# Experiments for the MSP430 Starter Kit

*Mark Buccini* *Mixed Signal Products*

**ABSTRACT**

The MSP430 is a 16-bit RISC-like mixed signal controller. This application report provides several experiments for the entry-level MSP-STK430×320 starter kit.

**Contents**



**List of Figures**

# 1 Introduction

The MSP-STK430x320 starter kit (STK) is a low-cost, entry-level demonstration board. The STK is designed to allow easy exploration of some of the MSP430x32x device capabilities. This report provides several STK experiments with complete and proven software to accelerate evaluation of the MSP430x32x. All the source code for the experiments provided in this report can be compiled within the TI ADT430 simulation environment. The ADT430 outputs a *.txt*-formatted code that can be downloaded and executed completely out of the STK's MSP430P325 RAM. The STK's MSP430P325 has a preprogrammed monitor which is used to communicate with the STK and download the *.txt*. Code executes on the STK at full speed. With all experiments performed out of RAM, an STK can be used over and over. The experiments in Section 2, *Getting Started—Experiments for the STK and ADT430,* can be conducted on the STK demo board or simulated in software using the ADT430 simulation environment. The experiments in Section 3, *Experiments Specific to the STK,* can only be performed on the STK demo board itself. A few of the experiments require external components, all of which can be conveniently purchased from many electronic parts distributors.

**NOTE:** It is important: that the user review all the documentation included with the STK. Current copies of the *MSP430 Architecture and Module Users Guide*, and the *MSP430x32x Family Data Sheet* should constitute good references.

## 1.1 Basics

- The STK demo board powers directly from a PC serial port when an RS232 cable is attached to the PC and the PC serial port is active.
- It is recommended to jumper the STK for 3-volt operation to save power. See the *Starter Kit Evaluation Kit Manual* for details.
- Source code specifically targeted to run on the STK demo board can not be directly simulated in the ADT430 simulation environment. The monitor on the preprogrammed MSP430P325 on the STK relocates the MSP430 interrupt vectors from ROM address 0FFFEh to RAM address 03FEh. Proper software simulation requires interrupt vectors located at 0FFFEh.

## 1.2 STK Hardware

An MSP-STK430x320 demo board contains the following major components to be used with the experiments in this report. See Figure 1.

- U3—MSP430P325IPM preprogrammed with a ROM monitor
- U1—TSP7701 adjustable low-dropout voltage regulator
- D5—TLS252 light-to-voltage converter
- U5—LM385-2-5 voltage reference

- J3—Reset button
- J4—User button
- J2—DB9 RS232 connector
- LCD

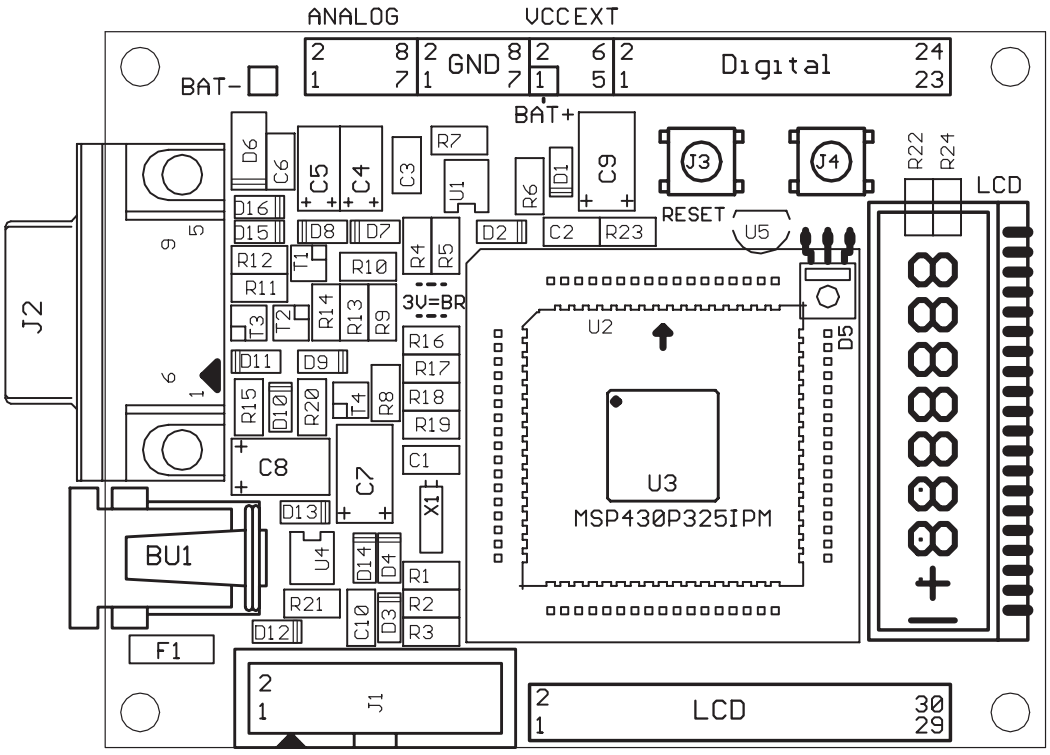


**Figure 1. MSP-STK320×320**

## 1.3 Using External Power

The STK demo board does not require an external power supply. Power is usually supplied directly from a PC serial port. For most experiments, the STK can be directly powered from an RS232 cable connected to an active PC serial port. Power from J2, the RS232 connection, is regulated to either 5 V or 3 V by the TPS7701 installed. To regulate an external power source (such as a 9-volt battery) the negative potential can be connected to pin 5, and the positive source can be connected to pin 4 of J2. In this way the power source is regulated via the TPS7701. An unregulated external power supply can be directly connected to Gnd and ExtVcc on the STK. To get started, and for most of the experiments, just use the PC's serial port to power the STK. A power supply should not be connected to BU1 (see Figure 1) for any of the experiments in this report; this could result in inadvertent permanent programming of the STK's MSP430P325 OTP.

**Important:** An external power source should be used to achieve very accurate analog measurements using the STK demo board,. The noise generated by the RS232 connection will strongly degrade precision analog measurements.

### 1.4 Connections to External Components

All of the MSP430P325's signal pins are brought to the headers on the outside edges of the STK, see Figure 1. For the experiments in this report, all connections with external components are done to digital headers 1–24 and analog headers 1–7.

### 1.5 Use of TI Standard Peripheral Definitions

All examples in this report use TI MSP430 standard peripheral definitions. The file (std_def.asm) is included with the ADT430 simulation environment, or is available from TI's web site: *www.ti.com*. At compilation time, the file std_def.asm must be in the same working directory as the source code examples included. The following assembler directive copies std_def.asm to the source:

```
        .include   "STD_DEF.ASM"    ; Standard Definitions Used
```

The programmer of MSP430 products is encouraged to use TI standard peripheral definitions to promote commonality with the *MSP430 Architecture Guide and Module Library* and simplify code debugging.

## 2 Getting Started—Experiments for the STK and ADT430

All the experiments in this section can be run on the STK demo board or simulated in the ADT430 simulation environment. The user typically receives visual feedback on the LCD installed on the STK or the *demo.lcd* when linked to the ADT430. The LCD on the STK and *demo.lcd* are identical. When source code is compiled, simply choose the conditional assembler directive at the top of the source code from the following:

```
;               1   Run on STK Demo Board
;               0   Run on ADT430 Software Simulator
;              ---
STK     .set    0
```

### 2.1 HELLO

See *Software Listing stk_1.asm*

This first experiment will simply flash HELL0 on the STK's or ADT430's LCD. The LCD is a great way to provide feedback during STK experiments. The first task in an MSP430 program is usually to take care of the stack pointer (SP). The SP must be initialized by software. A good place to locate the SP is at the top of available RAM as the stack *pushes* down. Any subroutine call or applicable interrupt uses the stack. The code line below shows how to initialize the SP using the value *Stack,* which is defined in the source code:

```
RESET     mov     #Stack,SP         ; Initialize stack pointer
```

Next, the software watchdog must be dealt with. The MSP430 watchdog comes up active; if you are not sure what to do, turn it off. When accessing the watchdog control register for any reason, it is important to include the password 0A500h or a system RESET will occur. The example code line below, showing how to turn off the watchdog, uses TI Standard MSP430 peripheral definitions:

```
SetupWDT  mov    #WDTPW+WDTHOLD,&WDTCTL   ; Stop Watchdog Timer
```

In the source code for this experiment, the LCD control register (LCDCTL) and the basic timer register (BTCTL) are configured to support the 4-mux LCD installed. In *Mainloop*, the letters H E L L O are exclusive-ORed directly in the LCD-display RAM. This has the effect of turning the letters on and off. A *software wait* loop is entered to slow things down by moving the value *Delay* to CPU register R15, and then decrements the register until it becomes zero. Any available CPU register from R5 through R15 can be used. The choice of R15 is arbitrary. Notice that the conditional assembly statement at the top of the source selects a different value of *Delay* for the STK and the ADT430. This speeds up software simulation runs. The MSP430 chip itself is much faster than software simulation on a PC.

## 2.2 A Simple Counter

See *Software Listing stk_cnt.asm*

In this experiment, a simple one-digit BCD counter is provided. A single BCD digit counts from 0–9 and is displayed on the STK LCD. In this and all additional experiments, the MSP430 peripheral setup is moved to an *Init_Sys* subroutine to keep code tidy and easy to follow. Also, the software *wait* routine now pushes the *Delay* value to the top of the stack (TOS) and directly decrements the *Delay* value on the TOS. The MSP430 instruction set can directly address the stack pointer (SP). The ability to address the SP is a powerful feature and allows many arguments and variables to be pushed onto the stack and passed to other routines. Being able to count in BCD is also very useful, especially in applications that require a numeric user interface. The following instructions directly add the carry bit, in decimal format (BCD), to *Counter*:

```
setc            ; Set Carry
dadc.b Counter  ; Decimal add carry to Counter
```

A binary-to-BCD conversion routine is not needed, since the software is directly counting in BCD.

## 2.3 Using Interrupts and Low-Power Modes—Basic Timer

See *Software Listing stk_bt1.asm*

It is often better to use interrupts to direct the program flow. Interrupts are very useful in low-power applications where CPU activity should be kept to a minimum. Polling pins or testing for events wastes CPU cycles and power. In this example, the MSP430 runs normally in LPM3 and is interrupted by the Basic Timer at a programmed interval. The Basic Timer ISR wakes up the CPU, which completes *Mainloop* by toggling two bits on the LCD. Notice how the Basic Timer interrupt service routine (ISR) wakes up the system by changing the operating mode configuration in the status register (SR). Any ISR places the program counter (PC) and SR on the TOS and returns the system to active inside the already active ISR. If the SR on the TOS is not modified, the system returns to LPM3 on exit from the ISR. The SR on the TOS is modified inside the Basic Timer ISR with the code line below:

```
BT_Isr bic #LPM3,0(SP)      ; Clear SR LMP3 Bits, on TOS
```

Exiting the Basic Timer ISR now places a modified SR in the CPU, which forces the system to exist LPM3 and return to active. Notice that the Basic Timer interrupt interval is different for the STK and for the ADT430 simulation environment. A much shorter interval is used with software simulation to speed up execution. The software simulator can not simulate an exact time interval, and is actually much slower.

## 2.4 Interrupt-Driven Counter Using the Watchdog as an Interval Timer

See *Software Listing stk_wdt.asm*

This experiment combines low-power modes, interrupts, and counters. The Watchdog Timer is configured as an interval timer counting down and displaying on the LCD (99 – 00). The code line below has the effect of decimally subtracting one (BCD) from *Counter*:

```
dadd.b  #099h,&Counter      ; BCD Decrement by one
```

Also important is that *Counter* is defined in the source as a RAM location. The MSP430 instruction set can directly access the RAM without using any CPU registers as an intermediate step. Actually, the MSP430 can directly access all memory locations with all instructions without using CPU registers. On every watchdog interrupt, the ISR counts down the counter and displays the value on the STK LCD. When configured as an interval timer, the watchdog uses a dedicated interrupt vector 0FFF4h. As an interval timer, the watchdog does not completely *reset* the MSP430. Several intervals and clock sources are selectable for the watchdog. Notice that different intervals are used for the STK demo board and software simulator in the source code:

```
                .if STK = 1
SetupWDT    mov     #WDT_ADLY_1000,&WDTCTL   ; Watchdog 1s  Timer
            .else
SetupWDT    mov     #WDT_ADLY_1_9,&WDTCTL   ; Watchdog 1.9ms  Timer
            .endif
```

## 2.5 Interrupt-Driven RTC Using the Basic Timer

See *Software Listing stk_rtc.asm*

This program demonstrates a second and minute real-time clock (RTC). A second and minute value is counted and displayed on the STK LCD. The program runs normally in LPM3 with the Basic Timer clocked by the 32,768 Hz-watch crystal. The Basic Timer interrupts at exactly 1-second intervals (on the STK), and returns the MSP430 to an active mode which allows the program to run through *Mainloop*. *Mainloop* calls *Clock* and *Display* subroutines, and then returns to LPM3. RAM bytes SEC and MIN are used to store the second and minute values. Different Basic Timer interrupt-interval values are used for the simulator and the STK. This is selected with conditional assembly. On the STK, an exact 1-second interrupt interval is used. Using the ADT430 software simulator, a fast interrupt is used allowing a faster simulation run.

# 3 Experiments Specific to the STK

All of the experiments presented in the following section can only be tested (as coded) on the STK demo board, but not under the ADT430 simulation environment. The reason for this is that they require external hardware to stimulate a response from the MSP430, such as a button on P0.0. Buttons and other external circuitry are not connected to the software simulator. It is possible to link external stimulus files to any MSP430 I/O using the ADT430 simulation environment, but this is not covered in this report. These experiments can be assembled using the ADT430 simulator, but can not be run under the ADT430 simulation environment – they are coded to run on the STK.

## 3.1 Polling P0.0 User Button

See *Software Listing stk_p0_0.asm*

A button is connected to I/O pin P0.0 on the STK demo board and is normally pulled high. Pressing the button causes P0.0 to go low. Any readable memory-mapped register anywhere in the MSP430 memory map can be directly tested using the bit test (*bit*) instruction. It is not necessary to move an operand to a CPU register for manipulation. P0.0 is tested directly using:

```
Mainloop    bit.b   #P0IN_0,&P0IN      ; P0.0 Pressed?
```

In this case, the logical result of the bit test is placed in the carry bit, which directs a conditional jump that follows the bit test. If the button is pressed, *On* is displayed on the STK LCD; if the button not pressed, *Off* is displayed.

## 3.2 Interrupt-Driven Menu Using P0.0 User Button

See *Software Listing stk_key.asm*

In this experiment, pressing the button connected to P0.0 causes an interrupt. P0.0 ISR increments the branch index pointer *Index*. Each time a button is pressed the index register gets doubled-incremented in the P0.0 ISR, causing the program to branch to the next mode as it goes through *Mainloop:*

```
Mainloop    br      Table(Index)        ; Branch to mode
            .even
Table       .word   OFF
            .word   Mode1
            .word   Mode2
            .word   Mode3
```

The program will display OFF, *1*, *2,* or *3* on the STK LCD to indicate the mode of operation. Using branch tables accelerates program execution by eliminating flag polling—the program knows exactly where to go. This is a type of event-driven programming technique. *Index* must be a CPU register, since indexed addressing, which does require a CPU register, is used.

## 3.3 Using the 14-Bit ADC to Read the TSL252 Light Sensor

See *Software Listing stk_252.asm*

This experiment reads the value of relative-light of a TSL252 sensor on the STK, and displays this value on the LCD. The TSL252 installed on the STK is connected to the 14-bit ADC channel A3, and powered by I/O pins P0.6 and P0.7. Additionally, an LM285–25 voltage reference is installed and connected to pin Svcc and is powered by I/O P0.5. This program normally runs in LPM3. The Basic Timer interrupts the system and activates code execution every 250 ms, which samples the TSL252 voltage and displays the value on STK LCD.

## 3.4 XBUF as a Tone Generator

See *Software Listing stk_xbuf.asm*

All MSP430x3xx family members include a simple external-clock buffer (XBUF), which is a simple square-wave generator. This experiment uses XBUF to output a tone burst every 125 ms. XBUF can output MCLK, ACLK, ACLK/2, or ACLK/4. The following operation will output ACLK/4, which is 32,768/4, or 8,192 Hz:

```
        mov.b   #CBSEL_ACLK_DIV4+CBE,&CBCTL ; 8khz on XBUF
```

Connect a low-power buzzer (such as Radio Shack™ P/N 273–074) with the positive terminal to XBUF and the negative to ground. When the program is running, a short-tone burst can be heard every 125 ms. The XBUF feature is simple but useful with buzzers and D/A converters, or as a clock source for another device. In this experiment, the Basic Timer is used to define the 125-ms interval. The clock-buffer control register (CBCTL) which configures the XBUF is a write-only register. This means that only move operations can be used on CBCTL, as bit manipulation instructions are invalid (bit manipulation is a form of read/write).

## 3.5 Multitasking and Using TPx as Outputs

See *Software Listing stk_led2.asm*

In this experiment, two simple tasks are managed simultaneously. I/O P0.0 is polled in *Mainloop*, and the watchdog is used in the background as a 250–ms interval timer. Also, the four Timer/Port pins (TPx) are used as digital outputs. If the Timer/Port is not used as an A/D converter, the six available TPx pins can be used as digital outputs with 3-state capability. The two timer/counters of the Timer/Port are also available (but not used in this experiment). Connect two low-current LEDs (LED1 and LED2 (such as Radio Shack™ P/N 276–310) to the STK's. Connect LED1 with the anode to TP0 and the cathode to TP1, and LED2 with the anode to TP2 and the cathode to TP3. LED1 will toggle continuously at 250 ms, timed by the watchdog interrupt. LED2 will only light if the button on P0.0 is pressed.

## 3.6 Using the Timer/Port to Measure a Thermistor

See *Software Listing stk_temp.asm*

This experiment measures temperature and displays the value on the STK LCD. The Timer/Port is used to perform a slope-type A/D conversion comparing a temperature-sensitive thermistor to a 10-kΩ resistor. The concept is described in the 1998 *MSP430 Metering Application Report*. A 0.1 µf capacitor is charged to a fixed voltage and discharged through a reference resistor. The external device connection to the MSP is described below:

```
;                  ------
;       +-(10K)---|TP3
;       |          |
;       +-(Rx)----|TP1
;       |          |
;       +---------|CIN
;       |          |
;      --- 0.1uf   |
;      ---         |
;       |          |
;       +---------|Gnd
;                  ------
```

The discharge time is measured and stored. The capacitor is recharged and discharged through a thermistor. The discharge times are directly proportional to the resistance. All the exponential components of the RC charge/discharge cancel themselves out. A look-up table is used to translate resistance to temperature. For even greater accuracy, the comparator on the MSP430P325 can also be used. The thermistor used in this experiment is a Radio Shack™ P/N 271-110A.

Radio Shack is a registered trademark of Tandy Corporation.

# 4 Summary

While the MSP–STK430x320 is not a complete development tool, many *proof of concept* experiments can be evaluated. This report provides several such experiments and hopes to familiarize STK users with the MSP430x32x capabilities. Once familiar with the experiments in this report, users can modify the code and try different ideas of their own.

Any code written and debugged on an STK can be ported to other MSP430 development tools and stand-alone device applications.

# 5 References

1. *Starter Kit Evaluation Kit Manual*, 1996
2. *MSP430 Family Architecture and Module Library*, 1996
3. *MSP430 Application Report*, 1998, Lutz Bierl

# Appendix A Software Listings

## A.1 Software Listing stk_1.asm

```
                .include     ”STD_DEF.ASM”         ; Standard Definitions Used
;***********************************************************************
;   MSP–STK430x320 and ADT430 Simulator Demonstration Program
;   Blink ”HELLO” on STK LCD
;
;               1   Run on STK Demo Board
;               0   Run on ADT430 Software Simulator
;              ---
STK     .set    0
;
;   ** MUST SET BREAKPOINT ON ”JMP Mainloop” for ADT430 Simulator **
;
;   M. Buccini
;   Americas Sales and Marketing
;   Texas Instruments, Inc
;   October 1999
;***********************************************************************
RAM_orig    .set    00200h              ; RAM start
ROM_orig    .set    0C000h              ; MSP430x325 ROM start
;                                       ;
            .if STK = 1                 ;
Stack       .set    003DEh              ; STK Stackpointer
I_vectors   .set    003FFh              ; STK relocated RAM interrupt vectors

Delay       .set    0FFFFh              ; Software Delay used for STK
Main        .equ    RAM_orig+20h        ; Program STK/EVK RAM start address
            .else                       ;
Stack       .set    RAM_orig+200h       ; Stack Pointer, top of ’x325 RAM
I_vectors   .set    0FFFFh              ; MSP430 interrupt vectors location
Delay       .set    10                  ; Software Delay used on simulator
Main        .equ    ROM_orig            ; Program ROM start address
            .endif                      ;
;
;-----------------------------------------------------------------------
            .sect ”MAIN”,Main           ;
;-----------------------------------------------------------------------
RESET       mov     #Stack,SP           ; Initialize stackpointer
;           Setup Modules
SetupWDT    mov     #WDTPW+WDTHOLD,&WDTCTL          ; Stop Watchdog Timer
SetupLCD    mov.b   #LCDON+LCD4MUX+LCDP2,&LCDCTL    ; STK LCD 4Mux, S0–S17
SetupBT     mov.b   #BTFRFQ1,&BTCTL                 ; STK LCD frame frequency
;           Clear  LCD Display RAM
ClearLCD    mov     #15,R15             ; Clear 15 bytes of LCD RAM
Clear1      clr.b   LCDM1–1(R15)        ; Clear LCD RAM to clear
            dec     R15                 ; All 15 bytes of LCD RAM?
            jnz     Clear1              ; Not done?
;           Mainloop
Mainloop    xor.b   #0BAh,&LCDM1+5      ; ”H”
            xor.b   #0ADh,&LCDM1+4      ; ”E”
            xor.b   #0A4h,&LCDM1+3      ; ”L”
```

```
            xor.b   #0A4h,&LCDM1+2      ; ”L”
            xor.b   #0B7h,&LCDM1+1      ; ”O”
Wait        mov     #Delay,R15          ; Delay value to Register R15
Wait1       dec     R15                 ; Decrement Register
            jnz     Wait1               ; Delay Over?
            jmp     Mainloop            ; Again
;-------------------------------------------------------------------------
            .sect   ”Int_Vect”,I_vectors–1  ; MSP430 Reset Vector
;-------------------------------------------------------------------------
            .word   RESET               ; POR, ext. Reset, Watchdog
```

## A.2 Software Listing stk_cnt.asm

```
                .include     "STD_DEF.ASM"         ; Standard Definitions Used
;*******************************************************************************
;   MSP-STK430x320 and ADT430 Simulator Demonstration Program
;   1 Digit BCD Counter Displayed on STK LCD
;
;               1   Run on STK Demo Board
;               0   Run on ADT430 Software Simulator
;              ---
STK     .set    0
;
;   ** MUST SET BREAKPOINT ON "jmp Mainloop" for ADT430 Simulator **
;   Description: This program demonstrates a one digit BCD counter.
;
;   M. Buccini
;   Americas Sales and Marketing
;   Texas Instruments, Inc
;   October 1999
;*******************************************************************************
RAM_orig    .set    00200h              ; RAM start
ROM_orig    .set    0C000h              ; MSP430x325 ROM start
            .if STK = 1                 ;
Stack       .set    003DEh              ; STK Stackpointer
I_vectors   .set    003FFh              ; STK relocated RAM interrupt vectors

Delay       .set    0FFFFh              ; Software Delay used for STK
Main        .equ    RAM_orig+20h        ; Program STK/EVK RAM start address
            .else                       ;
Stack       .set    RAM_orig+200h       ; Stack Pointer, top of 'x325 RAM
I_vectors   .set    0FFFFh              ; MSP430 interrupt vectors location
Delay       .set    10                  ; Software Delay used on simulator
Main        .equ    ROM_orig            ; Program ROM start address
            .endif                      ;
;
;            CPU Register Used
Counter     .equ    R5                  ; Register Used as BCD Digit Counter
;
;-------------------------------------------------------------------------------
            .sect "MAIN",Main           ;
;-------------------------------------------------------------------------------
RESET       mov     #Stack,SP           ; Initialize Stackpointer
            call    #Init_Sys           ; Initialize System
;
;           Mainloop                    ;
Mainloop    call    #Count              ; Count on LCD
            call    #Wait               ; Delay
            jmp     Mainloop            ; Again
;-------------------------------------------------------------------------------
Count       ; Subroutine - Increment Counter and Display 1 Digit on STK LCD
;-------------------------------------------------------------------------------
            setc                        ; Set Carry
            dadc.b  Counter             ; Decimal add carry to Counter
            and.b   #0Fh,Counter        ; Only low BCD digit in Counter
```

```
            mov.b   LCD_Tab(Counter),&LCDM1+3 ; Move low digit to LCD RAM
            ret                         ; Done
;-------------------------------------------------------------------------------
Wait        ; Subroutine – Software delay routine
;-------------------------------------------------------------------------------
            push    #Delay              ; Delay value –-> TOS
Wait1       dec     0(SP)               ; Decrement TOS
            jnz     Wait1               ; Delay over?
            incd    SP                  ; Clean up stack
            ret                         ; Done
;-------------------------------------------------------------------------------
Init_Sys    ; Subroutine – Configure Modules and Control Registers
;-------------------------------------------------------------------------------
SetupWDT    mov     #WDTPW+WDTHOLD,&WDTCTL          ; Stop Watchdog Timer
SetupLCD    mov.b   #LCDON+LCD4MUX+LCDP2,&LCDCTL    ; STK LCD 4Mux, S0–S17
SetupBT     mov.b   #BTFRFQ1,&BTCTL                 ; STK LCD frame frequency
;
ClearLCD    mov     #15,Counter         ; Clear 15 bytes of LCD RAM
Clear1      mov.b   #0,LCDM1–1(Counter)     ; Move #0 to LCD RAM to clear
            dec     Counter             ; All 15 bytes of LCD RAM?
            jnz     Clear1              ; Not done?
;***        CLR     Counter             ; Counter Cleared Above!
            ret                         ; Setup complete
;-------------------------------------------------------------------------------
;           MSP–STK/EVK LCD Definitions
;-------------------------------------------------------------------------------
a           .equ    01h
b           .equ    02h
c           .equ    10h
d           .equ    04h
e           .equ    80h
f           .equ    20h
g           .equ    08h
h           .equ    40h
;
LCD_Tab     .byte   a+b+c+d+e+f         ; displays ”0”
            .byte   b+c                 ; displays ”1”
            .byte   a+b+d+e+g           ; displays ”2”
            .byte   a+b+c+d+g           ; displays ”3”
            .byte   b+c+f+g             ; displays ”4”
            .byte   a+c+d+f+g           ; displays ”5”
            .byte   a+c+d+e+f+g         ; displays ”6”
            .byte   a+b+c               ; displays ”7”
            .byte   a+b+c+d+e+f+g       ; displays ”8”
            .byte   a+b+c+d+f+g         ; displays ”9”
;
;-------------------------------------------------------------------------------
            .sect   ”Int_Vect”,I_vectors–1  ; MSP430 Reset Vector
;-------------------------------------------------------------------------------
            .word   RESET               ; POR, ext. Reset, Watchdog
```

## A.3 Software Listing stk_bt1.asm

```
                .include     "STD_DEF.ASM"         ; Standard Definitions Used
;***********************************************************************
;   MSP430 STK and Simulator Demonstration Program
;   Basic Timer used as 125ms Wall Clock
;
;               1   Run on STK Demo Board
;               0   Run on ADT430 Software Simulator
;              ---
STK     .set    0
;
;   ** MUST SET BREAKPOINT ON "JMP Mainloop" for Simulator **
;
;   Description:  This program demonstrates how to use the Basic Timer as
;   a 125ms wall clock. System runs in LPM3, BTInt returns program to active
on
;   interrupt and toggle two LCD segments.
;
;   M. Buccini
;   Americas Sales and Marketing
;   Texas Instruments, Inc
;   October 1999
;***********************************************************************
RAM_orig    .set    00200h              ; RAM start
ROM_orig    .set    0C000h              ; 32x ROM start
            .if STK = 1                 ;
Stack       .set    003DEh              ; STK Stackpointer
I_vectors   .set    003FFh              ; STK relocated RAM interrupt vectors
Main        .equ    RAM_orig+20h        ; Program STK/EVK RAM start address
            .else                       ;
Stack       .set    RAM_orig+200h       ; Stack Pointer, top of 'x325 RAM
I_vectors   .set    0FFFFh              ; MSP430 interrupt vectors location
Main        .equ    ROM_orig            ; Program ROM start address
            .endif                      ;
;
;-----------------------------------------------------------------------
            .sect "MAIN",Main
;-----------------------------------------------------------------------
RESET       mov     #Stack,SP           ; Initialize Stackpointer
            call    #Init_Sys           ; Initialize System
                                        ;
Mainloop    bis     #LPM3,SR            ; Set SR bits for LPM3
            xor.b   #08h,&LCDM1+3
            xor.b   #08h,&LCDM1+4
            JMP     Mainloop            ; Endless Loop
                                        ;
;-----------------------------------------------------------------------
Init_Sys    ; Subroutine to Configure Modules and Control Registers
;-----------------------------------------------------------------------
SetupWDT    mov     #WDTPW+WDTHOLD,&WDTCTL          ; Stop Watchdog Timer
SetupLCD    mov.b   #LCDON+LCD4MUX+LCDP2,&LCDCTL    ; STK LCD 4Mux, S0–S17
;
            .if STK = 1
```

```
SetupBT     mov.b   #BTIP0+BTIP1+BTDIV+BTFRFQ1,&BTCTL   ; STK LCD 125ms Int.
            .else
SetupBT     mov.b   #BTIP0+BTFRFQ1,&BTCTL               ; STK LCD fast
            .endif
            bis.b   #BTIE,&IE2          ; Enable Basic Timer interrupt
            eint                        ; Enable interrupts
;
;           Clear LCD Display RAM
ClearLCD    clr.b   &LCDM1              ; Clear LCD RAM
            clr.b   &LCDM1+1            ; Clear LCD RAM
            clr.b   &LCDM1+2            ; Clear LCD RAM
            clr.b   &LCDM1+3            ; Clear LCD RAM
            clr.b   &LCDM1+4            ; Clear LCD RAM
            clr.b   &LCDM1+5            ; Clear LCD RAM
            clr.b   &LCDM1+6            ; Clear LCD RAM
            clr.b   &LCDM1+7            ; Clear LCD RAM
            ret                         ;
                                        ;
;-------------------------------------------------------------------------------
BT_Isr      ; Basic Timer ISR, CPU is returned to active on RETI
;-------------------------------------------------------------------------------
            bic     #LPM3,0(SP)         ; Clear SR LMP3 Bits, on top of stack
            reti                        ;
                                        ;
;-------------------------------------------------------------------------------
        .sect   ”Int_Vect”,I_vectors–31 ; MSP430x32x Interrupt Vectors
;-------------------------------------------------------------------------------
        .word   RESET                   ; Port0, bit 2 to bit 7
        .word   BT_Isr                  ; Basic Timer
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; Timer Port
        .word   RESET                   ; EOC from ADC
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; Watchdog/Timer, Timer mode
        .word   RESET                   ; no source
        .word   RESET                   ; Address of UART handler
        .word   RESET                   ; P0.0
        .word   RESET                   ; NMI, Osc. fault
        .word   RESET                   ; POR, ext. Reset, Watchdog
```

## A.4 Software Listing stk_wdt.asm

```
                .include     ”STD_DEF.ASM”         ; Standard Definitions Used
;***********************************************************************
;   MSP–STK430x320 and ADT430 Simulator Demonstration Program
;   WDT Interval Timer used for 99 Count Down Displayed on STK LCD
;
;               1   Run on STK Demo Board
;               0   Run on ADT430 Software Simulator
;              ---
STK     .set    1
;
;   ** SET BREAKPOINT AT ”reti” in WDTInt ISR for ADT430 Simulator **
;
;   Description:  This program demonstrates a 99 second count down using the
;   WDT configured as interval timer. In conditional assembly, a different
;   value for the WDT interrupt interval is used for the STK and ADT430
;   Simulator – a shorter interval is used on ADT430 for fast simulation
;
;   M. Buccini
;   Americas Sales and Marketing
;   Texas Instruments, Inc
;   October 1999
;***********************************************************************
RAM_orig    .set    00200h              ; RAM start
ROM_orig    .set    0C000h              ; MSP430x325 ROM start
;
            .if STK = 1                 ;
Stack       .set    003DEh              ; STK Stackpointer
I_vectors   .set    003FFh              ; STK relocated RAM interrupt vectors
Main        .equ    RAM_orig+20h        ; Program STK/EVK RAM start address
            .else                       ;
Stack       .set    RAM_orig+200h       ; Stack Pointer, top of ’x325 RAM
I_vectors   .set    0FFFFh              ; MSP430 interrupt vectors location
Main        .equ    ROM_orig            ; Program ROM start address
            .endif                      ;
;           RAM Used
Counter     .equ    021Eh               ; Byte for counting of seconds
;
;-----------------------------------------------------------------------
            .sect ”MAIN”,Main           ;
;-----------------------------------------------------------------------
RESET       mov     #Stack,SP           ; Initialize Stackpointer
            call    #Init_Sys           ; Initialize System
                                        ;
Mainloop    jmp     Mainloop            ; Endless Loop
                                        ;
;-----------------------------------------------------------------------
Init_Sys    ; Configure Modules and Control Registers
;-----------------------------------------------------------------------
            bis.b   #WDTIE,&IE1         ; Enable Watchdog Timer
;
           .if STK = 1
SetupWDT    mov     #WDT_ADLY_1000,&WDTCTL   ; WDT is a 1s Interval Timer
```

```
            .else
SetupWDT    mov     #WDT_ADLY_1_9,&WDTCTL   ; WDT is a 1.9ms  Interval Timer
            .endif
;
SetupBT     mov.b   #BTFRFQ1,&BTCTL     ; STK LCD frame frequency
SetupLCD    mov.b   #LCDON+LCD4MUX+LCDP2,&LCDCTL    ; STK LCD 4Mux, S0–S17
ClearRAM    clr.b   Counter             ; Clear byte Counter
            eint                        ; Enable interrupts
ClearLCD    mov     #15,R15             ; 15 LCD mem locations to clear
Clear1      mov.b   #0,LCDM1–1(R15)     ; Write zeros in LCD RAM locations
            dec     R15                 ; All LCD mem clear?
            jnz     Clear1              ; More LCD mem to clear go
            ret                         ;
                                        ;
;-------------------------------------------------------------------------------
WDT_Isr     ; WDT ISR counts down Decimally Counter and Displays on STK LCD
;-------------------------------------------------------------------------------
            dadd.b  #099h,&Counter      ; BCD Decrement by one
;                                       ; Fall through to Display
;-------------------------------------------------------------------------------
Display     ; Display Counter values on STK LCD
            ; CPU Registers R14 and R13 Used Temporarily and not saved.
;-------------------------------------------------------------------------------
            mov.b   Counter,R14         ; Seconds BCD value moved to R14
OutLCD      mov.b   R14,R13             ; Copy BCD value in R14 to R13
            rra.b   R13                 ; Right Shift
            rra.b   R13                 ;
            rra.b   R13                 ;
            rra.b   R13                 ; 4 times to expose high nibble
            and.b   #0Fh,R14            ; Only low nibble now in R14
            and.b   #0Fh,R13            ; Only high nibble now in R13
            mov.b   LCD_Tab(R14),&LCDM1+3   ; Low nibble to LCD
            mov.b   LCD_Tab(R13),&LCDM1+4   ; High nibble to LCD, over left
            reti
                                        ;
;-------------------------------------------------------------------------------
;           STK LCD Type
;-------------------------------------------------------------------------------
a           .equ    01h
b           .equ    02h
c           .equ    10h
d           .equ    04h
e           .equ    80h
f           .equ    20h
g           .equ    08h
h           .equ    40h

LCD_Tab     .byte   a+b+c+d+e+f         ; displays ”0”
            .byte   b+c                 ; displays ”1”
            .byte   a+b+d+e+g           ; displays ”2”
            .byte   a+b+c+d+g           ; displays ”3”
            .byte   b+c+f+g             ; displays ”4”
            .byte   a+c+d+f+g           ; displays ”5”
```

```
            .byte   a+c+d+e+f+g         ; displays "6"
            .byte   a+b+c               ; displays "7"
            .byte   a+b+c+d+e+f+g       ; displays "8"
            .byte   a+b+c+d+f+g         ; displays "9"
;-------------------------------------------------------------------------------
        .sect   "Int_Vect",I_vectors-31 ; MSP430x32x Interrupt Vectors
;-------------------------------------------------------------------------------
        .word   RESET                   ; Port0, bit 2 to bit 7
        .word   RESET                   ; Basic Timer
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; Timer Port
        .word   RESET                   ; EOC from ADC
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; Timer_A
        .word   RESET                   ; Timer_A
        .word   WDT_Isr                 ; Watchdog/Timer, Timer mode
        .word   RESET                   ; no source
        .word   RESET                   ; Address of UART handler
        .word   RESET                   ; P0.0
        .word   RESET                   ; NMI, Osc. fault
        .word   RESET                   ; POR, ext. Reset, Watchdog
```

## A.5 Software Listing stk_rtc.asm

```
                .include     ”STD_DEF.ASM”         ; Standard Definitions Used
;***********************************************************************
;   MSP–STK430x320 and ADT430 Simulator Demonstration Program
;   RTC Using Basic Timer
;
;
;               1   Run on STK Demo Board
;               0   Run on ADT430 Software Simulator
;              ---
STK     .set    0
;
;   ** MUST SET BREAKPOINT ON ”jmp Mainloop” for ADT430 Simulator **
;
;   Description:  This program demonstrates a Second and Minute RTC.
;
;   M. Buccini
;   Americas Sales and Marketing
;   Texas Instruments, Inc
;   October 1999
;***********************************************************************
RAM_orig    .set    00200h              ; RAM start
ROM_orig    .set    0C000h              ; MSP430x325 ROM start
;
           .if STK = 1
I_vectors   .set    003FFh              ; STK Interrupt vectors in RAM
Stack       .set    003DEh              ; STK Stackpointer
Main        .equ    RAM_orig+20h        ; Program STK/EVK RAM start address
           .else
I_vectors   .set    0FFFFh              ; Device Interrupt vectors
Stack       .set    RAM_orig+200h       ; Stack Pointer, top of 325 of RAM
Main        .equ    ROM_orig            ; Program ROM start address
            .endif
;
;           RAM Used
SEC         .equ    021Eh               ; Byte for counting of seconds
MIN         .equ    021Fh               ; Byte for counting of minutes
;
;-----------------------------------------------------------------------
            .sect ”MAIN”,Main
;-----------------------------------------------------------------------
RESET       mov     #Stack,SP           ; Initialize stackpointer
            call    #Init_Sys           ; Prepare LCD an basic timer
;                                       ;
Mainloop    bis     #LPM3,SR            ; Set SR bits for LPM3
            call    #Clock              ; Update Clock
            call    #Display            ; Update LCD display
            jmp     Mainloop            ; Endless Loop
                                        ;
;-----------------------------------------------------------------------
Clock       ; Subroutine to Update clock SEC and MIN
;-----------------------------------------------------------------------
            setc                        ; Entry every second
```

```
            dadc.b  SEC                 ; Increment seconds, add carry bit
            cmp.b   #060h,SEC           ; One minute elapsed?
            jlo     Clockend            ; No, return (C = 0)
            clr.b   SEC                 ; Yes, clear seconds (C = 1)
            dadc.b  MIN                 ; Increment minutes if set carry
            cmp.b   #060h,MIN           ; Sixty minutes elapsed?
            jlo     Clockend            ; No
            clr.b   MIN                 ; Clear minutes
Clockend    ret                         ;
                                        ;
;-------------------------------------------------------------------------------
Display     ; Subroutine to Display Clock values SEC, MIN to STK LCD.
            ; CPU Registers R15, R14 and R13 Used Temporarily and not saved.
;-------------------------------------------------------------------------------
            mov.b   #LCDM1+1,R15        ; R15 points to first LCD location
            mov.b   SEC,R14             ; Seconds BCD value moved to R14
            call    #OutLCD             ; Shift will shift BCD value to LCD
            mov.b   #08h,0(R15)         ; Insert "-" between min and sec
            inc.b   R15                 ;
            mov.b   MIN,R14             ; Minutes BCD value to R5
                                        ;
OutLCD      mov.b   R14,R13             ; Copy BCD value in R14 to R13
            rra.b   R13                 ; Right Shift
            rra.b   R13                 ;
            rra.b   R13                 ;
            rra.b   R13                 ; 4 times to expose high nibble
            and.b   #0Fh,R14            ; Only low nibble now in R5
            and.b   #0Fh,R13            ; Only high nibble now in R6
            mov.b   LCD_Tab(R14),0(R15) ; Low nibble to LCD
            mov.b   LCD_Tab(R13),1(R15) ; High nibble to LCD, over left
            incd.b  R15                 ; Double inc R15, LCD mem pointer
            ret                         ;
                                        ;
;-------------------------------------------------------------------------------
Init_Sys    ; Subroutine to Configure Modules and Control Registers
;-------------------------------------------------------------------------------
SetupWDT    mov     #WDTPW+WDTHOLD,&WDTCTL          ; Stop Watchdog Timer
SetupLCD    mov.b   #LCDON+LCD4MUX+LCDP2,&LCDCTL    ; STK LCD 4Mux, S0-S17
            .if STK = 1
SetupBT     mov.b   #BTIP1+BTIP2+BTDIV+BTFRFQ1,&BTCTL ; True 1 second interrupt
            .else
SetupBT     mov.b   #BTIP0+BTFRFQ1,&BTCTL   ; Fast Interrupt for ADT430
            .endif
            bis.b   #BTIE,&IE2          ; Enable Basic Timer interrupt
            clr.b   &BTCNT1             ; Clear BT counter 1
            clr.b   &BTCNT2             ; Clear BT counter 2
ClearRAM    clr     SEC                 ; Word Operation Clears SEC and MIN
            eint                        ; Enable interrupts
;
ClearLCD    mov     #15,R15             ; 15 LCD mem locations to clear
Clear1      mov.b   #0,LCDM1-1(R15)     ; Write zeros in LCD RAM locations
            dec     R15                 ; All LCD mem clear?
            jnz     Clear1              ; More LCD mem to clear go
```

```
            ret                         ;
                                        ;
;-------------------------------------------------------------------------------
BT_Isr      ; Basic Timer ISR, CPU is returned to active on RETI
;-------------------------------------------------------------------------------
            bic     #LPM3,0(SP)         ; Clear SR LMP3 Bits, on top of stack
            reti                        ;
                                        ;
;-------------------------------------------------------------------------------
;           STK LCD Type
;-------------------------------------------------------------------------------
a           .equ    01h
b           .equ    02h
c           .equ    10h
d           .equ    04h
e           .equ    80h
f           .equ    20h
g           .equ    08h
h           .equ    40h

LCD_Tab     .byte   a+b+c+d+e+f         ; displays ”0”
            .byte   b+c                 ; displays ”1”
            .byte   a+b+d+e+g           ; displays ”2”
            .byte   a+b+c+d+g           ; displays ”3”
            .byte   b+c+f+g             ; displays ”4”
            .byte   a+c+d+f+g           ; displays ”5”
            .byte   a+c+d+e+f+g         ; displays ”6”
            .byte   a+b+c               ; displays ”7”
            .byte   a+b+c+d+e+f+g       ; displays ”8”
            .byte   a+b+c+d+f+g         ; displays ”9”
;-------------------------------------------------------------------------------
        .sect   ”Int_Vect”,I_vectors-31 ; MSP430x32x Interrupt Vectors
;-------------------------------------------------------------------------------
        .word   RESET                   ; Port0, bit 2 to bit 7
        .word   BT_Isr                  ; Basic Timer
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; Timer Port
        .word   RESET                   ; EOC from ADC
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; Timer_A
        .word   RESET                   ; Timer_A
        .word   RESET                   ; Watchdog/Timer, Timer mode
        .word   RESET                   ; no source
        .word   RESET                   ; Address of UART handler
        .word   RESET                   ; P0.0
        .word   RESET                   ; NMI, Osc. fault
        .word   RESET                   ; POR, ext. Reset, Watchdog
```

## A.6 Software Listing stk_p0_0.asm

```
                .include     ”STD_DEF.ASM”         ; Standard Definitions Used
;***********************************************************************
;   MSP-STK430x320 Demonstration Program
;   Polling P0.0 and Displaying ”Off” or ”On on the STK LCD.
;
;   *This program runs only on STK not the ADT430 Simulator*
;
;   Description:  This program demonstrates how to poll P0.0 on the STK.
;
;   M. Buccini
;   Americas Sales and Marketing
;   Texas Instruments, Inc
;   October 1999
;***********************************************************************
RAM_orig    .set    00200h              ; RAM start
ROM_orig    .set    0C000h              ; MSP430x325 ROM start
I_vectors   .set    003FFh              ; STK Interrupt vectors in RAM
Stack       .set    003DEh              ; STK Stackpointer
Main        .equ    RAM_orig+20h        ; Program STK/EVK RAM start address
;-----------------------------------------------------------------------
            .sect ”MAIN”,RAM_orig+20h
;-----------------------------------------------------------------------
RESET       mov     #Stack,SP           ; Initialize stackpointer
            call    #Init_Sys           ; Initialize Peripherals

Mainloop    bit.b   #P0IN_0,&P0IN      ; P0.0 Pressed?
            jc      OFF                ; Jump if P0.0 is not pressed (high)
;
;           Display Modes
ON          mov.b   #0B7h,&LCDM1+4      ; ”O” Display ”On”
            mov.b   #098h,&LCDM1+3      ; ”n”
            clr.b   &LCDM1+2            ; clear
            jmp     Mainloop
OFF         mov.b   #0B7h,&LCDM1+4      ; ”O” Display ”OFF”
            mov.b   #0A9h,&LCDM1+3      ; ”F”
            mov.b   #0A9h,&LCDM1+2      ; ”F”
            jmp     Mainloop
;-----------------------------------------------------------------------
Init_Sys    ; Subroutine to Configure Modules and Control Registers
;-----------------------------------------------------------------------
SetupWDT    mov     #WDTPW+WDTHOLD,&WDTCTL          ; Stop Watchdog Timer
SetupLCD    mov.b   #LCDON+LCD4MUX+LCDP2,&LCDCTL    ; STK LCD 4Mux, S0-S17
SetupBT     mov.b   #BTFRFQ1,&BTCTL                 ; STK LCD frame frequency
;
ClearLCD    mov     #15,R15             ; 15 bytes of LCD RAM to clear
Clear1      clr.b   LCDM1-1(R15)        ; Clear LCD display RAM
            dec     R15                 ; All 15 bytes
            jnz     Clear1              ; Not done?
            ret
;-----------------------------------------------------------------------
            .sect   ”Int_Vect”,I_vectors-1  ; MSP430 Reset Vector
;-----------------------------------------------------------------------
```

```
                .word   RESET               ; POR, ext. Reset, Watchdog
```

## A.7 Software Listing stk_key.asm

```
                .include     ”STD_DEF.ASM”         ; Standard Definitions Used
;***********************************************************************
;   MSP430 STK Demonstration Program
;   P0.0 Interrupt Driven Key Pad
;
;   *Program coded to run only on STK not in the ADT430 Simulator*
;
;   Description:  This program demonstrates a key interface utilizing the
;   STK’s button connected to P0.0.  Each Press of the button will scroll to
;   the next mode.  Modes are ”OFF”, ”1”, ”2” and ”3”.
;
;   M. Buccini
;   Americas Sales and Marketing
;   Texas Instruments, Inc
;   October 1999
;***********************************************************************
RAM_orig    .set    00200h              ; RAM start
ROM_orig    .set    0C000h              ; MSP430x325 ROM start
;
I_vectors   .set    003FFh              ; STK Interrupt vectors in RAM
Stack       .set    003DEh              ; STK Stackpointer
Main        .equ    RAM_orig+20h        ; Program STK/EVK RAM start address
;           CPU Registers Used
Index       .equ    R5                  ; Index for branching
;
;-----------------------------------------------------------------------
            .sect ”MAIN”,Main
;-----------------------------------------------------------------------
RESET       mov     #Stack,SP           ; Initialize stackpointer
            call    #Init_Sys           ; Setup Peripherals

Mainloop    br      Table(Index)        ; Branch to mode
            .even
Table       .word   OFF
            .word   Mode1
            .word   Mode2
            .word   Mode3
;-----------------------------------------------------------------------
;   Display Modes
;-----------------------------------------------------------------------
OFF         mov.b   #0B7h,LCDM1+4       ; ”O” Display ”OFF”
            mov.b   #0A9h,LCDM1+3       ; ”F”
            mov.b   #0A9h,LCDM1+2       ; ”F”
            jmp     Mainloop
;
Mode1       mov.b   #12h,&LCDM1+3       ; Display ”1”
            jmp     ModeX               ; Common Part
;
Mode2       mov.b   #8Fh,&LCDM1+3       ; Display ”2”
            jmp     ModeX               ; Common Part
;
Mode3       mov.b   #1Fh,&LCDM1+3       ; Display ”3”
```

```
ModeX       mov.b   #00h,&LCDM1+4       ; Clear
            mov.b   #00h,&LCDM1+2       ; Clear
            jmp     Mainloop            ; Wait for next time interval
;
;-------------------------------------------------------------------------------
Init_Sys    ; Subroutine to Configure Modules and Control Registers
;-------------------------------------------------------------------------------
SetupWDT    mov     #WDTPW+WDTHOLD,&WDTCTL          ; Stop Watchdog Timer
SetupLCD    mov.b   #LCDON+LCD4MUX+LCDP2,&LCDCTL    ; STK LCD 4Mux, S0–S17
SetupBT     mov.b   #BTFRFQ1,&BTCTL                 ; STK LCD frame frequency
            bis.b   #P0IE_0,&IE1        ; Enable P0.0 Interrupt
            clr     Index               ; Clear mode branching register
            eint                        ; Enable interrupts
ClearLCD    mov     #15,R15             ; 15 bytes of LCD RAM to clear
Clear1      mov.b   #0,LCDM1–1(R15)     ; Move #0 to LCD display RAM
            dec     R15                 ; All 15 bytes
            jnz     Clear1              ; Not done?
            ret
;-------------------------------------------------------------------------------
P0_0_Isr    ; P0.0 ISR, Increment Index register.
;-------------------------------------------------------------------------------
            incd    Index               ; Double increment branch index
            cmp     #08,Index           ; Out of range of branch table?
            jl      Debounce            ; Jump if still in range less than 3
            clr     Index               ; Reset mode index to 0
Debounce    bic.b   #P0IE_0,&IE1        ; Disable P0.0 Interrupt
            eint                        ; Enable other interrupt during
                                        ; debounce!!
            push    #020000             ; Load a SW delay to TOS
Delay       dec     0(SP)               ; DEC SW Delay on TOS
            jnz     Delay               ; Delay Done?
            incd    SP                  ; Clean up Stack
;  *P0.0 interrupt flag is re–cleared in case key bounce*
            bic.b   #P0IFG_0,&IFG1      ; re–clear P0.0 flag in case key
bounce
            bis.b   #P0IE_0,&IE1        ; re–enable P0.0 Interrupt
            reti                        ; flag is cleared on interrupt also.
;-------------------------------------------------------------------------------
            .sect   ”Int_Vect”,I_vectors–31 ; MSP430x32x Interrupt Vectors
;-------------------------------------------------------------------------------
            .word   RESET               ; Port0, bit 2 to bit 7
            .word   RESET               ; Basic Timer
            .word   RESET               ; no source
            .word   RESET               ; no source
            .word   RESET               ; Timer Port
            .word   RESET               ; EOC from ADC
            .word   RESET               ; no source
            .word   RESET               ; no source
            .word   RESET               ; no source
            .word   RESET               ; no source
            .word   RESET               ; Watchdog/Timer, Timer mode
            .word   RESET               ; no source
            .word   RESET               ; Address of UART handler
```

```
                .word   P0_0_Isr            ; P0.0
                .word   RESET               ; NMI, Osc. fault
                .word   RESET               ; POR, ext. Reset, Watchdog
```

## A.8 Software Listing stk_252.asm

```
_CPU_   .set        3                       ; ID32X
        .include     ”STD_DEF.ASM”          ; Standard Definitions Used
;*******************************************************************************
;   MSP–STK430x320 Demonstration Program
;   TSL252 Light Meter
;
;   *This program runs only on STK not the ADT430 Simulator*
;
;   Description:  This program demonstrates using the MSP430x32x 14–bit ADC to
;   read the STK’s on–board TSL252 Light to Voltage Converter.  Programmed
;   specifically for MSP–STK430x320. Office Lights and noise from a PC RS232
;   powered demo DO cause the light readings to fluctuate.
;
;   M. Buccini
;   Americas Sales and Marketing
;   Texas Instruments, Inc
;   October 1999
;*******************************************************************************
RAM_orig    .set    00200h              ; RAM start
ROM_orig    .set    0C000h              ; MSP430x325 ROM start
;
I_vectors   .set    003FFh              ; STK Interrupt vectors in RAM
Stack       .set    003DEh              ; STK Stackpointer
Delay       .set    0FFFFh              ; Software Delay used for STK
Main        .equ    RAM_orig+20h        ; Program STK/EVK RAM start address
;
;-------------------------------------------------------------------------------
            .sect ”MAIN”,Main
;-------------------------------------------------------------------------------
RESET       mov     #Stack,SP           ; Initialize Stackpointer
            call    #Init_Sys           ; Initialize System
;
Mainloop    bis     #LPM3,SR            ; Enter LPM3, stop here
            call    #Read252            ; Read, convert and display ADC value
            jmp     Mainloop
;
;-------------------------------------------------------------------------------
Read252     ; Subroutine to Power up 14–bit ADC and then read value on A3.
;-------------------------------------------------------------------------------
            mov     #ADIN_A3+ADCSRCOFF+ADAUTO,&ACTL    ;A3, auto, SVcc
            bis     #ADSOC,&ACTL        ; Start next ADC conversion
            bis     #CPUOFF,SR          ; Turn off CPU, quite ADC, stop here
            xor.b   #04h,&LCDM1+7       ; Blink ”+” on STK LCD, show activity
            mov     &ADAT,R12           ; Get ADC–value in R12
;            ret                         ; Wait for next time interval
;
;-------------------------------------------------------------------------------
BIN2BCD     ; Subroutine converts Binary Number (R12) –> Packed BCD (R13/R14)
            ; CPT Registers used R12, R13, R14, R15 and not saved.
;-------------------------------------------------------------------------------
            mov     #16,R15             ; Loop Counter
            clr     R14                 ; 0 –> RESULT MSD
```

```
            clr     R13                 ; 0 -> RESULT LSD
BIN1        rla     R12                 ; Binary MSB to carry
            dadd    R13,R13             ; RESULT x2 LSD
            dadd    R14,R14             ; MSD
            dec     R15                 ; Through?
            jnz     BIN1                ; Not through
;            ret
;
;-------------------------------------------------------------------------------
DISBCD5     ; Subroutine displays 4 packed BCD in R13, 1 digit in R14 --> LCD.
            ; CPU registers used R13, R14, R15 and not saved.
;-------------------------------------------------------------------------------
            mov     #LCDM1+1,R15        ; Point to LCD display RAM
DIS1        push    R13                 ; Push Packed BCD number
            rra     0(SP)               ; Rotate a nibble (BCD digit)
            rra     0(SP)               ; Rotate a nibble (BCD digit)
            rra     0(SP)               ; Rotate a nibble (BCD digit)
            rra     0(SP)               ; Rotate a nibble (BCD digit)
            and     #0Fh,R13            ; Expose single digit
            mov.b   LCD_Tab(R13),0(R15)  ; Move digit to LCD display RAM
            pop     R13
            inc     R15                 ; Point to next LCD display RAM
            cmp     #LCDM1+5,R15        ; All digits displayed?
            jl      DIS1                ; Still in 4 digit range?
            and     #0Fh,R14            ; Display 1 digit from R14
            mov.b   LCD_Tab(R14),0(R15)    ; Digit to LCD display RAM
            ret
;
;-------------------------------------------------------------------------------
Init_Sys    ; Subroutine sets up  Modules and Controls Registers
;-------------------------------------------------------------------------------
SetupWDT    mov     #WDTPW+WDTHOLD,&WDTCTL          ; Stop Watchdog Timer
SetupLCD    mov.b   #LCDON+LCD4MUX+LCDP2,&LCDCTL    ; STK LCD 4Mux, S0-S17
SetupBT     mov.b   #BTIP2+BTDIV+BTFRFQ1,&BTCTL  ; STK LCD,.250ms Interrupt
            bis.b   #BTIE+ADIE,&IE2     ; Enable Basic Timer, ADC interrupts
SetupP0     mov.b   #P0DIR_7+P0DIR_6+P0DIR_5,&P0DIR     ; P0.7,6,5 outputs
            bis.b   #P0OUT_7+P0OUT_6+P0OUT_5,P0OUT      ; Set P0.7,6,5
            eint                        ; Enable interrupts
ClearLCD    mov     #15,R15             ; 15 bytes of LCD RAM to clear
Clear1      mov.b   #0,LCDM1-1(R15)     ; Move #0 to LCD display RAM
            dec     R15                 ; All 15 bytes
            jnz     Clear1              ; Not done?
            ret                         ;
;
;-------------------------------------------------------------------------------
ADC14_Isr   ; 14-bit ADC ISR powers off 14-bit ADC, fall through to BT_ISR
BT_Isr      ; Basic Timer ISR exit any LPMx mode
;-------------------------------------------------------------------------------
            bis     #ADPD,&ACTL         ; Power down ADC
            mov     #GIE,0(SP)          ; Clear LPMx and enable int. on RETI
            reti
;
;-------------------------------------------------------------------------------
```

```
;           MSP-STK/EVK LCD Definitions
;-------------------------------------------------------------------------------
a           .equ    01h
b           .equ    02h
c           .equ    10h
d           .equ    04h
e           .equ    80h
f           .equ    20h
g           .equ    08h
h           .equ    40h
;
LCD_Tab     .byte   a+b+c+d+e+f         ; displays "0"
            .byte   b+c                 ; displays "1"
            .byte   a+b+d+e+g           ; displays "2"
            .byte   a+b+c+d+g           ; displays "3"
            .byte   b+c+f+g             ; displays "4"
            .byte   a+c+d+f+g           ; displays "5"
            .byte   a+c+d+e+f+g         ; displays "6"
            .byte   a+b+c               ; displays "7"
            .byte   a+b+c+d+e+f+g       ; displays "8"
            .byte   a+b+c+d+f+g         ; displays "9"
;-------------------------------------------------------------------------------
        .sect   "Int_Vect",I_vectors-31 ; MSP430x32x Interrupt Vectors
;-------------------------------------------------------------------------------
        .word   RESET                   ; Port0, bit 2 to bit 7
        .word   BT_Isr                  ; Basic Timer
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; Timer Port
        .word   ADC14_Isr               ; EOC from ADC
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; Watchdog/Timer, Timer mode
        .word   RESET                   ; no source
        .word   RESET                   ; Address of UART handler
        .word   RESET                   ; P0.0
        .word   RESET                   ; NMI, Osc. fault
        .word   RESET                   ; POR, ext. Reset, Watchdog
```

## A.9 Software Listing stk_xbuf.asm

```
                .include     ”STD_DEF.ASM”         ; Standard Definitions Used
;***********************************************************************
;   MSP430 STK and Simulator Demonstration Program
;   XBUF used to send Tone Burst
;
;
;   ** MUST SET BREAKPOINT ON ”JMP Mainloop” for Simulator **
;
;   Description:  This program demonstrates using XBUF to provide a tone
burst.
;   Basic Timer used to define interval for tone burst.
;
;   M. Buccini
;   Americas Sales and Marketing
;   Texas Instruments, Inc
;   October 1999
;***********************************************************************
**
RAM_orig    .set    00200h              ; RAM start
ROM_orig    .set    0C000h              ; 32x ROM start
Stack       .set    003DEh              ; STK Stackpointer
I_vectors   .set    003FFh              ; STK relocated RAM interrupt vectors
Main        .equ    RAM_orig+20h        ; Program STK/EVK RAM start address
Delay       .set    05000               ; Software Delay used for STK
;
;-----------------------------------------------------------------------
            .sect ”MAIN”,Main
;-----------------------------------------------------------------------
RESET       mov     #Stack,SP           ; Initialize Stackpointer
            call    #Init_Sys           ; Initialize System
                                        ;
Mainloop    bis     #LPM3,SR            ; Set SR bits for LPM3
            mov.b   #CBSEL_ACLK_DIV4+CBE,&CBCTL ; 8khz on XBUF
            call    #Wait
            mov.b   #00h,&CBCTL         ; XBUF off
            jmp     Mainloop            ; Endless Loop
                                        ;
Wait        push    #Delay              ; Delay value –-> TOS
Wait1       dec     0(SP)               ; Decrement TOS
            jnz     Wait1               ; Delay Over?
            incd    SP                  ; Clean up stack
            ret
;-----------------------------------------------------------------------
Init_Sys    ; Subroutine to Configure Modules and Control Registers
;-----------------------------------------------------------------------
SetupWDT    mov     #WDTPW+WDTHOLD,&WDTCTL          ; Stop Watchdog Timer
SetupLCD    mov.b   #LCDON+LCD4MUX+LCDP2,&LCDCTL    ; STK LCD 4Mux, S0–S17
;
SetupBT     mov.b   #BTIP0+BTIP2+BTDIV+BTFRFQ1,&BTCTL   ; STK LCD 500ms Int.
            bis.b   #BTIE,&IE2          ; Enable Basic Timer interrupt
            eint                        ; Enable interrupts
            ret                         ;
```

```
                                        ;
;-------------------------------------------------------------------------
BT_Isr      ; Basic Timer ISR, CPU is returned to active on RETI
;-------------------------------------------------------------------------
            bic     #LPM3,0(SP)         ; Clear SR LMP3 Bits, on top of stack
            reti                        ;
                                        ;
;-------------------------------------------------------------------------
        .sect   ”Int_Vect”,I_vectors–31 ; MSP430x32x Interrupt Vectors
;-------------------------------------------------------------------------
        .word   RESET                   ; Port0, bit 2 to bit 7
        .word   BT_Isr                  ; Basic Timer
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; Timer Port
        .word   RESET                   ; EOC from ADC
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; Watchdog/Timer, Timer mode
        .word   RESET                   ; no source
        .word   RESET                   ; Address of UART handler
        .word   RESET                   ; P0.0
        .word   RESET                   ; NMI, Osc. fault
        .word   RESET                   ; POR, ext. Reset, Watchdog
```

## A.10 Software Listing stk_led2.asm

```
                .include     ”STD_DEF.ASM”         ; Standard Definitions Used
;*******************************************************************************
;   MSP–STK430x320 Demonstration Program
;   Multitasking LED outputs
;
;   Description: This program will toggle TP0 @ 250ms timed by the WDT
;   configured as an interval timer and set TP2 only if the button on the STK
;   is pressed.  P0.0 button is normally pulled high on STK.
;
;   M. Buccini
;   Americas Sales and Marketing
;   Texas Instruments, Inc
;   October 1999
;*******************************************************************************
RAM_orig    .set    00200h              ; RAM start
ROM_orig    .set    0C000h              ; MSP430x325 ROM start
;
I_vectors   .set    003FFh              ; STK Interrupt vectors in RAM
Stack       .set    003DEh              ; STK Stackpointer
Main        .equ    RAM_orig+20h        ; Program STK/EVK RAM start address
;-------------------------------------------------------------------------------
            .sect ”MAIN”,Main
;-------------------------------------------------------------------------------
RESET       mov     #Stack,SP           ; Initialize Stackpointer
            call    #Init_Sys           ; Initialize System

MainLoop    bit.b   #P0IN_0,&P0IN       ; P0.0 Low?
            jc      Off                 ; P0.0 high –-> LED off
On          bis.b   #TPD_2,&TPD         ; Set TP2 LED on
            jmp     MainLoop            ; Again
Off         bic.b   #TPD_2,&TPD         ; Reset TP2
            jmp     MainLoop            ; Again
;-------------------------------------------------------------------------------
Init_Sys    ; Subroutine sets up  Modules and Controls Registers
;-------------------------------------------------------------------------------
SetupWDT    mov     #WDT_ADLY_250,&WDTCTL           ; WDT, ACLK, 250ms Timer
            bis.b   #WDTIE,IE1                      ; Enable WDT interrupt
SetupTP     mov.b   #TPE_3+TPE_2+TPE_1+TPE_0,&TPE   ; Enable TPD.0–3
            eint                        ; Enable interrupts
            ret                         ; Setup complete
                                        ;
;-------------------------------------------------------------------------------
WDT_Isr     ; Watchdog Timer ISR: Toggle TP0
;-------------------------------------------------------------------------------
            xor.b   #TPD_0,&TPD         ; Toggle TP0
            reti                        ;
                                        ;
;-------------------------------------------------------------------------------
        .sect   ”Int_Vect”,I_vectors–31 ; MSP430x32x Interrupt Vectors
;-------------------------------------------------------------------------------
        .word   RESET                   ; Port0, bit 2 to bit 7
        .word   RESET                   ; Basic Timer
```

```
.word   RESET                   ; no source
.word   RESET                   ; no source
.word   RESET                   ; Universal Timer/Port
.word   RESET                   ; EOC from ADC
.word   RESET                   ; no source
.word   RESET                   ; no source
.word   RESET                   ; no source
.word   RESET                   ; no source
.word   WDT_Isr                 ; Watchdog/Timer, Timer mode
.word   RESET                   ; no source
.word   RESET                   ; Address of UART handler
.word   RESET                   ; P0.0
.word   RESET                   ; NMI, Osc. fault
.word   RESET                   ; POR, ext. Reset, Watchdog
```

## A.11 Software Listing stk_temp.asm

```
                .include     ”STD_DEF.ASM”         ; Standard Definitions
Used
;*****************************************************************************
;   MSP430 Starter Kit Demonstration Program
;   Using Timer/Port used as 16–bit Slope ADC
;
;   *This program runs only on STK not the ADT430 Simulator*
;
;   Description: This program demonstrates reading a thermistor value,
;   converting to temperature in oF and displaying on STK LCD.
;
;                ------
;       +–(10K)---|TP3
;       |         |
;       +–(Rx)----|TP1
;       |         |
;       +---------|CIN
;       |         |
;      --- 0.1uf  |
;      ---        |
;       |         |
;       +---------|Gnd
;                 ------
;
;       Rx= P/N 271–110A, available from Radio Shack
;
;   M. Buccini
;   Americas Sales and Marketing
;   Texas Instruments, Inc
;   April 1999
;*****************************************************************************
RAM_orig    .set    00200h              ; RAM start
ROM_orig    .set    0C000h              ; MSP430x325 ROM start
;
I_vectors   .set    003FFh              ; STK Interrupt vectors in RAM
Stack       .set    003DEh              ; STK Stackpointer
Main        .equ    RAM_orig+20         ; Program STK/EVK RAM start address
;
;           CPU Registers Used for Integer Math
IRBIT       .equ     R15                ; Bit test register
IROP1       .equ     R14                ; First operand
IROP2L      .equ     R13                ; Second operand low word
IROP2H      .equ     R12                ; Second operand high word
IRACCL      .equ     R11                ; Result low word
IRACCH      .equ     R10                ; Result high word
RESULT      .equ     R9                 ; temp storage (CAN USE RAM)
TEMP        .equ     R8                 ; temp storage (CAN USE RAM)

;           Bit Definitions Used
REF         .equ     08h                ; Reference Resistor on TP3
Sensor      .equ     02h                ; Thermistor on TP1
;
```

```
;-------------------------------------------------------------------------
            .sect "MAIN",Main           ;
;-------------------------------------------------------------------------
RESET       mov     #Stack,SP           ; Initialize stackpointer
            call    #Init_Sys             ; Prepare Periferals
                                        ;
Mainloop    bis     #LPM3,SR            ; Enter LPM3, stop, save power
            call    #Measure            ; Measure Temperature
            jmp     Mainloop            ;
                                        ;
;-------------------------------------------------------------------------
;   Measure: Reference, Sensor and Branch to Calculate Subroutine
;-------------------------------------------------------------------------
Measure     mov     #REF,R6             ; Configure TP for REF measurements
            call    #Charge             ; Charge CAP and Measure Reference
            mov     RESULT,TEMP         ; Save REF discharge Time
            mov     #Sensor,R6          ; Configure TP for Sensor measurement
            call    #Charge             ; Charge CAP and Measure Sensor
            br      #Calculate          ; Do math and return to Mainloop
;
;           Capacitor is charged up with reference resistor
Charge      mov.b   #REF,&TPD           ; REF high to charge CAP
            clr.b   &TPCNT2             ; TPCNT2 Cleared
            clr.b   &TPCTL              ; TPCNT1 Disabled
            mov.b   #TPSSEL2+REF,&TPE   ; TPCNT2 On, CLK2=ACLK, REF Output
            bis     #LPM3,SR            ; Stop, save power
            clr.b   &TPE                ; Charge complete, REF to HiZ
;
;           Measure Discharge Time through TPx,  TPx passed in R6
Discharge   mov.b   #B16,&TPD           ; 16b counter, All TP lines low
            clr.b   &TPCNT1             ; Clear Timer
            clr.b   &TPCNT2             ; Clear Timer
            mov.b   #TPSSEL1+TPSSEL0+ENA+ENB,&TPCTL ;16-bit CLK=MCLK, CIN
            bis.b   R6,&TPE             ; R6 has REF or Sensor
;
;           Wait Until ADC complete EN1=1 Interrupt, MCLK remains on
            bis     #CPUOFF,SR          ; CPUOFF, Stop, save power
            bic.b   R6,&TPE             ; Disable Discharge and Continue
;
;           Store 16-bit ADC Result
Store       mov.b   &TPCNT2,RESULT      ; Store High Byte
            swpb    RESULT              ; SWAP bytes in RESULT
            mov.b   &TPCNT1,R5          ; Temp store low byte
            add     R5,RESULT           ; 16-bit result
;
;           Check for Error in ADC Result
            cmp     #100,RESULT         ; Is RESULT too small?
            jl      MError              ; Jump to MError
            ret                         ; RESULT OK
MError      mov     #00AAh,R14          ; Flag Error with "--"
            incd    SP                  ; Stack House Cleaning
            jmp     Display             ; Display Error
                                        ;
```

```
;-------------------------------------------------------------------------
Calculate   ; Calculate Temperature and fall through to Display
;-------------------------------------------------------------------------
            mov     RESULT,IROP1
            mov     #10000,IROP2L
;
;-------------------------------------------------------------------------
MPYU        ; UNSIGNED MULTIPLY SUBROUTINE: IROP1 x IROP2L -> IRACM/IRACL
            ; Used Register IROP1, IROP2L, IROP2M, IRACL, IRACM, IRBT
;-------------------------------------------------------------------------
            CLR     IRACCL              ; 0 -> LSBs RESULT
            CLR     IRACCH              ; 0 -> MSBs RESULT
MACU        CLR     IROP2H              ; MSBs MULTIPLIER
            MOV     #1,IRBIT            ; BIT TEST REGISTER
MPY2        BIT     IRBIT,IROP1         ; TEST ACTUAL BIT
            JZ      MPY1                ; IF 0: DO NOTHING
            ADD     IROP2L,IRACCL       ; IF 1: ADD MULTIPLIER TO RESULT
            ADDC    IROP2H,IRACCH
MPY1        RLA     IROP2L              ; MULTIPLIER x 2
            RLC     IROP2H              ;
            RLA     IRBIT               ; NEXT BIT TO TEST
            JNC     MPY2                ; IF BIT IN CARRY: FINISHED
;           RET
            mov     TEMP,IROP1
            mov     IRACCL,IROP2L
            mov     IRACCH,IROP2H
;-------------------------------------------------------------------------
DIVIDE      ; UNSIGNED DIVISION SUBROUTINE 32-BIT BY 16-BIT
            ; IROP2M|IROP2L : IROP1 -> IRACL   REMAINDER IN IROP2M
            ; RETURN: CARRY = 0: OK    CARRY = 1: QUOTIENT > 16 BITS
            ; Used Register IROP1, IROP2L, IRACL, IRBT, IROP2M
;-------------------------------------------------------------------------
            CLR     IRACCL              ; Clear Accumulator Low
            MOV     #17,IRBIT           ; Initialize Loop Counter
DIV1        CMP     IROP1,IROP2H        ;
            JLO     DIV2
            SUB     IROP1,IROP2H
DIV2        RLC     IRACCL
            JC      DIV4                ; Error: result > 16 bits
            DEC     IRBIT               ; Decrement loop counter
            JZ      DIV4                ; Is 0: terminate w/o error
            RLA     IROP2L
            RLC     IROP2H
            JNC     DIV1
            SUB     IROP1,IROP2H
            SETC
            JMP     DIV2
DIV4;       RET                         ; Error indication in C
;
;-------------------------------------------------------------------------
RES2F       ; Subroutine Converts Sensor Resistance to Degrees F for
            ; Display.  Table search done directly with BCD subtraction.
            ; IRACCL has Sensor Resistance Value
```

```
;-------------------------------------------------------------------------
            clr     R12                 ; R12 Used as Pointer
            mov     #99h,R14            ; R14 Used as Temperature Counter
CMPR        cmp     RES_TAB(R12),IRACCL ; Compare Table Value to Sensor
            JL      Display             ; Jump if Sensor < Table
            incd    R12                 ; Increment word Pointer
            dadd.b  #99h,R14            ; Subtract 1 from Counter
            jmp     CMPR                ;
                                        ;
;-------------------------------------------------------------------------
Display     ; Subroutine to Display "'F" and Temperature value on LCD
            ; Temperature in BCD is passed in R14
;-------------------------------------------------------------------------
     mov.b   #0A9h,&LCDM1+1      ; "F" to display
            xor.b   #02Bh,&LCDM1+2      ; "o" blinked on display

            mov     R14,R12             ; Copy binary number to R12
            rra     R14                 ; Rotate a nibble (digit)
            rra     R14
            rra     R14
            rra     R14
            and     #0Fh,R12            ; Expose single digit
            and     #0Fh,R14            ; Expose single digit
            mov.b   LCD_Tab(R12),&LCDM1+4 ; Move digit to LCD display RAM
            mov.b   LCD_Tab(R14),&LCDM1+5 ;  Move digit to LCD display RAM
            ret
;-------------------------------------------------------------------------
Init_Sys    ; Subroutine sets up  Modules and Controls Registers
;-------------------------------------------------------------------------
SetupWDT    mov     #WDTPW+WDTHOLD,&WDTCTL          ; Stop Watchdog Timer
SetupLCD    mov.b   #LCDON+LCD4MUX+LCDP2,&LCDCTL    ; STK LCD 4Mux, S0-S17
SetupBT     mov.b   #BT_ADLY_1000+BTFRFQ1,&BTCTL   ; STK LCD, 1s int.
            mov.b   #BTIE+TPIE,&IE2     ; Enable Basic Timer, TP interrupts
            eint                        ; Enable interrupts
ClearLCD    mov     #15,R15             ; 15 LCD mem locations to clear
Clear1      mov.b   #0,LCDM1-1(R15)     ; Write zeros in LCD RAM locations
            dec     R15                 ; All LCD mem clear?
            jnz     Clear1              ; More LCD mem to clear go
            ret
;-------------------------------------------------------------------------
TP_Isr      ; Timer/Port and Basic Timer ISR, System Returned Active on RETI
;-------------------------------------------------------------------------
            clr.b   &TPCTL              ; Clear TP interrupt flags, TP Off
BT_Isr      mov     #GIE,0(SP)          ; System Active, Interrupts Enabled
            reti                        ;
                                        ;
;-------------------------------------------------------------------------
;   Resistance table, K ohms to 99-59 degrees F.
;-------------------------------------------------------------------------
RES_TAB    .even                        ; Evenly aligned
           .word   6446                 ;99 F
           .word   6570
           .word   6693
```

```
            .word   6817                ;96 F
            .word   6941                ;95 F
            .word   7093
            .word   7245
            .word   7398
            .word   7550
            .word   7703                ;90 F
            .word   7856
            .word   8008
            .word   8161
            .word   8313
            .word   8500                ;85 F
            .word   8688
            .word   8875
            .word   9063
            .word   9250
            .word   9438                ;80 F
            .word   9625
            .word   9813
            .word   10000
            .word   10232
            .word   10464               ;75 F
            .word   10697
            .word   10929
            .word   11161
            .word   11393
            .word   11626               ;70 F
            .word   11858
            .word   12090
            .word   12378
            .word   12666
            .word   12953               ;65 F
            .word   13241               ;64 F
            .word   13529
            .word   13817
            .word   14104
            .word   14392
            .word   14680               ;59 F
;
;-------------------------------------------------------------------------
;           STK LCD Type
;-------------------------------------------------------------------------
a           .equ    01h
b           .equ    02h
c           .equ    10h
d           .equ    04h
e           .equ    80h
f           .equ    20h
g           .equ    08h
h           .equ    40h

LCD_Tab     .byte   a+b+c+d+e+f         ; displays ”0”
            .byte   b+c                 ; displays ”1”
```

```
            .byte   a+b+d+e+g           ; displays ”2”
            .byte   a+b+c+d+g           ; displays ”3”
            .byte   b+c+f+g             ; displays ”4”
            .byte   a+c+d+f+g           ; displays ”5”
            .byte   a+c+d+e+f+g         ; displays ”6”
            .byte   a+b+c               ; displays ”7”
            .byte   a+b+c+d+e+f+g       ; displays ”8”
            .byte   a+b+c+d+f+g         ; displays ”9”
            .byte   g                   ; displays ”–”
;
;-----------------------------------------------------------------------
        .sect   ”Int_Vect”,I_vectors–31 ; MSP430x32x Interrupt Vectors
;-----------------------------------------------------------------------
        .word   RESET                   ; Port0, bit 2 to bit 7
        .word   BT_Isr                  ; Basic Timer
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   TP_Isr                  ; Timer Port
        .word   RESET                   ; EOC from ADC
        .word   RESET                   ; no source
        .word   RESET                   ; no source
        .word   RESET                   ; Timer_A
        .word   RESET                   ; Timer_A
        .word   RESET                   ; Watchdog/Timer, Timer mode
        .word   RESET                   ; no source
        .word   RESET                   ; Address of UART handler
        .word   RESET                   ; P0.0
        .word   RESET                   ; NMI, Osc. fault
        .word   RESET                   ; POR, ext. Reset, Watchdog
```

## IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

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

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Copyright © 1999, Texas Instruments Incorporated