IMPLEMENTATION OF A MSP430-BASED DIGITAL THERMOMETER USING THE SLOPE ADC OF THE TIMER PORT MODULE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

in

Electrical Engineering

By

SANTOSH PALLAV SAHU, UMA SHANKAR ROSHAN & UMESH KUMAR AGRAWAL

Department of Electrical Engineering

National Institute of Technology

Rourkela

2007

IMPLEMENTATION OF A MSP430-BASED DIGITAL THERMOMETER USING THE SLOPE ADC OF THE TIMER PORT MODULE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology
in
Electrical Engineering

By
SANTOSH PALLAV SAHU, UMA SHANKAR ROSHAN & UMESH KUMAR AGRAWAL

Under the guidance of

Prof. J.K.SATAPATHY

Department of Electrical Engineering

National Institute of Technology

Rourkela

2007

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled **"Implementation of a MSP430-based digital thermometer using the slope ADC of the timer port module "** submitted by **Santosh Pallav Sahu, Roll No: 10302049, Uma Shankar Roshan, Roll No: 10302059 and Umesh Kumar Agrawal**, **Roll No: 10302040** in the partial fulfillment of the requirement for the degree of **Bachelor of Technology** in **Electrical Engineering**, National Institute of Technology, Rourkela , is an authentic work carried out by them under my supervision.

To the best of my knowledge the matter embodied in the thesis has not been submitted to any other university/institute for the award of any degree or diploma.

Professor J.K.Satapathy

Date

Department of Electrical Engineering

National Institute of Technology

Rourkela-769008

ACKNOWLEDGMENT

We avail this opportunity to extend our hearty indebtedness to our guide **Professor J. K. Satapathy**, Electrical Engineering Department, for his valuable guidance, constant encouragement and kind help at different stages for the execution of this dissertation work.

We also express our sincere gratitude to **Dr. P. K. Nanda**, Head of the Department, Electrical Engineering, for providing valuable departmental facilities.

Submitted by:

Santosh Pallav Sahu
Roll No: 10302049
National Institute of Technology
Rourkela

Uma Shankar Roshan
Roll No: 10302059
National Institute of Technology
Rourkela

Umesh Kumar Agrawal
Roll No: 10302040
National Institute of Technology
Rourkela

CONTENTS

ABSTRACT

This report describes the slope A/D measurement of a resistance and the ease with which it can be applied to MSP430 microcontrollers. It describes a digital thermometer design that uses the slope ADC capabilities of the Timer Port module on the MSP430x3xx microcontrollers. It is used more generally as a reference on how to connect resistive sensors and reference resistors to the Timer Port module. All MSP430x3xx devices include the Timer Port module. The module allows several resistive sensors and reference resistors to be connected in an application. Unused module pins can be used as independent outputs.

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-relationship is linear, which means the calculation is easier and less-costly to implement in a microcontroller than for the exponential V-t relationship.

The thermometer has been simulated by using a variable resistance instead of a thyristor. In addition care has been taken to optimize the power consumption by forcing the microcontroller to several low-power modes during the operation. The combination of the Timer Port module, the 16-bit CPU, and the ultra low power design provide unmatched MIPS per watt performance. The set up can be extended to provide a low power thermostat.

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

BACKGROUND

OBJECTIVE

1.1 INTRODUCTION

This report describes a digital thermometer design that uses the slope ADC capabilities of the Timer Port module on the MSP430x3xx microcontrollers. It is used more generally as a reference on how to connect resistive sensors and reference resistors to the Timer Port module. All MSP430x3xx devices include the Timer Port module. The module allows several resistive sensors and reference resistors to be connected in an application. Unused module pins can be used as independent outputs.

The project was done on the evaluation kit 'EVK-MSP430S330' supplied by Texas Instruments which is meant to act as a development kit using the microcontroller 'PMS430E337AHFD'- an EPROM version of the family ' MSP430337A'. The corresponding programs were developed in assembly language using 'IAR –KICKSTART WORKBENCH' supplied with the kit.

1.2 BACKGROUND

The MSP430 is a 16-bit RISC-based microcontroller that uses advanced timing and design features, as well as a highly orthogonal structure, to deliver a processing core that is both powerful and very flexible. These features allow theMSP430 to consume only 400 mA in active mode in a typical 3-V system. TheMSP430, typically using only 2 mA in standby mode, can wake up to fully synchronized active mode in a maximum of 6 ms. The MSP430 subfamilies incorporate various mixes of peripheral modules which result in highly integrated systems. Figure 1.1 shows a block diagram of the MSP430x32x.

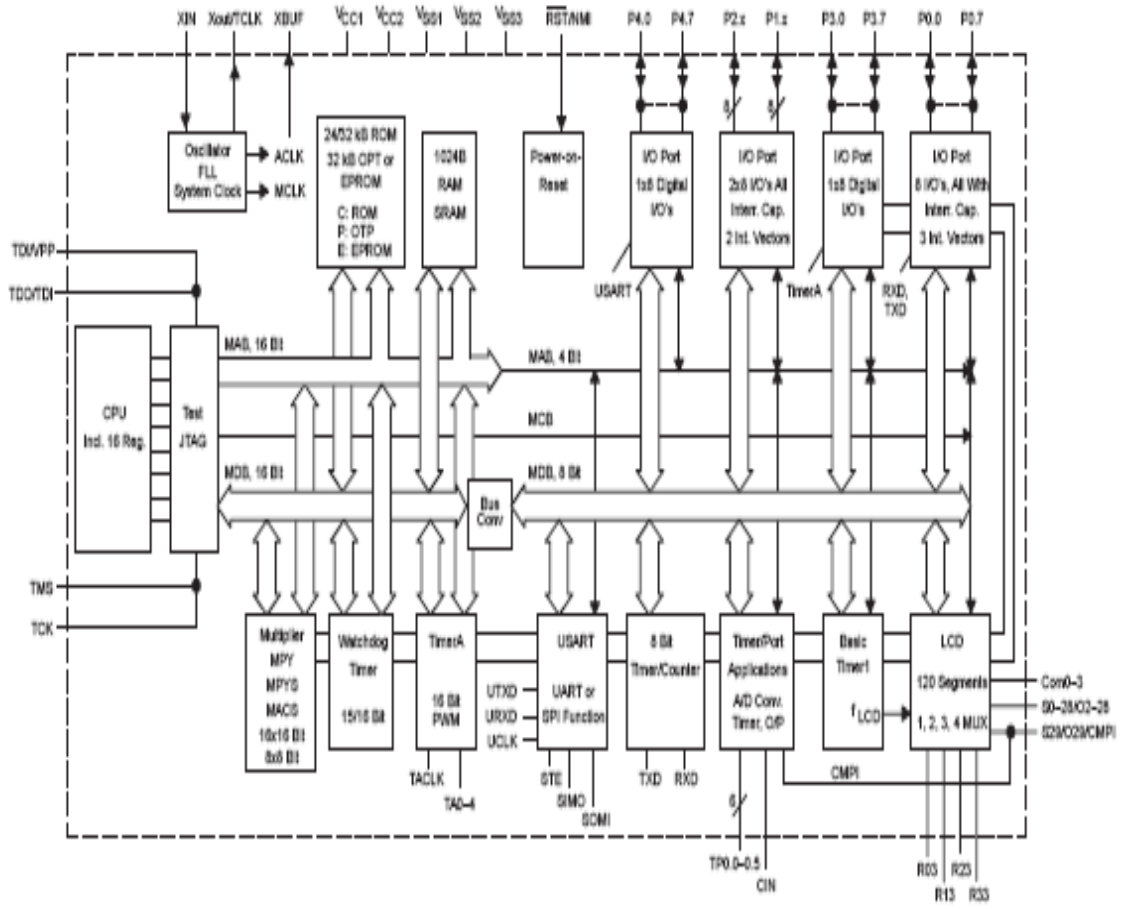


Fig 1.1

1.3 OBJECTIVE

To devise a method to incorporate ultra low power consumption in thermometer or thermostat devices using ADC capabilities of the timer port module of MSP430 microcontrollers without including any external A/D converters. The report describes a program to optimize power consumption by forcing the microcontroller to several sleep states during the operation .

Chapter 2

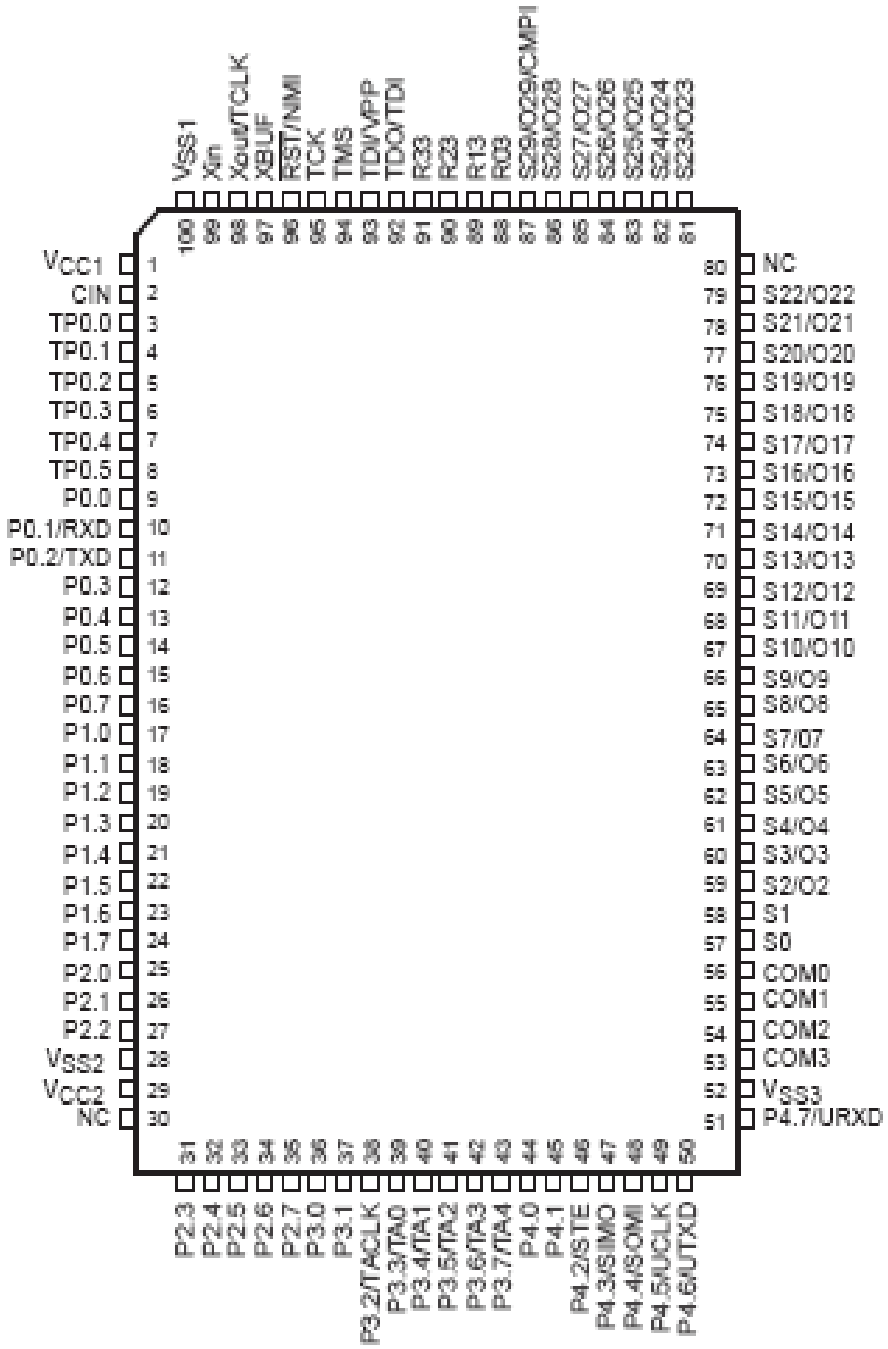
OVERVIEW OF MSP430P337A MICROCONTROLLERS

SALIENT FEATURES
PIN DIAGRAM & TERMINAL FUNCTIONS
DESCRIPTION

2.1 SALIENT FEATURES

- Low Supply Voltage Range 2.5 V – 5.5 V
- Low Operation Current, 400 μA at 1 MHz,3 V
- Ultra low-Power Consumption:
  - – Standby Mode: 2 mA
  - – RAM Retention Off Mode: 0.1 mA
- Five Power-Saving Modes
- Wake-Up From Standby Mode in 6 ms
- 16-Bit RISC Architecture, 300 ns Instruction Cycle Time
- Single Common 32 kHz Crystal, Internal System Clock up to 3.8 MHz
- Integrated LCD Driver for up to 120 Segments
- Integrated Hardware Multiplier Performs Signed, Unsigned on Multiply, and MAC Operations for Operands up to 16 × 16 Bits
- Serial Communication Interface (USART),
- Select Asynchronous UART or Synchronous SPI by Software
- Slope A/D Converter Using External Components
- 16-Bit Timer With Five Capture/Compare Registers
- Serial Onboard Programming
- Programmable Code Protection by Security Fuse
- Family Members Include:
  - – MSP430C336 – 24 KB ROM, 1 KB RAM
  - – MSP430C337 – 32 KB ROM, 1 KB RAM
  - – MSP430P337A – 32 KB OTP, 1 KB RAM
- EPROM Version Available for Prototyping:
  - – PMS430E337A
- Available in the Following Packages:
  - – 100 Pin Quad Flat-Pack (QFP)
  - – 100 Pin Ceramic Quad Flat-Pack (CFP)
    (EPROM Version)

2.2 PIN DIAGRAM OF PMS430E337AHFD



NC – No internal connection

Fig 2.1

TERMINAL FUNCTIONS

TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
CIN	2	I	Input port. CIN is used as an enable for counter TPCNT1 – (Timer/Port).
COM0–3	56–53	O	Common outputs. COM0-3 are used for LCD backplanes – LCD
P0.0	9	I/O	General-purpose digital I/O
P0.1/RXD	10	I/O	General-purpose digital I/O, receive digital Input port – 8-Bit Timer/Counter
P0.2/TXD	11	I/O	General-purpose digital I/O, transmit data output port – 8-Bit Timer/Counter
P0.3–P0.7	12–16	I/O	Five general-purpose digital I/Os, bit 3-7
P1.0–P1.7	17–24	I/O	Eight general-purpose digital I/Os, bit 0-7
P2.0–P2.7	25–27, 31–35	I/O	Eight general-purpose digital I/Os, bit 0-7
P3.0, P3.1	36,37	I/O	Two general-purpose digital I/Os, bit 0 and bit 1
P3.2/TACLK	38	I/O	General-purpose digital I/O, clock input – Timer_A
P3.3/TA0	39	I/O	General-purpose digital I/O, capture I/O, or PWM output port – Timer_A CCR0
P3.4/TA1	40	I/O	General-purpose digital I/O, capture I/O, or PWM output port – Timer_A CCR1
P3.5/TA2	41	I/O	General-purpose digital I/O, capture I/O, or PWM output port – Timer_A CCR2
P3.6/TA3	42	I/O	General-purpose digital I/O, capture I/O, or PWM output port – Timer_A CCR3
P3.7/TA4	43	I/O	General-purpose digital I/O, capture I/O, or PWM output port – Timer_A CCR4
P4.0	44	I/O	General-purpose digital I/O, bit 0
P4.1	45	I/O	General-purpose digital I/O, bit 1
P4.2/STE	46	I/O	General-purpose digital I/O, slave transmit enable – USART/SPI mode
P4.3/SIMO	47	I/O	General-purpose digital I/O, slave in/master out – USART/SPI mode
P4.4/SOMI	48	I/O	General-purpose digital I/O, master in/slave out – USART/SPI mode
P4.5/UCLK	49	I/O	General-purpose digital I/O, external clock input – USART
P4.6/UTXD	50	I/O	General-purpose digital I/O, transmit data out – USART/UART mode
P4.7/URXD	51	I/O	General-purpose digital I/O, receive data in – USART/UART mode
R03	88	I	Input port of fourth positive (lowest) analog LCD level (V5) – LCD
R13	89	I	Input port of third most positive analog LCD level (V3 of V4) – LCD
R23	90	I	Input port of second most positive analog LCD level (V2) – LCD
R33	91	O	Output of most positive analog LCD level (V1) – LCD
$\overline{\text{RST}}$/NMI	96	I	Reset input or non-maskable interrupt input port
S0	57	O	Segment line S0 – LCD
S1	58	O	Segment line S1 – LCD
S2/O2–S5/O5	59–62	O	Segment lines S2 to S5 or digital output ports, O2-O5, group 1 – LCD
S6/O6–S9/O9	63–66	O	Segment lines S6 to S9 or digital output ports O6-O9, group 2 – LCD
S10/O10–S13/O13	67–70	O	Segment lines S10 to S13 or digital output ports O10-O13, group 3 – LCD
S14/O14–S17/O17	71–74	O	Segment lines S14 to S17 or digital output ports O14-O17, group 4 – LCD
S18/O18–S21/O21	75–78	O	Segment lines S18 to S21 or digital output ports O18-O21, group 5 – LCD
S22/O22–S25/O25	79, 81–83	O	Segment line S22 to S25 or digital output ports O22-O25, group 6 – LCD
S26/O26–S29/O29/CMPI	84–87	O	Segment line S26 to S29 or digital output ports O26-O29, group 7 – LCD. Segment line S29 can be used as comparator input port CMPI – Timer/Port
TCK	95	I	Test clock. TCK is the clock input port for device programming and test.
TDI/VPP	93	I	Test data input. TDI/VPP is used as a data input port or input for programming voltage.

TERMINAL NAME	NO.	I/O	DESCRIPTION
TMS	94	I	Test mode select. TMS is used as an input port for device programming and test.
TDO/TDI	92	I/O	Test data output port. TDO/TDI data output or programming data input terminal
TP0.0	3	O	General-purpose 3-state digital output port, bit 0 – Timer/Port
TP0.1	4	O	General-purpose 3-state digital output port, bit 1 – Timer/Port
TP0.2	5	O	General-purpose 3-state digital output port, bit 2 – Timer/Port
TP0.3	6	O	General-purpose 3-state digital output port, bit 3 – Timer/Port
TP0.4	7	O	General-purpose 3-state digital output port, bit 4 – Timer/Port
TP0.5	8	I/O	General-purpose 3-state digital input/output port, bit 5 – Timer/Port
V_{CC1}	1		Positive supply voltage
V_{CC2}	29		Positive supply voltage
V_{SS1}	100		Ground reference
V_{SS2}	28		Ground reference
V_{SS3}	52		Ground reference
XBUF	97	O	System clock (MCLK) or crystal clock (ACLK) output
Xin	99	I	Input port for crystal oscillator
Xout/TCLK	98	I/O	Output terminal of crystal oscillator or test clock input

Table 2.1

2.3 DESCRIPTION

2.3.1 Processing unit:

The processing unit is based on a consistent and orthogonal designed CPU and instruction set. This design structure results in a RISC-like architecture, highly transparent to the application development, which is distinguished by ease of programming. All operations other than program-flow instructions consequently are performed as register operations in conjunction with seven addressing modes for source and four modes for destination operand.

2.3.2 CPU registers:

The CPU has sixteen registers that provide reduced instruction execution time. This reduces the register-to-register operation execution time to one cycle of the processor frequency. Four of the registers are reserved for special use as a program counter, a stack pointer, a status register, and a constant generator. The remaining registers are available as general-purpose registers. Peripherals are connected to the CPU using a data address and control bus and can be handled easily with all instructions for memory manipulation

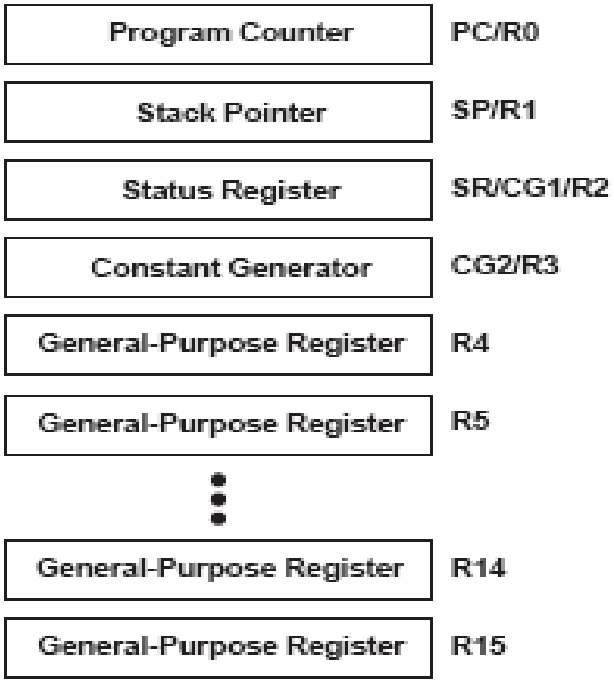


Fig 2.2

2.3.3 Operation modes and interrupts

The MSP430 operating modes support various advanced requirements for ultra low-power and ultra low-energy consumption. This is achieved by the intelligent management of the operations during the different module operation modes and CPU states. The requirements are fully supported during interrupt event handling. An interrupt event awakens the system from each of the various operating modes and returns with the RETI instruction to the mode that was selected before the interrupt event. The clocks used are ACLK and MCLK.ACLK is the crystal frequency and MCLK, a multiple of ACLK, is used as the system clock.

The following five operating modes are supported:

 Active mode (AM). The CPU is enabled with different combinations of active peripheral modules.

 Low-power mode 0 (LPM0). The CPU is disabled, peripheral operation continues, ACLK and MCLK signals are active, and loop control for MCLK is active.

 Low-power mode 1 (LPM1). The CPU is disabled, peripheral operation continues, ACLK and MCLK signals are active, and loop control for MCLK is inactive.

 Low-power mode 2 (LPM2). The CPU is disabled, peripheral operation continues, ACLK signal is active, and MCLK and loop control for MCLK are inactive.

 Low-power mode 3 (LPM3). The CPU is disabled, peripheral operation continues, ACLK signal is active, MCLK and loop control for MCLK are inactive, and the dc generator for the digital controlled oscillator (DCO)(MCLK generator) is switched off.
 Low-power mode 4 (LPM4). The CPU is disabled, peripheral operation continues, ACLK signal is inactive (crystal oscillator stopped), MCLK and loop control for MCLK are inactive, and the dc generator for the DCO is switched off.

The special function registers (SFR) include module-enable bits that stop or enable the operation of the specific peripheral module. All registers of the peripherals may be accessed if the operational function is stopped or enabled; however, some peripheral current-saving functions are accessed through the state of local register bits. An example is the enable/disable of the analog voltage generator in the LCD peripheral, which is turned on or off using one register bit. The most general bits that influence current consumption and support fast turn on from low power operating modes are located in the status register (SR). Four of these bits control the CPU and the system clock generator: SCG1, SCG0, OscOff, and CPUOff.

Fig 2.3

2.3.4 Interrupt:

Software determines the activation of interrupts through the monitoring of hardware set interrupt flag status bits, the control of specific interrupt enable bits in SRs, the establishment of interrupt vectors, and the programming of interrupt handlers. The interrupt vectors and the power-up starting address are located in ROM address locations 0FFFFh through 0FFE0h. Each vector contains the 16-bit address of the appropriate interrupt handler instruction sequence.

2.3.5 ROM memory organization:

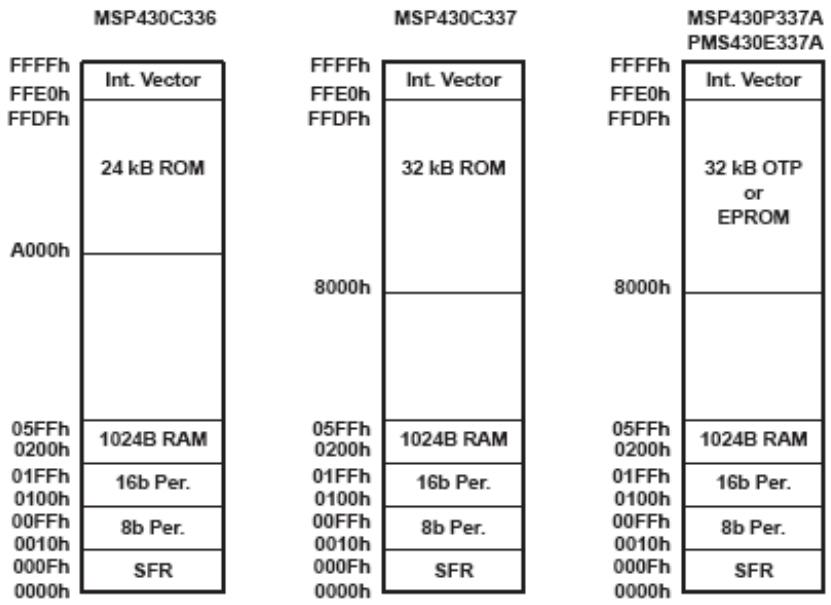


Fig 2.4

2.3.6 Peripherals:

Peripherals that are connected to the CPU through a data, address, and controls bus can be handled easily with instructions for memory manipulation.

2.3.6 Oscillator and system clock:

Two clocks are used in the system: the system (master) clock (MCLK) and the auxiliary clock (ACLK). The MCLK is a multiple of the ACLK. The ACLK runs with the crystal oscillator frequency. The special design of the oscillator supports the feature of low current consumption and the use of a 32 768 Hz crystal. The crystal is connected across two terminals without any other external components required.

The oscillator starts after applying VCC, due to a reset of the control bit (OscOff) in the status register (SR). It can be stopped by setting the OscOff bit to a 1. The enabled clock signals ACLK, ACLK/2, ACLK/4, or MCLK are accessible for use by external devices at output terminal XBUF.

The controller system clocks have to deal with different requirements according to the application and system condition. Requirements include:

- High frequency in order to react quickly to system hardware requests or events
- Low frequency in order to minimize current consumption, EMI, etc.
- Stable frequency for timer applications e.g., real-time clock (RTC)
- Enable start-stop operation with minimum delay to operation function

These requirements cannot all be met with fast frequency high-Q crystals or with RC-type low-Q oscillators. This Compromise and selected for the MSP430, uses a low-crystal frequency, which is multiplied to achieve the desired nominal operating range:

$f_{(system)} = (N+1) f_{(crystal)}$

The crystal frequency multiplication is achieved with a frequency locked loop (FLL) technique. The factor N is set to 31 after a power-up clear condition. The FLL technique, in combination with a digital controlled oscillator (DCO), provides immediate start-up capability together with long term crystal stability. The frequency variation of the DCO with the FLL inactive is typically 330 ppm , which means that with a cycle time of 1 μs the maximum possible variation is 0.33 ns. For more precise timing, the FLL can be used, which forces longer cycle times if the previous cycle time was shorter than the selected one. This switching of cycle times makes it possible to meet the chosen system frequency over a long period of time. The start-up operation of the system clock depends on the previous machine state. During a PUC, the DCO is reset to its lowest possible frequency. The control logic starts operation immediately after recognition of PUC.

2.3.7 Multiplication:

The multiplication operation is supported by a dedicated peripheral module. The module performs 16x16, 16x8,8x16, and 8x8 bit operations. The module is capable of supporting signed and unsigned multiplication as well as signed and unsigned multiply and accumulates operations. The result of an operation can be accessed immediately after the operands have been loaded into the peripheral registers. No additional clock cycles are required.

2.3.8 Digital I/O:

Five eight-bit I/O ports (P0 thru P4) are implemented. Port P0 has six control registers, P1 and P2 have seven control registers, and P3 and P4 modules have four control registers to give maximum flexibility of digital input/output to the application:

- Individual I/O bits are independently programmable.
- Any combination of input, output, and interrupt conditions is possible.
- Interrupt processing of external events is fully implemented for all eight bits of the P0, P1, and P2 ports.
- Read/write access is available to all registers by all instructions.

The seven registers are:

 Input register contains information at the pins
 Output register contains output information
 Direction register controls direction
 Interrupt edge select contains input signal change necessary for interrupt
 Interrupt flags indicate if interrupt(s) are pending
 Interrupt enable contains interrupt enable pins
 Function select determines if pin(s) used by module or port

These registers contain eight bits each with the exception of the interrupt flag register and the interrupt enable register which are 6 bits each. The two least significant bit (LSBs) of the interrupt flag and enable registers are located in the special function register (SFR). Five interrupt vectors are implemented, one for Port P0.0, one for Port P0.1, one commonly used for any interrupt event on Port P0.2 to Port P0.7, one commonly used for any interrupt event on Port P1.0 to Port P1.7, and one commonly used for any interrupt event on Port P2.0 to PortP2.7.

2.3.9 LCD drive:

The liquid crystal displays (LCDs) for static, 2-, 3-, and 4-MUX operation can be driven directly. The operation of the controller LCD logic is defined by software through memory-bit manipulation. The LCD memory is part of the LCD module, not part of data memory. Eight mode and control bits define the operation and current consumption of the LCD drive. The information for the individual digits can be easily obtained using table programming techniques combined with the proper addressing mode. The segment information is stored into LCD memory using instructions for memory manipulation.

The drive capability is defined by the external resistor divider that supports analog levels for 2-, 3-, and 4-MUX operation. Groups of the LCD segment lines can be selected for digital output signals. The MSP430x33x configuration has four common lines, 30 segment lines, and four terminals for adjusting the analog levels.

2.3.10 Basic Timer1:

The Basic Timer1 (BT1) divides the frequency of MCLK or ACLK, as selected with the SSEL bit, to provide low-frequency control signals. This is done within the system by one central divider, the Basic Timer1, to support low current applications. The BTCTL control register contains the flags which control or select the different operational functions. When the supply voltage is applied or when a reset of the device (RST/NMI pin), a watchdog

overflow, or a watchdog security key violation occurs, all bits in the register hold undefined or unchanged status. The user software usually configures the operational conditions on the BT during initialization.

The Basic Timer1 has two eight bit timers which can be cascaded to a sixteen bit timer. Both timers can be read and written by software. Two bits in the SFR address range handle the system control interaction according to the function implemented in the Basic Timer1. These two bits are the Basic Timer1 interrupt flag (BTIFG) and the Basic Timer1 interrupt enable (BTIE) bit.

2.3.11 Watchdog Timer:

The primary function of the Watchdog Timer (WDT) module is to perform a controlled system restart after software upset has occurred. If the selected time interval expires, a system reset is generated. If this watchdog function is not needed in an application, the module can work as an interval timer, which generates an interrupt after the selected time interval.

The Watchdog Timer counter (WDTCNT) is a 15/16-bit up counter which is not directly accessible by software. The WDTCNT is controlled using the Watchdog Timer control register (WDTCTL), which is an 8-bit read/write register. Writing to WDTCTL, in both operating modes (watchdog or timer) is only possible by using the correct password in the high-byte. The low-byte stores data written to the WDTCTL. The high-byte password is 05Ah.If any value other than 05Ah is written to the high-byte of the WDTCTL, a system reset PUC is generated. When the password is read its value is 069h. This minimizes accidental write operations to the WDTCTL register. In addition to the Watchdog Timer control bits, there are two bits included in the WDTCTL that configure the NMI pin.

2.3.12 USART:

The universal synchronous/asynchronous interface is a dedicated peripheral module which provides serial communications. The USART supports synchronous SPI (3 or 4 pin) and asynchronous UART communications protocols, using double buffered transmit and receive channels. Data streams of 7 or 8 bits in length can be transferred at a rate determined by the program, or by a rate defined by an external clock. Low-power applications are optimized by UART mode options which allow for the receipt of only the first byte of a complete frame. The applications software then decides if the succeeding data is to be processed. This option reduces power consumption.

Two dedicated interrupt vectors are assigned to the USART module, one for the receive and one for the transmit channel.

2.3.13 Timer/Port:

The Timer/Port module has two 8-Bit Timer/Counters, an input that triggers one counter and six digital outputs with 3-state capability. Both counters have an independent clock selector for selecting an external signal or one of the internal clocks (ACLK or MCLK). One of the counters has an extended control capability to halt, count continuously, or gate the counter by selecting one of two external signals. This gate signal sets the interrupt flag if an external signal is selected and the gate stops the counter. Both timers can be read to and written from by software. The two 8-Bit Timer/Counters can be cascaded to form a 16-bit counter. A common interrupt vector is implemented. The interrupt flag can be set by three events in the 8-Bit Timer/Counter mode (gate signal or overflow from the counters) or by two events in the 16-bit counter mode (gate signal or overflow from the MSB of the cascaded counter).

2.3.14 slope A/D conversion:

Slope A/D conversion is accomplished with the Timer/Port module using external resistor(s) for reference (R_{ref}), using external resistor(s) to the measured (R_{meas}), and an external capacitor. The external components are driven by software in such a way that the internal counter measures the time that is needed to charge or discharge the capacitor. The reference resistor's (R_{ref}) charge or discharge time is represented by N_{ref} counts. The unknown resistors (R_{meas}) charge or discharge time is represented by N_{meas} counts. The unknown resistor's value R_{meas} is the value of R_{ref} multiplied by the relative number of counts (N_{meas}/N_{ref}). This value determines resistive sensor values that correspond to the physical data, for example temperature, when an NTC or PTC resistor is used.

2.3.15 Timer A

The Timer A module (see Figure1) offers one sixteen bit counter and five capture/compare registers. The timer clock source can be selected to come from an external source TACLK (SSEL=0), the ACLK (SSEL=1), or MCLK (SSEL=2 or SSEL=3). The clock source can be divided by one, two, four, or eight. The timer can be fully controlled (in word mode) since it can be halted, read, and written. It can be stopped or run continuously. It can count up or count up/down using one compare block to determine the period. The five capture/compare blocks are configured by the application software to run in either capture or compare mode. The capture mode is primarily used to measure external or internal events with any combination of positive, negative, or both edges of the clock. The clock can also be stopped

in capture mode by software. One external event (CCISx=0) per capture block can be selected. If CCISx=1, the ACLK is the capture signal; and if CCISx=2 or CCISx=3, software capture is chosen. The compare mode is primarily used to generate timing for the software or application hardware or to generate pulse-width modulated output signals for various purposes like D/A conversion functions or motor control. An individual output module, which can run independently of the compare function or is triggered in several ways, is assigned to each of the five capture/compare registers. Two interrupt vectors are used by the Timer_A module. One individual vector is assigned to capture/compare block CCR0 and one common interrupt vector is assigned to the timer and the other four capture/compare blocks. The five interrupt events using the common vector are identified by an individual interrupt vector word. The interrupt vector word is used to add an offset to the program counter to continue the interrupt handler software at the correct location. This simplifies the interrupt handler and gives each interrupt event the same interrupt handler overhead of 5 cycles.

2.3.16 8-Bit Timer/Counter

The 8-bit interval timer supports three major functions for applications:

- Serial communication or data exchange
- Plus counting or plus accumulation
- Timer

The 8-Bit Timer/Counter peripheral includes the following major blocks: an 8-bit up-counter with preload register, an 8-bit control register, an input clock selector, an edge detection (e.g. start bit detection for asynchronous protocols), and an input and output data latch, triggered by the carry-out-signal from the 8-Bit Timer/Counter. The 8-Bit Timer/Counter counts up with an input clock, which is selected by two control bits from the control register. The four possible clock sources are MCLK, ACLK, the external signal from terminal P0.1, and the signal from the logical AND of MCLK and terminal P0.1. Two counter inputs (load, enable) control the counter operation. The load input controls load operations. A write-access to the counter results in loading the content of the preload register into the counter. The software writes or reads the preload register with all instructions. The preload register acts as a buffer and can be written immediately after the load of the counter is completed. The enable input enables the count operation. When the enable signal is set to high, the counter will count-up each time a positive clock edge is applied to the clock input of the counter.

Electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS		MIN	NOM	MAX	UNIT
$I_{(AM)}$	Active mode	C336/7	T_A= –40°C +85°C,	V_{CC} = 3 V		400	500	µA
			T_A= –40°C +85°C,	V_{CC} = 5 V		800	900	
		P337A	T_A= –40°C +85°C,	V_{CC} = 3 V		570	700	
			T_A= –40°C +85°C,	V_{CC} = 5 V		1170	1250	
$I_{(CPUOff)}$	Low power mode, (LPM0,1)	C336/7	T_A= –40°C +85°C,	V_{CC} = 3 V		50	70	µA
			T_A= –40°C +85°C,	V_{CC} = 5 V		100	130	
		P337A	T_A= –40°C +85°C,	V_{CC} = 3 V		50	70	
			T_A= –40°C +85°C,	V_{CC} = 5 V		100	130	
$I_{(LPM2)}$	Low power mode, (LPM2)		T_A= –40°C +85°C,	V_{CC} = 3 V		7	12	µA
			T_A= –40°C +85°C,	V_{CC} = 5 V		18	25	
$I_{(LPM3)}$	Low power mode, (LPM3)		T_A= –40°C	V_{CC} = 3 V		2.0	3.5	µA
			T_A= 25°C			2.0	3.5	
			T_A= 85°C			1.6	3.5	
			T_A= –40°C	V_{CC} = 5 V		5.2	10	
			T_A= 25°C			4.2	10	
			T_A= 85°C			4.0	10	
$I_{(LPM4)}$	Low power mode, (LPM4)		T_A= –40°C	V_{CC} = 3 V/5 V		0.1	0.8	µA
			T_A= 25°C			0.1	0.8	
			T_A= 85°C			0.4	1.5	

Table 2.2

Variation of system frequency with supply voltage

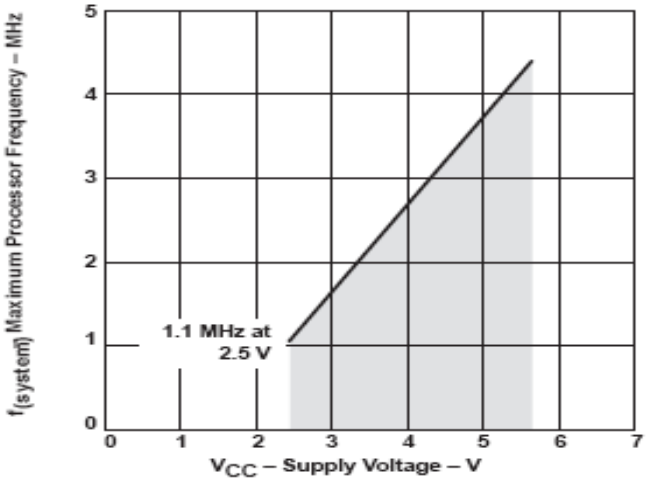


Fig 2.5

2.3.17 Instruction set :

The instruction set for this register-register architecture provides a powerful and easy-to-use assembly language. The instruction set consists of 51 instructions with three formats and seven addressing modes. Table 2.3 provides a summation and example of the three types of instruction formats; the address modes are listed in Table 2.4.

Dual operands, source-destination	e.g. ADD R4,R5	R4 + R5 → R5
Single operands, destination only	e.g. CALL R8	PC → (TOS), R8→ PC
Relative jump, un-/conditional	e.g. JNE	Jump-on equal bit = 0

Table 2.3

Instructions that can operate on both word and byte data are differentiated by the suffix .B when a byte operation is required.

Examples: Instructions for word operation:

MOV EDE,TONI

ADD #235h,&MEM

PUSH R5

SWPB R5

Instructions for byte operation:

MOV.B EDE,TONI

ADD.B #35h,&MEM

PUSH.B R5

ADDRESS MODE	S	D	SYNTAX	EXAMPLE	OPERATION
Register	√	√	MOV Rs,Rd	MOV R10,R11	R10 → R11
Indexed	√	√	MOV X(Rn),Y(Rm)	MOV 2(R5),6(R6)	M(2+R5) → M(6+R6)
Symbolic (PC relative)	√	√	MOV EDE,TONI		M(EDE) → M(TONI)
Absolute	√	√	MOV &MEM,&TCDAT		M(MEM) → M(TCDAT)
Indirect	√		MOV @Rn,Y(Rm)	MOV @R10,Tab(R6)	M(R10) → M(Tab+R6)
Indirect autoincrement	√		MOV @Rn+,Rm	MOV @R10+,R11	M(R10) → R11 R10 + 2→ R10
Immediate	√		MOV #X,TONI	MOV #45,TONI	#45 → M(TONI)

NOTE 1: S = source, D = destination.

Table 2.4

Computed branches (BR) and subroutine calls (CALL) instructions use the same address modes as the other instructions. These addressing modes provide indirect addressing, ideally suited for computed branches and calls. The full use of this programming capability permits a program structure different from conventional 8- and 16-bit controllers. For example, numerous routines can easily be designed to deal with pointers and stacks instead of using flag type programs for flow control.

Chapter 3

HARDWARE INTERFACE-I

MSP- EVK4430S330 SCEMATIC
MSP-EVK430S330 PART PLACEMENT
DEVELOPMENT FLOW
PROJECT SETTINGS
OTHER INFORMATIONS

3.1 MSP-EVK430S330 SCHEMATIC:

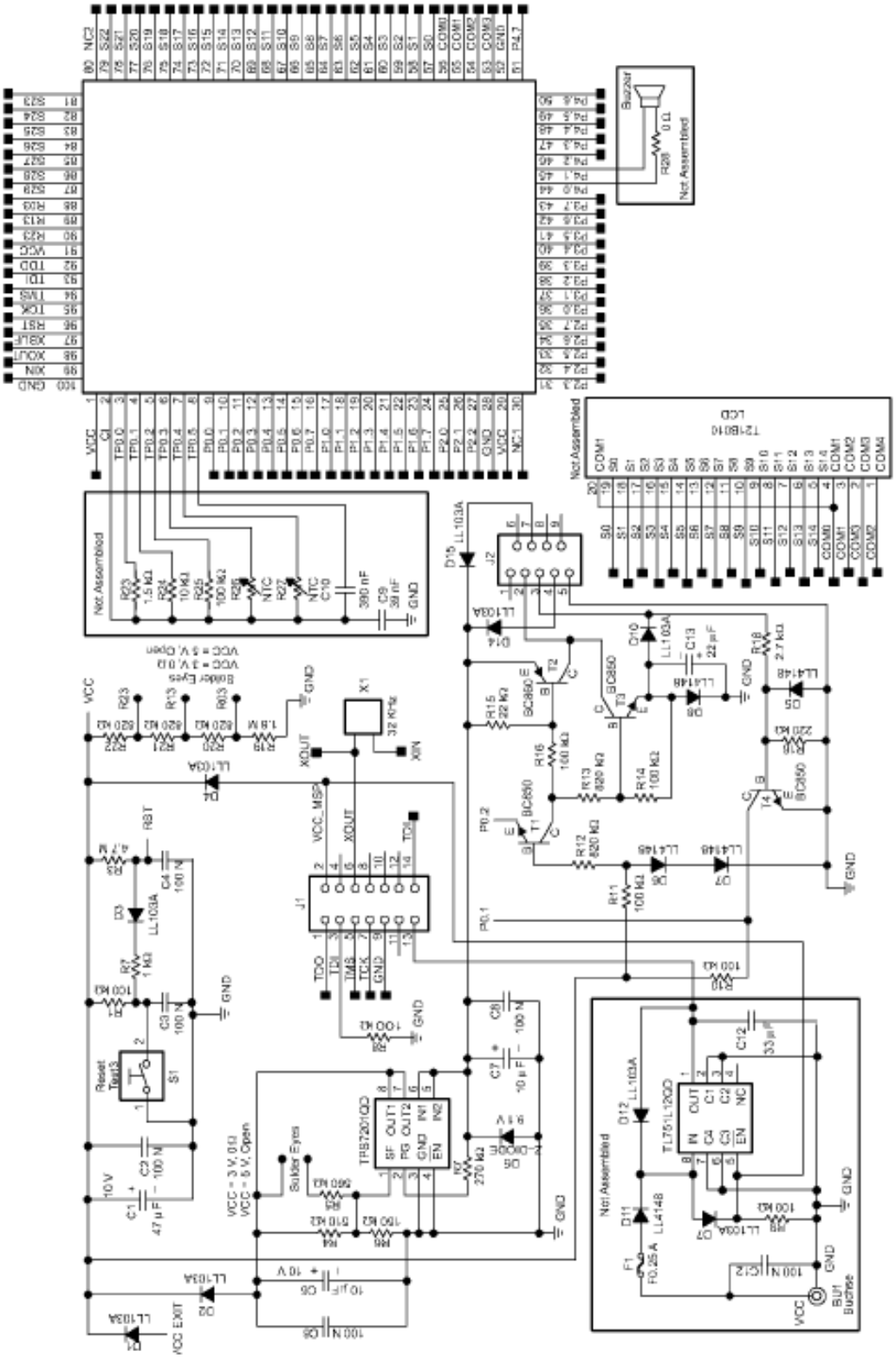


Fig 3.1

3.2 MSP-EVK430S330 PART PLACEMENT:

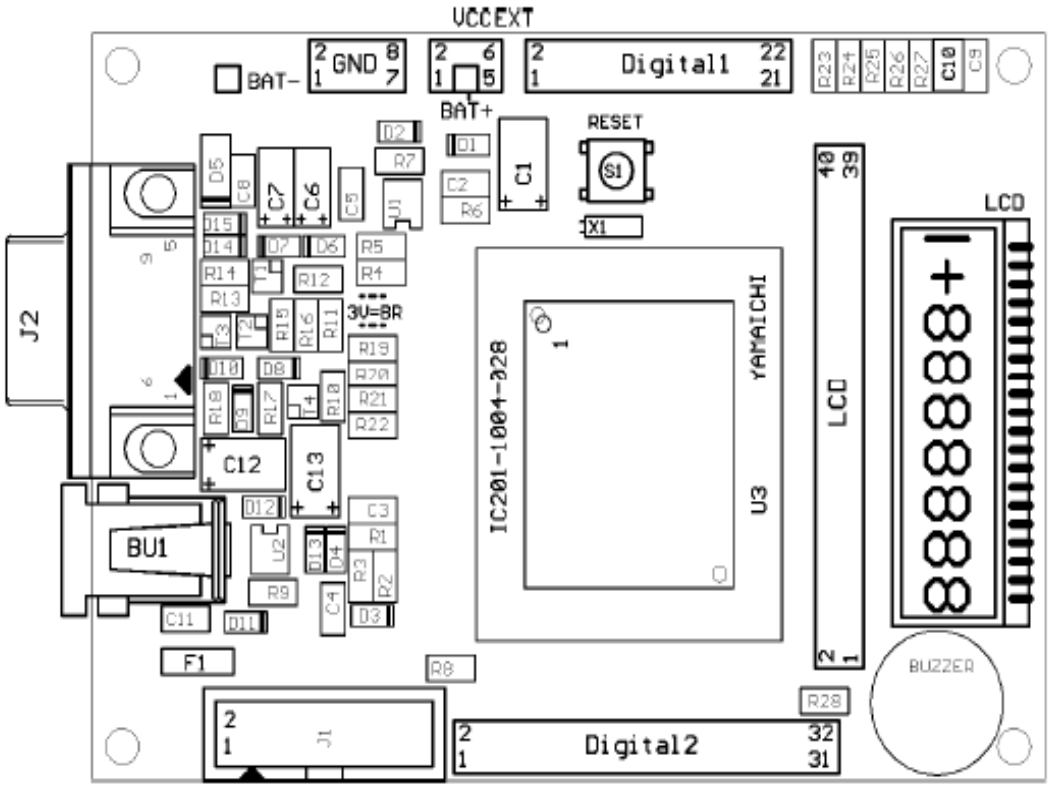


Fig 3.2

3.3 DEVELOPEMENT FLOW

Applications are developed in assembler and/or C using the Workbench, and they are debugged using CSPY.C-SPY can be configured to operate with the EVK, or with a software simulation of the MSP430device. When targeting the EVK, applications are best downloaded into the device RAM memory where they can be run and debugged (using breakpoints and single step). Also, applications can be downloaded into RAM very quickly. Using the Serial Programming Adapter, it is possible to program applications into the device EPROM. Breakpoints and single step cannot be used in EPROM, and changes to the EPROM may require that the EPROM be erased (which requires many minutes of exposure to UV light). It is greatly easier to develop an application in RAM than in EPROM. However, unlike EPROM, RAM memory is volatile (i.e., its contents is lost when device power is removed).
C-SPY operates in conjunction with the ROM-Monitor. The ROM-Monitor is an application what executes in the MSP430 on the target EVK. Note: A ROM-Monitor **must** be present in the EVK MSP430 if it is to be controlled by C-SPY. Basic C-SPY commands (read memory, read registers, single step, etc.) are sent to the ROM-Monitor where they are executed. The results of the command execution are returned to C-SPY where they are displayed. C-SPY and the ROM-Monitor communicate using a 4800 baud serial interface.

Using Kickstart

The Kickstart development environment is limited. The following restrictions are in place:
1. The C compiler has no support for floating-point arithmetic, and it will not generate assembly code output.
2. The linker will link a maximum of 2K bytes code originating from C source (but an unlimited amount of code originating from assembler source).
3. C-SPY does not support code profiling.
4. The IAR Simulator will input a maximum of 400 C source lines (but an unlimited number of assembler source lines). The TI Simulator has no such limitations. A “full” (i.e., unrestricted) version of the software tools can be purchased from IAR. A “mid-featured” tool set – called Baseline, with an 8K byte C code size limitation and no floating-point arithmetic – is also available.

3.4 PROJECT SETTINGS

1. Choose the "–v0, 310/320 series (no hardware multiplier)" when developing with MSP430 devices without a hardware multiplier. Choose the "–v1, 330 series (hardware multiplier present)" when developing with MSP430 devices with a hardware multiplier. (GENERAL, TARGET)

2. Enable Debug Information in the compiler. (ICC430, DEBUG)

3. Enable Generate Debug Information in the assembler. (A430, CODE GENERATION)

4. Enable Debug Info in the linker Format section. (XLINK, OUTPUT)

5. Override the XCL File Name. Refer to System Files below. (XLINK, INCLUDE)

6. Override and select the correct Chip Description for C-SPY. Refer to System Files below. (CSPY, SETUP)

7. Configure the ROM-monitor: Suppress download of ROM-Monitor, and Remap interrupt vectors of the Application. Refer to the IAR ROM-Monitor Supplement for an explanation of these settings.

8. Select the C-SPY driver: Select Simulator to debug on the simulator. Select ROM-Monitor to debug on the EVK. (CSPY, SETUP). Select the active serial port in SERIAL PORT and configure the settings in the ROM-monitor tab: 4800-Even-8-2-None (addition information about the settings can be found in the ROM-Monitor Supplement).

9. The ROM-Monitor makes use of R4. When using the C-compiler, select Exclude R4 (ICC430, CODEGENERATION)

10. Avoid the use of absolute pathnames when referencing files. Instead, use the relative pathname keywords $TOOLKIT_DIR$ and $PROJ_DIR$. Refer to the IAR documentation for a description of these keywords. The use of relative pathnames will permit projects to be moved easily, and projects will not require modification when IAR systems are upgraded (say, from Kickstart, or Baseline, to full).

System Files:

The following configuration and special files are provided to facilitate development of MSP430 applications under Kickstart/MSP-EVK430S3x0:

1. Linker control file for point 5. above that supports assembler development in the '32x device:$TOOLKIT_DIR$\icc430\Lnk430KSrom_320A.xcl

Linker control file for point 5. above that supports assembler development in the '33x device.: $TOOLKIT_DIR$\icc430\Lnk430KSrom_330A.xcl

2. Linker control file for point 5. above that supports C development in the '32x device:
$TOOLKIT_DIR$\icc430\Lnk430KSrom_320.xcl
Linker control file for point 5. above that supports C development in the '33x device:
$TOOLKIT_DIR$\icc430\Lnk430KSrom_330.xcl
3. Chip Description file for point 6. above that supports debugging the ‘32x device:
$TOOLKIT_DIR$\cw430\msp430E325.ddf
4. Chip Description file for point 6. above that supports debugging the ‘33x device:
$TOOLKIT_DIR$\cw430\msp430E337.ddf
5. Device definition “#include” files:
$TOOLKIT_DIR$\inc\msp430x32x.h
$TOOLKIT_DIR$\inc\msp430x33x.h
6. C library files:
$TOOLKIT_DIR$\lib\cl430ks.r43 // For ‘3xx devices without hardware multiplier.
$TOOLKIT_DIR$\lib\cl430ksm.r43 // For ‘3xx devices with hardware multiplier.
// For ‘3xx devices without hardware multiplier, Position Independent Code.
$TOOLKIT_DIR$\lib\cl430ks_pic.r43
// For ‘3xx devices with hardware multiplier, Position Independent Code.
$TOOLKIT_DIR$\lib\cl430ksm_pic.r43

3.5 OTHER INFORMATIONS

1. The state of the machine (registers, memory, etc.) is undefined following a reset. The only exception to the above statement is that the PC is loaded with the word at 0xfffe (i.e., the reset vector).
2. A common MSP430 “mistake” is to fail to disable the Watchdog mechanism; the Watchdog is enabled by default, and it will reset the device if not disabled or properly handled by your application. Refer to Known Problems 15.
3. C-SPY is capable of downloading data into RAM memory (without using a PRGS). A PRGS is required to download/program EPROM memory.
4. C-SPY is capable of debugging applications that utilize interrupts and low power modes. It is not possible to single step beyond an instruction that enables a low power mode as the instruction effectively turns off the device. Refer to Known Problems 20.
5. C-SPY is incapable of accessing the device registers and memory while the device is running. The user must stop the device in order to access device registers and memory.

6. When adding source files to a project, do not add files that are #included by source files that have already been added to the project (say, an .h file within a .c or .s43 file). These files will be added to the project file hierarchy automatically.

7. In assembler, enclosing a string in double-quotes ("string") automatically prepends a zero byte to thestring. Enclosing a string in single-quotes ('string') does not.

8. When using the compiler or the assembler, if the last character of a source line is backslash (\), thes ubsequent carriage return/line feed is ignored (i.e., it is as if the current line and the next line are a single line). When used in this way, the backslash character is a "Line Continuation" character.

9. C-SPY implements breakpoints and single step by temporarily replacing user instructions with trap (a.k.a. breakpoint) instructions. For this reason, breakpoints and single step can only occur while the CPU is executing from RAM memory. It is not possible to set a breakpoint or single step through EPROM.

The MSP-EVK430S320 has 352 bytes of RAM available for the user program.

The MSP-EVK430S330 has 864 bytes of RAM available for the user program.

It is possible to develop a total system with the EVK by using a modular approach to work with the limited resources. Program fragments can be developed and tested within the available RAM, and then programmed into EPROM once they are complete. The linker (XLINK) is then used to manage references to the completed program fragments.

C-SPY does provide a non-real time data breakpoint mechanism; it is possible to associate with an address breakpoint an expression that C-SPY evaluates when the breakpoint is hit. If the expression is FALSE, program execution resumes. And, if the expression is TRUE, C-SPY halts the program execution and displays the machine state. The breakpoint expression can be arbitrarily complex. Refer to the C-SPY documentation for a description of this data breakpoint mechanism.

10. The TI Simulator for Kickstart fully simulates the MSP430, including all peripheral modules and interrupts.

11. The linker output format **must be** "Debug info" or "Debug info with terminal I/O" (.d43) for use with C-SPY. If the linker output format is .txt, the .d43 file is not updated and subsequent C-SPY sessions will utilize the existing .d43 file when present. Thus, you can loose synchronization between the source and the code being debugged. Do not launch C-SPY when you have "Other" output file formats elected for the XLINK options.

12. Position Independent versions of the C libraries are provided. The libraries are named cl430ks_pic.r43 (no support for hardware multiply) and cl430ksm_pic.r43 (support for hardware multiply).

13. Within the C libraries that support devices with a hardware multiplier (cl430ksm.r43, cl430ksm_pic.r43), interrupts are disabled during critical sections of the floating-point functions.

14. Within C-SPY, (most) state information (breakpoint settings, etc.) can be preserved by selecting OPTIONS->SETTINGS->WINDOW SETTINGS->GENERAL->RESTORE STATES). It is noted the feature does not take effect until the next C-SPY session (i.e., C-SPY must be stopped and restarted to enable this feature).

15. It is possible to mix assembler and C programs within the Workbench. TI is developing an application note describing how this is achieved.

Chapter 4

HARDWARE INTERFACE –II

LCD CONNECTIONS
LCD CONTROLLER/DRIVER FEATURES
LOOK UP TABLE

4.1 LCD CONNECTIONS

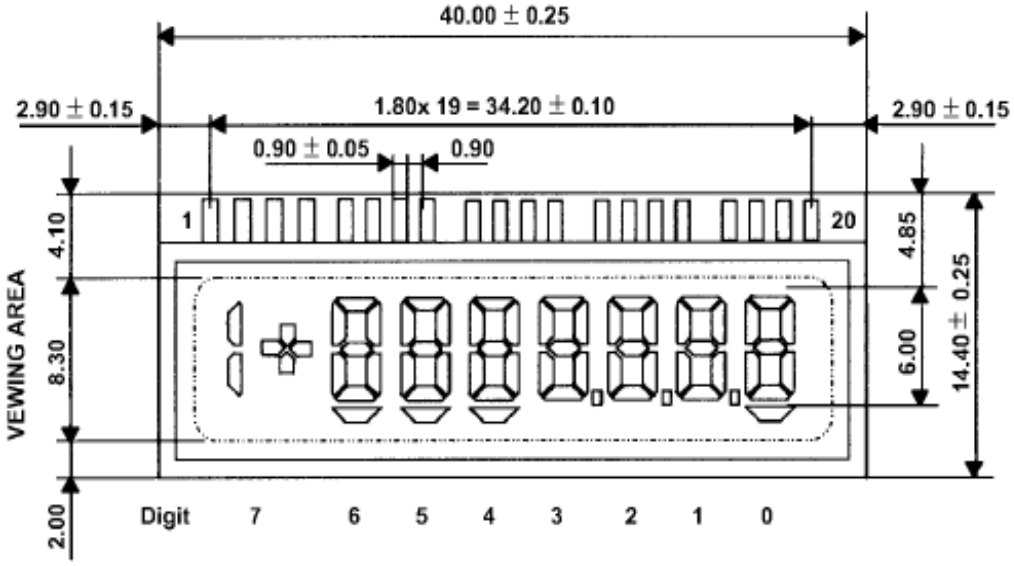
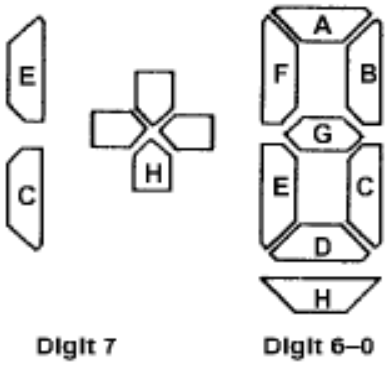


Fig 4.1

LCD is connected to EVK as follows:

PIN NO.	COM1	COM2	COM3	COM4
1	–	–	COM3	–
2	–	–	–	COM4
3	–	COM2	–	–
4	COM1	–	–	–
5	–	–	–	–
6	6C	6F	6H	6E
7	6A	6B	6D	6G
8	5C	5F	5H	5E
9	5A	5B	5D	5G
10	4C	4F	4H	4E
11	4A	4B	4D	4G
12	3C	3F	3H	3E
13	3A	3B	3D	3G
14	2C	2F	2H	2E
15	2A	2B	2D	2G
16	1C	1F	1H	1E
17	1A	1B	1D	1G
18	0C	0F	0H	0E
19	0A	0B	0D	0G
20	COM1	–	–	–

Table 4.1



4.2 LCD CONTROLLER/DRIVER FEATURES

The LCD controller/driver features are:

_ Display memory

_ Automatic signal generation

_ Support for 4 types of LCDs:

_ Static

_ 2 MUX, 1/2 bias

_ 3 MUX, 1/3 bias

_ 4 MUX, 1/3 bias

_ Multiple frame frequencies

_ Unused segment outputs may be used as general-purpose outputs.

_ Unused display memory may be used as normal memory

_ Operates using the basic timer with the auxiliary clock (ACLK).

4.2.1 LCD TIMING GENERATION

LCD Timing Generation

The LCD controller uses the fLCD signal from the Basic Timer1 to generate the timing for common and segment lines. The frequency fLCD of signal is generated from ACLK. Using a 32,768-Hz crystal ,the fLCD frequency can be 1024 Hz, 512 Hz, 256 Hz, or 128 Hz. Bits FRFQ1and FRFQ0 allow the correct selection of frame frequency. The proper frequency fLCD depends on the LCD's requirement for framing frequency and LCD multiplex rate, and is calculated by:

$f_{LCD} = 2 \times \text{MUX rate} \times f_{Framing}$

A 3 MUX example follows:

LCD data sheet: $f_{Framing}$ = 100 Hz .... 30 Hz

FRFQ: $f_{LCD} = 6 \times f_{Framing}$

$f_{LCD} = 6 \times 100$ Hz = 600 Hz ... 6×30 Hz = 180 Hz

Select f_{LCD}: 1024 Hz, 512 Hz, 256 Hz, or 128 Hz

$f_{LCD} = 32{,}768/128 = 256$ Hz FRFQ1 = 1; FRFQ0 = 0

4.2.2 LCD CONTROL REGISTER

The LCD control register contents define the mode and operating conditions.The LCD module is byte structured and should be accessed using byte instructions (suffix .B). All LCD control register bits are reset with a PUC signal.

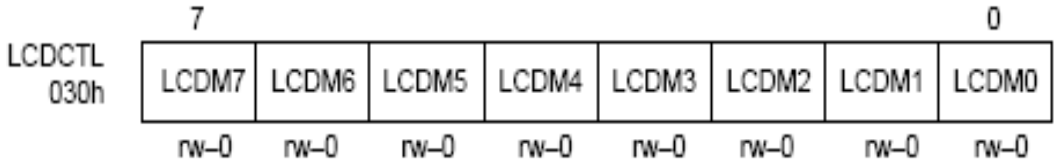


Fig 4.2.

LCDM4	LCDM3	LCDM2	Display Mode
X	X	0	All segments are deselected. The port outputs remain stable. This supports flashing LCD applications.
0	0	1	Static mode
0	1	1	2 MUX mode
1	0	1	3 MUX mode
1	1	1	4 MUX mode

Table 4.2

The primary function of the LCDM2 bit is to support flashing or blinking the LCD. The LCDM2 bit is logically ANDed with each segment's display memory value to turn each LCD segment on or off. WhenLCDM2=1, each LCD segment is on or off according to the LCD display memory. When LCDM2=0, each LCD segment is off, therefore blanking the LCD.

4.2.3 LCD MEMORY

The LCD memory map is shown in Figure 4.3. Each individual memory bit corresponds to one LCD segment. To turn on an LCD segment the memory bit is simply set. To turn off an LCD segment, the memory is reset. The mapping of each LCD segment in an application depends on the connections between the '430 and the LCD and on the LCD pinout .Examples for each of the four modes follow including an LCD with pin out, the'430-to-LCD

connections, and the resulting data mapping.

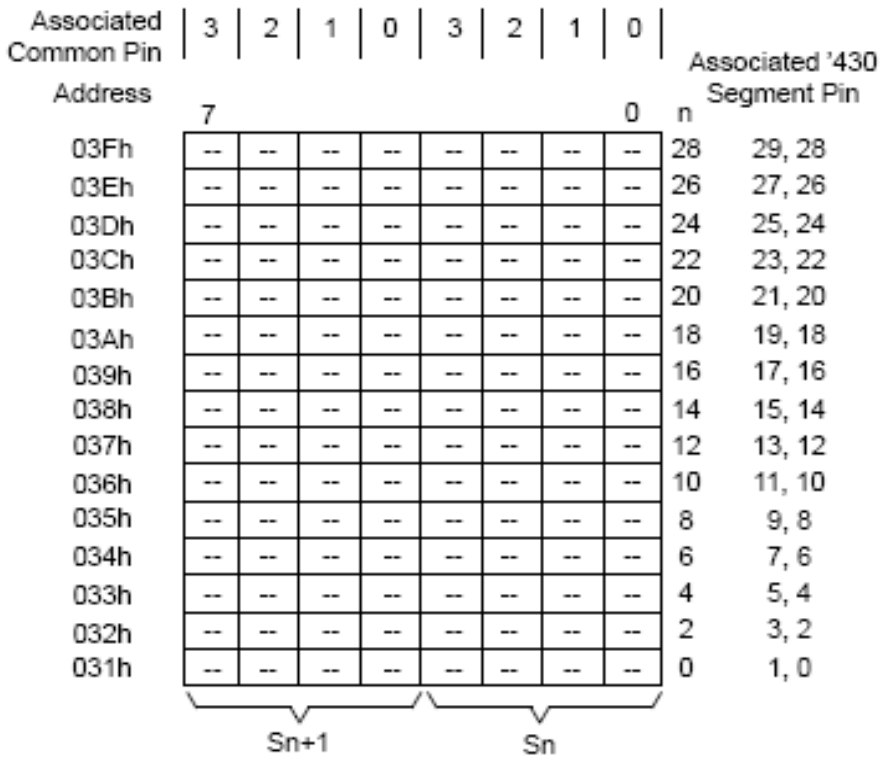


Fig 4.3

4.3 LOOK UP TABLE

```
a       equ   01h
b       equ   02h
c       equ   10h
d       equ   04h
e       equ   80h
f       equ   20h
g       equ   08h
h       equ   40h
```

character definitions

LCD_Tab

```
DB   a+b+c+d+e+f                ; displays "0"
DB   b+c                        ; displays "1"
DB   a+b+d+e+g                  ; displays "2"
DB   a+b+c+d+g                  ; displays "3"
DB   b+c+f+g                    ; displays "4"
DB   a+c+d+f+g                  ; displays "5"
DB   a+c+d+e+f+g                ; displays "6"
DB   a+b+c                      ; displays "7"
DB   a+b+c+d+e+f+g              ; displays "8"
DB   a+b+c+d+f+g                ; displays "9"
DB   0                          ; displays ":" blank
DB   g                          ; displays ";" -
DB   a+d+e+f                    ; displays "<" [
DB   d+g                        ; displays "="
DB   a+b+c+d                    ; displays ">" ]
DB   a+b+e+g                    ; displays "?"
DB   a+b+d+e+f+g                ; displays "@"
DB   a+b+c+e+f+g                ; displays "A"
DB   c+d+e+f+g                  ; displays "B" b
DB   a+d+e+f                    ; displays "C"
DB   b+c+d+e+g                  ; displays "D" d
DB   a+d+e+f+g                  ; displays "E"
DB   a+e+f+g                    ; displays "F"
DB   a+c+d+e+f+g                ; displays "G"
DB   b+c+e+f+g                  ; displays "H"
DB   b+c                        ; displays "I"
DB   b+c+d+e                    ; displays "J"
DB   0                          ; displays "K"
DB   d+e+f                      ; displays "L"
DB   a+b+c+e+f                  ; displays "M"
DB   c+e+g                      ; displays "N" n
DB   c+d+e+g                    ; displays "O" o
DB   a+b+e+f+g                  ; displays "P"
```

```
DB   0                          ; displays "Q"
DB   e+g                        ; displays "R" r
DB   a+c+d+f+g                  ; displays "S"
DB   d+e+f+g                    ; displays "T" t
DB   c+d+e                      ; displays "U" u
DB   0                          ; displays "V"
DB   0                         ; displays "W"
DB   0                         ; displays "X"
DB   b+c+d+f+g                  ; displays "Y"
DB   a+b+d+e+g                  ; displays "Z" 2
```

CHAPTER 5

IMPLEMENTATION OF DIGITAL THERMOMETER

SLOPE A/D TECHNIQUE USING MSP430

HARDWARE IMPLEMENTATION

ASSEMBLYLANGUAGE PROGRAM

5.1 SLOPE A/D TECHNIQUE USING MSP430

5.1.1 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 Fig.5.1 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 Cm through it.

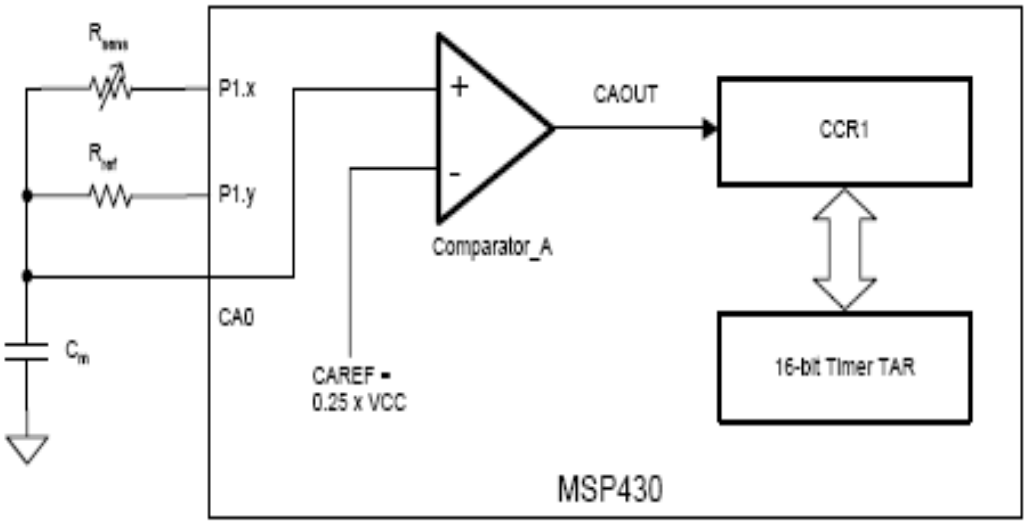


Fig 5.1

To measure the resistor value R_{sens}, capacitor C_m is first charged to the digital I/O high Voltage ($V_{OH} \cong V_{CC}$) by outputting a high on either P1.x or P1.y. After configuring the timer, the capacitor is discharged through via R_{sens} 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)* V_{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 τ = R_{sens} C_m .The value within this range depends on the accuracy required.

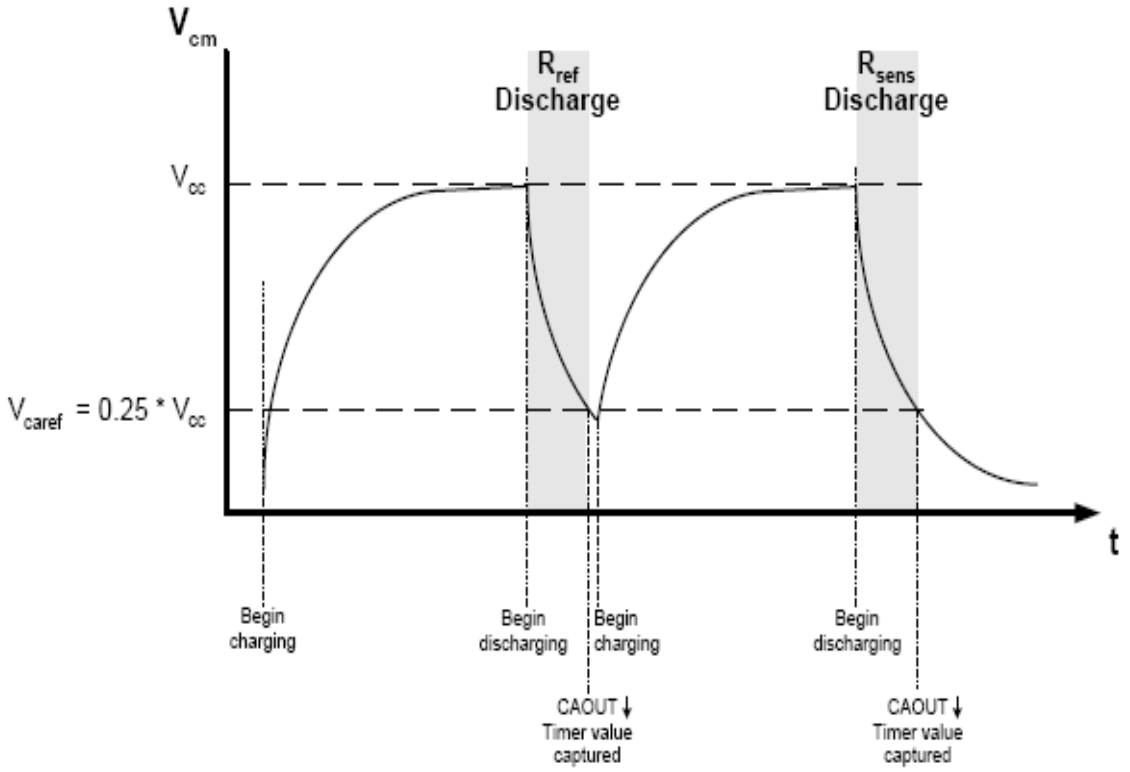


Fig 5.2

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 an additional code and execution cycle 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 divides per measurement.

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

$V(t)=V_0e^{-t/RC}$ (equation 1)

Given these specific and known values:

$V(t) = V_{caref}$

$V_0 = V_{OH} \cong V_{CC}$

$R = R_{sens}$

$C = C_m$

this produces:

$V_{caref}= V_{CC}\, e^{-t}_{sens}{}^{/}\, R_{sens}\, C_m$ *(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 Rref 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:

$R_{sens} /t_{sens} = R_{ref} / t_{ref}$ *(equation 3)*

Therefore, if the capacitor discharges 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.

5.2 HARDWARE IMPLEMENTATION

5.2.1 HARDWARE INTERFACING:

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 hardware interface circuit is simply a thermistor (Radio Shack #271–110), a 10-k□reference resistor, and a 0.1-□F capacitor. The components connect directly to the MSP430 as shown in Figure 2. An LCD display must also be connected if a visual readout of the measurements is desired. The circuit performs a measurement by charging the capacitor to approximately V_{CC} , then discharging it through the reference resistor, while counting the number of internal clock cycles it takes until the CIN input goes low. The capacitor is charged to near V_{CC} again and then discharged through the thermistor, while counting the internal clock cycles required. The unknown resistance value of the thermistor can then be determined by taking a ratio of clock cycles required to discharge the capacitor via the thermistor, versus the number required to discharge via the known reference resistor value then multiplying the result by the value of the reference resistor. Software routines calculate the actual value of the thermistor, equate the value to a corresponding temperature, convert it to degrees Fahrenheit, and display the value on the LCD. Even though the last reading is constantly displayed, the MSP430 spends the majority of its time in low power mode 3 (LPM3). This time could be used to make additional measurements, to communicate with other components, or to perform calculations. The three components used to make the temperature measurement can be connected directly to a Texas Instruments MSP430 starter kit (STK) or evaluation kit (EVK). All of the other required connections, including those for the LCD, are already in place on the STK and EVK boards. The attached code is sized to fit completely into the 512 bytes of RAM memory that is available on the STK and EVK boards, which are based on the MSP430x325 devices. The code can be loaded into RAM through the serial port of a PC, using the interface included with the boards.

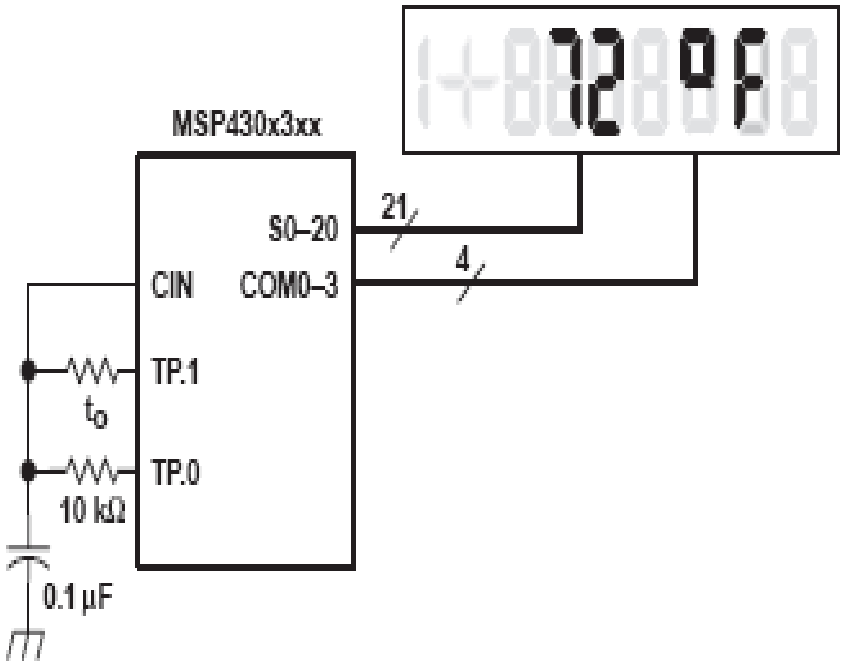


Fig 5.3

5.2 .2 APPLICATION INFORMATION:

The formula to measure the discharge time of the capacitor is:

$t = -R \times C \times \ln (V_{ref} / V_{CC})$

$t = N \times t_{clock}$ (N is the number of clocks cycles)

$N \times t_{clock} = -R \times C \times \ln (V_{ref} / V_{CC})$

$N = -R \times C \times f_{clock} \times \ln (V_{ref} / V_{CC})$

The values of C, fclock, and Vref/Vcc are known. The value of the resistive sensor can be determined by the following formula, since the value of the reference resistor is a stable and known value.

$N_{sensor} / N_{ref} = [-R_{sensor} \times C \times f_{clock} \times \ln (V_{ref} / V_{CC})] \setminus [-R_{ref} \times C \times f_{clock} \times \ln (V_{ref} / V_{CC})]$

$N_{sensor} / N_{ref} = R_{sensor} / R_{ref}$

$R_{sensor} = R_{ref} \times N_{sensor} / N_{ref}$

The formula resulting from the circuit in FIG 5.2 would be equal to:

$R_{sensor} = 10 \times N_{sensor} / N_{ref}$

5.3 ASSEMBLY LANGUAGE PROGRAM FOR DIGITAL THERMOMETER

```
;*************************************************************
;msp430-based DIGITAL THERMOMETER using slope a/d technique to measure
sensor resistance
;*************************************************************
; REGISTERS USED TO SUPPORT CALCULATION OF SENSOR RESISTANCE
;*************************************************************
#define MLTPLR_HW   R5
#define TEN_K       R6
#define BITTEST     R7
#define MRESLT_HW   R8
#define MRESLT_LW   R9
#define LPCNTR      R10
#define RESULT      R11
;*************************************************************
; DEFINITION SECTION FOR TIMER PORT ADC
;*************************************************************
BTLOAD      EQU       035H ; LOAD ACTUAL 0.5 SECOND INTERRUPT
TPCTL       EQU       04BH ; TIMER PORT CONTROL REGISTER (04BH)
TPSSEL0     EQU       040H ; CLK SOURCE 0=CMP, 1=ACLK (BIT 6 OF TPCTL)
ENB         EQU      020H ; CONTROLS EN1 OF TPCNT1; 1(+ENA=1)=CMP (BIT 5 OF
;TPCTL)
ENA         EQU      010H ; CONTROLS EN1 OF TPCNT1; 1(+ENB=1)=CMP (BIT 4 OF
;TPCTL)
EN1         EQU      008H ; ENABLE FOR TPCNT1 READ ONLY (BIT 3; FO TPCTL)
RC2FG       EQU      004H ; RIPPLE CARRY TPCNT2 (BIT 2 OF TPCTL)
EN1FG       EQU      001H ; EN1 FLAG BIT (BIT 0 OF TPCTL)
TPIE        EQU      004H ; TIMER PORT INTERRUPT ENABLE (BIT 3 OF IE2)
TPCNT1      EQU      04CH ; COUNTER LOW BYTE
TPCNT2      EQU      04DH ; COUNTER HIGH BYTE
TPD         EQU      04EH ; TP DATA REGISTER (0-5=TP OUTPUT; DATA, 6=CPON,
;7=B16=2-8B OR 1-16B CNTR)
B16         EQU      080H ; SEPARATE TIMERS (0), OR 1-16 BIT; TIMER (1)
CPON        EQU      040H ; COMP OFF (0), COMP ON (1)
TPDMAX      EQU      002H ; BIT POSITION OUTPUT TPD.MAX; (2=BIT1=TPD.1)
TPE         EQU      04FH ; TP DATA ENABLE REGISTER (0-5=TPD; ENABLES, 6-
;7=TPCNT2 CLK)
MSTACK      EQU      03D2H ; RESULT STACK - 1ST WORD
PRESET      EQU      0E8H ; PRESET TPCNT2 FOR CHARGING OF C, COUNT; STOPS
;WHEN TPCNT2 OVERFLOWS, VALUE ALLOWS; CAP TO CHARGE FOR 6 RC TIME
CONSTANTS
;*************************************************************************
; CONTROL REGISTER DEFININTIONS
;*************************************************************************
```

```
IE1         EQU      0H ; INTERRUPT ENABLE REGISTER 1
IE2         EQU      01H ; INTERRUPT ENABLE REGISTER 2
P01IE       EQU      08H ; P0.1 INTERRUPT ENABLE IN IE1
BTIE        EQU      080H ; BASIC TIMER INTERRUPT ENABLE IN IE2
IFG1        EQU      02H ; INTERRUPT FLAG REGISTER 1
IFG2        EQU      03H ; INTERRUPT FLAG REGISTER 2
LCDCTL      EQU      030H ; LCD CONTROL REGISTER
LCDM        EQU      031H ; FIRST LCD DISPLAY MEM LOCATION
BTCTL       EQU      040H ; BASIC TIMER CONTROL REGISTER
BTCNT1      EQU      0046H ; BASIC TIMER COUNTER 1
BTCNT2      EQU      0047H ; BASIC TIMER COUNTER 2
WDTCTL      EQU      0120H ; WATCHDOG CONTROL REGISTER
WDTHOLD     EQU      080H ; PATTERN TO HOLD WATCHDOG
WDT_KEY     EQU      05A00H ; KEY TO ACCESS WATCHDOG
WDT_STOP    EQU      05A80H ; WATCHDOG HOLD+KEY;
GIE         SET      8H ; GENERAL INTERRUPT ENABLE
CPUOFF      SET      10H ; BIT TO TURN CPU OFF
OSCOFF      SET      20H ; BIT TO TURN OSCILLATOR OFF
SCG0        SET      40H ; SYS CLK GENERATOR CONTROL BIT 0
SCG1        SET      80H ; SYS CLK GENERATOR CONTROL BIT 1
LPM0        SET      CPUOFF ; BITS TO SET FOR LOW POWER MODE 0
LPM1        SET      SCG0+CPUOFF ; " " " " " " " 1
LPM2        SET      SCG1+CPUOFF ; " " " " " " " 2
LPM3        SET      SCG1+SCG0+CPUOFF; " " " " " " " 3
LPM4        SET      OSCOFF+CPUOFF ; " " " " " " " 4

            RSEG CSTACK
            DS   0
;-------------------------------------------------------------------------------
; RESET PROGRAM
;-------------------------------------------------------------------------------
            RSEG RAMCODE
RESET       MOV.W #SFE(CSTACK),SP
;-------------------------------------------------------------------------------
; SETUP UP PERIPHERALS
;-------------------------------------------------------------------------------
SETUP
SETUPINT    MOV.B #P01IE,&IE1 ; ENABLE P0.1/UART FOR RS232 MONITOR
            MOV.B #BTIE+TPIE,&IE2 ; ENABLE B.TIMER, & TMR. PORT INTRPTS.
            CLR.B &IFG1 ; CLEAR ANY INTERRUPT FLAGS
            CLR.B &IFG2 ; CLEAR ANY INTERRUPT FLAGS
            EINT ; ENABLE INTERRUPTS
SETUPWDT    MOV #WDT_STOP,&WDTCTL ; STOP WATCHDOG TIMER
SETUPLCD    MOV.B #0FFH,&LCDCTL ; STK LCD, ALL SEG, 4MUX
SETUPBT     MOV.B #BTLOAD,&BTCTL ; LOAD BASIC TIMER WITH INTERRUPT FREQ
            CLR.B &BTCNT1 ; CLEAR BT COUNTER 1
```

```
            CLR.B &BTCNT2 ; CLEAR BT COUNTER 2

CLEARLCD    MOV #15,R6 ; 15 LCD MEM LOCATIONS TO CLEAR
CLEAR1      MOV.B #0,LCDM-1(R6) ; WRITE ZEROS IN LCD RAM LOCATIONS
            DEC R6 ; ALL LCD MEM CLEAR?
            JNZ CLEAR1 ; MORE LCD MEM TO CLEAR GO
;*******************************************************************************
; BEGIN MAIN PROGRAM
;*******************************************************************************
BEGIN       BIS #LPM3,SR ; SET SR BITS FOR LPM3
;*******************************************************************************
; MEASUREMENT SUBROUTINE WITHOUT INTERRUPT. TP.2-.5 ARE NOT USED
; AND THEREFORE OVERWRITTEN. ONLY TPD.0 & 1 USED.
; INITIALIZATION: STACK INDEX = 0, START WITH TPD.1
; 16-BIT TIMER, MCLK, CIN ENABLES COUNTING
;*******************************************************************************
MEASURE     PUSH.B #TPDMAX ;PUSH TO STACK FOR LATER USE
            CLR R8 ;INDEX FOR RESULT STACK
MEASLOP     MOV.B #(TPSSEL0*3)+ENA,&TPCTL ;TPCNT1 CLK=MCLK, EN1=1
;*******************************************************************************
; CAPACITOR C IS CHARGED UP FOR >5 TAU. N-1 OUTPUTS ARE USED
;*******************************************************************************
            MOV.B #081H,&TPD ;1-16BIT COUNTER, SELECT CHARGE OUTPUTS
            MOV.B #01H,&TPE ;ENABLE CHARGE OUTPUTS
            MOV.B #PRESET,&TPCNT2 ;LOAD NEG. CHARGE TIME
            BIS #CPUOFF,SR ;LOW POWER MODE TO SAVE POWER
            MOV.B @SP,&TPE ;ENABLE ONLY ACTUAL SENSOR
            CLR.B &TPCNT2
;*******************************************************************************
; SWITCH ALL INTERRUPTS OFF TO ALLOW NON-INTERRUPTED START OF
; TIMER AND CAPACITOR DISCHARGE
;*******************************************************************************
            DINT ;DISABLE INTERRUPTS-ALLOW NEXT 2
            CLR.B &TPCNT1 ;CLEAR LOW BYTE OF TIMER
            BIC.B @SP,&TPD ;SWITCH ACTUAL SENSOR TO LOW
            MOV.B #(TPSSEL0*3)+ENA+ENB,&TPCTL ;TPCNT1 CLK=MCLK, ENABLE CIN
;INPUT
            EINT ;ENABLE INTERRUPTS-COMMON START
            BIS #CPUOFF,SR ;CPU OFF TO SAVE POWER
;*******************************************************************************
; EN=0: END OF CONVERSION: STORE 2X8 BIT RESULT ON MSTACK
; ADDRESS NEXT SENSOR: IF NO OTHER SENSOR END REACHED
;*******************************************************************************
            MOV.B &TPCNT1,MSTACK(R8) ;STORE RESULT ON STACK
            MOV.B &TPCNT2,MSTACK+1(R8) ;STORE HIGH BYTE IN NEXT STACK BYTE
L$301       INCD R8 ;ADDRESS NEXT WORD
```

```
            RRA.B @SP ;NEXT OUTPUT TPD.X
            JNC MEASLOP ;IF C=1 - FINISHED
            INCD SP ;HOUSEKEEPING-TPDMAX OFF STACK

;*******************************************************************************
; CALCULATE RESISTANCE OF SENSOR
;*******************************************************************************
; UNSIGNED MULTIPLY SUBROUTINE: MSTACK X TEN_K -> MRESLT_HW/MRESLT_LW
; USED REGISTERS MSTACK, TEN_K, MLTPLR_HW, MRESLT_LW, MRESLT_HW, BITTEST
; UNSIGNED MULTIPLY AND ACCUMULATE SUBROUTINE:
; (MSTACK X TEN_K) + MRESLT_HW|MRESLT_LW -> MRESLT_HW|MRESLT_LW
;*******************************************************************************
CALC_RES
            MOV #10000,TEN_K ; MOVE 10,000 DECIMAL INTO TEN_K
MPYU        CLR MRESLT_LW ; 0 -> LSBS RESULT
            CLR MRESLT_HW ; 0 -> MSBS RESULT
MACU        CLR MLTPLR_HW ; MSBS MULTIPLIER
            MOV #1,BITTEST ; BIT TEST REGISTER
L$002       BIT BITTEST,MSTACK ; TEST ACTUAL BIT
            JZ L$01 ; IF 0: DO NOTHING
            ADD TEN_K,MRESLT_LW ; IF 1: ADD MULTIPLIER TO RESULT
            ADDC MLTPLR_HW,MRESLT_HW
L$01        RLA TEN_K ; MULTIPLIER X 2
            RLC MLTPLR_HW ;
            RLA BITTEST ; NEXT BIT TO TEST
            JNC L$002 ; IF BIT IN CARRY: FINISHED
;*******************************************************************************
; UNSIGNED DIVISION SUBROUTINE 32-BIT BY 16-BIT
; REGISTERS USED (MSTACK+2), MRESLT_LW, RESULT, LPCNTR, MRESLT_HW
; MRESLT_HW MRESLT_LW / (MSTACK+2) -> RESULT REMAINDER IN MRESLT_HW
; RETURN: CARRY = 0: OK CARRY = 1: QUOTIENT > 16 BITS
;*******************************************************************************
DIVIDE      CLR RESULT ; CLEAR RESULT
            MOV #17,LPCNTR ; INITIALIZE LOOP COUNTER
DIV1        CMP MSTACK+2,MRESLT_HW ;
            JLO DIV2
            SUB MSTACK+2,MRESLT_HW
DIV2        RLC RESULT
            JC RES_2_F ; ERROR: RESULT > 16 BITS
            DEC LPCNTR ; DECREMENT LOOP COUNTER
            JZ DIV3 ; IS 0: TERMINATE W/O ERROR
            RLA MRESLT_LW
            RLC MRESLT_HW
            JNC DIV1
            SUB MSTACK+2,MRESLT_HW
            SETC
```

```
            JMP DIV2
DIV3        CLRC ; NO ERROR, C = 0
;*******************************************************************************
; CONVERT RESISTANCE OF SENSOR TO DEGREES F FOR DISPLAY
;*******************************************************************************
RES_2_F
            CLR R12 ;POINTS TO VALUE IN RESISTANCE TABLE
            MOV #064H,R13 ;MOVE MINIMUM TEMP-1 INTO TEMP INDICATOR
            JMP FIRST_CMP ;AVOID ADDING 1 ON FIRST COMPARE
CHECK_R     INCD R12 ;INCREMENT RESISTANCE TABLE POINTER
            DADD #1,R13 ;DECIMAL INCREMENT COUNTER
FIRST_CMP   CMP RESIS_TAB(R12),RESULT ;COMPARE TABLE VALUE TO; CALCULATED
RESISTANCE
            JNC CHECK_R ;JUMP IF RSENSOR < TABLE VALUE @ POINTER

;*******************************************************************************
; DISPLAY "F" AND DEGREE SIGN ON LCD
;*******************************************************************************
DISPLAY         MOV #LCDM+1,R14
                MOV.B #10,R15;
                MOV.B LCD_TAB(R15),0(R14)
                MOV #LCDM+2,R14
                MOV.B #11,R15;
                MOV.B LCD_TAB(R15),0(R14)
;*******************************************************************************
; DISPLAY BCD NUMBER IN R13 ON LCD
;*******************************************************************************
            MOV R13,R12 ;COPY BCD NUMBER TO R12
            MOV #LCDM+4,R14 ;LOWER DIGIT LCD MEMORY LOCATION INTO R14
            BIC #0FFF0H,R13 ;BLANK OFF ALL BUT LOWEST DIGIT
            MOV.B LCD_TAB(R13),0(R14) ;LOWER DIGIT TO LCD
            MOV R12,R13 ;REPLACE VALUE OF R13
            RRA R13 ;ROTATE VALUE RIGHT FOUR TIMES
            RRA R13
            RRA R13
            RRA R13
            BIC #0FFF0H,R13 ;BLANK OFF ALL BUT LOWEST DIGIT
            MOV.B LCD_TAB(R13),1(R14) ;UPPER DIGIT TO LCD
            JMP BEGIN ;LOOP BACK TO BEGINNING OF PROGRAM
;*******************************************************************************
; BASIC TIMER INTERRUPT SERVICE ROUTINE:
; CPU IS RETURNED TO ACTIVE BY CLEARING LPM3 BITS ON SR ON THE STACK,
;*******************************************************************************
BTINT       BIC #LPM3,0(SP) ; CLEAR SR LPM3 BITS, ON TOP OF STACK
            RETI
;*******************************************************************************
```

```
; TIMER PORT INTERRUT SERVICE ROUTINE:
; SYSTEM RETURNED TO ACTIVE ON RETI
;*******************************************************************************
TPINT       CLR.b &TPCTL ; CLEAR TP INTERRUPT FLAGS, TP OFF
            BIC #LPM3,0(SP) ; CLEAR SR LMP3 BITS, ON TOP OF STACK
            RETI
;*******************************************************************************
; STK LCD TYPE
;*******************************************************************************
LCD_TYPE
A           EQU       01H
B1          EQU       02H
C           EQU       10H
D           EQU       04H
E           EQU       80H
F           EQU       20H
G           EQU       08H
H           EQU       40H
LCD_TAB
           DB A+B1+C+D+E+F ; DISPLAYS ”0”
           DB B1+C ; DISPLAYS ”1”
           DB A+B1+D+E+G ; DISPLAYS ”2”
           DB A+B1+C+D+G ; DISPLAYS ”3”
           DB B1+C+F+G ; DISPLAYS ”4”
           DB A+C+D+F+G ; DISPLAYS ”5”
           DB A+C+D+E+F+G ; DISPLAYS ”6”
           DB A+B1+C ; DISPLAYS ”7”
           DB A+B1+C+D+E+F+G ; DISPLAYS ”8”
           DB A+B1+C+D+F+G ; DISPLAYS ”9”
           DB A+E+F+G;DISPLAYS” F”
           DB  A+B1+F+G;DISPLAYS” O”

; RESISTANCE VALUES 65-99 DEGREES F. VALUES = K OHMS X1000 - TO 3 DECIMAL
;PLACES
;*******************************************************************************
            EVEN ; FOLLOWING SECTION MUST BE EVENLY ALIGNED
RESIS_TAB
            DW 12953 ;65 F
            DW 12666
            DW 12378
            DW 11858
            DW 11393
            DW 11161
            DW 10929
            DW 10697
            DW 10464 ;75 F
            DW 10232
```

```
            DW 10000
            DW 9625
            DW 9438 ;80 F
            DW 9250
            DW 9063
            DW 8875
            DW 8688
            DW 8500 ;85 F
            DW 8313
            DW 8161
            DW 8008
            DW 7856
            DW 7703 ;90 F
            DW 7551
            DW 7398
            DW 7246
            DW 7093
            DW 6941 ;95 F
            DW 6817
            DW 6694
            DW 6570
            DW 6446 ;99 F
;*******************************************************************************
; INTERRUPT VECTORS
;*******************************************************************************
        EVEN ; FOLLOWING SECTION MUST BE EVENLY ALIGNED
        COMMON INTVEC
        DW RESET ; PORT0, BIT 2 TO BIT 7
        DW BTINT ; BASIC TIMER
        DW RESET ; NO SOURCE
        DW RESET ; NO SOURCE
        DW RESET ; NO SOURCE
        DW TPINT ; TIMER PORT
        DW RESET ; NO SOURCE
        DW RESET ; NO SOURCE
        DW RESET ; NO SOURCE
        DW RESET ; NO SOURCE
        DW RESET ; WATCHDOG/TIMER, TIMER MODE
        DW RESET ; NO SOURCE
        DW RESET ; ADDRESS OF UART HANDLER
        DW RESET ; P0.0
        DW RESET ; NMI, OSC. FAULT
        DW RESET ; POR, EXT. RESET, WATCHDOG
        END
```

RESULT

The program was successfully compiled and linked. The resulting 452 bytes of executable code was downloaded into RAM memory of MSP430 device using C-SPY. The program was successfully debugged and run after being downloaded into the RAM memory of the MSP430 device and the corresponding temperature was displayed on the LCD screen. (eg- temperature of 72° F was displayed for a sensor resistance of 10KΩ).

Fig 5.4

A simulation of the negative temperature coefficient (NTC) thermistor was made by using a variable resistance (pot of resistance range 0-20KΩ).

Some problems faced:

1. The MEMORY utility of C-SPY can be used to view the RAM and the EPROM memory. The MEMORY utility of C-SPY can be used to modify the RAM; the EPROM cannot be modified using the MEMORY utility. The EPROM memory can only be programmed using the PRGS (after the EPROM is erased).

2. Direct assembler programs will not function correctly on the actual device because the Watchdog mechanism is active. The programs need to be modified to disable the Watchdog mechanism. The Watchdog mechanism is disabled with the C statement: "WDTCTL = 0x5a80;", or "mov #5a80h, &WDTCTL" in assembler.

3. GO OUT is not available while debugging assembler files. GO OUT operates like GO while debugging assembler files.

4**.** The following cryptic error message is output by the linker when a C.xcl file is incorrectly used in an assembler project:

Error[e46]: Undefined external "main" referred in CSTARTUP

Warning[w52]: More than one definition for the byte at address 0xfffe in common segment INTVEC. It is defined in module "CSTARTUP" as well as in module "..."

The solution to this problem is to use the correct A.xcl file (for assembler).

5. The IAR tutorial assumes full version of workbench, while it is not.

CONCLUSION

The Timer Port is a very versatile module that is available on MSP430x3xx microcontrollers. It is capable of supporting a wide variety of resistive sensor and reference resistor combinations. Components can be directly connected to the Timer Port to form complete sensor systems with a minimum of hardware interfacing. The combination of the Timer Port module, the 16-bit CPU, and the ultra low power design provide unmatched MIPS per watt performance.

An extension of the design can be used to incorporate additional control features into the design; thereby the set-up can be used as a low power thermostat.

REFERENCES

1. Glasford Glenn, Analog Electronic Circuit, PHI,1st edition, A-D converters.pp-439-464
2. Millman Jacob & Halkias Christos, Integrated Electronics, TMH ,1st edition , Analog & digital Circuits & systems, pp-665-667
3. Gaonkar Ramesh, Microprocessor, Architecture, Programming and application with the 8085,PRI Private limited, 5th edition,pp-633-636.
4. http://en.wikipedia.org/wiki/Thermistor
5. www.ti.com/sc/msp430
6. MSP430 Family Architecture Guide and Module Library User's Guide, SLAUE10B, Texas Instruments Incorporated, 1996.
7. MSP430 Family Metering Applications Report, SLAAE10C, 1998
8. http://www.iar.com/p89661/p89661_eng.php#ev
9. MSP430 EVK CD ROM, Literature,Application notes, slaa038
10. MSP430 EVK CD ROM, Literature, Data sheets, MSP439X3XX, slas227a
11. MSP430 EVK CD ROM, Literature, User's Guide ,Kickstart Environment, EVK_IAR_Users Guide
12. MSP430 EVK CD ROM, Literature, User's Guide, Kickstart Environment, a430