

## AN520 C-code example for MS56xx pressure sensors

### INTRODUCTION

This application note describes the communication between a microcontroller and MEAS Switzerland's MS56XX pressure sensor modules series using SPI and I2C protocol. In these code examples we explain: microcontroller initialization, commands construction and calculation of pressure and temperature from the obtained data. For more information on specific pressure and temperature algorithm please refer to the specific MS56xx sensor datasheet.

The code in these examples is developed for the ATMEL ATmega644p microcontroller. The compiler used is gcc from the WinAVR 20080610 bundle available freely on the Internet. All examples presented here can be implemented on any microcontroller with hardware implementation of the SPI and I2C protocol.

### SPI PROTOCOL C-CODE EXAMPLE

Figure below shows an easy way to use MS56xx sensors with the SPI bus. Please note a minimum of 100nF decoupling capacitor for the sensor and the PS pin (protocol select) connected to ground that defines the usage of the SPI protocol.

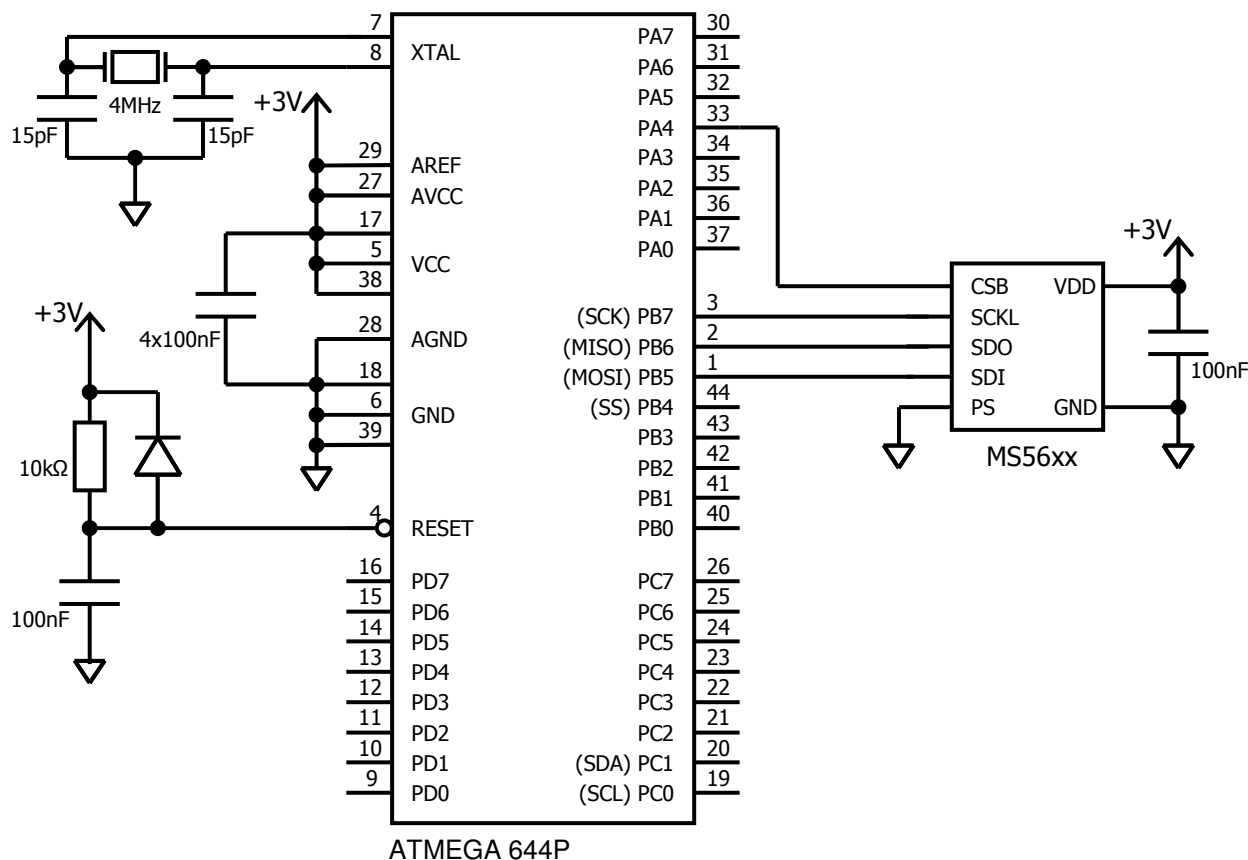


Figure 1: Circuit diagram of the hardware used for the SPI C-code testing

## AN520 C-code example for MS56xx pressure sensors

```

/*!
/*! @file an520_SPI.c,v
/*!
/*! Copyright (c) 2009 MEAS Switzerland
/*!
/*!
/*!
/*! @brief This C code is for starter reference only. It is written for the
/*! MEAS Switzerland MS56xx pressure sensor modules and Atmel Atmega644p
/*! microcontroller.
/*!
/*! @version 1.0 $Id: an520_SPI.c,v 1.0
/*!
/*! @todo

//_____ M A C R O S

#define TRUE 1
#define FALSE 0

#define F_CPU 4000000UL // 4 MHz XTAL

#define CMD_RESET      0x1E // ADC reset command
#define CMD_ADC_READ  0x00 // ADC read command
#define CMD_ADC_CONV  0x40 // ADC conversion command
#define CMD_ADC_D1    0x00 // ADC D1 conversion
#define CMD_ADC_D2    0x10 // ADC D2 conversion
#define CMD_ADC_256   0x00 // ADC OSR=256
#define CMD_ADC_512   0x02 // ADC OSR=512
#define CMD_ADC_1024  0x04 // ADC OSR=1024
#define CMD_ADC_2048  0x06 // ADC OSR=2056
#define CMD_ADC_4096  0x08 // ADC OSR=4096
#define CMD_PROM_RD   0xA0 // Prom read command

#define csb_hi() (_SFR_BYTE(PORTA) &= ~_BV(3)) // setting CSB low
#define csb_lo() (_SFR_BYTE(PORTA) |=  _BV(3)) // setting CSB high

//_____ I N C L U D E S

#include <stdio.h>
#include <util/delay.h>
#include <avr/io.h>
#include <math.h>

//_____ D E F I N I T I O N S

void spi_send(char cmd);
void cmd_reset(void);
unsigned long cmd_adc(char cmd);
unsigned int cmd_prom(char coef_num);
unsigned char crc4(unsigned int n_prom[]);

//*****
/*! @brief send 8 bit using SPI hardware interface
/*!
/*! @return 0
//*****
void spi_send(char cmd)
{
    SPDR= cmd; // put the byte in the SPI hardware buffer and start sending
    while (bit_is_clear(SPSR, 7)); // wait that the data is sent
}

```

## AN520 C-code example for MS56xx pressure sensors

```

//*****
//! @brief send reset sequence
//!
//! @return 0
//*****

void cmd_reset(void)
{
    csb_lo();           // pull CSB low to start the command
    spi_send(CMD_RESET); // send reset sequence
    _delay_ms(3);      // wait for the reset sequence timing
    csb_hi();          // pull CSB high to finish the command
}

//*****
//! @brief preform adc conversion
//!
//! @return 24bit result
//*****
unsigned long cmd_adc(char cmd)
{
    unsigned int ret;
    unsigned long temp=0;
    csb_lo();           // pull CSB low
    spi_send(CMD_ADC_CONV+cmd); // send conversion command
    switch (cmd & 0x0f) // wait necessary conversion time
    {
        case CMD_ADC_256 : _delay_us(900); break;
        case CMD_ADC_512 : _delay_ms(3);   break;
        case CMD_ADC_1024: _delay_ms(4);   break;
        case CMD_ADC_2048: _delay_ms(6);   break;
        case CMD_ADC_4096: _delay_ms(10);  break;
    }

    csb_hi();           // pull CSB high to finish the conversion
    csb_lo();           // pull CSB low to start new command
    spi_send(CMD_ADC_READ); // send ADC read command
    spi_send(0x00);      // send 0 to read 1st byte (MSB)
    ret=SPDR;
    temp=65536*ret;
    spi_send(0x00);      // send 0 to read 2nd byte
    ret=SPDR;
    temp=temp+256*ret;
    spi_send(0x00);      // send 0 to read 3rd byte (LSB)
    ret=SPDR;
    temp=temp+ret;
    csb_hi();           // pull CSB high to finish the read command
    return temp;
}

//*****
//! @brief Read calibration coefficients
//!
//! @return coefficient
//*****

```

## AN520 C-code example for MS56xx pressure sensors

```

unsigned int cmd_prom(char coef_num)
{
    unsigned int ret;
    unsigned int rC=0;

    csb_lo(); // pull CSB low
    spi_send(CMD_PROM_RD+coef_num*2); // send PROM READ command
    spi_send(0x00); // send 0 to read the MSB
    ret=SPDR;
    rC=256*ret;
    spi_send(0x00); // send 0 to read the LSB
    ret=SPDR;
    rC=rC+ret;
    csb_hi(); // pull CSB high
    return rC;
}

//*****
//! @brief calculate the CRC code for details look into AN521
//!
//! @return crc code
//*****

unsigned char crc4(unsigned int n_prom[])
{
    int cnt; // simple counter
    unsigned int n_rem; // crc reminder
    unsigned int crc_read; // original value of the crc
    unsigned char n_bit;
    n_rem = 0x00;
    crc_read=n_prom[7]; //save read CRC
    n_prom[7]=(0xFF00 & (n_prom[7])); //CRC byte is replaced by 0
    for (cnt = 0; cnt < 16; cnt++) // operation is performed on bytes
    {
        // choose LSB or MSB
        if (cnt%2==1) n_rem ^= (unsigned short) ((n_prom[cnt]>>1]) & 0x00FF);
        else n_rem ^= (unsigned short) (n_prom[cnt]>>1]>>8);
        for (n_bit = 8; n_bit > 0; n_bit--)
        {
            if (n_rem & (0x8000))
            {
                n_rem = (n_rem << 1) ^ 0x3000;
            }
            else
            {
                n_rem = (n_rem << 1);
            }
        }
    }
    n_rem= (0x000F & (n_rem >> 12)); // // final 4-bit reminder is CRC code
    n_prom[7]=crc_read; // restore the crc_read to its original place
    return (n_rem ^ 0x00);
}

//*****
//! @brief main program
//!
//! @return 0
//*****

```

## AN520 C-code example for MS56xx pressure sensors

```

int main (void)
{

    unsigned long D1;           // ADC value of the pressure conversion
    unsigned long D2;           // ADC value of the temperature conversion
    unsigned int C[8];          // calibration coefficients
    double P;                   // compensated pressure value
    double T;                   // compensated temperature value
    double dT;                  // difference between actual and measured temperature
    double OFF;                 // offset at actual temperature
    double SENS;               // sensitivity at actual temperature

    int i;
    unsigned char n_crc;        // crc value of the prom

    DDRA = 0xFE;                // prepare the port A
    DDRB = 0xBF;                // SDO input
    DDRC = 0x00;                // I2C pins as input
    DDRD = 0x82;                // prepare the port D, RX out and TX out;
    PORTA= 0x10;
    PORTD= 0x20;

    //SPI settings:master, mode 0, fosc/4
    SPCR=(1<<SPE)|(1<<MSTR);
    //alternative SPI settings: master, mode 3, fosc/4
    //SPCR=(1<<SPE)|(1<<MSTR)|(1<<CPOL)|(1<<CPHA);

    cmd_reset();                // reset the module after powerup
    for (i=0;i<8;i++){ C[i]=cmd_prom(i);} // read calibration coefficients
    n_crc=crc4(C);

    while(TRUE)                 // loop without stopping
    {

        D1=cmd_adc(CMD_ADC_D1+CMD_ADC_256); // read uncompensated pressure
        D2=cmd_adc(CMD_ADC_D2+CMD_ADC_4096); // read uncompensated temperature

        // calculate 1st order pressure and temperature (MS5607 1st order algorithm)
        dT=D2-C[5]*pow(2,8);
        OFF=C[2]*pow(2,17)+dT*C[4]/pow(2,6);
        SENS=C[1]*pow(2,16)+dT*C[3]/pow(2,7);

        T=(2000+(dT*C[6])/pow(2,23))/100;
        P=((D1*SENS)/pow(2,21)-OFF)/pow(2,15))/100;

        // place to use P, T, put them on LCD, send them trough RS232 interface...
    }

    return 0;
}

```

## AN520 C-code example for MS56xx pressure sensors

### I2C PROTOCOL C-CODE EXAMPLE

Figure below shows an easy way to use MS56xx sensors with the I2C bus. Please note a minimum of 100nF decoupling capacitor for the sensor and the PS pin (protocol select) connected to VDD that defines the usage of the I2C protocol. In this example the CSB is connected to ground which defines the I2C address to 0xEE.

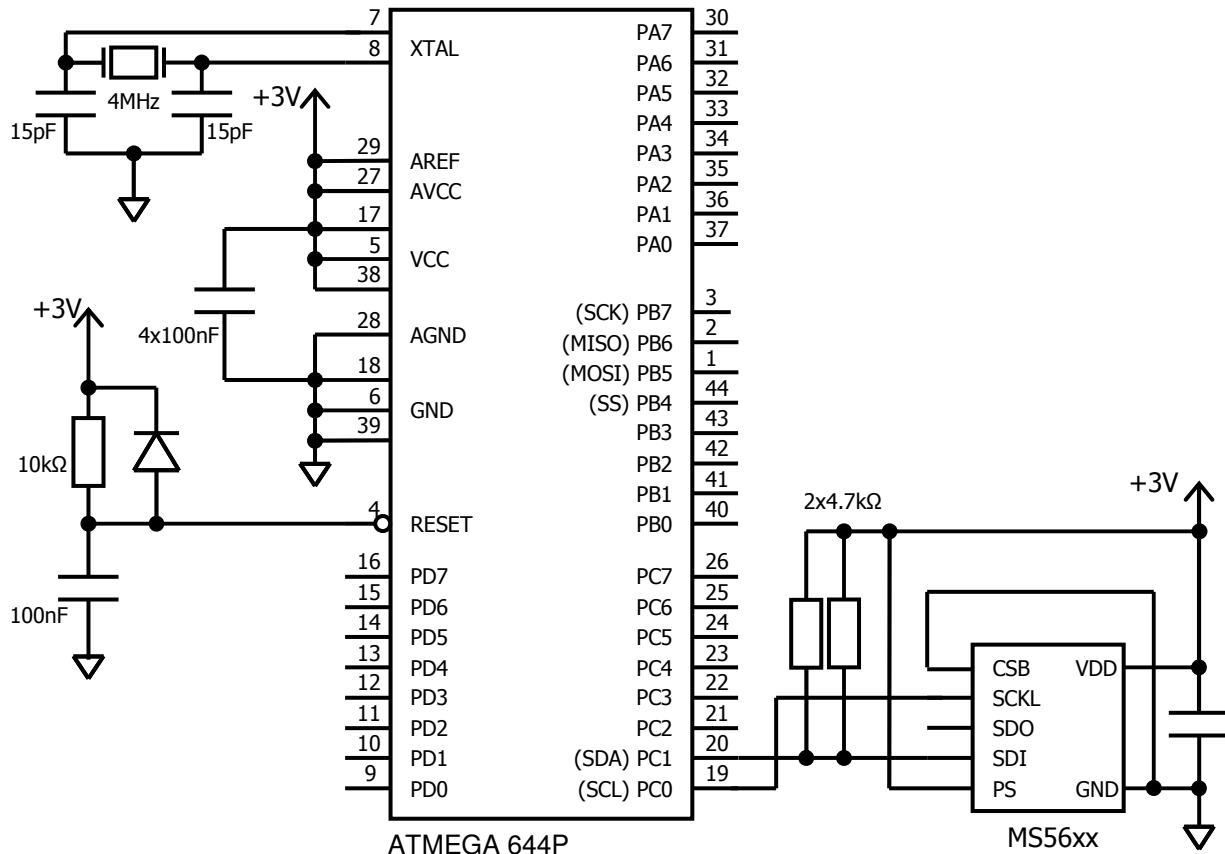


Figure 2: Circuit diagram of the hardware used for the I2C C-code testing

```

//!
//! @file an520_I2C.c,v
//!
//! Copyright (c) 2009 MEAS Switzerland
//!
//!
//! @brief This C code is for starter reference only. It is written for the
//! MEAS Switzerland MS56xx pressure sensor modules and Atmel Atmega644p
//! microcontroller.
//!
//! @version 1.0 $Id: an520_I2C.c,v 1.0
//!
//! @todo

```

## AN520 C-code example for MS56xx pressure sensors

```
//_____ M A C R O S

#define TRUE 1
#define FALSE 0

#define F_CPU      4000000UL    // 4 MHz external XTAL
#define SCL_CLOCK  100000L     // I2C clock in Hz
#define ADDR_W     0xEF        // Module address write mode
#define ADDR_R     0xEF        // Module address read mode

#define CMD_RESET  0x1E        // ADC reset command
#define CMD_ADC_READ 0x00      // ADC read command
#define CMD_ADC_CONV 0x40      // ADC conversion command
#define CMD_ADC_D1  0x00      // ADC D1 conversion
#define CMD_ADC_D2  0x10      // ADC D2 conversion
#define CMD_ADC_256 0x00      // ADC OSR=256
#define CMD_ADC_512 0x02      // ADC OSR=512
#define CMD_ADC_1024 0x04     // ADC OSR=1024
#define CMD_ADC_2048 0x06     // ADC OSR=2048
#define CMD_ADC_4096 0x08     // ADC OSR=4096
#define CMD_PROM_RD 0xA0      // Prom read command

//_____ I N C L U D E S

#include <stdio.h>
#include <util/delay.h>
#include <util/twi.h>
#include <avr/io.h>
#include <math.h>

//_____ D E F I N I T I O N S

unsigned char i2c_start(unsigned char address);
void i2c_stop(void);
unsigned char i2c_write( unsigned char data );
unsigned char i2c_readAck(void);
unsigned char i2c_readNak(void);
void cmd_reset(void);
unsigned long cmd_adc(char cmd);
unsigned int cmd_prom(char coef_num);
unsigned char crc4(unsigned int n_prom[]);

//*****
//! @brief send I2C start condition and the address byte
//!
//! @return 0
//*****
unsigned char i2c_start(unsigned char address)
{
    unsigned char    twst;

    TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN); // send START condition
    while(!(TWCR & (1<<TWINT))); // wait until transmission completed
    twst = TW_STATUS & 0xF8;// check value of TWI Status Register. Mask prescaler bits.
    if ( (twst != TW_START) && (twst != TW_REP_START)) return 1;
    TWDR = address; // send device address
    TWCR = (1<<TWINT) | (1<<TWEN);
    // wait until transmission completed and ACK/NACK has been received
    while(!(TWCR & (1<<TWINT)));
    twst = TW_STATUS & 0xF8;
}
```

## AN520 C-code example for MS56xx pressure sensors

```

// check value of TWI Status Register. Mask prescaler bits.
if ( (twst != TW_MT_SLA_ACK) && (twst != TW_MR_SLA_ACK) ) return 1;
return 0;
}

//*****
//! @brief send I2C stop condition
//!
//! @return none
//*****
void i2c_stop(void)
{
    /* send stop condition */
    TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);
    // wait until stop condition is executed and bus released
    while(TWCR & (1<<TWSTO)); //THIS MAKES PROBLEM FOR IS2402
}

//*****
//! @brief send I2C stop condition
//!
//! @return 0
//*****
unsigned char i2c_write(unsigned char data)
{
    unsigned char twst;
    TWDR = data; // send data to the previously addressed device
    TWCR = (1<<TWINT) | (1<<TWEN);
    while(!(TWCR & (1<<TWINT))); // wait until transmission completed
    twst = TW_STATUS & 0xF8; // check value of TWI Status Register. Mask prescaler bits
    if( twst != TW_MT_DATA_ACK) return 1;
    return 0;
}

//*****
//! @brief read I2C byte with acknowledgment
//!
//! @return read byte
//*****
unsigned char i2c_readAck(void)
{
    TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);
    while(!(TWCR & (1<<TWINT)));
    return TWDR;
}

//*****
//! @brief read I2C byte without acknowledgment
//!
//! @return read byte
//*****
unsigned char i2c_readNak(void)
{
    TWCR = (1<<TWINT) | (1<<TWEN);
    while(!(TWCR & (1<<TWINT)));
    return TWDR;
}

//*****
//! @brief send command using I2C hardware interface
//!

```



## AN520 C-code example for MS56xx pressure sensors

```

//! @return none
//*****
void i2c_send(char cmd)
{
    unsigned char ret;

    ret = i2c_start(ADDR_W); // set device address and write mode
    if ( ret )
        {//failed to issue start condition, possibly no device found */
            i2c_stop();
        }
    else
        {// issuing start condition ok, device accessible
            ret=i2c_write(cmd);
            i2c_stop();
        }
}

//*****
//! @brief send reset sequence
//!
//! @return none
//*****
void cmd_reset(void)
{
    i2c_send(CMD_RESET); // send reset sequence
    _delay_ms(3); // wait for the reset sequence timing
}

//*****
//! @brief preform adc conversion
//!
//! @return 24bit result
//*****
unsigned long cmd_adc(char cmd)
{
    unsigned int ret;
    unsigned long temp=0;

    i2c_send(CMD_ADC_CONV+cmd); // send conversion command
    switch (cmd & 0x0f) // wait necessary conversion time
    {
        case CMD_ADC_256 : _delay_us(900); break;
        case CMD_ADC_512 : _delay_ms(3); break;
        case CMD_ADC_1024: _delay_ms(4); break;
        case CMD_ADC_2048: _delay_ms(6); break;
        case CMD_ADC_4096: _delay_ms(10); break;
    }
    i2c_send(CMD_ADC_READ);
    ret = i2c_start(ADDR_R); // set device address and read mode
    if ( ret )
        {//failed to issue start condition, possibly no device found
            i2c_stop();
        }
    else
        {//issuing start condition ok, device accessible
            ret = i2c_readAck(); // read MSB and acknowledge
            temp=65536*ret;
            ret = i2c_readAck(); // read byte and acknowledge
            temp=temp+256*ret;
            ret = i2c_readNak(); // read LSB and not acknowledge
            temp=temp+ret;
            i2c_stop(); // send stop condition
        }
}

```

## AN520 C-code example for MS56xx pressure sensors

```

    return temp;
}

//*****************************************************************************
//! @brief Read calibration coefficients
//!
//! @return coefficient
//*****************************************************************************
unsigned int cmd_prom(char coef_num)
{
    unsigned int ret;
    unsigned int rC=0;

    i2c_send(CMD_PROM_RD+coef_num*2);        // send PROM READ command
    ret = i2c_start(ADDR_R);                 // set device address and read mode
    if ( ret )
        {///failed to issue start condition, possibly no device found
            i2c_stop();
        }
    else
        {///issuing start condition ok, device accessible
            ret = i2c_readAck();               // read MSB and acknowledge
            rC=256*ret;
            ret = i2c_readNak();              // read LSB and not acknowledge
            rC=rC+ret;
            i2c_stop();
        }
    return rC;
}

//*****************************************************************************
//! @brief calculate the CRC code
//!
//! @return crc code
//*****************************************************************************
unsigned char crc4(unsigned int n_prom[])
{
    int cnt;                                // simple counter
    unsigned int n_rem;                     // crc reminder
    unsigned int crc_read;                  // original value of the crc
    unsigned char n_bit;
    n_rem = 0x00;
    crc_read=n_prom[7];                     //save read CRC
    n_prom[7]=(0xFF00 & (n_prom[7]));       //CRC byte is replaced by 0
    for (cnt = 0; cnt < 16; cnt++)          // operation is performed on bytes
        {/// choose LSB or MSB
            if (cnt%2==1) n_rem ^= (unsigned short) ((n_prom[cnt]>>1]) & 0x00FF);
            else n_rem ^= (unsigned short) (n_prom[cnt]>>1]>>8);
            for (n_bit = 8; n_bit > 0; n_bit--)
                {
                    if (n_rem & (0x8000))
                        {
                            n_rem = (n_rem << 1) ^ 0x3000;
                        }
                    else
                        {
                            n_rem = (n_rem << 1);
                        }
                }
        }
    n_rem= (0x000F & (n_rem >> 12));        // final 4-bit reminder is CRC code
}

```

## AN520 C-code example for MS56xx pressure sensors

```

n_prom[7]=crc_read;                // restore the crc_read to its original place
return (n_rem ^ 0x0);
}

//*****
//! @brief main program
//!
//! @return 0
//*****
int main (void)
{
    unsigned long D1;                // ADC value of the pressure conversion
    unsigned long D2;                // ADC value of the temperature conversion
    unsigned int C[8];               // calibration coefficients
    double P;                        // compensated pressure value
    double T;                        // compensated temperature value
    double dT;                       // difference between actual and measured temperature
    double OFF;                      // offset at actual temperature
    double SENS;                    // sensitivity at actual temperature
    int i;
    unsigned char n_crc;             // crc value of the prom

    // setup the ports
    DDRA = 0xFE;
    DDRB = 0x0F;                    //SPI pins as input
    DDRC = 0x03;                    // I2C pins as output
    DDRD = 0x82;                    // RS out and tx out;

    PORTA = 0x1F;                   // I2C pin high
    PORTB = 0xF0;
    PORTC = 0x01;
    PORTD = 0x00;

    // initialize the I2C hardware module
    TWSR = 0;                       // no prescaler
    TWBR = ((F_CPU/SCL_CLOCK)-16)/2; // set the I2C speed

    D1=0;
    D2=0;
    cmd_reset();                    // reset IC
    for (i=0;i<8;i++){ C[i]=cmd_prom(i);} // read coefficients
    n_crc=crc4(C);                  // calculate the CRC
    for(;;)                          // loop without stopping
    {
        D2=cmd_adc(CMD_ADC_D2+CMD_ADC_4096); // read D2
        D1=cmd_adc(CMD_ADC_D1+CMD_ADC_4096); // read D1

        // calculate 1st order pressure and temperature (MS5607 1st order algorithm)
        dT=D2-C[5]*pow(2,8);
        OFF=C[2]*pow(2,17)+dT*C[4]/pow(2,6);
        SENS=C[1]*pow(2,16)+dT*C[3]/pow(2,7);

        T=(2000+(dT*C[6])/pow(2,23))/100;
        P=((D1*SENS)/pow(2,21)-OFF)/pow(2,15))/100;

        // place to use P, T, put them on LCD, send them trough RS232 interface...
    }

    return 0;
}

```

## AN520 C-code example for MS56xx pressure sensors

### CRC CODE NOTES

The CRC code is calculated and written in factory with the LSB byte in the prom `n_prom[7]` set to 0x00 (see Coefficient table below). It is thus important to clear those bytes from the calculation buffer before proceeding with the CRC calculation itself:

```
n_prom[7]=(0xFF00 & (n_prom[7]));          //CRC byte is replaced by 0
```

As a simple test of the CRC code, the following coefficient table could be used:

```
unsigned int nprom[] = {0x3132,0x3334,0x3536,0x3738,0x3940,0x4142,0x4344,0x4500};
```

the resulting calculated CRC should be 0xB.

A d	D B 5	D B 4	D B 3	D B 2	D B 1	D B 0	D B 9	D B 8	D B 7	D B 6	D B 5	D B 4	D B 3	D B 2	D B 1	D B 0
0	16 bit reserved for manufacturer															
1	Coefficient 1 (16 bit unsigned)															
2	Coefficient 2 (16 bit unsigned)															
3	Coefficient 3 (16 bit unsigned)															
4	Coefficient 4 (16 bit unsigned)															
5	Coefficient 5 (16 bit unsigned)															
6	Coefficient 6 (16 bit unsigned)															
7										0	0	0	0			CRC (0x0)

Table 1: Memory PROM mapping

## AN520 C-code example for MS56xx pressure sensors

### REVISION HISTORY

Date	Revision	Type of changes
18.06.2009	00	Initial document
12.01.2010	01	Change to MEAS logo and layout; addition of CRC code notes

### FACTORY CONTACTS

#### NORTH AMERICA

Measurement Specialties, Inc.  
1000 Lucas Way  
Hampton, VA 23666  
Tel: +1 800 555 1551  
Fax: +1 757 766 4297  
e-mail: [sales@meas-spec.com](mailto:sales@meas-spec.com)  
Website: [www.meas-spec.com](http://www.meas-spec.com)

#### EUROPE

MEAS Switzerland SA  
Ch. Chapons-des-Prés 11  
CH-2022 Bevaix  
Tel: +41 32 847 9550  
Fax: + 41 32 847 9569  
e-mail: [sales@intersema.ch](mailto:sales@intersema.ch)  
Website: [www.intersema.ch](http://www.intersema.ch)

#### ASIA

Measurement Specialties (China), Ltd.  
No. 26 Langshan Road  
Shenzhen High-Tech Park (North)  
Nanshan District, Shenzhen, 518107  
China  
Tel: +86 755 3330 5088  
Fax: +86 755 3330 5099  
e-mail: [pfc.cs.asia@meas-spec.com](mailto:pfc.cs.asia@meas-spec.com)  
Website: [www.meas-spec.com](http://www.meas-spec.com)

The information in this sheet has been carefully reviewed and is believed to be accurate; however, no responsibility is assumed for inaccuracies. Furthermore, this information does not convey to the purchaser of such devices any license under the patent rights to the manufacturer. Measurement Specialties, Inc. reserves the right to make changes without further notice to any product herein. Measurement Specialties, Inc. makes no warranty, representation or guarantee regarding the suitability of its product for any particular purpose, nor does Measurement Specialties, Inc. assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Typical parameters can and do vary in different applications. All operating parameters must be validated for each customer application by customer's technical experts. Measurement Specialties, Inc. does not convey any license under its patent rights nor the rights of others.