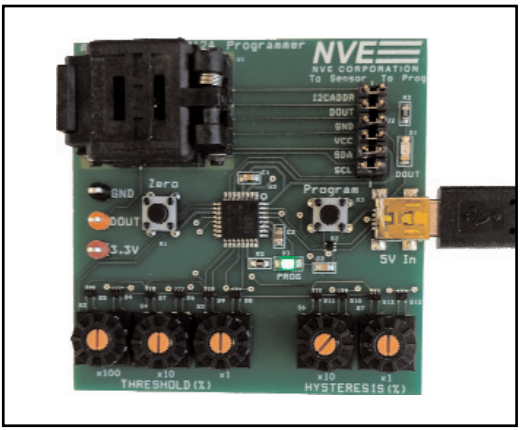
NVE CORPORATION

AG955-07E
Thumbwheel Programmer for SM12x-Series Smart Sensors



SB-00-092

Overview

This self-contained module programs the SM12x digital output ("DOUT") threshold and hysteresis without a computer or customer microcontroller. The board has an I²C master microcontroller to connect to the sensor.

A custom TDFN-6 socket accommodates the SM12x sensor. Miniature rotary thumbwheel switches set the threshold and hysteresis, and a pushbutton allows rezeroing the sensor. There are three digits of resolution for the threshold and two for hysteresis. The module is powered by a small wall-mounted power supply (included).

Jumpers allow the board to be used as a self-contained programmer, a programmer for a breakout board such as an NVE AG958-07E, or the socket can be used to connect a sensor to the customer's own electronics.

This Module Includes

- A 2.5 by 2.5-inch (64 x 64 mm) circuit board with:
  - A 2.5 x 2.5 mm TDFN6 socket for a SM12x Smart Magnetometer
  - A preprogrammed, onboard microcontroller
  - Two sets of thumbwheel switches for digital threshold and hysteresis
  - Two LEDs indicating the sensor's output and successful programming

- A 5-volt wall-mount power supply

SM12x Smart Magnetometer Features

- Factory calibrated for sensitivity, offset, and linearity; can be recalibrated
- 1 mT (SM124) or 4 mT (SM125) linear range
- Elegant I²C connections
- Can detect magnets more than 50 mm away
- Digital threshold output
- 7-bit output resolution
- In-plane sensitivity—more usable than Hall effect sensors
- Wide 2.2 to 3.6 V supply range
- Ultraminiature 2.5 mm x 2.5 mm x 0.8 mm TDFN6 package

Visit **www.nve.com** for complete product specifications.

Programmer Operation

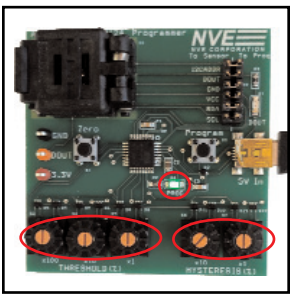
⇒ Verify jumpers or a sensor connector are in place.

⇒ Place an SM12x-10E sensor in the socket or connect the board to an SM12x breakout board (part number AG958-07E or AG961-07E).

⇒ Connect the five-volt power supply.

⇒ If desired, rezero the sensor by pressing the "Zero" button with no field present. This overrides the factory calibration offset.

⇒ The green LED turns off briefly, then on to indicate successful rezeroing.



Thumbwheels and successful programming LED.

⇒ Set the "threshold" and "hysteresis" thumbwheels to the desired values. Thumbwheel settings are percentages of the sensor's linear range. So for example, "50" represents 50% of the linear range, or 0.5 mT for an SM124.

⇒ The threshold can be between 1 and approximately 120%, and hysteresis from 1 to 99%. A hysteresis setting greater than the threshold invokes a latching mode.

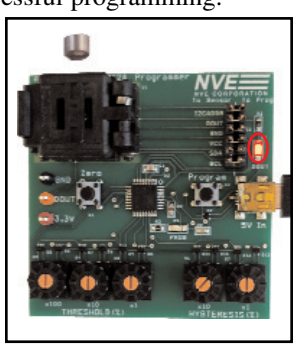
⇒ Press the "Program" button.

⇒ The green LED turns off, then on to indicate successful programming.

⇒ The green LED will flash for one second if programming was unsuccessful. Verify the threshold and hysteresis settings are in the allowable range and check the sensor seating in the socket.

⇒ The orange LED on the sensor Digital Output ("DOUT") can provide a gross check by activating the sensor with a magnet while the sensor is still in the socket.

⇒ Remove the sensor from the socket or disconnect the breakout board and put it to work.



DOUT indication with a magnetic field applied.

Visit **nve.com** or **YouTube/NveCorporation** for a demonstration.

Thumbwheel Switches

There are two sets of thumbwheel switches on the board:

- **Threshold**
  Sets the field at which the output turns on.
  The range is 1 to approximately 120%.

- **Hysteresis**
  Sets the hysteresis of the threshold. The range is 1 to 99%. A hysteresis setting greater than the threshold invokes a latching mode.

Settings are percentages of the sensor's linear range. For example, "50" represents 50% of the linear range (0.5 mT for an SM124; 2 mT for an SM125).

Buttons

There are two pushbutton switches:

- **Zero**
  Overrides the factory calibration offset by zeroing the magnetometer.

- **Program**
  Programs the Threshold and Hysteresis values set on the thumbwheels.

LEDs

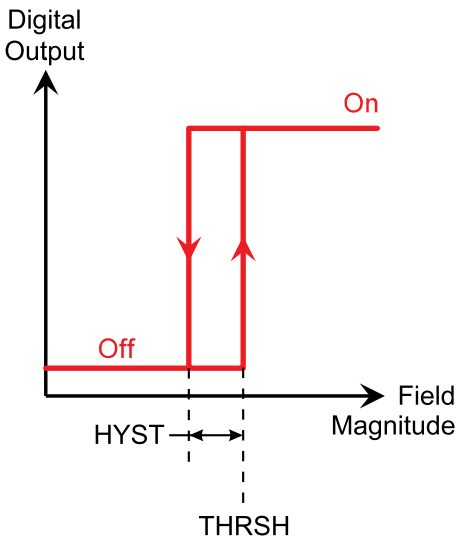
There are two LEDs:

- **Digital Output** **(orange LED)**
  - Connected to the sensor's digital output (DOUT), the LED turns on when the output is HIGH.

- **Status** **(green LED)**
  - Flashes on power-up to indicate a successful programmer board power-on reset and microcontroller boot.
  - Solid ON indicates successful rezeroing or programming, verified by reading the parameters back from the sensor.
  - Flashes for one second if programming was unsuccessful.

The Threshold and Hysteresis Parameters

This board uses the SM12x “default” comparator mode (it does not support the “Window Comparator” mode). DOUT goes HIGH when the sensor field exceeds a threshold (THRSH; also known as THRSH_L), then LOW when the field magnitude drops below the threshold minus hysteresis as illustrated below:



Nonvolatile
Threshold and hysteresis parameters are stored in the sensor’s nonvolatile memory, and can be set for life if desired.

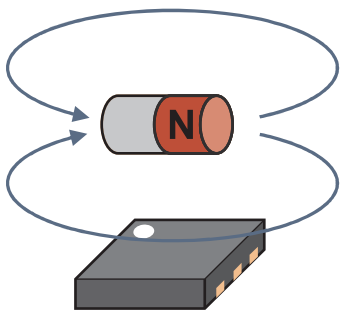
Continuously Updated
DOUT is continuously updated at high speed and runs independently of the I²C interface.

Latching Mode
If Hysteresis is set to be greater than Threshold, DOUT will latch ON the first time the field exceeds the threshold. Once latched, the output can be reset by cycling the sensor power. The latching mode is useful for fault detection and safety shutoffs.

Convenient Direction of Magnetic Sensitivity

In-Plane Sensitivity
The SM12x Digital Output ("DOUT") turns on and off in response to magnetic fields. Unlike Hall effect or other sensors, the direction of sensitivity is in the plane of the package, which is more convenient.

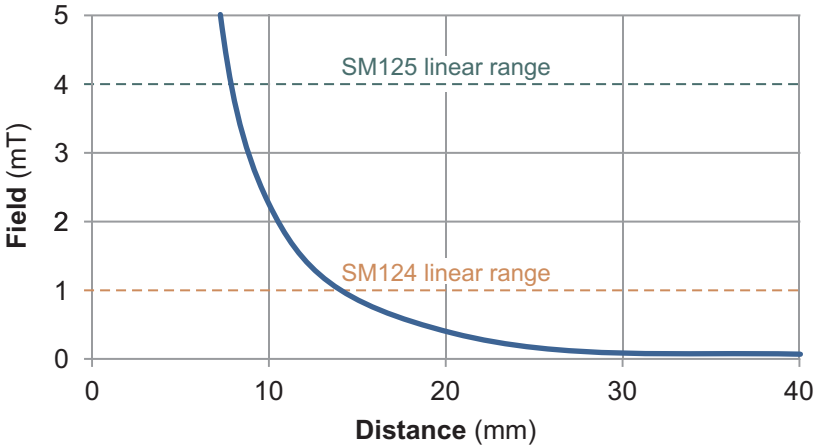
The diagram below shows the magnet orientation to activate the sensor:



Omnipolar
SM12x magnetometers are "omnipolar," meaning the output turns ON with a field of either magnetic polarity. This simplifies system design since magnet polarity is often unknown.

Magnet-to-Sensor Distance

Typical operating distances are illustrated in this graph for an inexpensive 6 mm diameter by 4 mm thick ferrite disk magnet:



Larger and stronger magnets allow farther operate and release distances. For more calculations, use our axial disc magnetic field versus distance Web application at:

www.nve.com/spec/calculators.php#tabs-Axial-Disc-Magnet-Field

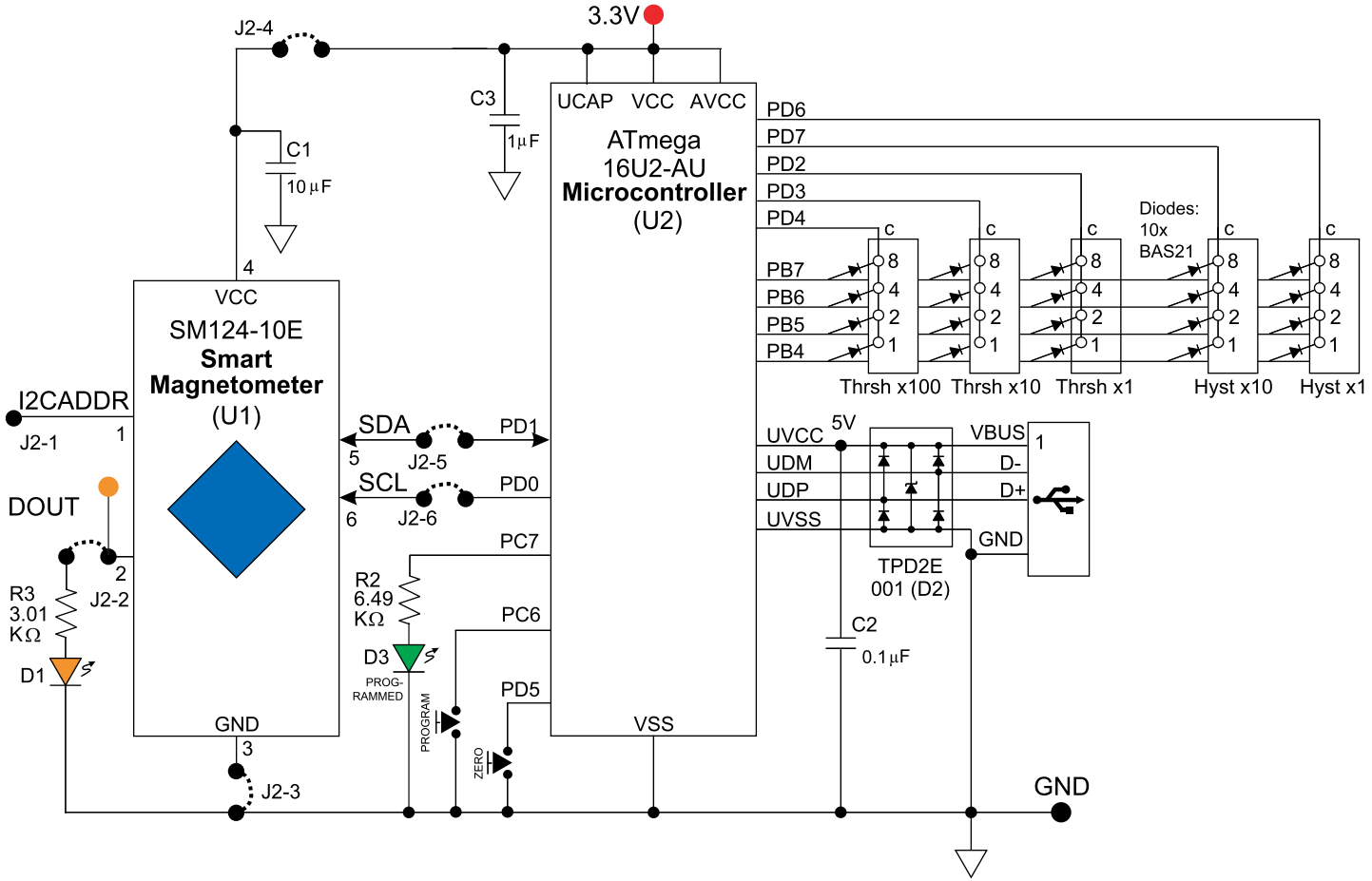
Responsibilities That Come With High Sensitivity

With low thresholds, care should be taken to account for the earth's magnetic field, which is typically about 0.05 mT.

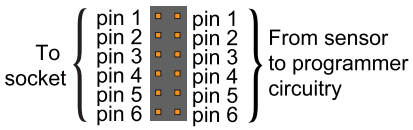
For low-field applications we recommend ultrasmall bypass capacitors such as 0201 (0603 metric) or 01005 (0402 metric), since they contain less ferromagnetic material than larger components.

Also, materials with remnant fields (permanent magnetization) such as steel should be avoided near the sensor. This board, for example, uses brass rather than steel screws and nuts to attach the socket to the circuit board. Brass, nylon, or austenitic stainless steel (such as 18-8), are recommended for hardware near the sensor.

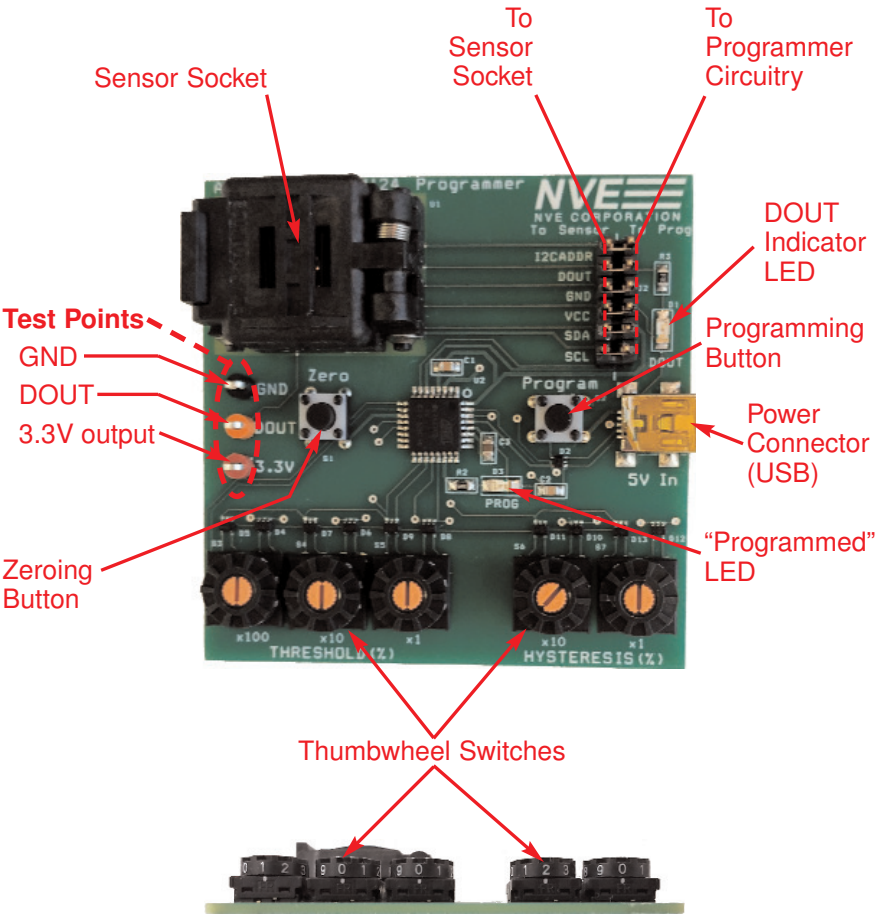
Schematic Diagram



J2 Detail



Board Layout



Bill of Materials

Part Number	Manufacturer	Reference Designator	Description
N/A	Custom	U1	TDFN6 SOCKET FOR SM124-10E
ATMEGA16U2-AUR	Microchip Technology	U2	IC MCU 8BIT 16KB FLASH 32TQFP
JLMK107BBJ106MALT	Taiyo Yuden	C1	CAP CER 10UF 10V X5R 0603
885012207016	Wurth Electronics Inc.	C2	CAP CER 0.1UF 10V X7R 0805
GRM21BR71C105KA01L	Murata Electronics	C1, C3	CAP CER 1UF 16V X7R 0805
APT3216LSECK/J4-PRV	Kingbright	D1	LED ORANGE CLEAR CHIP SMD
APT3216SGC	Kingbright	D3	LED GREEN CLEAR CHIP SMD
TPD2E001DRLR	Texas Instruments	D2	TVS DIODE 5.5V SOT5
BAS21DW5T1G	ON Semiconductor	D4-D13	DIODE ARRAY GP 250V 200MA SOT353
RMCF0805FT6K49	Stackpole Electronics	R2	RES 6.49K OHM 1% 1/8W 0805
RMCF0805FT3K01	Stackpole Electronics	R3	RES 3.01K OHM 1% 1/8W 0805
FR01SR10P	NKK Switches	S3-S7	SWITCH ROTARY DIP BCD 100 MA 5V
1825910-6	TE Connectivity ALCO	S1, S2	SWITCH TACTILE SPST-NO 0.05A 24V
690-005-299-043	EDAC Inc.	J1	CONN RCPT USB2.0 MINI B SMD R/A
TSW-106-14-T- D	Samtec Inc.	J2	CONN HEADER 12POS .100" DUAL TIN
QPC02SXGN-RC	Sullins Connector	J2 (6)	CONN JUMPER SHORTING .100" GOLD
5005	Keystone Electronics	J3	PC TEST POINT COMPACT RED
5006	Keystone Electronics	J4	PC TEST POINT COMPACT BLACK
5008	Keystone Electronics	J5	PC TEST POINT COMPACT ORANGE

Programmer Firmware

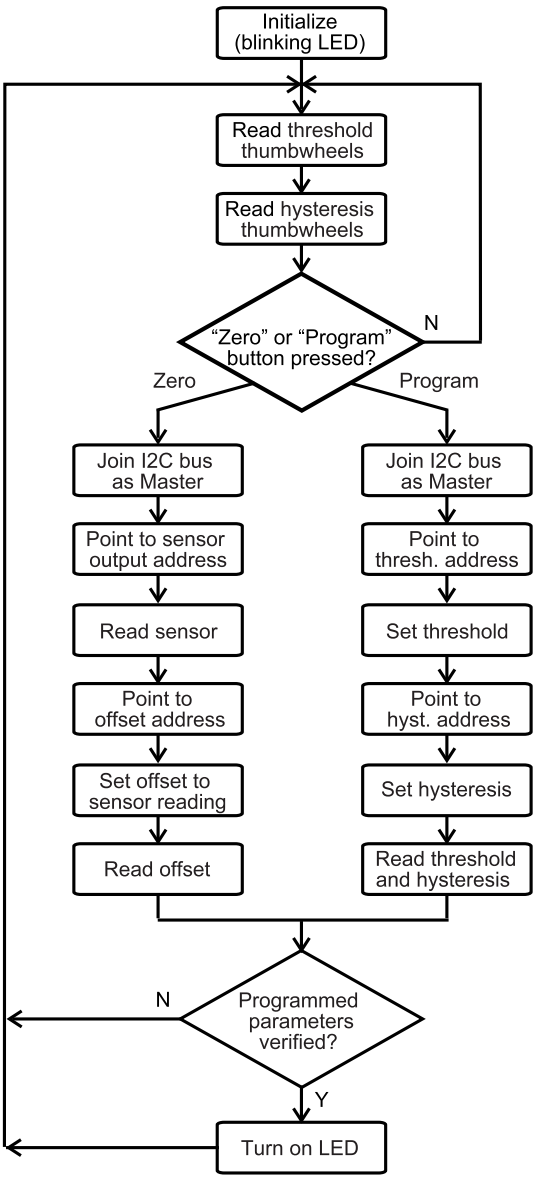
```
/* SM124 Thumbwheel Programmer
 * For SM124 version with widow comparator (lot codes 1932xx and higher)
 */

#define I2C_TIMEOUT 1000
#define I2C_PULLUP 1
#define SDA_PORT PORTD
#define SDA_PIN 1 // = A4
#define SCL_PORT PORTD
#define SCL_PIN 0 // = A5
unsigned char MEMLOC = 0x0A;
#define ADDRLEN 1 // address length, usually 1 or 2 bytes
#define I2C_WRITE   0
#define I2C_READ    1
#define F_CPU 8000000UL

#include <avr/io.h>
#include "SoftI2CMaster.h"
#include <util/delay.h>

#define LOC_HYST_ORIG 0x21
#define LOC_OFFSET_ORIG 0x21
#define LOC_FIELD 0x00
#define LOC_THRESHOLD 0x20

#define ORIGINAL_LOT_SHIFT 0
#define WINDOW_LOT_SHIFT 1
#define BAD_LOT_CODE -1

double holder;
unsigned char field=0, uncal=0;
unsigned char threshold=0;
unsigned char hysteresis=0;
int zeroSwitch= 0b00100000; //"Zero" button previous state
int pgmSwitch = 0b01000000; //"Program" button previous state
int digit=1; //Digit counter
int g_i2c_addr,iterator;

void setup() {
  DDRC = 0b10000000; //Set PC7 as output for green LED (indicates successful programming)
  DDRB = 0b00001111; //Set PB4-PB7 as inputs (for thumbwheels)
  PORTB = 0b11110000; //Enable PB4-PB7 pullups
  DDRD = 0b11011111; //Set PD2-PD4; PD6-PD7 as outputs for thumbwheel selection (low true)
  DDRD = 0b11011111; //Set PD5 as input for "Zero" button
  PORTD = 0b00100011; //Enable PD5 pullup
  DDRC = 0b10111111; //Set PC6 as input for "Program" button
  PORTC = 0b01000000; //Enable PC6 pullup
  //Flash the green LED to indicate successful bootup
  for(digit=1; digit<6; digit++) { //Toggle LED
    _delay_ms(200);
    PORTC ^= 0b10000000;
  }
  _delay_ms(200);
  I2c_init();
}

int read_location(unsigned char location, unsigned char * value) {
  unsigned char buffer; unsigned int attempts = 0;
  while(!i2c_start((g_i2c_addr << 1) | I2C_WRITE)) {  // write the location
    if(attempts++ > 100) {i2c_stop();return -1;}
    i2c_stop();
  }
  i2c_write(location);
  i2c_stop();
  attempts = 0;
  while(!i2c_start((g_i2c_addr << 1) | I2C_READ)) { // read from the location
    if(attempts++ > 100) {i2c_stop();return -1;}
    i2c_stop();
  }
  buffer = i2c_read(1);
  i2c_stop();
  *value = buffer;
  return 0;
}

int write_location(unsigned char location, unsigned char value) {
  unsigned int attempts = 0;
  while(!i2c_start((g_i2c_addr << 1) | I2C_WRITE)) {
    if(attempts++ > 100) {i2c_stop();return -1;}
    i2c_stop();
  }
  i2c_write(location);
  i2c_write(value);
  i2c_stop();
  _delay_ms(20);
  return 0;
}

char get_sensor_lot_shift(void) {
  unsigned char l1,l2,l3,l4,l5,l6;
  unsigned long long total;

  if(read_location(0x80, &l1)) return BAD_LOT_CODE;
  if(read_location(0x81, &l2)) return BAD_LOT_CODE;
  if(read_location(0x82, &l3)) return BAD_LOT_CODE;
  if(read_location(0x83, &l4)) return BAD_LOT_CODE;
  if(read_location(0x84, &l5)) return BAD_LOT_CODE;
  if(read_location(0x85, &l6)) return BAD_LOT_CODE;

  total = ((unsigned long long) l1 << (5 * 8)) +
      ((unsigned long long) l2 << (4 * 8)) +
      ((unsigned long long) l3 << (3 * 8)) +
      ((unsigned long long) l4 << (2 * 8)) +
      ((unsigned long long) l5 << (1 * 8)) +
      ((unsigned long long) l6 << (0 * 8));

  if(total >= 0x313933323030)
    return WINDOW_LOT_SHIFT;
  else
    return ORIGINAL_LOT_SHIFT;
```



```
  return WINDOW_LOT_SHIFT;
}

int get_address(){
  for(iterator = 0; iterator < ((0xFF >> 1) + 1); iterator++)
  {
    if(i2c_start((iterator << 1) | I2C_WRITE)) {
      g_i2c_addr=iterator;
      i2c_stop();
      return g_i2c_addr;
    }
    i2c_stop();
  }
  return -1;
}

int main(void) {
  char lot_shift;
  unsigned char passed;
  unsigned char read_buffer;
  unsigned char offset_buffer;
  unsigned char original_offset_buffer;
  unsigned char new_offset_buffer;
  unsigned char new_threshold;
  unsigned char new_hysteresis;

  setup();
    while (1) {
    if(PIND & 0b00100000 ^ zeroSwitch) { //"Zero" button changed
      if(zeroSwitch) {
        passed = 0;

        PORTC &= 0b01111111; //Turn off LED pending verification
        _delay_ms(200); //Just to see the LED turn off when it is working

        if(get_address()==-1) goto ZERO_EXIT; // get address
        lot_shift = get_sensor_lot_shift(); // get lot
        if(lot_shift == BAD_LOT_CODE) goto ZERO_EXIT;
        // store original offset in case it needs to be reverted
        if(read_location(LOC_OFF_ORIG + lot_shift, &original_offset_buffer)) goto ZERO_EXIT
        if(read_location(LOC_FIELD, &read_buffer)) goto ZERO_EXIT; // read the field
        while(field_buffer == 0){ // if the field is 0 then it may have been truncated; increase the offset until the rea
          if(read_location(LOC_OFF_ORIG + lot_shift, &offset_buffer)) goto ZERO_EXIT; // read the offset
          if((signed char)offset_buffer > 100) { // if its unreasonably large something went wrong
            write_location(LOC_OFF_ORIG + lot_shift, original_offset_buffer); //reset back to original
            goto ZERO_EXIT;
          }
          if(write_location(LOC_OFF_ORIG + lot_shift, offset_buffer + 10)) goto ZERO_EXIT;// write shifted offset
          if(read_location(LOC_FIELD, &field_buffer)) goto ZERO_EXIT;// read field again
        }
        if(read_location(LOC_OFF_ORIG + lot_shift, &offset_buffer)) goto ZERO_EXIT;// read the offset
        new_offset_buffer = (signed char) offset_buffer - field_buffer; // change the offset by the reading
        if(write_location(LOC_OFF_ORIG + lot_shift, new_offset_buffer)) goto ZERO_EXIT;// write it
        if(read_location(LOC_OFF_ORIG + lot_shift, &offset_buffer)) goto ZERO_EXIT;
        if(offset_buffer != new_offset_buffer) goto ZERO_EXIT;// make sure its the same
        passed = 1;
ZERO_EXIT:
        if(passed) { //Turn on LED if passed
          PORTC |=0b10000000;
        }
        else { // Flash LED if failed
          PORTC |=0b10000000;
          _delay_ms(1000);
          PORTC &= 0b01111111;
        }
      }
      zeroSwitch ^= 0b00100000; //Update switch
    }
    if((PINC & 0b01000000 ^ pgmSwitch)) { //"Program" button changed
      if(pgmSwitch) { //Button pressed (not released)
        passed = 0;
        PORTC &= 0b01111111; //Turn off LED pending verification
        _delay_ms(200); //Just to see the LED turn off when it is working
        if(get_address()==-1) goto PROGRAM_EXIT; // get address
        lot_shift = get_sensor_lot_shift(); // get lot
        if(lot_shift == BAD_LOT_CODE) goto PROGRAM_EXIT;// if the function failed

        threshold=0; hysteresis=0; //Prepare to read thumbwheels
        PORTD = (PORTD | 0b11011100) & 0xFB; //Set PD2 low; other thumbwheel selections high
        for(digit=1; digit<999; digit*=10) { //Step through the three threshold digit thumbwheels
          _delay_us(1); //Let input stabilize
          threshold += ((PINB >> 4)^0xF)*digit; //Read each digit
          PORTD = ((PORTD << 1) & 0x1C) | 4 | (PORTD & 0xE3); //Select next thumbwheel
        }
        PORTD = (PORTD | 0b11011100) & 0xBF; //Set PD6 low; other thumbwheel selections high
        for(digit=1; digit<99; digit*=10) { //Step through the two hysteresis thumbwheels
          _delay_us(1); //Let input stabilize
          hysteresis += ((PINB >> 4)^0xF)*digit; //Read and invert digits
          PORTD = (PORTD | 0b11011100) & 0x7F; //Set PD7 low; other thumbwheel selections high
        }

        if(write_location(LOC_THRESHOLD, threshold)) goto PROGRAM_EXIT;
        if(write_location(LOC_HYST_ORIG + lot_shift, hysteresis)) goto PROGRAM_EXIT;
        if(read_location(LOC_THRESHOLD, &new_threshold)) goto PROGRAM_EXIT;
        if(read_location(LOC_HYST_ORIG + lot_shift, &new_hysteresis)) goto PROGRAM_EXIT;
        if((threshold == new_threshold) && (hysteresis == new_hysteresis)) passed = 1;
PROGRAM_EXIT:
        if(passed) { //Turn on LED if verified
          PORTC |=0b10000000;
        }
        else { // Flash LED if failed
          PORTC |=0b10000000;
          _delay_ms(1000);
          PORTC &= 0b01111111;
        }
      }
      pgmSwitch ^= 0b01000000; //Update switch state
    }
    }
}
```

E-mail us for the firmware:
sensor-apps@nve.com

Programmer Flowchart



In Case of Difficulty

- **The board doesn't turn on (no LED activity).**
  - Check the power supply connection.

- **The green verification LED doesn't turn on after programming.**
  - Verify the threshold and hysteresis settings are in the allowable range.
  - Check the sensor seating in the socket.
  - Verify that all jumpers to the socket or a connector to a sensor is in place.
  - Reboot the microcontroller by cycling the power. The sensor should be in the socket when power is applied so the board can establish communications with the sensor.

- **Sensor threshold output never turns on.**
  - Verify the threshold settings is in the allowable range (1 to 120 for the threshold and 1 to 99 for hysteresis). The sensor may not reach thresholds that are significantly beyond the 0 to 100 linear range.
  - Verify the hysteresis is set to less than the threshold.
  - Verify that all jumpers are in place.

- **Sensor threshold output never turns off.**
  - Verify the threshold and hysteresis settings are in the allowable range.
  - Verify the hysteresis is set less than the threshold (more hysteresis than the threshold will cause the output to latch on).
  - If the latching mode is being invoked (hysteresis > threshold), the sensor power must be cycled to reset its output.
  - For low thresholds (less than approximately 0.5 mT), verify there are no stray magnetic fields, ensure there are no remnant fields from sockets or other fixturing, and change the orientation for avoid a contribution from the earth's magnetic field.

We're here to help: ***sensor-apps@nve.com***

NVE CORPORATION

Limited Warranty and Liability
Information in this document is believed to be accurate and reliable. However, NVE does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. In no event shall NVE be liable for any indirect, incidental, punitive, special or consequential damages (including, without limitation, lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Right to Make Changes
NVE reserves the right to make changes to information published in this document including, without limitation, specifications and product descriptions at any time and without notice.

Use in Life-Critical or Safety-Critical Applications
Unless NVE and a customer explicitly agree otherwise in writing, NVE products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical devices or equipment. NVE accepts no liability for inclusion or use of NVE products in such applications and such inclusion or use is at the customer's own risk. Should the customer use NVE products for such application whether authorized by NVE or not, the customer shall indemnify and hold NVE harmless against all claims and damages.

Applications
Applications described in this document are illustrative only. NVE makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their applications and products using NVE products, and NVE accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NVE product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customers. Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. NVE does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customers. The customer is responsible for all necessary testing for the customer's applications and products using NVE products in order to avoid a default of the applications and the products or of the application or use by customer's third party customers. NVE accepts no liability in this respect.

An ISO 9001 Certified Company

NVE Corporation
11409 Valley View Road
Eden Prairie, MN 55344-3617

Manual No.: SB-00-092 Rev. B; 11/25/19