SPI - BarmetricPressureSensor, DigitalPotControl

Example Code

```
/*
 SCP1000 Barometric Pressure Sensor Display

 Shows the output of a Barometric Pressure Sensor on a
 Uses the SPI library. For details on the sensor, see:
 http://www.sparkfun.com/commerce/product_info.php?products_id=8161
 http://www.vti.fi/en/support/obsolete_products/pressure_sensors/

 This sketch adapted from Nathan Seidle's SCP1000 example for PIC:
 http://www.sparkfun.com/datasheets/Sensors/SCP1000-Testing.zip

 Circuit:
 SCP1000 sensor attached to pins 6, 7, 10 - 13:
 DRDY: pin 6
 CSB: pin 7
 MOSI: pin 11
 MISO: pin 12
 SCK: pin 13

 created 31 July 2010
 modified 14 August 2010
 by Tom Igoe
 */

// the sensor communicates using SPI, so include the library:
#include <SPI.h>

//Sensor's memory register addresses:
const int PRESSURE = 0x1F;      //3 most significant bits of pressure
const int PRESSURE_LSB = 0x20;  //16 least significant bits of pressure
const int TEMPERATURE = 0x21;   //16 bit temperature reading
const byte READ = 0b11111100;     // SCP1000's read command
const byte WRITE = 0b00000010;   // SCP1000's write command

// pins used for the connection with the sensor
// the other you need are controlled by the SPI library):
const int dataReadyPin = 6;
const int chipSelectPin = 7;

void setup() {
  Serial.begin(9600);

  // start the SPI library:
  SPI.begin();

  // initialize the  data ready and chip select pins:
```

```
  pinMode(dataReadyPin, INPUT);
  pinMode(chipSelectPin, OUTPUT);

  //Configure SCP1000 for low noise configuration:
  writeRegister(0x02, 0x2D);
  writeRegister(0x01, 0x03);
  writeRegister(0x03, 0x02);
  // give the sensor time to set up:
  delay(100);
}

void loop() {
  //Select High Resolution Mode
  writeRegister(0x03, 0x0A);

  // don't do anything until the data ready pin is high:
  if (digitalRead(dataReadyPin) == HIGH) {
    //Read the temperature data
    int tempData = readRegister(0x21, 2);

    // convert the temperature to celsius and display it:
    float realTemp = (float)tempData / 20.0;
    Serial.print("Temp[C]=");
    Serial.print(realTemp);


    //Read the pressure data highest 3 bits:
    byte  pressure_data_high = readRegister(0x1F, 1);
    pressure_data_high &= 0b00000111; //you only needs bits 2 to 0

    //Read the pressure data lower 16 bits:
    unsigned int pressure_data_low = readRegister(0x20, 2);
    //combine the two parts into one 19-bit number:
    //
    // 1/22/21: Fix bug dating back to the dark ages in example
    // pressure_data_high is a 16-bit datatype, if you leftshift 16 bits
    // you have 0. The fact that you then assign the result to a larger
    // variable that could fit those extra bits isn't the compiler's
    // concern.
    //
    // More than anything else, what this demonstrates is why
    // you should always enable warnings!
    long pressure = (((long)pressure_data_high << 16) | pressure_data_low) / 4;

    // display the temperature:
    Serial.println("\tPressure [Pa]=" + String(pressure));
  }
}

//Read from or write to register from the SCP1000:
unsigned int readRegister(byte thisRegister, int bytesToRead) {
  byte inByte = 0;           // incoming byte from the SPI
  unsigned int result = 0;   // result to return
  Serial.print(thisRegister, BIN);
```

```
  Serial.print("\t");
  // SCP1000 expects the register name in the upper 6 bits
  // of the byte. So shift the bits left by two bits:
  thisRegister = thisRegister << 2;
  // now combine the address and the command into one byte
  byte dataToSend = thisRegister & READ;
  Serial.println(thisRegister, BIN);
  // take the chip select low to select the device:
  digitalWrite(chipSelectPin, LOW);
  // send the device the register you want to read:
  SPI.transfer(dataToSend);
  // send a value of 0 to read the first byte returned:
  result = SPI.transfer(0x00);
  // decrement the number of bytes left to read:
  bytesToRead--;
  // if you still have another byte to read:
  if (bytesToRead > 0) {
    // shift the first byte left, then get the second byte:
    result = result << 8;
    inByte = SPI.transfer(0x00);
    // combine the byte you just got with the previous one:
    result = result | inByte;
    // decrement the number of bytes left to read:
    bytesToRead--;
  }
  // take the chip select high to de-select:
  digitalWrite(chipSelectPin, HIGH);
  // return the result:
  return (result);
}


//Sends a write command to SCP1000

void writeRegister(byte thisRegister, byte thisValue) {

  // SCP1000 expects the register address in the upper 6 bits
  // of the byte. So shift the bits left by two bits:
  thisRegister = thisRegister << 2;
  // now combine the register address and the command into one byte:
  byte dataToSend = thisRegister | WRITE;

  // take the chip select low to select the device:
  digitalWrite(chipSelectPin, LOW);

  SPI.transfer(dataToSend); //Send register location
  SPI.transfer(thisValue);  //Send value to record into register

  // take the chip select high to de-select:
  digitalWrite(chipSelectPin, HIGH);
}
```

```
/*
  Digital Pot Control

  This example controls an Analog Devices AD5206 digital potentiometer.
  The AD5206 has 6 potentiometer channels. Each channel's pins are labeled
  A - connect this to voltage
  W - this is the pot's wiper, which changes when you set it
  B - connect this to ground.

 The AD5206 is SPI-compatible,and to command it, you send two bytes,
 one with the channel number (0 - 5) and one with the resistance value for the
 channel (0 - 255).

 The circuit:
  * All A pins  of AD5206 connected to +5V
  * All B pins of AD5206 connected to ground
  * An LED and a 220-ohm resisor in series connected from each W pin to ground
  * CS - to digital pin 10  (SS pin)
  * SDI - to digital pin 11 (MOSI pin)
  * CLK - to digital pin 13 (SCK pin)

 created 10 Aug 2010
 by Tom Igoe

 Thanks to Heather Dewey-Hagborg for the original tutorial, 2005

*/


// inslude the SPI library:
#include <SPI.h>


// set pin 10 as the slave select for the digital pot:
const int slaveSelectPin = 10;

void setup() {
  // set the slaveSelectPin as an output:
  pinMode(slaveSelectPin, OUTPUT);
  // initialize SPI:
  SPI.begin();
}

void loop() {
  // go through the six channels of the digital pot:
  for (int channel = 0; channel < 6; channel++) {
    // change the resistance on this channel from min to max:
    for (int level = 0; level < 255; level++) {
      digitalPotWrite(channel, level);
      delay(10);
    }
    // wait a second at the top:
    delay(100);
```

```
    // change the resistance on this channel from max to min:
    for (int level = 0; level < 255; level++) {
      digitalPotWrite(channel, 255 - level);
      delay(10);
    }
  }

}

void digitalPotWrite(int address, int value) {
  // take the SS pin low to select the chip:
  digitalWrite(slaveSelectPin, LOW);
  //  send in the address and value via SPI:
  SPI.transfer(address);
  SPI.transfer(value);
  // take the SS pin high to de-select the chip:
  digitalWrite(slaveSelectPin, HIGH);
}
```

Result

Examples compile and upload successfully.

Messages

```
Sketch uses 6404 bytes (4%) of program storage space. Maximum is 131072 bytes.
Global variables use 376 bytes (2%) of dynamic memory, leaving 16008 bytes for
local variables. Maximum is 16384 bytes.


avrdude: Version 6.3-20201216
         Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
         Copyright (c) 2007-2014 Joerg Wunsch

         System wide configuration file is
"C:\Users\ivanFernandez\AppData\Local\Arduino15\packages\Microchip\hardware\megaav
r\1.0.0/avrdude.conf"

         Using Port                    : usb
         Using Programmer              : curiosity_updi
avrdude: Found CMSIS-DAP compliant device, using EDBG protocol
         AVR Part                      : AVR128DA48
         Chip Erase delay              : 0 us
         PAGEL                         : P00
         BS2                           : P00
         RESET disposition             : dedicated
         RETRY pulse                   : SCK
         serial program mode           : yes
         parallel program mode         : yes
         Timeout                       : 0
         StabDelay                     : 0
```

```
          CmdexeDelay                   : 0
          SyncLoops                     : 0
          ByteDelay                     : 0
          PollIndex                     : 0
          PollValue                     : 0x00
          Memory Detail                 :

                                  Block Poll               Page
Polled
            Memory Type Mode Delay Size  Indx Paged  Size   Size #Pages MinW  MaxW
ReadBack
            ----------- ---- ----- ----- ---- ------ ------ ---- ------ ----- -----
---------
            signature      0     0     0    0 no          3    0      0     0     0
0x00 0x00
            prodsig        0     0     0    0 no        125  125      0     0     0
0x00 0x00
            fuses          0     0     0    0 no          9   16      0     0     0
0x00 0x00
            fuse0          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            fuse1          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            fuse2          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            fuse4          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            fuse5          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            fuse6          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            fuse7          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            fuse8          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            lock           0     0     0    0 no          4    1      0     0     0
0x00 0x00
            data           0     0     0    0 no          0    0      0     0     0
0x00 0x00
            flash          0     0     0    0 no     131072  512      0     0     0
0x00 0x00
            eeprom         0     0     0    0 no        512   32      0     0     0
0x00 0x00

          Programmer Type : JTAGICE3_UPDI
          Description     : Microchip Curiosity in UPDI mode
          ICE hardware version: 0
          ICE firmware version: 1.17 (rel. 514)
          Serial number   : MCHP3280031800001901
          Vtarget         : 3.31 V
          JTAG clock megaAVR/program: 0 kHz
          JTAG clock megaAVR/debug:   0 kHz
          JTAG clock Xmega: 0 kHz
          PDI clock Xmega : 100 kHz
```

```
avrdude: Partial Family_ID returned: "    "
avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.01s

avrdude: Device signature = 0x1e9708 (probably avr128da48)
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed
         To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "0b11001001"
avrdude: writing fuse5 (1 bytes):

Writing | ################################################## | 100% 0.02s

avrdude: 1 bytes of fuse5 written
avrdude: verifying fuse5 memory against 0b11001001:
avrdude: load data fuse5 data from input file 0b11001001:
avrdude: input file 0b11001001 contains 1 bytes
avrdude: reading on-chip fuse5 data:

Reading | ################################################## | 100% 0.00s

avrdude: verifying ...
avrdude: 1 bytes of fuse5 verified
avrdude: reading input file "0x00"
avrdude: writing fuse7 (1 bytes):

Writing | ################################################## | 100% 0.02s

avrdude: 1 bytes of fuse7 written
avrdude: verifying fuse7 memory against 0x00:
avrdude: load data fuse7 data from input file 0x00:
avrdude: input file 0x00 contains 1 bytes
avrdude: reading on-chip fuse7 data:

Reading | ################################################## | 100% 0.00s

avrdude: verifying ...
avrdude: 1 bytes of fuse7 verified
avrdude: reading input file "0x00"
avrdude: writing fuse8 (1 bytes):

Writing | ################################################## | 100% 0.02s

avrdude: 1 bytes of fuse8 written
avrdude: verifying fuse8 memory against 0x00:
avrdude: load data fuse8 data from input file 0x00:
avrdude: input file 0x00 contains 1 bytes
avrdude: reading on-chip fuse8 data:

Reading | ################################################## | 100% 0.00s

avrdude: verifying ...
```

```
avrdude: 1 bytes of fuse8 verified
avrdude: reading input file
"C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/BarometricPressureSensor
.ino.hex"
avrdude: writing flash (6404 bytes):

Writing | ################################################## | 100% 0.79s

avrdude: 6404 bytes of flash written
avrdude: verifying flash memory against
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/BarometricPressureSensor.
ino.hex:
avrdude: load data flash data from input file
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/BarometricPressureSensor.
ino.hex:
avrdude: input file
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/BarometricPressureSensor.
ino.hex contains 6404 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 0.43s

avrdude: verifying ...
avrdude: 6404 bytes of flash verified

avrdude done.  Thank you.
```

```
Sketch uses 1604 bytes (1%) of program storage space. Maximum is 131072 bytes.
Global variables use 99 bytes (0%) of dynamic memory, leaving 16285 bytes for
local variables. Maximum is 16384 bytes.



avrdude: Version 6.3-20201216
         Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
         Copyright (c) 2007-2014 Joerg Wunsch

         System wide configuration file is
"C:\Users\ivanFernandez\AppData\Local\Arduino15\packages\Microchip\hardware\megaav
r\1.0.0/avrdude.conf"

         Using Port                    : usb
         Using Programmer              : curiosity_updi
avrdude: Found CMSIS-DAP compliant device, using EDBG protocol
         AVR Part                      : AVR128DA48
         Chip Erase delay              : 0 us
         PAGEL                         : P00
```

```
          BS2                           : P00
          RESET disposition             : dedicated
          RETRY pulse                   : SCK
          serial program mode           : yes
          parallel program mode         : yes
          Timeout                       : 0
          StabDelay                     : 0
          CmdexeDelay                   : 0
          SyncLoops                     : 0
          ByteDelay                     : 0
          PollIndex                     : 0
          PollValue                     : 0x00
          Memory Detail                 :

                                  Block Poll               Page
Polled
            Memory Type Mode Delay Size  Indx Paged  Size   Size #Pages MinW  MaxW
ReadBack
            ----------- ---- ----- ----- ---- ------ ------ ---- ------ ----- -----
---------
            signature      0     0     0    0 no          3    0      0     0     0
0x00 0x00
            prodsig        0     0     0    0 no        125  125      0     0     0
0x00 0x00
            fuses          0     0     0    0 no          9   16      0     0     0
0x00 0x00
            fuse0          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            fuse1          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            fuse2          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            fuse4          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            fuse5          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            fuse6          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            fuse7          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            fuse8          0     0     0    0 no          1    0      0     0     0
0x00 0x00
            lock           0     0     0    0 no          4    1      0     0     0
0x00 0x00
            data           0     0     0    0 no          0    0      0     0     0
0x00 0x00
            flash          0     0     0    0 no     131072  512      0     0     0
0x00 0x00
            eeprom         0     0     0    0 no        512   32      0     0     0
0x00 0x00

          Programmer Type : JTAGICE3_UPDI
          Description     : Microchip Curiosity in UPDI mode
          ICE hardware version: 0
```

```
          ICE firmware version: 1.17 (rel. 514)
          Serial number   : MCHP3280031800001901
          Vtarget         : 3.31 V
          JTAG clock megaAVR/program: 0 kHz
          JTAG clock megaAVR/debug:   0 kHz
          JTAG clock Xmega: 0 kHz
          PDI clock Xmega : 100 kHz

avrdude: Partial Family_ID returned: "    "
avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.01s

avrdude: Device signature = 0x1e9708 (probably avr128da48)
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed
         To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "0b11001001"
avrdude: writing fuse5 (1 bytes):

Writing | ################################################## | 100% 0.02s

avrdude: 1 bytes of fuse5 written
avrdude: verifying fuse5 memory against 0b11001001:
avrdude: load data fuse5 data from input file 0b11001001:
avrdude: input file 0b11001001 contains 1 bytes
avrdude: reading on-chip fuse5 data:

Reading | ################################################## | 100% 0.00s

avrdude: verifying ...
avrdude: 1 bytes of fuse5 verified
avrdude: reading input file "0x00"
avrdude: writing fuse7 (1 bytes):

Writing | ################################################## | 100% 0.02s

avrdude: 1 bytes of fuse7 written
avrdude: verifying fuse7 memory against 0x00:
avrdude: load data fuse7 data from input file 0x00:
avrdude: input file 0x00 contains 1 bytes
avrdude: reading on-chip fuse7 data:

Reading | ################################################## | 100% 0.00s

avrdude: verifying ...
avrdude: 1 bytes of fuse7 verified
avrdude: reading input file "0x00"
avrdude: writing fuse8 (1 bytes):

Writing | ################################################## | 100% 0.02s

avrdude: 1 bytes of fuse8 written
avrdude: verifying fuse8 memory against 0x00:
```

```
avrdude: load data fuse8 data from input file 0x00:
avrdude: input file 0x00 contains 1 bytes
avrdude: reading on-chip fuse8 data:

Reading | ################################################## | 100% 0.00s

avrdude: verifying ...
avrdude: 1 bytes of fuse8 verified
avrdude: reading input file
"C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/DigitalPotControl.ino.he
x"
avrdude: writing flash (1604 bytes):

Writing | ################################################## | 100% 0.79s

avrdude: 1604 bytes of flash written
avrdude: verifying flash memory against
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/DigitalPotControl.ino.hex
:
avrdude: load data flash data from input file
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/DigitalPotControl.ino.hex
:
avrdude: input file
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/DigitalPotControl.ino.hex
contains 1604 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 0.43s

avrdude: verifying ...
avrdude: 1604 bytes of flash verified

avrdude done.  Thank you.
```

Notes

1. Each of the sketches compiled and uploaded successfully to the AVR128DA48 board. This concludes testing of the SPI examples within the Team 25 core.