MSP430F4270 Altimeter Demo

Andreas Dannenberg *MSP430*

ABSTRACT

This application report describes the implementation of an altimeter using a fully-integrated MCU solution with the MSP430F42x0 family of microcontrollers. An air pressure sensor is directly interfaced with the MCU-integrated 16-bit sigma/delta data converter and used for altitude determination. Other implemented functions are air pressure measurement, temperature measurement and real-time clock (RTC).

Background

The air pressure decreases by about 1mb each 8m increase in altitude at altitudes around sea level. This pressure change is caused by the weight of the atmosphere itself and can be used to determine the altitude. Assuming a sea-level pressure of p_0 = 1013.25mb and a constant air temperature of 0°C, the following formula can be used to model this dependency for altitudes of up to 100km:

$$p_h = p_0 \times e^{-\frac{h}{7990m}} \qquad \text{(Equation 1)}$$

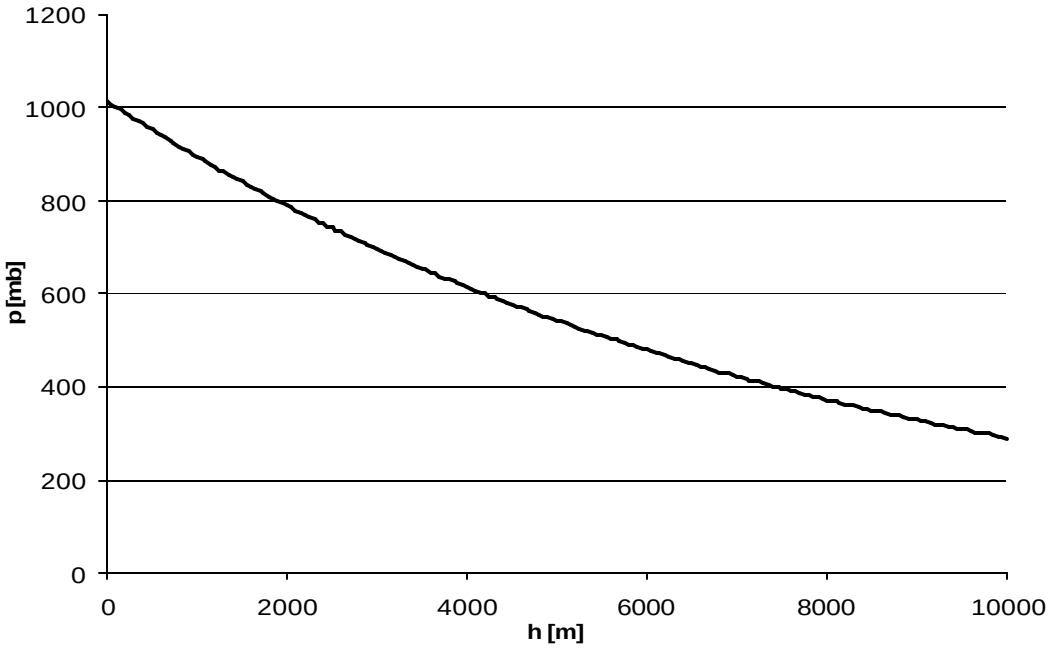


Figure 1. Air Pressure vs. Altitude

Solving this equation for h gives the altitude in meters as a function of the air pressure p_h and allows calculating the absolute altitude based on the difference in air pressure.

$$h = 18{,}400m \times \lg \frac{p_0}{p_h} \quad \text{(Equation 2)}$$

However, this method is not free of problems. With changing air pressure due to temperature changes and changes in the weather pattern, it is not unusual to see pressure changes in the lower single-digit millibar range, which could be interpreted as altitude changes in the range of several 10s of meters. For increased accuracy, a barometric altimeter typically needs to get calibrated several times a day, especially in changing weather conditions.

MSP430F4270 Solution

This demo application uses the MPXM2102 absolute pressure sensor. It provides a differential output voltage that is linear and directly proportional to the ambient pressure. The full-scale differential output voltage is 40mV with an air pressure of 1000mb using a sensor excitation voltage of 10V. Also, the output voltage decreases as air pressure decreases by 40µV/mb. In this application, the sensor is powered directly from a 3V battery source. As the sensor output voltage is ratio metric to its supply voltage, the new differential full-scale output voltage can be calculated as 3V / 10V x 40mV = 12mV. With 3V sensor excitation voltage, the output voltage decreases as air pressure decreases by 3V / 10V x 40µV/mb = 12µV/mb. For more information, refer to the device specific data sheet [1].

The design goal for this demo application is to provide an altitude display resolution of 1m for altitudes around sea level. For this, the pressure change for going from 0m to 1m can be calculated as:

$$\Delta p = 1013.25mb \times e^{0} - 1013.25mb \times e^{-\frac{1m}{7990m}} = 0.127mb \quad \text{(Equation 3)}$$

To resolve this pressure difference, an ADC needs to be able to resolve at least 0.127mb / 2 = 0.0635mb. This equals a required maximum LSB step size of 12µV/mb x 0.0635mb = 0.762µV.

In the given demo, the MSP430F42x0 microcontroller integrated SD16_A 16-bit sigma/delta ADC is used. Its electrical characteristics can be found in the MSP430F42x0 device data sheet [2], the module operation is described in the MSP430F4xx Family User's Guide [3]. Here, the SD16_A module is supplied with an external reference voltage of 1.27V at 3V supply voltage. Using the maximum gain setting of 32 results in a typical gain of 28.35. With a total of 16 bits (15 bits and one sign bit) from the SD16_A decimation filter output, this leads to an SD16_A LSB step voltage of:

$$V_{LSB} = \frac{V_{REF} / 2}{GAIN_{PGA} \times 2^{NrBits}} = \frac{1.269V / 2}{28.35 \times 2^{15}} = 0.683mV \quad \text{(Equation 4)}$$

This LSB step size voltage reflects an equivalent pressure change of 0.683µV / (12µV/mb) = 0.0569mb and is sufficient to fulfill the LSB step size requirement for altitudes from see level to up to approximately 6,000m. At 6,000m, 1m altitude difference results in a pressure difference of 0.0598mb. Note that this is a theoretic calculation assuming that all 15 bits can be used. In practice when using an SD16_A gain setting of 32, the effective number of bits output by the converter is less than this due to additional noise. Always refer to the device data sheet for SD16_A performance parameters [2]. In this demo application, averaging techniques are used to filter out noise and stabilize the conversion result that is used for the math calculations.

Using the calculated SD16_A LSB step size voltage and the MPXM2102 pressure to voltage transfer coefficient, the actual air pressure can be calculated as follows:

$$p = SD16\,\mathrm{Re}\,sult \times 0.683\frac{mV}{LSB} \times 0.08\overline{3}\frac{mb}{mV} \qquad \text{(Equation 5)}$$

Equation 5 is used in the altimeter firmware to calculate and display the current air pressure in millibar. Also, when combining equations 2 and 5, the following equation for altitude calculation can be obtained:

$$h = 18{,}400m \times \lg \frac{1013.25mb}{SD16\,\mathrm{Re}\,sult \times 0.683\frac{mV}{LSB} \times 0.08\overline{3}\frac{mb}{mV}} \qquad \text{(Equation 6)}$$

This equation is used in the altimeter demo firmware and gives the altitude in m for a given SD16_A output value.

Another function of this demo application is to display the ambient temperature. For this, the SD16_A integrated temperature sensor is used. Per device datasheet, it has a temperature transfer coefficient of 1.32mV/K (typ.). With the SD16_A operated in 16-bit mode and using the internal reference voltage of 1.20V (typ), the ambient temperature in ºC can be calculated according to equation 7:

$$T = SD16\,\mathrm{Re}\,sult \times 0.683\frac{mV}{LSB} \times \frac{1}{1.32}\frac{K}{mV} - 273K \qquad \text{(Equation 7)}$$

Hardware Implementation

To achieve a single-chip implementation, an integrated MCU solution is used. The MSP430F42x0 series of ultra-low power, Flash-based microcontrollers comes with an integrated 16-bit sigma-delta analog-to-digital converter (SD16_A). This data converter also features an on-chip programmable gain amplifier (PGA) which allows the amplification of incoming signals up to 32 times. Here, the MSP430F4270 was used, but other MSP430F42x0 family members can be used as well. Appendix A shows the schematic of the demo board that was used to develop this application.

To enable low-power standby operation, the used pressure transducer MPXM2102 is excited through MSP430 port pins P6.6 and P6.7 and can be disabled when not in use. With a supply voltage of 3V, the sensor is consuming about 1.8mA of current while excited. The pressure transducer differential output signal is connected directly to SD16_A channel A0 via P6.0 and P6.1.

The MSP430 SD16_A 16-bit sigma-delta ADC can operate with either using a built-in reference of 1.2V, or an externally connected reference voltage as used in this application example. Here, an external resistor divider is used to provide the reference voltage. With the pressure transducer powered from the same voltage source, this has the advantage of implementing a V_{CC}-independent, ratio metric measurement principle. If the pressure transducer would be powered from V_{CC}, and the internal voltage reference for the SD16_A module would be used, the measurement results would change as V_{CC} changes over the lifetime of the battery. Using the resistor divider as shown in Appendix A and a supply voltage of 3V, the reference voltage can be calculated as follows:

$$V_{REF} = Vcc \times \frac{R7}{R7 + R11} = 3V \times \frac{11K}{11K + 15K} = 1.269V$$

The R7/R11 divider ratio was chosen such that the generated reference voltage stays in the allowed $V_{REF(I)}$ range while V_{CC} is dropping from 3.0V to 2.5V. 2.5V is minimum supply voltage for the SD16_A module and for in-system Flash programming. For detailed SD16_A voltage ranges and other parameters, please refer to the MSP430F42x0 Data Sheet [2]. The capacitor C5 is used to stabilize either off-chip or on-chip reference voltage. The on-chip reference voltage is used for temperature measurement. In this case, the reference buffer is enabled to compensate for the load on the V_{REF} pin caused by R7 and R11.

The MSP430F4270 on-chip LCD driver enables direct interfacing to common LCD modules. In this application, the SBLCDA4, a 4-mux 7.1 digit LCD from SoftBaugh is used (http://www.softbaugh.com). The different voltages levels for driving the 4-mux displays are generated internally by the LCD_A module. Also, a capacitor is provided on pin 30 to enable the operation of the LCD_A charge pump. The use of the charge pump feature is optional, but it enables a higher LCD contrast by generating a V_{CC} -independent, user-selectable LCD drive voltage.

A 32.768kHz watch crystal is used as the system clock reference, to drive the LCD display, and to provide periodic wake-up from low-power modes during application operation. Furthermore, two push buttons (SW1 and SW2) are provided for the demo application operation and connected to port pins P1.0 and P1.1.

Software Implementation

The entire MSP430F4270 altimeter demo application is contained in a single C file and is provided in versions for use with TI Code Composer Essentials (file name: F4270_Altimeter_SB_CCE.c) and IAR Embedded Workbench for MSP430 (file name: F4270_Altimeter_SB_IAR.c).

After power-on reset, the MSP430 peripherals are initialized through calling Init_Sys(). This includes disabling the watchdog timer, configuring the LFXT1 oscillator load caps for the external watch crystal, initializing the LCD_A controller, the Basic Timer, as well as the I/O pins. Then, the calibration constants used for altitude, air pressure and temperature as well as the LCD_A contrast voltage setting and a miscellaneous flag register is loaded from Flash memory. Finally, interrupts are enabled and LPM3 is entered. From now on, the entire program flow is interrupt driven. The flow of the main() routine can be seen in Figure 2.

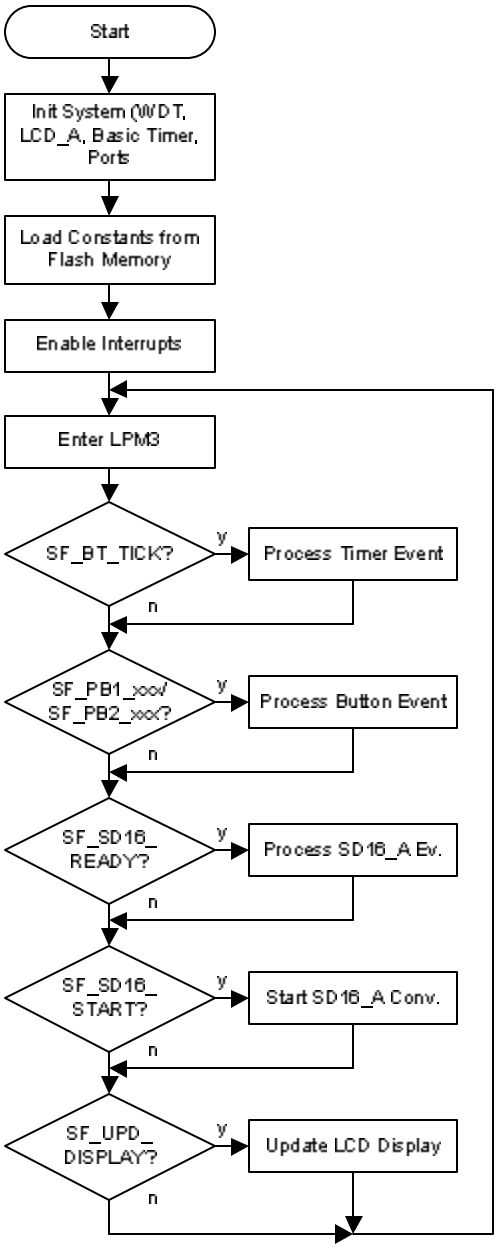


Figure 2. Flowchart main()

The Basic Timer interrupt service routine (ISR), the Watchdog timer ISR, and the SD16_A ISR can wake up the device from LPM3 and trigger event processing inside the main() event handler loop. The variable StatusFlags is used as a flag register to keep track of events to be processed. After one loop through the event handler, LPM3 is re-entered, and the device is waiting for another wakeup caused by an ISR.

The Basic Timer ISR (Figure 3) BT_ISR() is executed every second and wakes up the MCU from LPM3. If the main event handler detects that the flag SF_BT_TICK was set, the real-time clock (RTC) counter variables are advanced. When the software is operated in temperature measurement mode, a new SD16_A conversion sequence is requested by setting the flag SF_SD16_START. Also, when in altitude or air pressure measurement mode, the RTC display mode is re-enabled after the time counter has reached MODE_TIMEOUT by setting the program mode variable ProgramMode back to PM_DISPLAY_TIME. This is to save current consumption should the user leave the demo application running in either mode. Furthermore, in the case of time display, the flag SF_UPD_DISPLAY is set to request an LCD display update.

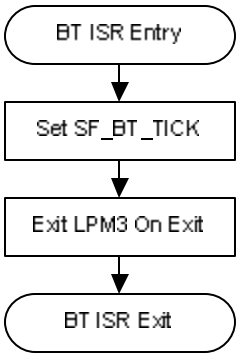


Figure 3. Flowchart Basic Timer ISR

Push-button events of SW1 and SW2 are handled by two ISRs, the PORT1_ISR() and the WDT_ISR(). The flowchart of these ISRs is shown in Figure 4. Only an initial push-button press will result in the execution of the port 1 ISR. After that, WDT_ISR() is called periodically every 16ms and used to poll the status of both buttons. For each of the buttons, there are status flags that indicate that a button was pressed (SF_PBx_PRESSED), is currently being held down (SF_PBx_DOWN), and got released (SF_PBx_RELEASED). Also, counters (PB1DownCtr and PB2DownCtr) are maintained that keeps track of how long each button was pressed. Every time WDT_ISR() is executed while a button is held down, the device wakes up from LPM3 and processes the button flags inside the main() event handler.

The main() event handler first checks if both buttons were held down for at least one second. In this case, the calibration mode (in the case of PM_MEASURE_A, PM_MEASURE_P, and PM_MEASURE_TEMP) or the set mode (in the case of PM_DISPLAY_TIME and PM_DISPLAY_CONTR) is entered. The LCD display is updated showing that the code now operates in calibration/set mode and the displayed value can be adjusted using SW1 and SW2. A one-point calibration method is used in altitude, air pressure, and temperature measurement modes. For this, an adjustable integer offset is added to the final result (CalConstA, CalConstP, CalConstT) it gets displayed. While in calibration/set mode, holding down both buttons for another second will store the calibration/set values into Flash memory using in-system programming techniques by calling StoreCalInFlash().

If button SW1 was pressed for less than 1s, the program mode is switched by cycling the variable ProgramMode from PM_MEASURE_A to PM_MEASURE_P to PM_DISPLAY_TIME to PM_MEASURE_TEMP to PM_DISPLAY_CONTR back to PM_MEASURE_A.

In the case a program mode was activated that involves the use of the SD16_A module, the function InitConversion() is called that sets up the data converter and the flag SF_SD16_START is set. The SD16_A is configured to use the input channel pair A0 and amplify the incoming signal with a gain of 32 using the module internal PGA when used for altitude and air pressure measurements. In temperature measurement mode, input channel pair A6 is used (connected to the internal temperature sensor) with a gain setting of 1. The converter is clocked by ACLK with a frequency of f_{MOD} = 32.768kHz and the continuous conversion mode is enabled. To obtain a more stable reading, 64 samples are averaged while in altitude or air pressure measurement mode to reduce noise. With an oversampling ratio of OSR = 1024, this process takes 64 x 1024 / 32.768kHz = 2s for one measurement. In temperature measurement mode, only one sample is taken. Detailed information on the SD16_A operation can be found in the MSP430x4xx Family User's Guide [3].

If a press of either button of less than 1s is detected while in calibration/set mode, the displayed value is incremented or decremented. Also, if either button is pressed for more than 1s while in calibration/set mode, the displayed value is incremented or decremented in larger steps. This feature is useful to "fast forward" should there be a larger difference between the actual and the displayed value, e.g., when setting the RTC.

If button SW2 is pressed for more than 1s while in temperature measurement mode (PM_MEASURE_TEMP), the unit used for temperature display is switched between °F and °C. A flag is used to store the current setting and is also programmed to Flash memory.

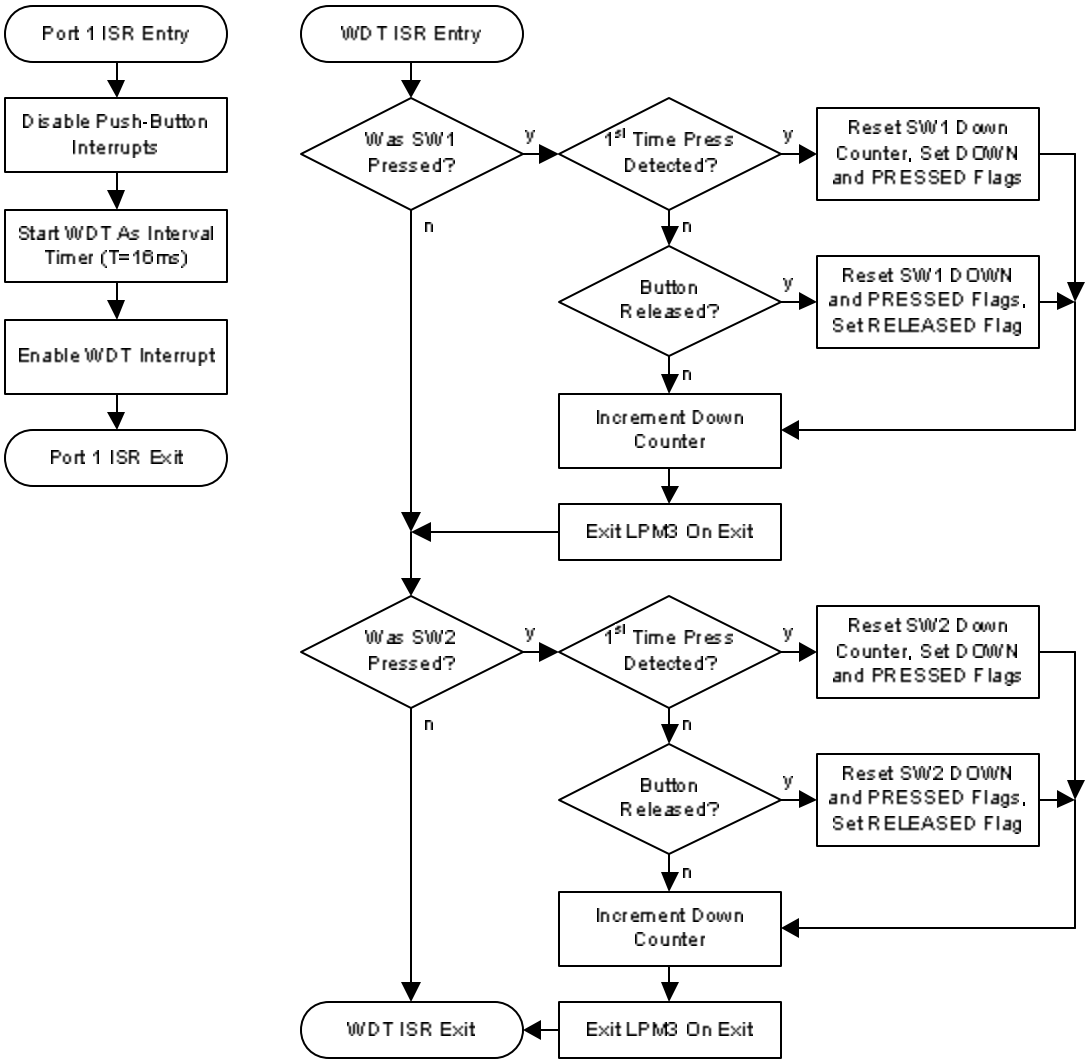


Figure 4. Flowchart Port 1 and WDT ISRs

The status flag SF_SD16_READY indicates that the current SD16_A conversion sequence is finished and the temporary result can be read out of SD16Temp. If the main event handler sees that this flag is set, it transfers the result to SD16Result and requests a display update by setting the flag SF_UPD_DISPLAY. Also, when in altitude or air pressure measurement mode, a new conversion sequence is requested immediately through setting of SF_SD16_START.

If the status flag SF_SD16_START is set, the main event handler starts a new SD16_A conversion sequence by calling StartNextConversion(). Depending on the current program mode, this function initializes conversion related variables and starts the SD16_A module by setting the SD16SC bit. If active, the SD16_A ISR is called upon each converted sample (Figure 5). The sample values are summed up in the variable SD16Temp for averaging. If SD16NrConversions samples were taken, the SD16_A is disabled and the ISR wakes up the MSP430 from LPM3. The new result is indicated by the flag SF_SD16_READY being set and can now be processed by the main event handler.

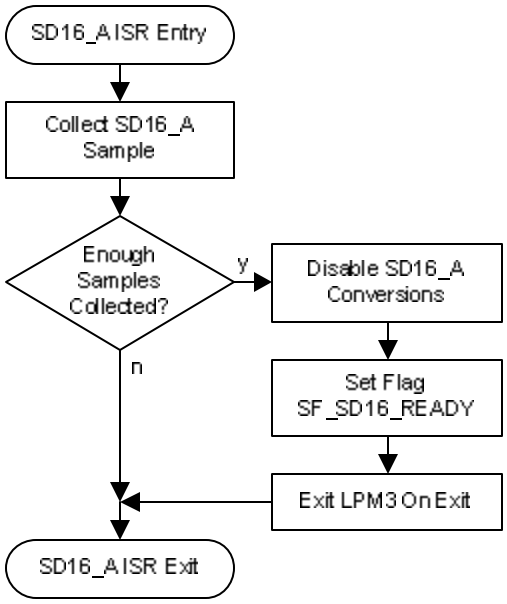


Figure 5. Flowchart SD16_A ISR

The last thing that is checked by the main event handler is the SF_UPD_DISPLAY status flag. If this flag is set, first the current LCD_A charge pump setting that is kept in the variable DispContr is programmed into the LCD_A module. This value can range from 0 (charge pump is off) to 15 (maximum charge pump voltage – maximum contrast). Then, depending on the current program mode, the LCD display is updated. This section of code also performs the necessary calculations to obtain altitude, air pressure and temperature values out of the raw SD16_A conversion result. The used formulas are described in the section “MSP430F4270 Solution”.

Demo Application Operation

On power-up, the firmware starts in RTC mode with the initial time of 12:00:00 (note that no colons are displayed). Pushing button SW1 cycles through the following operating modes in the order as listed here:

1.) RTC (displaying hours, minutes, and seconds)

2.) Temperature measurement (display shows temperature in either ‘°F’ or ‘°C’)

3.) LCD contrast control by adjusting the LCD_A charge pump voltage (display reads ‘CP’ and the current charge pump setting from OFF, 1, 2, 3, 4, …, 15)

4.) Altitude measurement (display reads ‘A’ and the altitude in meter)

5.) Air pressure measurement (display reads ‘P’ and air pressure in millibar)

In RTC as well as in temperature, altitude, and air pressure measurement modes, the displayed value can be set/calibrated by entering the set/calibration mode through simultaneously holding down SW1 and SW2 for at least 2 seconds. The displayed value can now be incremented by pressing SW2, and decremented by pressing SW1. Holding down either button will be treated as a repeated button press. By simultaneously holding down SW1 and SW2 for at least 2 seconds again, the calibrated value will be stored into Flash memory (expect in RTC set mode). Note that altitude, air pressure and temperature need to be calibrated based on the current altitude, weather conditions and temperature conditions after programming MSP430. Also, the altitude and air pressure values need to be recalibrated from time to time as outlined in the section "Background".

While in RTC or temperature measurement mode, the average board power consumption is around 3µA with the LCD_A charge pump disabled (set to "CP OFF") and the MSP430 operating in LPM3 most of the time. In this mode, the 32.768kHz oscillator is running, the LCD display enabled and the Basic Timer is running providing a 1s wakeup interval for RTC update and to obtain a new temperature reading. To reduce current consumption even further, the LCD display driver can be disabled by pressing SW2. The current consumption of the entire board now drops to about 1µA. Furthermore, the temperature display can be switched between °F and °C by holding down button SW2 while in temperature measurement mode. This setting gets stored into the device Flash memory.

While in altitude and air pressure measurement mode, the average board power consumption is around 2.8mA. This current draw results from the current consumed by the pressure sensor as well as the external reference voltage divider (about 2mA) and the SD16_A module that is operated continuously (about 0.8mA). Note that the MSP430 itself is operated in LPM3 and the CPU wakes up on demand only to process SD16_A results and update the display. To prevent draining the batteries, the software switches back to RTC mode when in altitude or air pressure measurement mode for more than 2 minutes.

Conclusion

This demo application shows a possible single-chip altimeter implementation using a fully-integrated MSP430F4270 microcontroller. If increased accuracy and stability is desired, the following improvements should be considered:

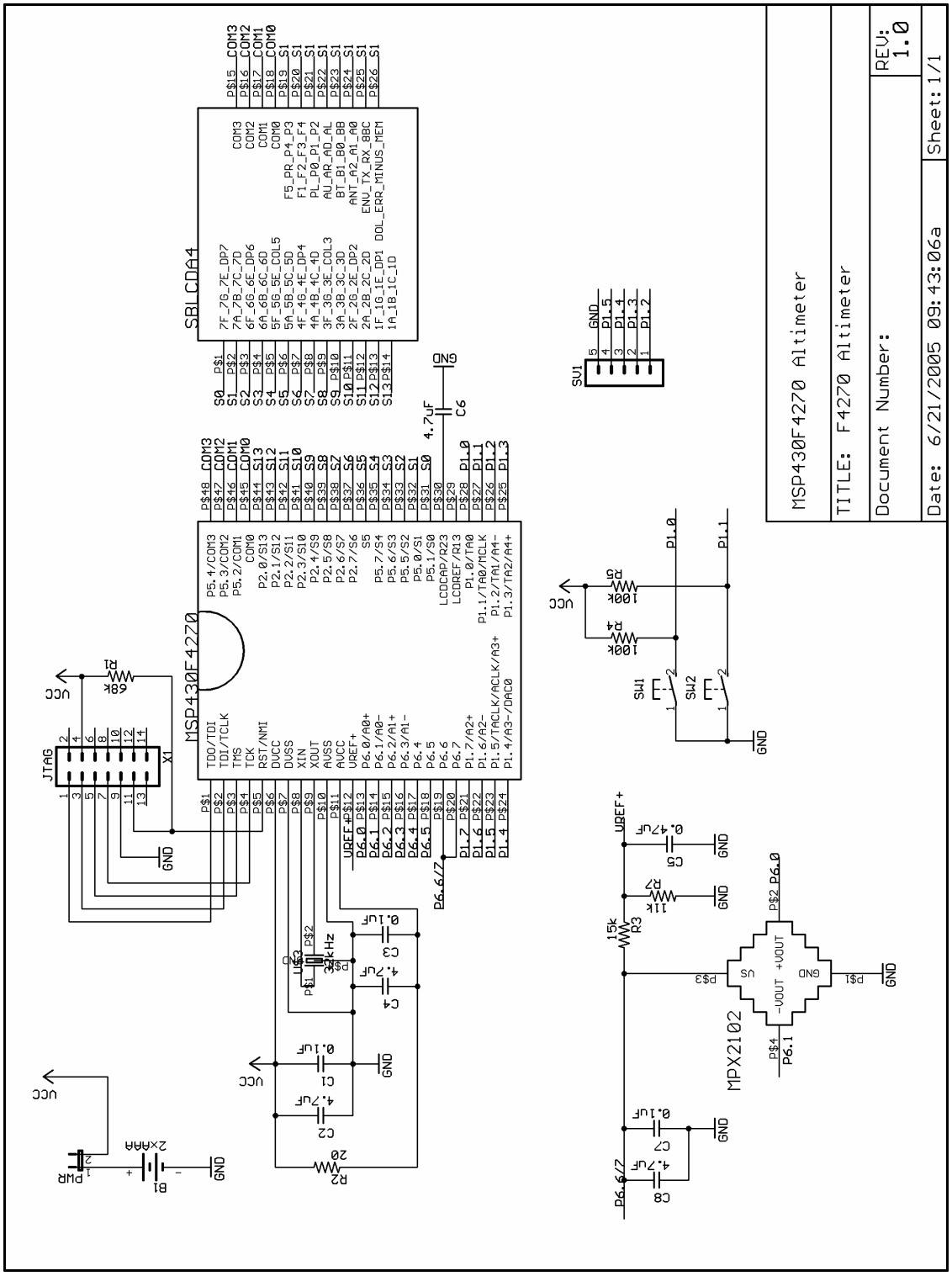
- Mounting the PCB inside a light-resistant enclosure. It was observed that the sensor readings are influenced by exposing it to light.
- Using an SD16_A gain setting of 1 for improved SINAD/ENOB performance and an external differential instrumentation OP-AMP to achieve a full-scale SD16_A input voltage. This way, the noise level is reduced, thus increasing the resolution and decreasing the amount of averaging needed.
- Using of a higher SD16_A modulator frequency to collect and average more conversion results.

References

1. *MPX2102 100kPa On-Chip Temperature Compensated & Calibrated Silicon Pressure Sensors Data Sheet* (Freescale Semiconductor)

2. *MSP430x42x0 Mixed Signal Microcontroller Data Sheet* (SLAS455)
3. *MSP430x4xx Family User's Guide* (SLAU056)

Appendix A. Application Schematic



Appendix B. Alternative Solution Using The TI LCD Display

Along with this application report, software, schematics and board files for an alternative solution are provided. The design files are located in the folder “Alternative_Solution” in the ZIP archive associated with this application report.

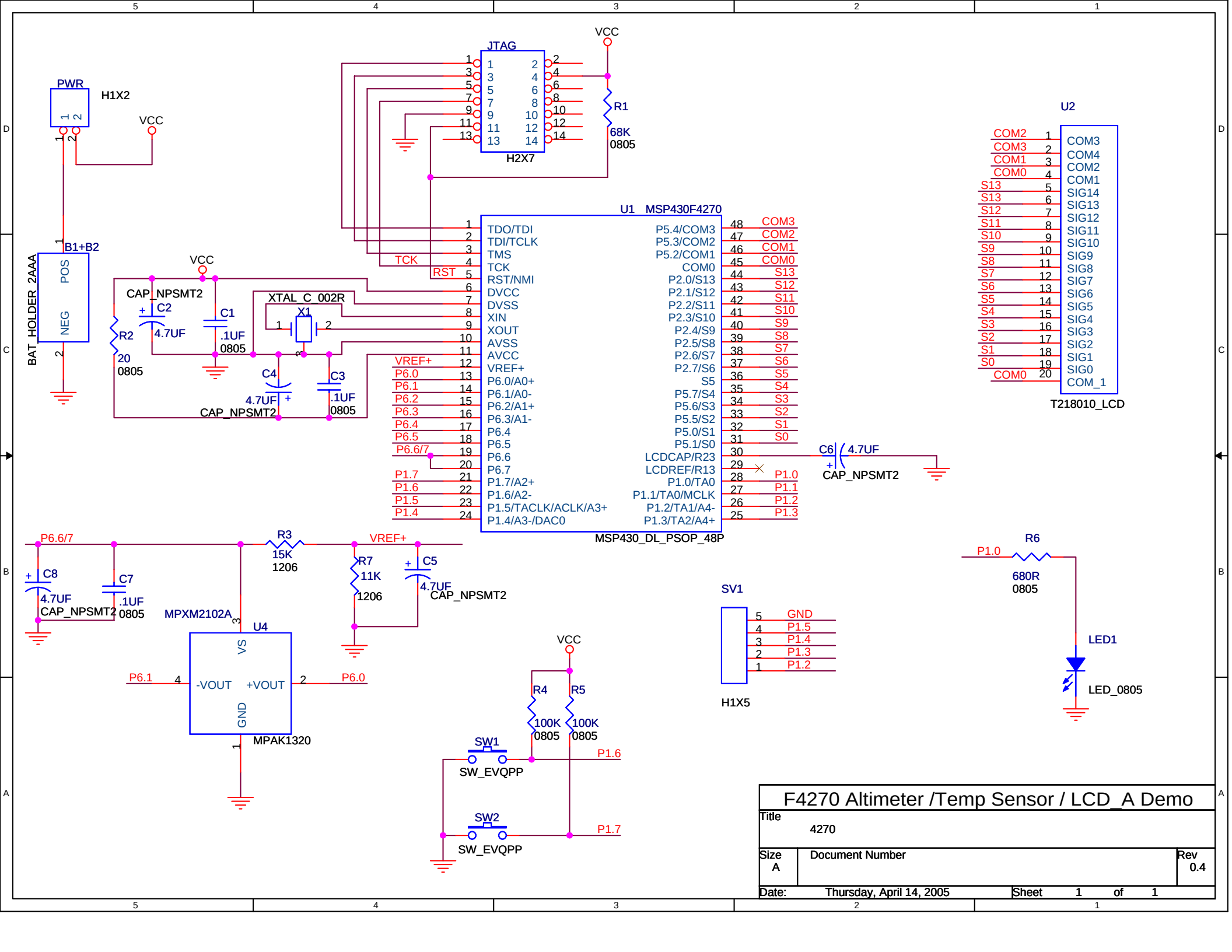
B.1 Differences To The Demo Application Using The SoftBaugh SBLCDA4 Display

The following items are different in the design of the alternative solution as described here in Appendix B and the demo application as described in the main part of this application report:

- The TI MSP-EVK430S320 4-Mux LCD display is used instead of the SoftBaugh SBLCDA4 display
- The push buttons SW1 and SW2 are connected to P1.6 and P1.7 instead of P1.0 and P1.1
- An LED has been added to P1.0
- The software has been modified to take the changed hardware into account.

All other explanations provided in this application report like software operational description and performance figures are identical.

B.2 Application Schematic (TI LCD Display)


JTAG
H2X7
VCC
R1
68K
0805
PWR
H1X2
B1+B2
BAT_HOLDER_2AAA
POS
NEG
U1 MSP430F4270
TDO/TDI
TDI/TCLK
TMS
TCK
RST/NMI
DVCC
DVSS
XIN
XOUT
AVSS
AVCC
VREF+
P6.0/A0+
P6.1/A0-
P6.2/A1+
P6.3/A1-
P6.4
P6.5
P6.6/7
P6.7/A2+
P6.6/A2-
P6.5/TACLK/ACLK/A3+
P6.4/A3-/DAC0
P5.4/COM3
P5.3/COM2
P5.2/COM1
COM0
P2.0/S13
P2.1/S12
P2.2/S11
P2.3/S10
P2.4/S9
P2.5/S8
P2.6/S7
P2.7/S6
S5
P5.7/S4
P5.6/S3
P5.5/S2
P5.0/S1
P5.1/S0
LCDCAP/R23
LCDREF/R13
P1.0/TA0
P1.1/TA0/MCLK
P1.2/TA1/A4-
P1.3/TA2/A4+
MSP430_DL_PSOP_48P
U2
T218010_LCD
CAP_NPSMT2
C1
.1UF
0805
C2
4.7UF
R2
20
0805
XTAL_C_002R
X1
C3
C4
C6
C7
C8
R3
15K
1206
R4
100K
0805
R5
R6
680R
R7
11K
C5
MPXM2102A
U4
MPAK1320
-VOUT
+VOUT
VS
GND
SW1
SW2
SW_EVQPP
SV1
H1X5
LED1
LED_0805
F4270 Altimeter /Temp Sensor / LCD_A Demo
Title
4270
Size
A
Document Number
Rev
0.4
Date:
Thursday, April 14, 2005
Sheet
1
of
1

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265