

Implementing An Ultralow-Power Thermostat With Slope A/D Conversion

Keith Quiring

MSP430 Applications

ABSTRACT

This application note describes slope A/D measurement of a resistance and the ease with which it can be implemented with the MSP430. To demonstrate this technique, two examples of ultra low power thermostats are presented, one using the MSP430F1111A and the other using the MSP430F413. Both examples implement the analog/digital conversion using the MSP430's on-chip comparator and timer.

1 Introduction

Analog-to-digital conversion is critical to any number of microcontroller applications. Real-world signals are often analog, and they must therefore be converted to digital before they can be analyzed by a processor. To make this conversion, various types of ADCs (analog-to-digital converters) can be used. However, if an ADC module isn't available on the MSP430 device in question, it's possible to do digitization with the integrated comparator and timer, using a technique known as *slope A/D conversion*.

In order to demonstrate the slope A/D conversion technique using the MSP430's on-chip comparator and timer, this document describes two thermostat designs. Each design uses the technique to measure the resistance of an external thermistor. Once the software has obtained the digitized temperature value, it compares it to a target "setpoint" value and decides how to adjust the temperature in the room. One design prioritizes low cost, while the other incorporates additional features. After some design description and commentary, a power analysis is provided for each design.

2 Description of Slope A/D Technique using MSP430

2.1 Overview

Slope A/D conversion is an analog-to-digital conversion technique that can be implemented with a comparator rather than a standalone ADC module or device. The technique is based on the charging/discharging of a capacitor with a known value. The number of clock cycles necessary to discharge the capacitor is then counted. Longer discharge times indicate larger voltages. The voltage is derived from the discharge time using the standard equation for capacitor discharge.

In addition to digitizing voltages, a variation of the technique can be used to measure resistance. This is valuable in measuring any component that can have varying resistance, such as potentiometers and various types of transducers. Unlike voltage measurement, where the key relationship is between voltage and time while the resistance is constant, the key relationship in resistance measurement is between resistance and time, while the initial voltage remains constant. The R-t relationship is linear, which means the calculation is easier and less-costly to implement in a microcontroller than for the exponential V-t relationship.

Fig. 1 shows the hardware configuration of a slope A/D resistance-measurement implementation using the MSP430. This circuit measures the resistance of R_{sens} by discharging capacitor C_m through it. The discharge through resistor R_{ref} is also measured, and will be used as discussed in Sec. 2.2.

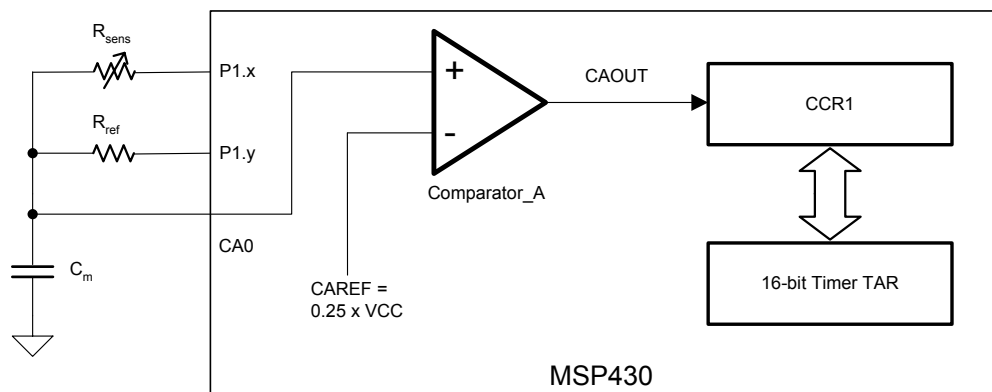


Figure 1. Measurement of Resistors

To measure the resistor value R_{sens} , capacitor C_m is first charged to the digital I/O high voltage ($V_{\text{OH}} \cong V_{\text{CC}}$) by outputting a high on either P1.x or P1.y. After configuring the timer, the capacitor is discharged through R_{sens} via P1.x by outputting a low level voltage. At the start of capacitor discharge, register TAR is cleared, and the timer is started. When the voltage across capacitor C_m reaches a comparator reference value V_{caref} of $(0.25) \cdot V_{\text{CC}}$, the negative edge of the comparator output CAOUT causes the TAR value to be captured in register CCR1. This value is the discharge time interval t_{sens} . The process is repeated for the reference resistor R_{ref} , which will be used to translate t_{sens} into the resistor value R_{sens} . C_m is given time t_c to charge, where t_c is between 5τ (for 1%) and 7τ (for 0.1%), where $\tau = R_{\text{ref}} \cdot C_m$. The value within this range depends on the accuracy required.

Fig. 2 shows the voltage V_{cm} across capacitor C_m during the two measurements.

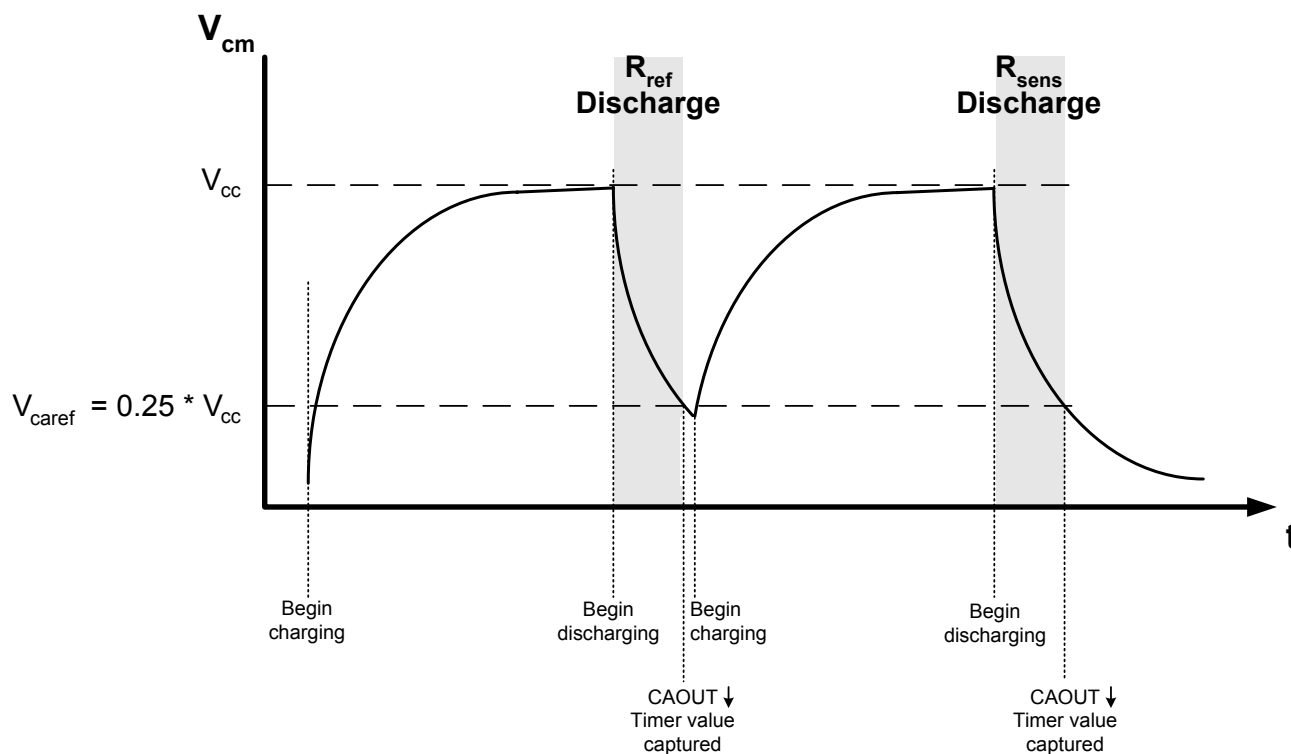


Figure 2. Voltage at C_m During Resistance Measurement

High accuracy can be obtained with this method, and it may broaden the number of MSP430 devices that fit a given application, since no ADC module is required. However, the tradeoff for this is that the sampling rate is limited due to the relatively long charge/discharge time, and there is additional code and execution cycles necessary to perform the measurement and calculation. As will be seen below, the execution cost of the calculation (and therefore power consumption cost) comes in the form of one multiply and one divide per measurement.

2.2 Calculation for Slope Measurement of a Resistance

From basic circuit theory, the voltage across a capacitor discharging through a resistor is:

$$V(t) = V_0 e^{-t/RC} \quad (\text{equation 1})$$

Given these specific and known values:

$$\begin{aligned} V(t) &= V_{\text{CAREF}} \\ V_0 &= V_{\text{OH}} \cong V_{\text{CC}} \\ R &= R_{\text{SENS}} \\ C &= C_{\text{m}} \end{aligned}$$

this produces:

$$V_{\text{caref}} = V_{\text{cc}} e^{-t_{\text{sens}} / R_{\text{sens}} C_{\text{m}}} \quad (\text{equation 2})$$

The only unknown in equation 2 is R_{SENS} , which means R_{SENS} can be calculated with this equation. However, it depends highly on the accuracy of C_{m} , which is a problem since most capacitors have relatively wide tolerance. Another problem with equation 2 is that it involves an exponential calculation, which is expensive either in execution cycles (if calculating) or memory (if using a lookup table).

One way to solve the C_{m} problem would be to use a very narrow-tolerance capacitor. However, another way to solve the problem is to measure the discharge of a reference resistor R_{ref} attached to the same capacitor, and using the resulting value to normalize R_{sens} . Not only does this provide an opportunity to remove C_{m} from the equation, but it also removes the exponential component, as will now be shown.

If C_{m} is discharged through such a resistor R_{ref} , the equation would be identical to equation 2 except with the new values of R_{ref} and t_{ref} . Since V_{CAREF} is equal in both cases, the right-handed portions of the R_{sens} and R_{ref} equations can be set equal to each other. When this compound equation is simplified, it produces a simple ratio:

$$\frac{R_{\text{SENS}}}{t_{\text{SENS}}} = \frac{R_{\text{REF}}}{t_{\text{REF}}} \quad (\text{equation 3})$$

Therefore, if the capacitor discharge timing sequence is performed on both R_{sens} and R_{ref} , with R_{ref} being a high-precision resistor of known value in the same range as R_{sens} , then the only unknown in equation 3 is R_{sens} . This allows a calculation of R_{sens} that is more accurate than equation 2. The equation 3 calculation also requires less power, since it can be performed with only a multiply and a divide. This is the calculation used in most slope A/D resistance-measurement applications.

3 Thermostat Implementation Using Slope A/D Conversion

The following are examples of using the method described above to build a thermostat. In both examples, the value of a thermistor is acquired using slope A/D conversion, which is then matched with a temperature value using a look-up table. Action is then taken to output the data to the user.

The first example is a low-cost design using the MSP430F1111A. In this design, the user sets the desired temperature using a potentiometer. LEDs momentarily light to show that the unit is cooling or heating. Slope A/D conversion is also used to determine the potentiometer setting.

The second example offers advanced features and uses the MSP430F41x. It drives an LCD to display either the current temperature, setpoint temperature, or time, depending on the mode. Buttons are utilized to change modes and set the values.

3.1 Low-Cost Thermostat Implementation using the MSP430F1111A

Fig. 3 portrays a simple thermostat using the MSP430F1111A. The software is designed around a main loop that executes once every five seconds, prompted by an interrupt. Between the interrupts, the device stays in low power mode three (LPM3) and draws very low current. (See the power analysis in section 3.1.3.)

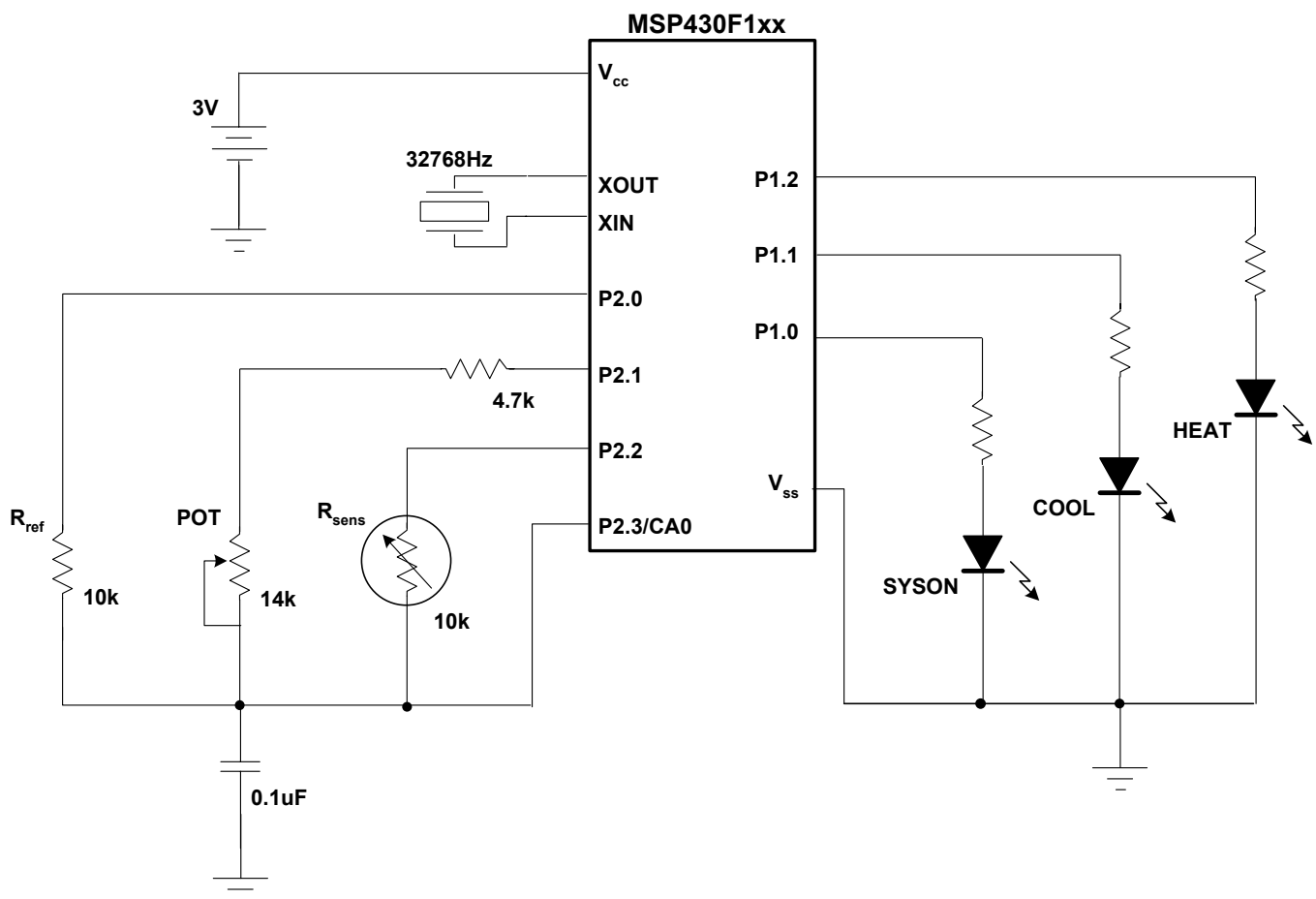


Figure 3. Low-Cost Thermostat Block Diagram

3.1.1 User Operation

The user selects the setpoint temperature using the potentiometer. The thermostat function then indicates whether the system needs to cool or heat the environment to match that temperature, using the “cool” and “heat” LEDs. At the same time, a “system on” LED is driven. If the current temperature matches the set-point temperature, only the “system on” LED is driven. In theory, the “heat” and “cool” LEDs could be used to drive heating/cooling equipment. The LEDs stay lit for only half a second.

In this design, the low end of the setpoint potentiometer is assigned to be 60° Fahrenheit, while the high end is assigned to 95° Fahrenheit. This can be user-defined in software.

There is no LCD display, so the only way to obtain the specific temperature value is to use a debugging environment to obtain the value from within the code. However, one can experiment with the device by locating the setpoint value (at which only the “system on” LED lights), then applying heat or cold to the thermistor and watching the resulting changes in the LEDs.

3.1.2 Design Description

A flowchart for the main loop, discharge measurement, and interrupt service routines for the low-cost implementation are shown in Fig. 4, 5, and 6 respectively.

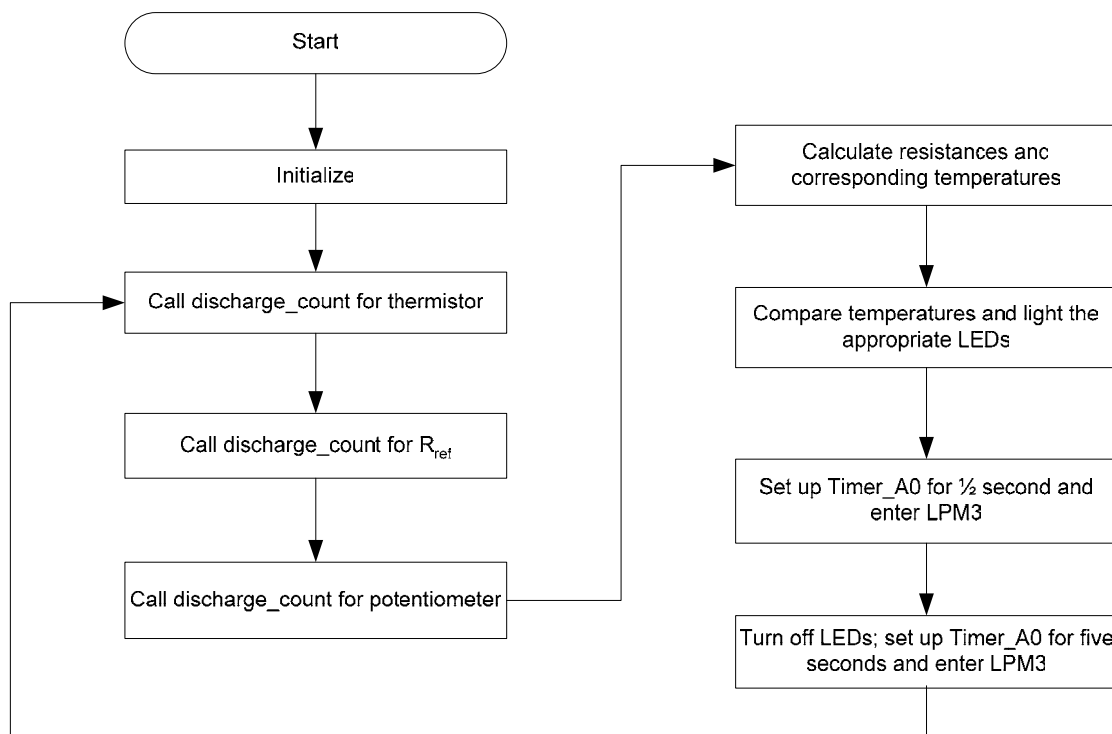


Figure 4. Main Loop Flowchart for Low-Cost Implementation

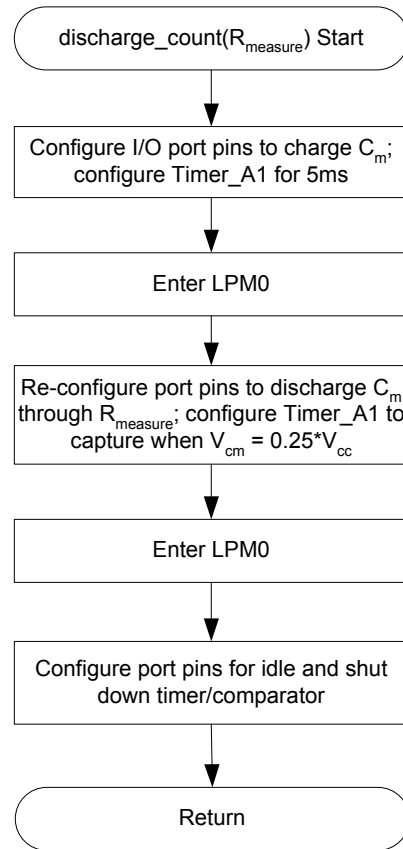


Figure 5. Discharge Measurement Flowchart for Low-Cost Implementation

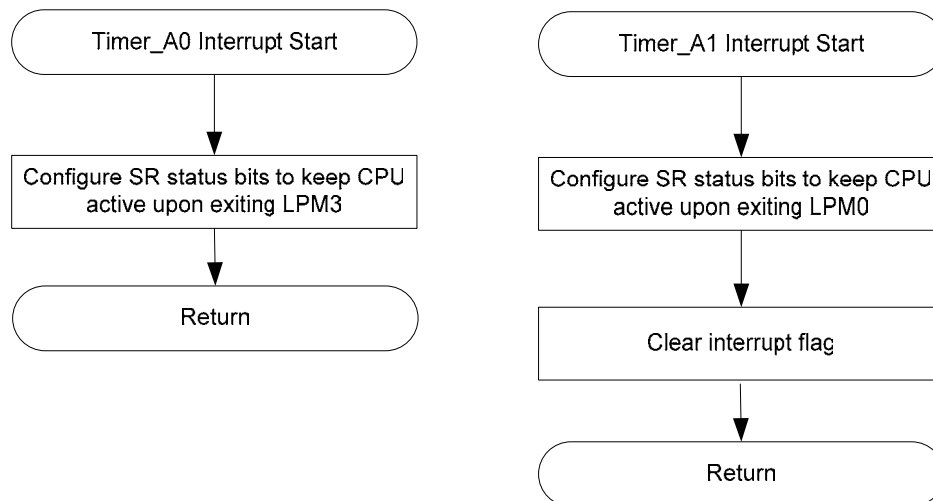


Figure 6. ISR Flowcharts for Low-Cost Implementation

The MSP430 is put to sleep several times: for five seconds between main loop runs; for half a second while the LEDs are lit; and while the capacitor is charging/discharging. In each case, the Timer_A module is used. For the former two, ACLK is the source clock, and therefore the sleep mode chosen is LPM3. For the latter, SMCLK is used (derived from the DCO) because of the higher frequency and greater slope ADC accuracy. Therefore, LPM0 is used in this case. A different CCR register is used in each case: CCR0 for LPM3 and CCR1 for LPM0. In the interrupt service routines, the respective low-power mode bits are cleared, such that when the ISR is exited, the device will remain active, returning operation to the place where it was put to sleep.

Every time the main loop runs, the MSP430 calculates the resistance value of the thermistor and potentiometer using the discharge times of R_{sens} and R_{ref} as described in Sec. 2. These resistance values are then converted to temperatures, using two different methods described below. Notice that all three resistor values are in the range of 10k: the reference resistor is a fixed, precise 10k resistor, and the possible resistance values of both the thermistor and the set-point resistor center around 10k. Because all three resistors use the same capacitor in the RC discharge measurement, assigning similar resistance values results in discharge times within the same range. This in turn allows use of the same timer clock setup for each measurement, and simple comparison of timer values.

The thermistor in this design is a BC Components 2322-640-54103, an NTC (negative temperature coefficient) thermistor device.

Conversion of the thermistor and setpoint potentiometer resistance values to temperature values utilize different approaches. For the thermistor, data is provided by the manufacturer to correlate resistance and temperature. For most NTC thermistors, the data is provided only in 5-deg Celsius increments. For the purposes of a residential thermostat, the resistance values for the temperatures between these points have been interpolated linearly in one-degree increments. This data is represented in the software as a table, indexed by one-degree values. The software determines the index at which the resistance value falls, and adds this to the minimum temperature value (60°) to produce the current temperature.

In contrast, the setpoint potentiometer resistance is converted with an equation that seeks to span the temperature range (65° to 90°) across the resistance range of the circuit (4.7k Ω to 14.7k Ω). The resistance range is evenly distributed among the available number of degrees (26). With 65° being represented by the bottom of the range, this leaves 25 segments to be represented by 10k. There are 25 400 Ω segments in this range. Keeping in mind that all resistances within the program code are represented as actual resistance divided by 100, a temperature value between 65-90° can be reached by subtracting 47 (4.7k Ω) from the potentiometer resistance and dividing by 4 (400 Ω). For example, a potentiometer resistance of 14.7k is represented within the program as 147; subtracting 47 and dividing by 4 results in 90°. Note that the potentiometer value was carefully chosen so that the divisor would be a value of 2^n . This allows the division to be handled by right-shift instructions, significantly reducing execution cycles and power draw.

To avoid the potential for rapid alteration between heat and cool when the current temperature is near the setpoint, a +/- 1 degree error is allowed when determining whether to “heat” or “cool”. If the current temperature is not within this range of the setpoint temperature, the appropriate heat/cool LED is driven.

An interrupt period of five seconds is sufficient because room temperature does not change quickly. This has a significant positive impact on power usage; for example, the battery will last approximately twice as long as if the measurement were performed every 2.5 seconds. A tradeoff to consider is that the user must wait longer for changes to take effect. This is not a problem when temperature is input via a switch (as in the advanced implementation below), since the switch drives an interrupt that wakes the device. In contrast, a rotation applied to the potentiometer does not generate an interrupt.

Note that when the resistors are not being measured, P2.1 is driven high. This has the effect of driving the other resistor port pins (assigned to be inputs) high as well. Floating inputs can cause the device to draw current in the hundreds of μA . By ensuring that all port pins are either outputs or inputs pulled high or low, this situation is prevented.

3.1.3 Power Analysis

A diagram portraying the power draw of the low-cost implementation is shown in Fig. 4. Time periods of similar lengths and current levels are given common labels. Table 1 shows the values associated with those labels, while Table 2 shows the final power calculation. Leakage current through the capacitors, diode, and digital I/Os are assumed to be negligible. The values shown represent a typical run through a main loop cycle; each run is dependent to some extent on the jumps and conditions taken, and in particular the varying values of the resistors, which can impact the multiplies and divides.

The values in the tables reflect the assembly version of the code. The assembly code is more power-efficient, mostly because of optimizations in the multiply and divide functions.

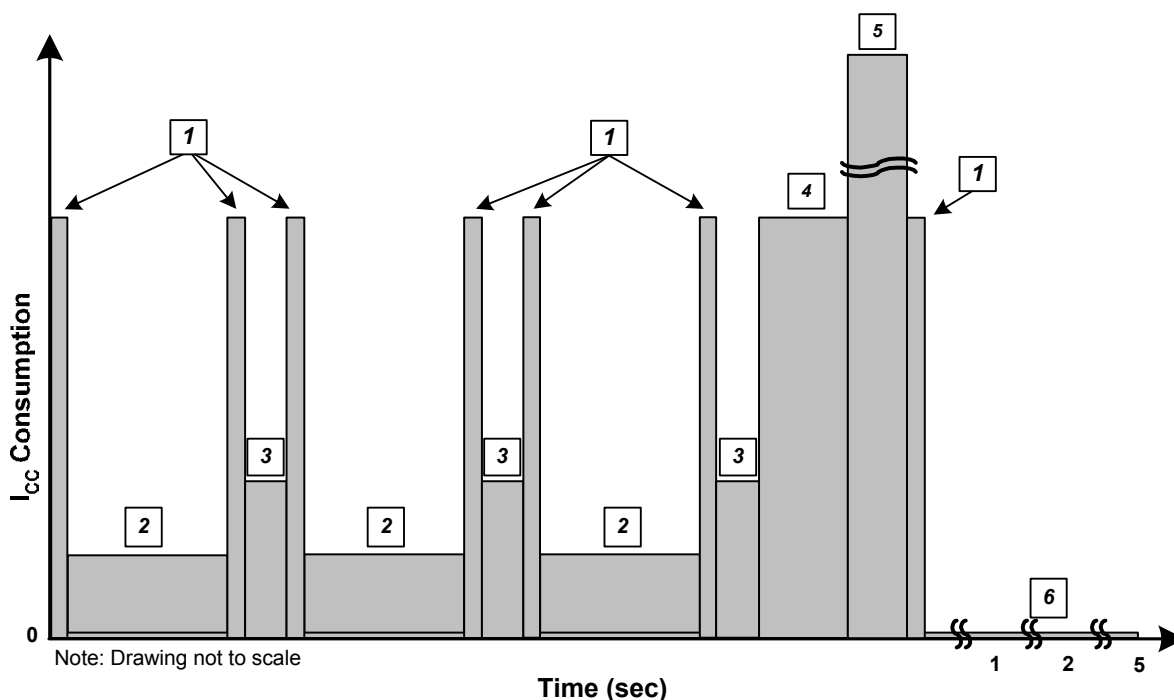


Figure 7. Power Draw over Time on the Low-Cost Implementation

Table 1. Label Definitions for Low-Cost Implementation

Label	Activity Description	Duration	Components Drawing Power	Current Draw (typ)
1	Active-mode main loop execution	51us† (each)	CPU active	300uA
2	Capacitor charging	5.000ms (each)	CPU in LPM0	55uA
3	Capacitor discharging	1.400ms (each)	Comp_A CPU in LPM0	45uA <u>55uA</u> 100uA
4	Main loop processing discharge data	1.414ms	CPU active	300uA
5	Drive LEDs	500ms	LED outputs CPU in LPM0	18036uA <u>55uA</u> 18091uA
6	Sleep	5000ms	CPU in LPM3	1.6uA

† Values of each occurrence vary slightly; this is the average value of each segment.

Table 2. Power Draw Calculation

Label	Instances	Current (typ)	Duration/ Instance	Total Time	uA-ms
1	7	300uA	0.051ms	0.357ms	107.100
2	3	55uA	5.000ms	15.000ms	825.000
3	3	100uA	1.400ms	4.200ms	420.000
4	1	300uA	1.414ms	1.414ms	424.200
5	1	18091uA	500.000ms	500.000ms	9,045,500.000
6	1	1.6uA	5000.000ms	5000.000ms	8000.000
With LED					
				5.521 sec	9.055 mA-sec
					1.640 mA/sec
Without LED					
				5.021 sec	9.776 uA-sec
					1.947 uA/sec

Note that the LED is the most significant source of current draw and has a major impact on the overall average. To show the low-power performance of the system with and without the LED, the data is calculated in both ways in Table 2. To obtain the “without LED” calculation, simply omit segment #5.

3.2 Advanced Thermostat Implementation Using the MSP430F413

The circuit in Fig. 2 is based on the MSP430F413 and features an LCD display, a real-time clock, and pushbutton setpoint input. The main loop for this implementation runs once every second, prompted by an interrupt. Between interrupts, the device is in LPM3 mode, resulting in very low power operation (see the power analysis in section 3.2.3).

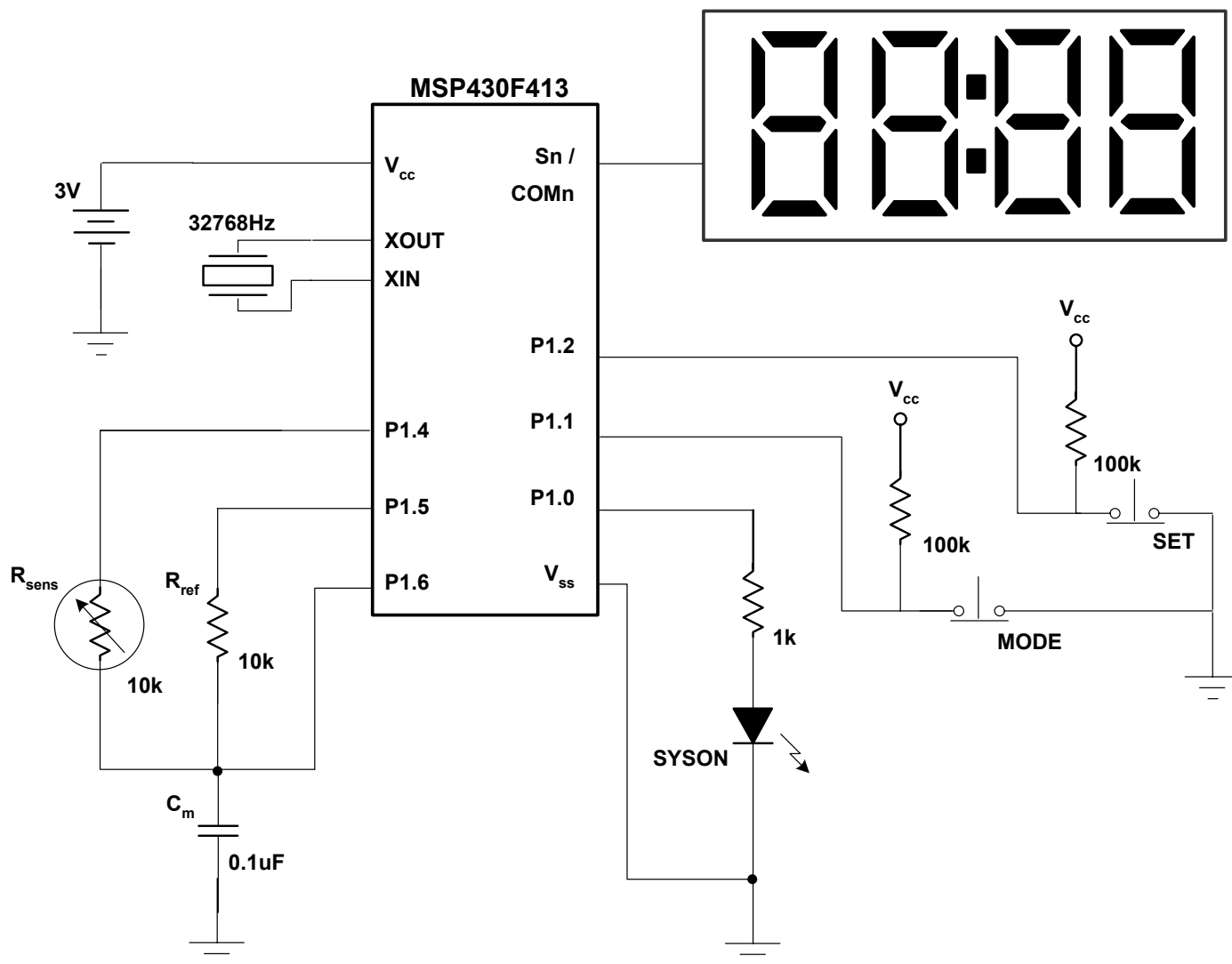


Figure 8. Advanced Thermostat Block Diagram

3.2.1 User Operation

The device can operate in one of four modes, selected by the “mode” switch. The modes are:

- *Thermostat.* Measures the current temperature and displays it on the LCD. If the current temperature is less than the setpoint temperature, the LED is driven to indicate the activation of “cool”.
- *Setpoint.* Allows the user to select the target temperature using the LCD and the “set” switch. Pressing the “set” switch repeatedly causes the setpoint temperature to proceed upward through the temperature range, rolling back to 60 once 95 has been reached. If the setpoint upon leaving this mode is different than the one in place when the mode was entered, then the new value is written to flash memory between 0x1000 and 0x10FF (“information memory”).
- *Time.* Displays the real-time clock on the LCD. The time can be changed by pressing the “set” switch; each press of the switch increments the time by one minute. After ten seconds of inactivity, the mode will automatically revert to “thermostat” mode.
- *Seconds.* Simply counts seconds and displays them on the LCD.

Pressing the “mode” switch repeatedly causes the mode to rotate through the above sequence.

3.2.2 Design Description

A flowchart for the main loop and interrupt service routines for the advanced implementation are shown in Fig. 9 and 10, respectively. The `discharge_count` function is the same as for the low-cost implementation, represented in Fig. 5.

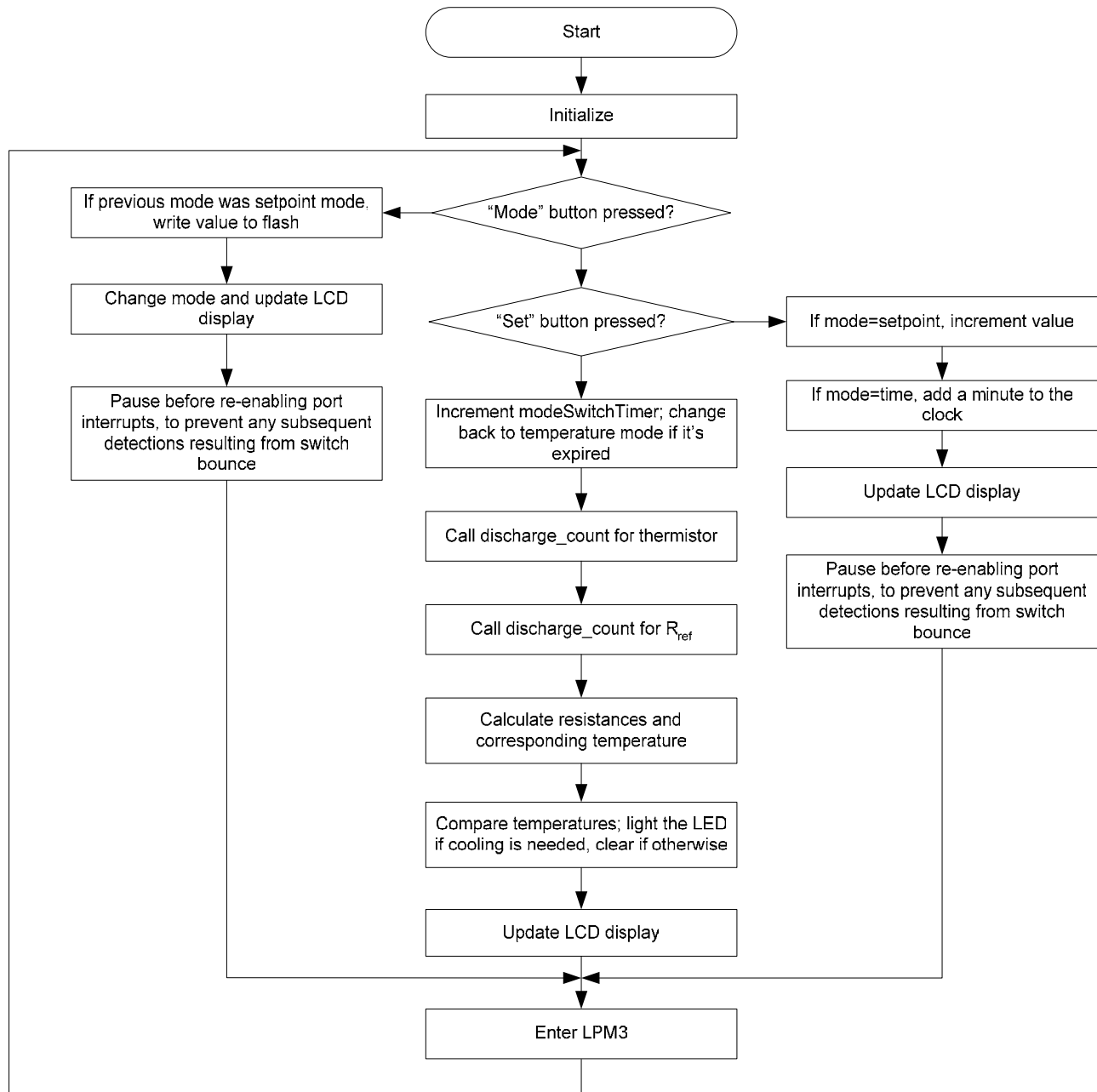


Figure 9. Main Loop Flowchart for Advanced Implementation

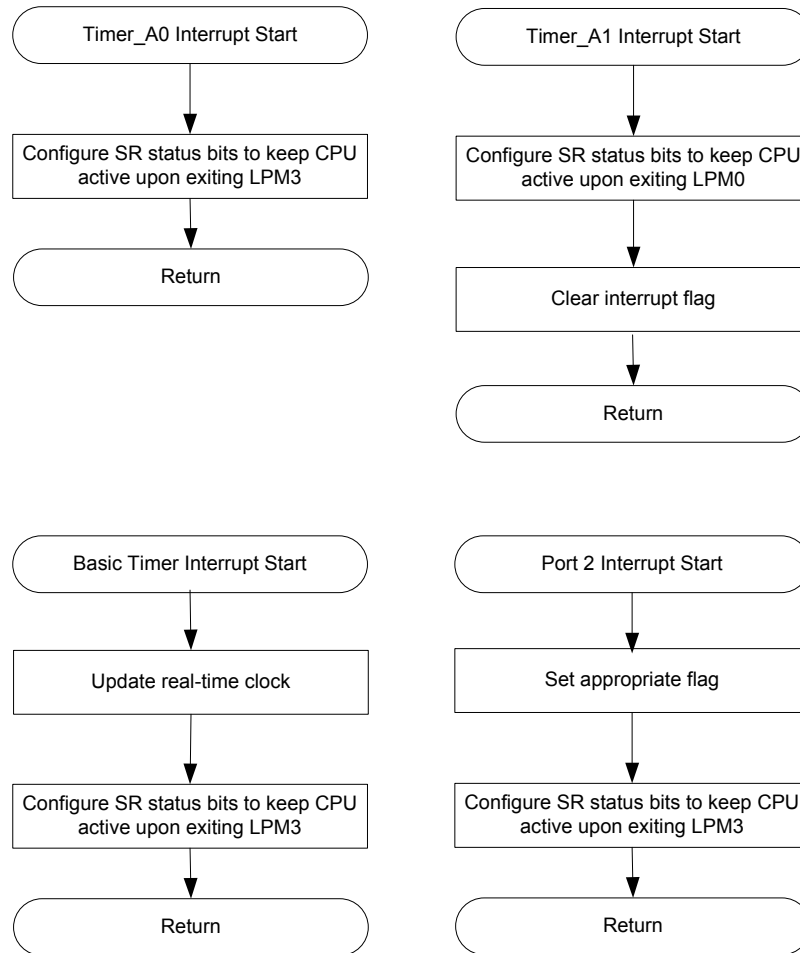


Figure 10. ISR Flowcharts for Advanced Implementation

Unlike the first thermostat example, which wakes up every five seconds, this one wakes up every second. The main reason for this is that the real-time clock needs to be updated every second. The device can also be woken between the one-second intervals by one of the two pushbutton switches.

The one-second interrupt is generated by one of the counters in the Basic Timer module, which is powered by the 32kHz watch crystal and continues to count while the device is in LPM3. Note that this differs from the MSP430F1111A implementation, since the 'F1xx family does not contain the Basic Timer. Using the Basic Timer frees Timer_A/B, if present, for functions that are more complex.

In the Basic Timer interrupt service routine, the low-power mode bits are cleared, such that when the ISR is exited, the device will remain active, prompting the start of one run through the main loop. Because the loop is executed for multiple purposes – sometimes to simply update the clock, sometimes to update the display, sometimes to measure the temperature, sometimes to handle the pushbuttons, etc. – it is necessary to check the state of the device within the main loop to determine what action needs to be taken. Once all appropriate actions are completed, the device is put back into LPM3 to await the next one-second interrupt or press of a button.

While the circuit is in set point mode the symbol “-|” is displayed on the LCD followed by the two-digit temperature set point. The symbol “-|” was chosen because the 3.5-digit LCD does not allow us to display “SP” alongside the two digit value of the set-point.

When the set point mode is exited, the new setpoint value is written to the information memory segment of flash. Rather than writing to a single location each time, it increments across the 256 bytes of flash, and when full, it erases and starts at the beginning. By doing this, the operating life of the flash is extended. The MSP430 flash memory can be erased 100,000 times at 25 degree C. This means that the set point could be changed reliably up to 25.6 million times.

3.2.3 Power Analysis

A diagram portraying the power draw of the advanced implementation is shown in Fig. 4. Time periods of similar lengths and current levels are given common labels. Table 3 shows the values associated with those labels, while Table 4 shows the final power calculation. Leakage current in the capacitors and digital I/Os is assumed to be negligible. The values shown represent a typical run through a main loop cycle; each run is dependent to some extent on the jumps and conditions taken, and in particular the varying values of the resistors, which can impact the multiplies and divides.

The values in the tables reflect the assembly version of the code. The assembly code is more power-efficient, mostly because of optimizations in the multiply and divide functions.

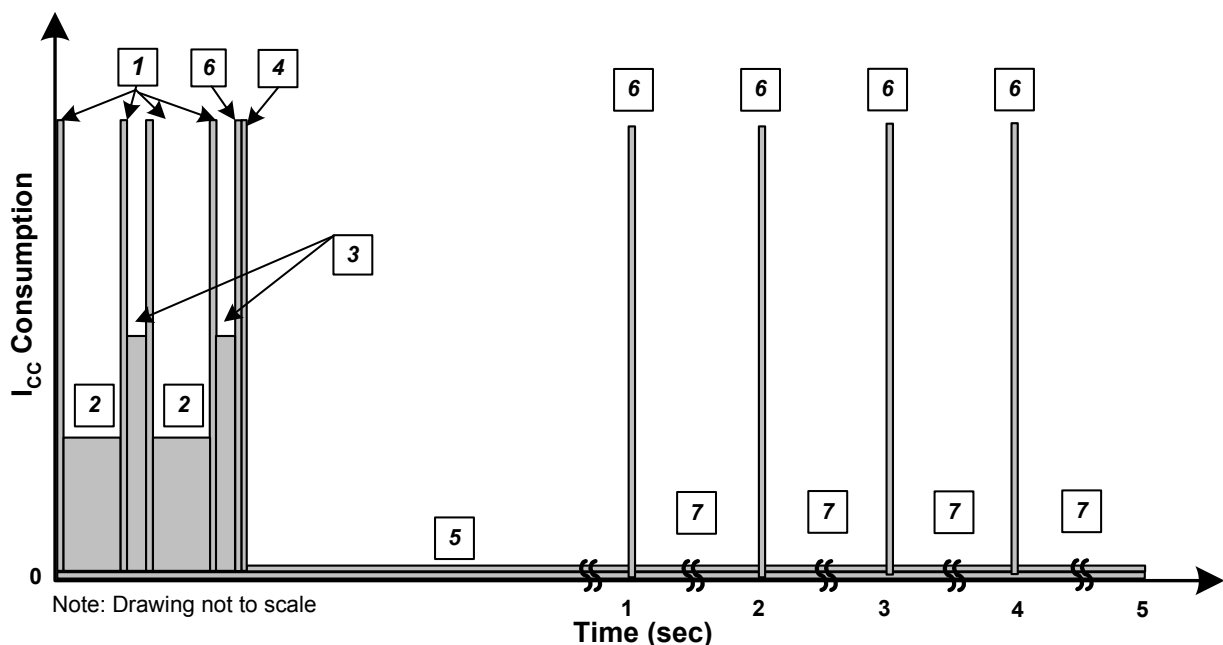


Figure 11. Power Draw over Time on the Advanced Implementation

Table 3. Label Definitions for Advanced Implementation

Label	Activity Description	Duration	Components Drawing Power	Current Draw (typ)
1	Active-mode main loop execution	67us† (each)	LCD display CPU active	1uA <u>300uA</u> 301uA
2	Capacitor charging	5.000ms (each)	LCD display CPU in LPM0	1uA <u>92uA</u> 93uA
3	Capacitor discharging	1.400ms (each)	Comp_A LCD display CPU in LPM0	45uA 1uA <u>92uA</u> 138uA
4	Main loop processing discharge data	730us	LCD display CPU active	1uA <u>300uA</u> 301uA
5	Sleep	985.955ms	LCD display CPU in LPM3	1uA <u>0.9uA</u> 1.9uA
6	Update clock/display every second	247us (each)	LCD display CPU active	1uA <u>300uA</u> 301uA
7	Sleep	999.753ms (each)	LCD display CPU in LPM3	1uA <u>0.9uA</u> 1.9uA

† Values of each occurrence vary slightly; this is the average value of each segment.

Table 4. Power Draw Calculation

Label	Instances	Current (typ)	Duration/ Instance	Total Time	uA-ms
1	4	301uA	.067ms	0.268ms	80.668
2	2	93uA	5.000ms	10.000ms	930.000
3	2	138uA	1.400ms	2.800ms	386.400
4	1	301uA	0.305ms	0.305ms	91.805
5	1	1.9uA	985.955ms	985.955ms	1873.315
6	5	301uA	0.672ms	3.360ms	1011.360
7	4	1.9uA	999.328ms	3997.312ms	7594.893
				5 sec	11.978 uA-sec
					2.394 uA/sec

In the advanced implementation, the LED is either on continuously or not at all. Therefore, the power calculation in Table 4 remains unchanged when the LED is off. If the LED is on, the current draw in every segment type is increased by 3mA, as is the per-second average.

References

1. *MSP430F41x data sheet (SLAS340)*
2. *MSP430F11x1 data sheet (SLAS241)*
3. *MSP430x1xx User's Guide (SLAU049)*
4. *MSP430x4xx User's Guide (SLAU056)*
5. *Economic Voltage Measurement with the MSP430 Family (SLAA061)*

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

Copyright © 2006, Texas Instruments Incorporated