IMPLEMENTATION OF A MSP430-BASED ULTRASONIC DISTANCE MEASUREMENT MODULE

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

Bachelor of Technology

in

Electrical Engineering

By

N.SIVA PRASAD & SURESH KUMAR

Department of Electrical Engineering

National Institute of Technology

Rourkela

2007

IMPLEMENTATION OF A MSP430-BASED ULTRASONIC DISTANCE MEASUREMENT MODULE

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

Bachelor of Technology
in
Electrical Engineering

By
N.SIVA PRASAD & SURESH KUMAR

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 Ultrasonic Distance Measurement Module "** submitted by **N.Siva Prasad, Roll No: 10302058 and Suresh Kumar** , **Roll No: 10302065** 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:

N.Siva Prasad
Roll No: 10302058
National Institute of Technology
Rourkela

Suresh Kumar
Roll No: 10302065
National Institute of Technology
Rourkela

CONTENTS

ABSTRACT

This application report describes a distance-measuring system based on ultrasonic sound utilizing the MSP430F413 ultralow-power microcontroller. The system transmits a burst of ultrasonic sound waves towards the subject and then receives the corresponding echo. The MSP430 integrated analog comparator Comparator_A is used to detect the arrival of the echo to the system. The time taken for the ultrasonic burst to travel the distance from the system to the subject and back to the system is accurately measured by the MSP430.

Assuming the speed of sound in air at room temperature to be 1100 ft/s, the MSP430 computes the distance between the system and the subject and displays it using a two-digit static LCD driven by its integrated LCD driver. The distance is displayed in inches with an accuracy of ±1 inch. The minimum distance that this system can measure is eight inches and is limited by the transmitter's transducer settling-time. The maximum distance that can be measured is ninety-nine inches. The amplitude of the echo depends on the reflecting material, shape, and size. Sound-absorbing targets such as carpets and reflecting surfaces less than two square feet in area reflect poorly. The maximum measurable range is lower for such subjects. If the amplitude of the echo received by the system is so low that it is not detectable by the Comparator_A, the system goes out of range. This is indicated by displaying the error message E.

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 INTRODUCTION

1.2 BACKGROUND

1.3 OBJECTIVE

1.1 INTRODUCTION

This application report describes a distance-measuring system based on ultrasonic sound utilizing the MSP430F413 ultralow-power microcontroller. The system transmits a burst of ultrasonic sound waves towards the subject and then receives the corresponding echo. The MSP430 integrated analog comparator Comparator_A is used to detect the arrival of the echo to the system. The time taken for the ultrasonic burst to travel the distance from the system to the subject and back to the system is accurately measured by the MSP430. 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 the MSP430 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.

Architecture of MSP430E337A

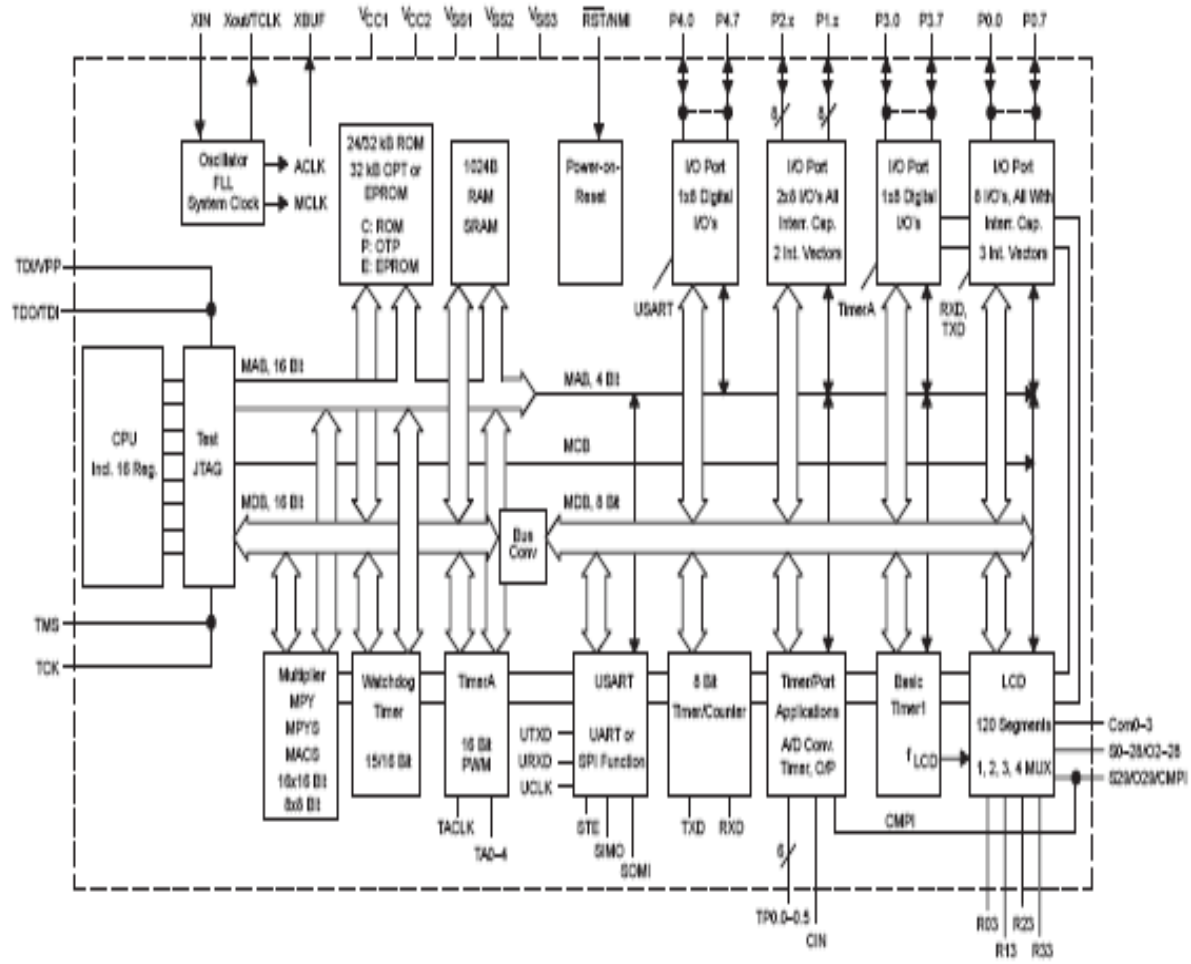


Fig 1.1

1.3 OBJECTIVE

To devise a method to incorporate ultra low power consumption in ultra sonic distance measurement modules using MSP430 microcontroller and ultrasonic transducers (transmitter and receiver) with the help of hex inverter and opamp (amplifier). The report describes a program to optimize power consumption by forcing the microcontroller to several sleep states during the operation.

Chapter 2

OVERVIEW OF "MSP430E337A" MICROCONTROLLERS

2.1 SALIENT FEATURES

2.2 PIN DIAGRAM & TERMINAL FUNCTIONS

2.3 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 MSP430E337AHFD

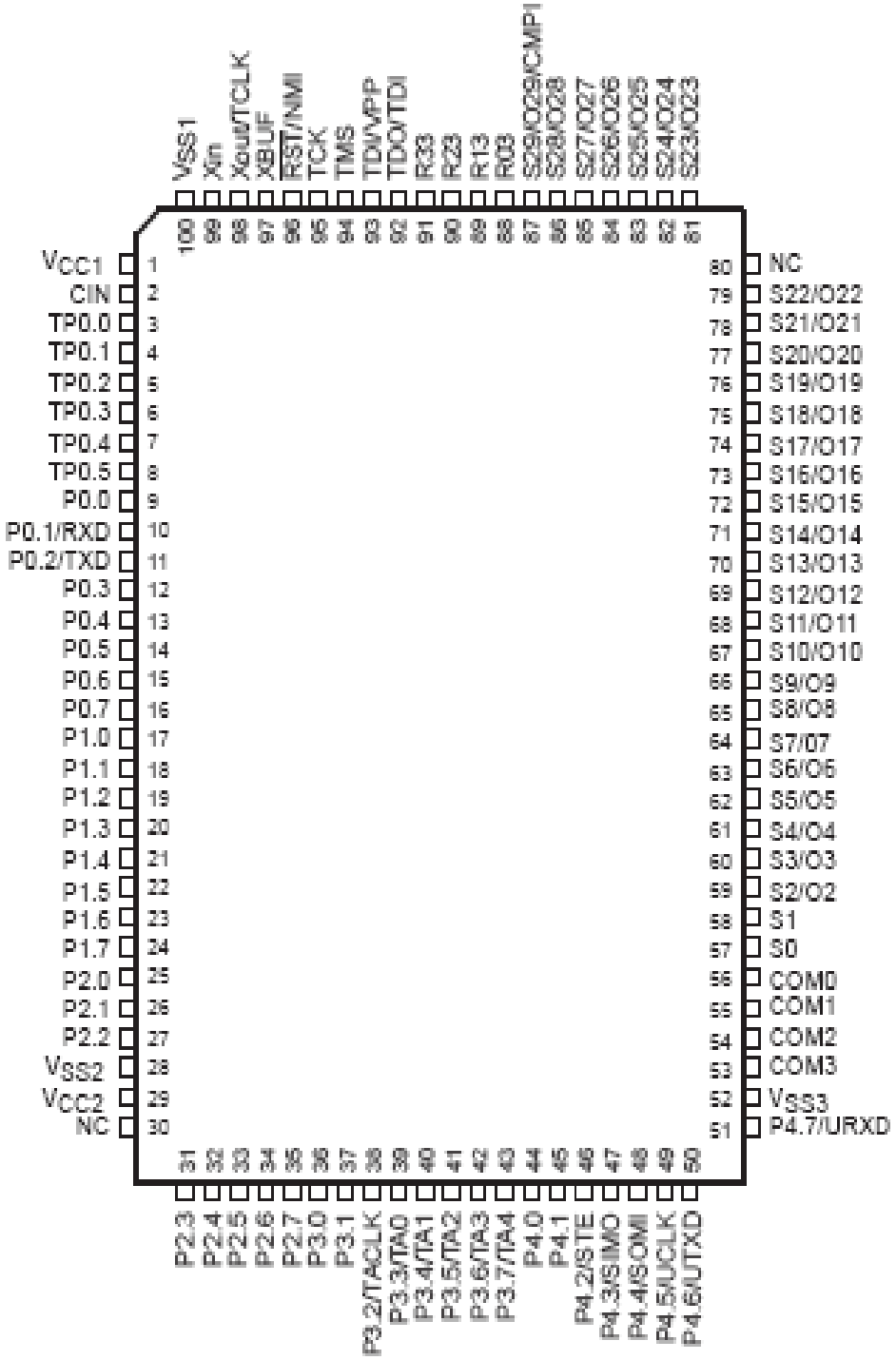


Fig 2.1

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.

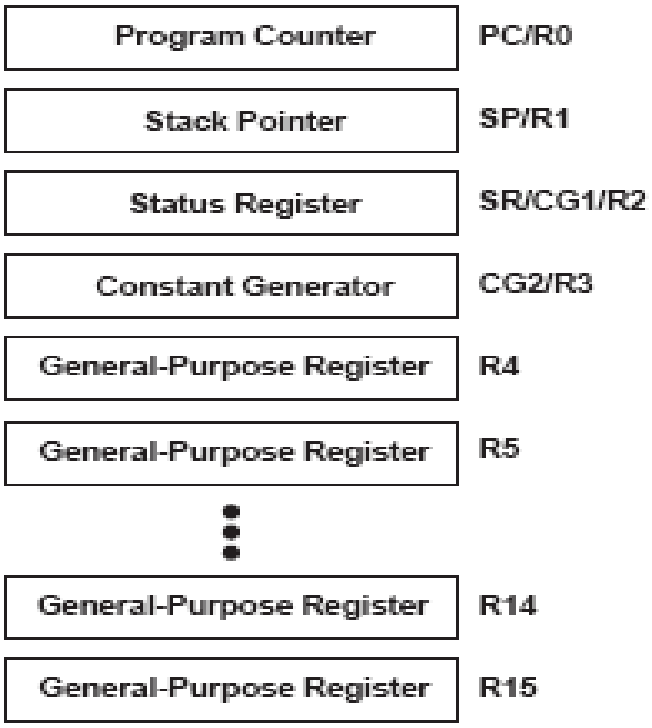


Fig 2.2

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.

Status Register

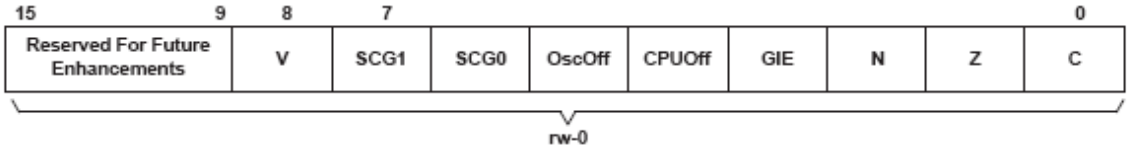


Fig 2.3

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:

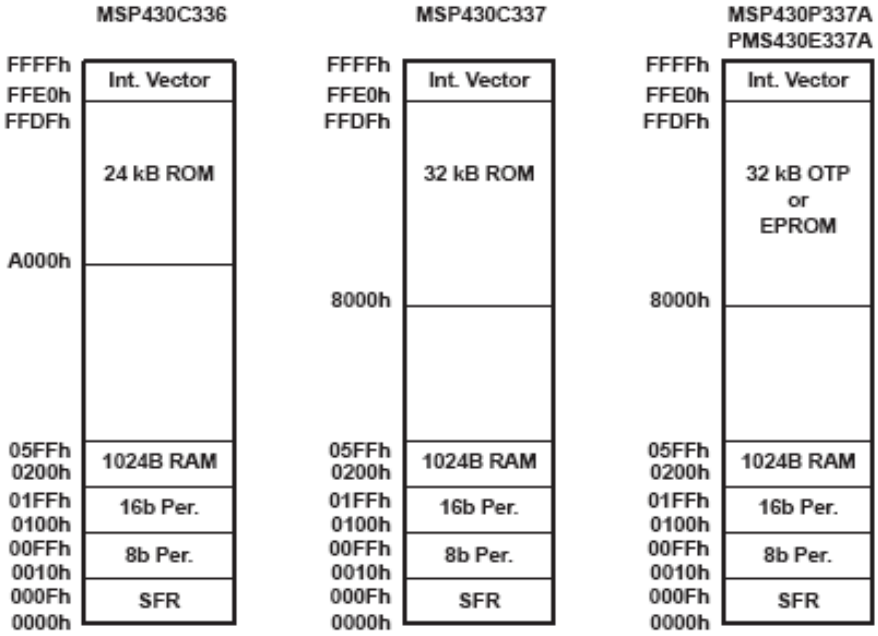


Fig 2.4

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

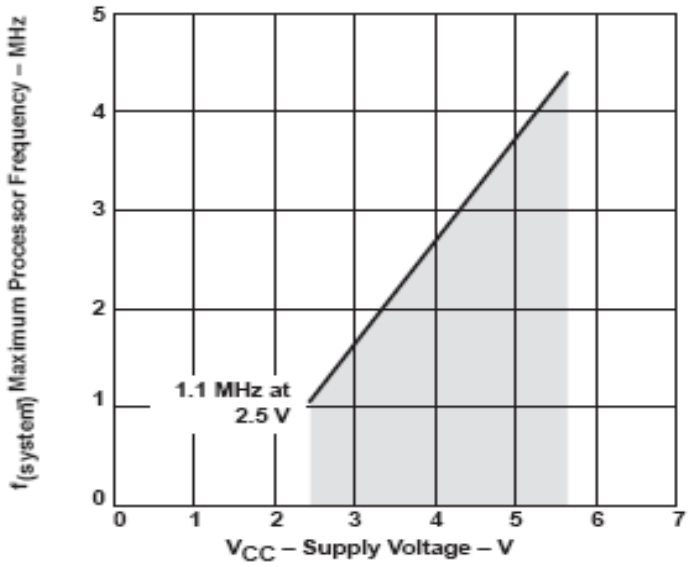


Fig 2.5

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

3.1 MSP- EVK4430S330 SCEMATIC

3.2 MSP-EVK430S330 PART PLACEMENT

3.3 DEVELOPMENT FLOW

3.4 PROJECT SETTINGS

3.5 OTHER INFORMATIONS

3.1 MSP-EVK430S330 SCHEMATIC:

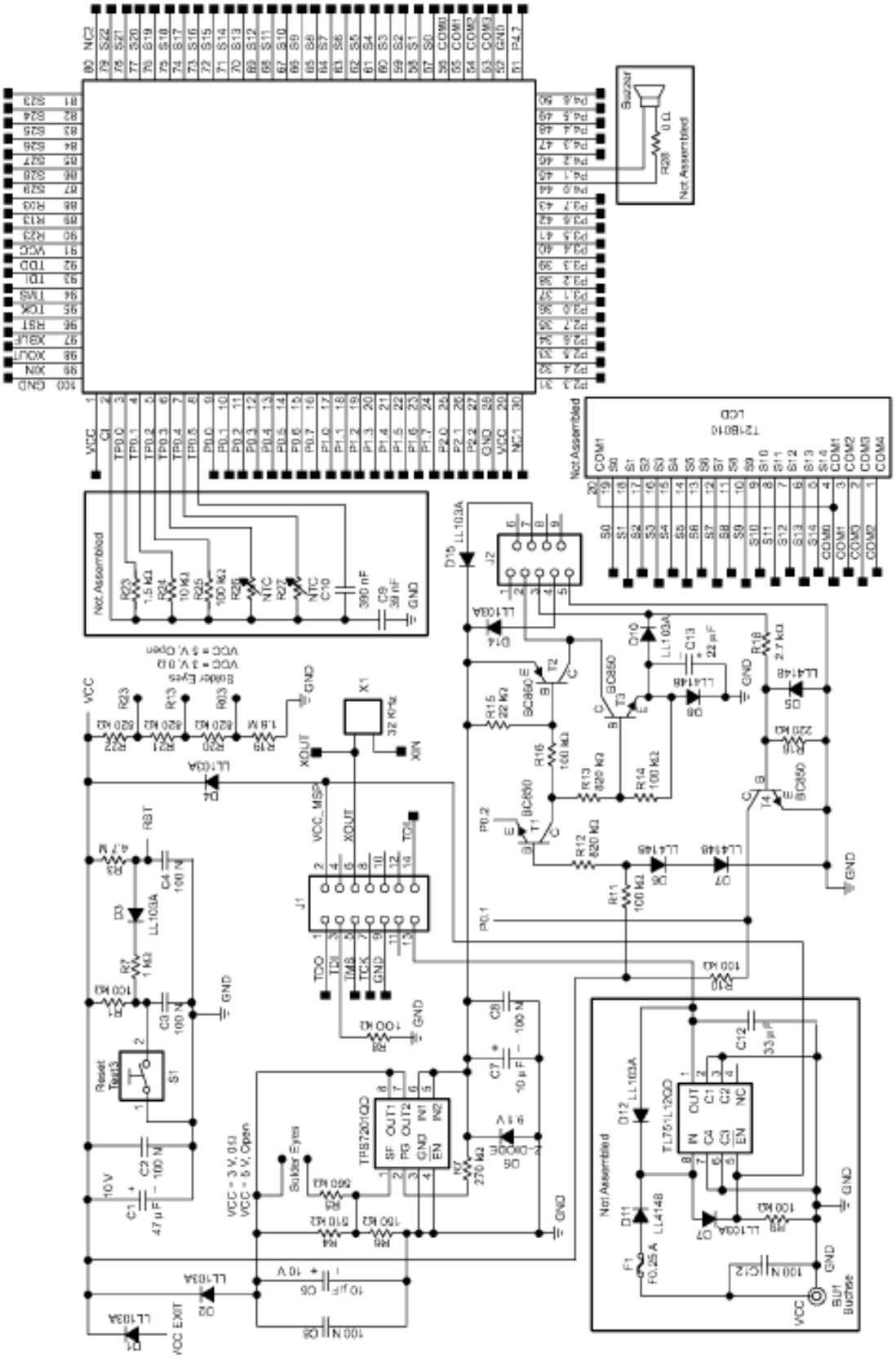


Fig 3.1

3.2 MSP-EVK430S330 PART PLACEMENT:

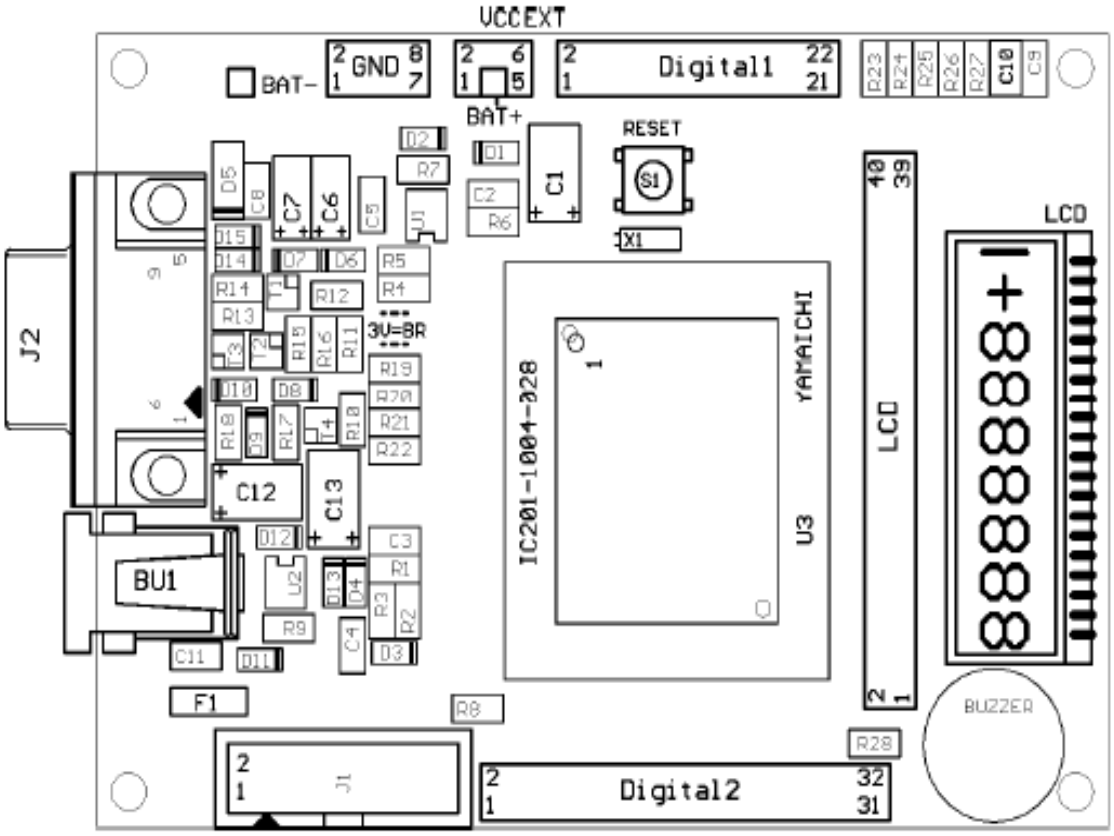


Fig 3.2

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

4.1 LCD CONNECTIONS

4.2 LCD CONTROLLER/DRIVER FEATURES

4.3 LOOK UP TABLE

4.1 LCD CONNECTIONS

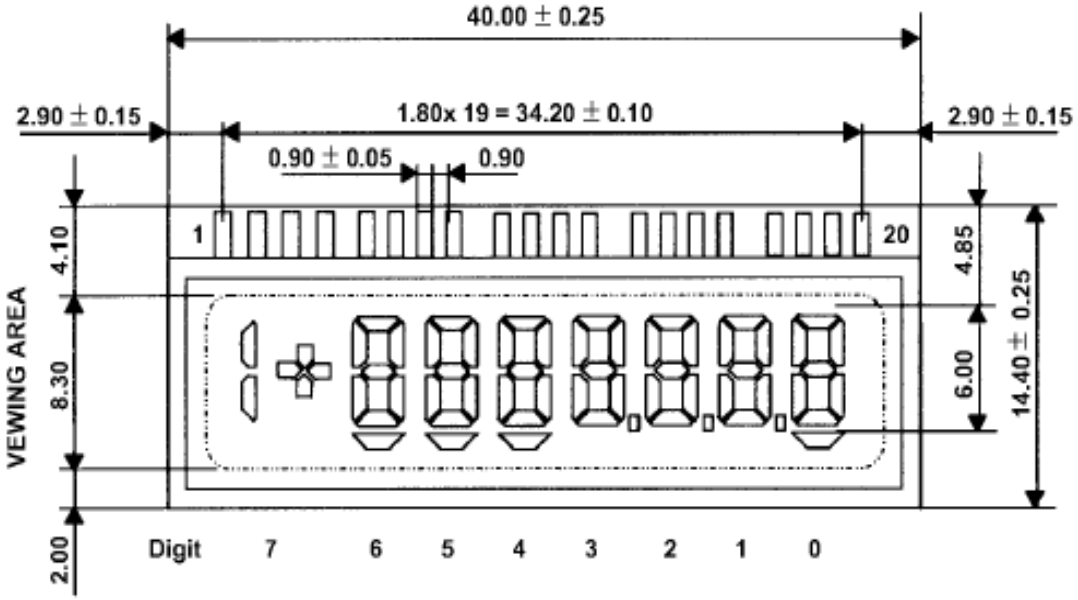
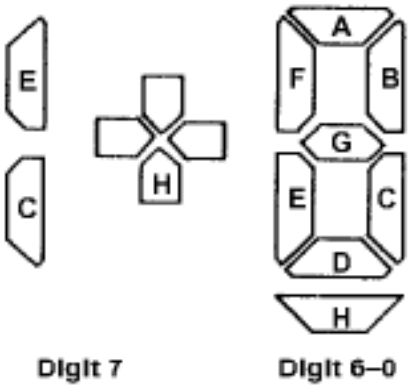


Fig 4.1

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 × 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.

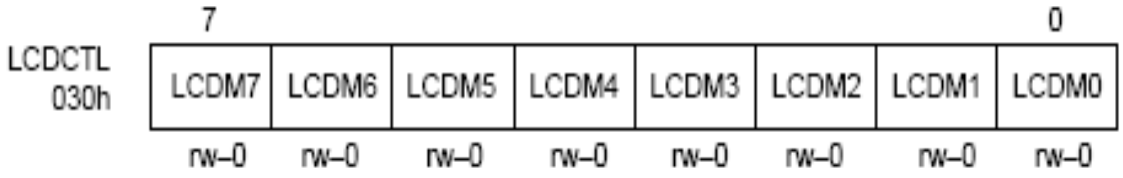


Fig 4.2

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 MSP430 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.

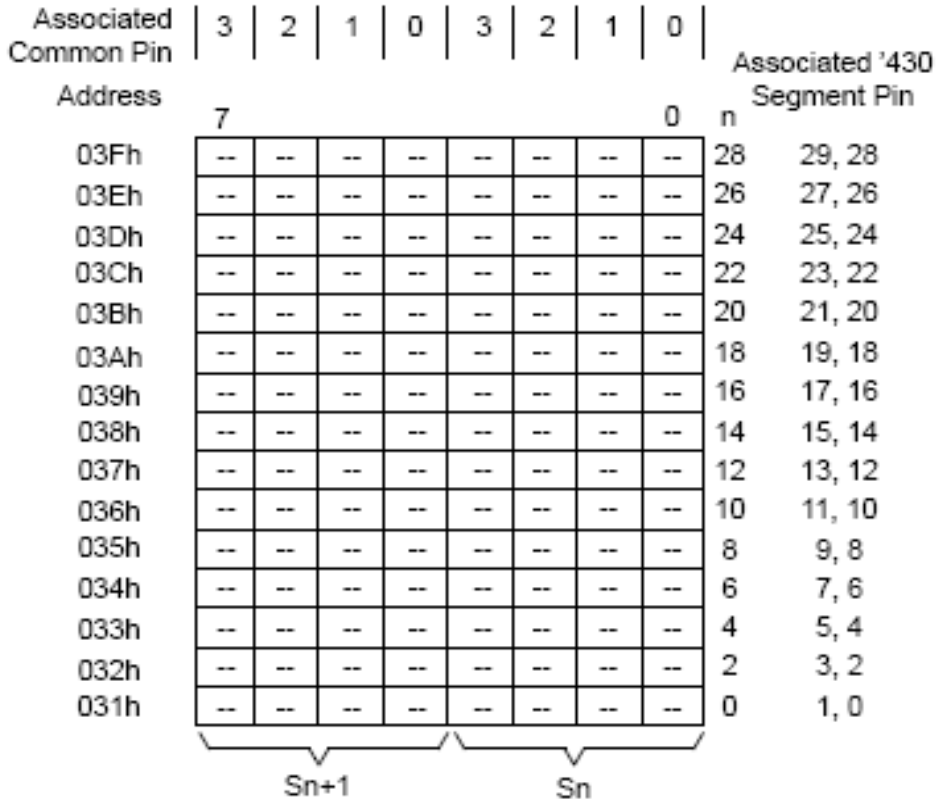


Fig 4.3

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"
DB      a+b+e+f+g               ; displays "P"
DB      0                       ; displays "Q"
DB      e+g                     ; displays "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 ULTRASONIC DISTANCE MEASUREMENT MODULE

5.1 THEORY OF OPERATION

5.2 HARDWARE IMPLEMENTATION

5.3 ULTRASONIC SOFTWARE DESCRIPTION

5.4 ASSEMBLYLANGUAGE PROGRAM

5.1 THEORY OF OPERATION

This application is based upon the reflection of sound waves. Sound waves are defined as longitudinal pressure waves in the medium in which they are traveling. Subjects whose dimensions are larger than the wavelength of the impinging sound waves reflect them. The reflected waves are called the echo. If the speed of sound in the medium is known and the time taken for the sound waves to travel the distance from the source to the subject and back to the source is measured, the distance from the source to the subject can be computed accurately. This is the measurement principle of this application. Here the medium for the sound waves is air, and the sound waves used are ultrasonic, since it is inaudible to humans.

Assuming that the speed of sound in air is 1100 feet/second at room temperature and that the measured time taken for the sound waves to travel the distance from the source to the subject and back to the source is *t* seconds, the distance *d* is computed by the formula $d=1100 \times 12 \times t$ inches. Since the sound waves travel twice the distance between the source and the subject, the actual distance between the source and the subject will be $d/2$.

5.2 HARDWARE IMPLEMENTATION

The devices used to transmit and receive the ultrasonic sound waves in this application are 40-kHz ceramic ultrasonic transducers. The MSP430 drives the transmitter transducer with a 12-cycle burst of 40-kHz square-wave signal derived from the crystal oscillator, and the receiver transducer receives the echo. The Timer_A in the MSP430 is configured to count the 40-kHz crystal frequency such that the time measurement resolution is 25 µs, which is more than adequate for this application. The measurement time base is very stable as it is derived from a Quartz-crystal oscillator. The echo received by the receiver transducer is amplified by an operational amplifier and the amplified output is fed to the Comparator_A input. The Comparator_A senses the presence of the echo signal at its input and triggers a capture of Timer_A count value to capture compare register CCR1. The capture is done exactly at the instant the echo arrives at the system. The captured count is the measure of the time taken for the ultrasonic burst to travel the distance from the system to the subject and back to the system. The distance in inches from the system to the subject is computed by the MSP430 using this measured time and displayed on a two-digit static LCD. Immediately after updating the display, the MSP430 goes to LPM3 sleep mode to save power.

The Basic Timer1 is programmed to interrupt the MSP430 every 205 milliseconds. The interrupt signal from the Basic Timer1 wakes up the MSP430 to repeat the measurement cycle and update the display.

Figure 5.3 shows the circuit schematic diagram of this application. The MSP430F413 (U1) is the core of this system. LCD1 is a two-digit low-voltage static LCD driven by the integrated LCD driver. R03 is connected to VSS, and R13 and R23 are left open for static-LCD-drive mode operation of the LCD peripheral. A 40-kHz crystal X1 is conveniently chosen for the low-frequency crystal oscillator to match the resonant frequency of the ultrasonic transducers used in this application. R12 serves as the pullup resistor for the reset line, and the integrated brownout-protection circuit takes care of brownout conditions. C9 provides power-supply decoupling to the MSP430 and is located close to the power supply lines of the device. A 14-pin box header (J1) allows JTAG interface to the MSP430 to provide in-circuit debugging and programming using the MSP430 flash emulation tool. LED1 is provided to indicate measurement cycles. Port pin P1.5 is configured to output the buffered 40-kHz square-wave ACLK required by the ultrasonic transmitter.

The output drive circuit for the transducer is powered directly from the 9-V battery and provides 18 VPP drive to the ultrasonic transmitter. The 18 VPP is achieved by a bridge configuration with hex inverter gates U4-CD4049. One inverter gate is used to provide a 180-degrees phase-shifted signal to one arm of the driver. The other arm is driven by the in-phase signal. This configuration doubles the voltage swing at the output and provides the required 18 VPP to the transmitter transducer. Two gates are connected in parallel so that each arm can provide adequate current drive to the transducer. Capacitors C6 and C7 block the dc to the transducer. Since the CD4049 operates on 9-V and the MSP430 operates on a VCC of 3.6 V, there is a logic level mismatch between the MSP430 and the output driver circuit.

Bipolar transistor Q1 acts as a logic-level shifter between these two logic levels. Operational amplifier U3 is the five-pin high-slew-rate TI operational amplifier TLV2771. This amplifier has a high-gain bandwidth and provides sufficiently high gain at 40 kHz. The operational amplifier is connected in an inverting amplifier configuration. R7 and R5 set the gain to 55 and C5 provides high-frequency roll off. R3 and R4 bias the non inverting input to a virtual mid rail for single-supply operation of the operational amplifier. The amplified ultrasonic signal swings above and below this virtual mid rail. The high Q of transducer RX1 provides selectivity and rejection of unwanted frequencies other than 40 kHz. The output of the operational amplifier is connected to the Comparator_A CA0 input of the MSP430 via port pin P1.6. The Comparator_A reference is internally selected to be 0.5VCC. When no ultrasonic echo is received, the voltage level at CA0 is slightly lower than the reference at CA1. When an echo is received, the voltage level increases above the reference and toggles the Comparator_A output CAOUT. R3 can be fine-tuned for the required sensitivity and the measurable range can be optimized.

The MSP430 and the ultrasonic signal amplifier circuit are powered by a regulated 3.6-V supply derived from the 9-V battery via TI LDO TPS77001. Resistors R1 and R2 program the regulator output voltage to 3.6 V. C1 and C2 are the recommended supply capacitors for correct functioning of the regulator. The transmitter driver is powered directly from the 9-V battery. Switch S1 functions as the power on switch for this application.

CIRCUIT SCHEMATIC

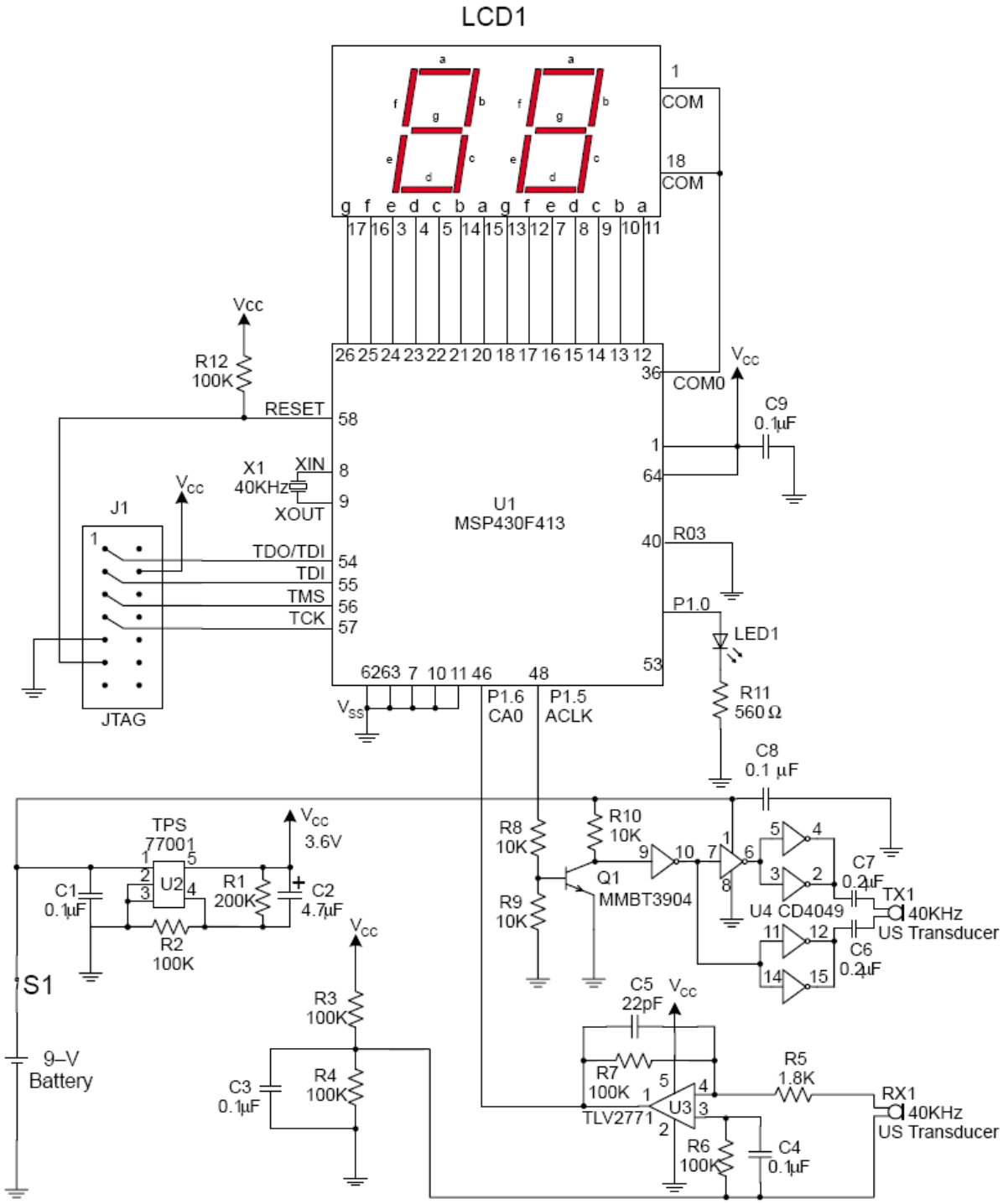


Fig 5.1

5.3 ULTRASONIC SOFTWARE DESCRIPTION

5.3.1 INITIALIZE DEVICE

This subroutine initializes and configures the peripherals used. The Watchdog Timer is disabled first. A software delay is provided to allow the low-frequency oscillator to stabilize. The FLL+ multiplier is set to 64 to produce an MCLK frequency of 2.56 MHz. P1.0 is configured as an output for the LED. The unused port pins are configured as outputs and port pin P1.5 is configured to output the 40-kHz buffered ACLK frequency. The Basic Timer1 is enabled and configured to provide a 150-Hz LCD frequency and to interrupt the CPU every 205 milliseconds to initiate a measurement cycle. The Comparator_A is configured with 0.5VCC internal reference and the CAPD bits are set to disable the input buffers for the comparator-input pins. The LCD module is turned on and configured for static-mode operation to drive the two-digit static LCD in the application. The LCD memory locations are cleared so that the initial LCD display is 00. The Basic Timer1 interrupt and the global interrupt enable are then enabled to allow the Basic Timer1 to periodically interrupt the CPU.

5.3.2 MAIN LOOP

Main loop updates the LCD with the value stored in the DIGITS buffer and then puts the MSP430 to LPM3 sleep mode. The MSP430 remains in sleep mode until a Basic Timer1 interrupt occurs and BT_ISR returns it to active mode. Now a measurement cycle is initiated. Timer_A is configured to 16-bit up mode and ACLK is selected as the clock source for Timer_A. CCR1 is set to the compare mode with a value of 12 so as to output a burst of 12 cycles of 40 kHz on P1.5. A 36-ACLK cycles delay follows to allow the output transducer to settle. This is realized by setting CCR1 to the compare mode with a value of 36. The MSP430 stays in LPM0 during these CCR1-compare wait states.

Now the system is set to receive the echo via the receiver transducer. The Comparator_A is configured to wait for the echo and it provides a capture interrupt at the instant the echo arrives. The Timer_A count is captured in capture-compare register CCR1. This value is the measure of the time it took the ultrasonic burst to travel the distance from the transmitter transducer to the subject and back to the receiver transducer. The count value is adjusted by adding 48 to compensate for the time lost in the 12-cycle burst and the 36-cycle transducer settling time delay. The adjusted value in CCR1 represents the exact time interval from the instant of the start

of the burst to the instant of the start of the echo at the system. Next, the math subroutine is called to compute the actual distance in inches and return the result. If the system is out of range, the echo signal is not received and the Comparator_A does not provide a capture interrupt. The MSP430 stays in LPM0 until the next Basic Timer1 interrupt wakes it up. The CAIFG bit in the CCTL1 control register is then tested to make sure that the echo was never received. To indicate this condition, a value of *0xBE* is stored in DIGITS to display an *E* on the LCD. The program finally loops back to Main loop to update the LCD and go back to LPM3 sleep mode. The next Basic Timer1 interrupt returns the MSP430 to active mode to repeat the program execution sequence.

5.3.3 MATHEMATICAL CALCULATIONS

The Math_Calc subroutine takes care of the mathematical calculations required by this application. The adjusted 16-bit value from CCR1 is stored in the variable *Result*. This value is the representation of the time it takes the ultrasonic burst to travel the distance from the system to the subject and back to the system. Since Timer_A counts in 25 µs steps, the equivalent value in time will be *Result X 25* µ*S*. Assuming the speed of sound as 1100 ft/s at room temperature, the *Result* from the Timer_A count works out to be six counts per inch of distance. Therefore, dividing the *Result* by six produces the required value of the distance in inches. To achieve the required precision with the available integer math of the MSP430, the 16-bit *Result* is first multiplied by 100 before dividing it by 6. This 16X16-bit multiplication is done by the subroutine Mul100. The 32-bit result is stored in the variables htX100_msw and htX100_lsw. This 32-bit result is then divided by 6 and the result is stored in the variable DIGITS. The value in DIGITS is in hexadecimal format. The hex2bcd subroutine converts this hexadecimal value to binary coded decimal (BCD) value, and the last two digits of the BCD number are discarded to compensate for the multiplication by 100 done earlier. The resulting two-digit value is returned to the variable DIGITS.

5.3.4 BT_ISR

The Basic Timer1 interrupt subroutine BT_ISR manipulates the bits in the status register SR residing in the stack such that the MSP430 returns to active mode on return from this ISR. This allows the MSP430 to continue to execute the code following the LPM3 instruction in Main loop.

5.3.5 DISPLAY

This subroutine updates the two-digit static LCD with the value in the variable DIGITS. The segment data for the static display is stored in look-up table LCD_Tab. The LCD memory is loaded with the required segment data by correlating the numbers in DIGITS and indexing to the required location in the LCD_Tab look-up table.

5.3.6 DELAY

This subroutine adds a 16-bit software delay. No registers are affected as the variable to be counted down by software is assigned to the top of stack (TOS). After the delay is timed out, the stack pointer (SP) is incremented back to the original value before returning from this subroutine.

5.4 ASSEMBLY LANGUAGE PROGRAM

```
#include “msp430x41x.h” ;              Standard Equations
;*******************************************************************************
; MSP430F413 Ultrasonic Distance Measurement Demonstration Program
;*******************************************************************************
; Register definitions
;*******************************************************************************
#define DIGITS        R11
#define Result        R10
#define IRBT          R9
#define IROP1         R4
#define IROP2L        R5
#define IROP2M        R6
#define IRACL         R7
#define IRACM         R8
;*******************************************************************************
; Variables definition
;*******************************************************************************
RSEG UDATA0
htX100_msw: DS 2 ;            word variable stored in RAM 200h & 201h
htX100_lsw: DS 2 ;            202h & 203h
;*******************************************************************************
RSEG CSTACK ;                 Directive to begin stack segment
DS 0
RSEG CODE ;                   Directive to begin code segment
RESET mov.w #SFE(CSTACK),SP ;         Define stack pointer
call #Init_Device ;           Initialize device
mov.w #0,DIGITS ;             Initialize DIGITS to ’0’
Mainloop
bic.b #CAON,&CACTL1 ;         Comparator_A OFF
call #Display ;               Display Data on LCD
bis.w #LPM3,SR ;              Wait in LPM3
```

```
;*******************************************************************************
; Start Ultrasonic Bursts and Take Measurements
;*******************************************************************************
clr.w &CCTL1 ;              Disable CCTL1
clr.w &TACTL ;              Disable timer_A
bis.b #BIT0,&P1OUT ;        LED ON
SetupTimerA mov.w #TASSEL0+TACLR+MC1,&TACTL; TACLK = ACLK, 16 bit up mode
bis.b #BIT5,&P1SEL ;        ACLK o/p on P1.5
mov.w #12,&CCR1 ;           12 cycle 40KHz burst
mov.w #CCIE,&CCTL1 ;        Compare mode,interrupt
bis.w #LPM0,SR ;            Wait for CCR1 interrupt
bic.b #BIT5,&P1SEL ;        ACLK o/p on P1.5 OFF
TimerCLR bis.w #TACLR,&TACTL
mov.w #36,&CCR1 ;           Delay for transducer to settle
mov.w #CCIE,&CCTL1 ;        Compare mode,interrupt
bis.w #LPM0,SR ;            Wait for CCR1 interrupt
bis.b #CAON,&CACTL1 ;       Comparator_A ON
bic.b #CAIFG,&CACTL1 ;      Enable Comparator_A interrupt flag
mov.w #CM0+CCIS0+SCS+CAP+CCIE,&CCTL1;          Pos edge, CCIB,Cap,interrupt
push &TAR ;                 TOS = TAR at Start of measurement
bis.w #LPM0,SR ;            Wait for CCR1 interrupt (Echo)
clr.w &CCTL1 ;              Disable CCTL1
bic.b #BIT0,&P1OUT ;        LED OFF
bit.b #CAIFG,&CACTL1 ;      Check for Echo not received
jz Next ;                   “out of range” condition
mov.w &CCR1,Result ;        Result = TAR (CCR1) at EOC
sub.w @SP+,Result ;         Result = time taken
add.w #48,Result ;          compensate 12Clks for the burst transmission time + 36Clks delay
;*******************************************************************************
; Measurement Done
;*******************************************************************************
call #Math_calc ;           Call Math subroutine
swpb DIGITS ;               Shift left by two digits for /100
jmp Mainloop ;              Next measurement cycle
```

```
Next mov.w #0beh,DIGITS ;          No echo received display 'E' error
jmp Mainloop
;*******************************************************************************
Init_Device ;                      Initialize MSP430x41x
;*******************************************************************************
mov.w #WDTPW+WDTHOLD,&WDTCTL ;     Stop WDT
bis.b #030h,&FLL_CTL0 ;    Turn on internal load capacitors for the XTAL to start oscillation
call #Delay ;                      Delay for oscillator to stabilize
mov.b #03fh,&SCFQCTL ;             MCLK = 40KhzX64 = 2.56Mhz
call #Delay ;                      Delay for FLL to stabilize
SetupP1 mov.b #000h,&P1OUT ;       Clear P1 output register
bis.b #0bfh,&P1DIR ;               Unused pins as o/p's
bis.b #040h,&P1SEL ;               Comp_A + i/p function
SetupP2 mov.b #000h,&P2OUT ;       Clear P2 output register
bis.b #0ffh,&P2DIR ;               Unused pins as o/p's
SetupP6 mov.b #000h,&P6OUT ;       Clear P6 output register
bis.b #0ffh,&P6DIR ;               Unused pins as o/p's
SetupBT mov.b #BTFRFQ0+BTFRFQ1+BTIP2+BTDIV,&BTCTL
                                   ;       Enable BT with 150Hz LCD freq.
                                   ;       and 205 millisecond interrupt
SetupCA mov.b #CAPD6,&CAPD ;  o/p buffer disable for comp i/p
mov.b #P2CA0,&CACTL2 ;             P1.6 to Comp + input
mov.b #CARSEL+CAREF1+CAON,&CACTL1 ;          Comp_A ON, 0.5Vcc int. reference
SetupLCD bis.b #LCDON+LCDSON+LCDSG0_7,LCDCTL
                                   ;       LCD module ON and in static mode
ClearLCD mov #15,R15 ;             15 LCD mem locations to clear
mov.b #LCDMEM,R14
Clear1 mov.b #0,0(R14) ;           Write zeros in LCD RAM locations
inc.b R14
dec R15 ;                          All LCD mem clear?
jnz Clear1 ;                       More LCD mem to clear go
bis.b #BTIE,&IE2 ;                 Enable Basic Timer interrupt
eint ;                             Enable interrupts
ret
```

```
;*******************************************************************************
BT_ISR ;                    Basic Timer ISR, CPU returns to active mode on RETI
;*******************************************************************************
bic #LPM3,0(SP) ;           Clear LPM3 bits on TOS
reti ;                      On return from interrupt
;*******************************************************************************
TAX_ISR;                    Common ISR for CCR1–4 and overflow
;*******************************************************************************
add.w &TAIV,PC ;            Add TA interrupt offset to PC
reti ;                      CCR0 – no source
jmp CCR1_ISR ;              CCR1
reti ;                      CCR2
reti ;                      CCR3
reti ;                      CCR4
TA_over reti ;              Timer_A overflow
CCR1_ISR bic.w #CCIFG,&CCTL1
bic.w #LPM0,0(SP) ;         Exit LPM0 on reti
reti ;
;*******************************************************************************
Display ;                   Subroutine to Display values DIGIT1 & DIGIT2
        ;                   CPU Registers used R15, R14, R13 and R12, not saved
;*******************************************************************************
mov.w #LCDM1,R15 ;          R15 points to first LCD location
mov.b DIGITS,R14 ;          LSD value moved to R14
OutLCD mov.b R14,R13 ;      Copy value in R14 to R13
rra.b R13 ;                 Right Shift
rra.b R13 ;                 four times to
rra.b R13 ;                 swap
rra.b R13 ;                 nibbles
and.b #0Fh,R14 ;            low nibble now in R14
and.b #0Fh,R13 ;            high nibble now in R13
mov.b LCD_Tab(R14),R12 ;    Low nibble to LCD digit 1
mov.b R12,0(R15) ;          Low nibble segments a & b to LCD
rra.w R12
```

```
inc.b R15
mov.b R12,0(R15) ;          Low nibble segments c & d to LCD
rra.w R12
inc.b R15
mov.b R12,0(R15) ;          Low nibble segments e & f to LCD
rra.w R12
inc.b R15
mov.b R12,0(R15) ;          Low nibble segments g & h to LCD
rra.w R12
inc.b R15
mov.b LCD_Tab(R13),R12 ; High nibble to LCD digit 2
mov.b R12,0(R15) ;          High nibble segments a & b to LCD
rra.w R12
inc.b R15
mov.b R12,0(R15) ;          High nibble segments c & d to LCD
rra.w R12
inc.b R15
mov.b R12,0(R15) ;          High nibble segments e & f to LCD
rra.w R12
inc.b R15
mov.b R12,0(R15) ;          High nibble segments g & h to LCD
rra.w R12
ret
;*****************************************************************************
; LCD Type Definition
;*****************************************************************************
;Segments definition
a       equ             001h
b       equ             010h
c       equ             002h
d       equ             020h
e       equ             004h
f       equ             040h
g       equ             008h
```

```
h       equ             080h
Blank   equ             000h
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      a+b+c+e+f+g ;       Displays ”A”
        db      Blank ;             Displays Blank
        db      a+d+e+f ;           Displays ”C”
        db      b+c+d+e+g ;         Displays ”D”
        db      a+d+e+f+g ;         Displays ”E”
        db      a+e+f+g ;           Displays ”F”
;*****************************************************************************
Delay;                      Software delay
;*****************************************************************************
push #0FFFFh ;              Delay to TOS
DL1 dec.w 0(SP) ;           Decrement TOS
jnz DL1 ;                   Delay over?
incd SP ;                   Clean TOS
ret ;                       Return from subroutine
;*****************************************************************************
Math_calc;                  Calculation Subroutine
;*****************************************************************************
mov.w #0h, DIGITS ;         Initialize DIGIT to 0
cmp.w #0h, Result ;         Check if Result count = 0
jeq calc_over ;             Exit if 0
call #Mul100 ;              Multiply Result count by 100
```

```
call #Divide ;                  Divide the result with #06d
call #Hex2bcd ;                 Convert 16bit binary to BCD number  Result xx.xx
calc_over ret ;                 Return from subroutine
;*****************************************************************************
Mul100 ;                        subroutine for multiplying Result with 100d
        ;                       inputs Result 16bit and constant 64h (100d) 16bit
        ;                       output 32bit htX100_msw & htX100_lsw
;*****************************************************************************
mov.w #100,IROP1 ;              Load IROP1 with 100 (multiplier)
mpyu clr.w htX100_lsw ;         Clear buffer for least Significant word
clr.w htX100_msw ;              Clear buffer for most Significant word
macu clr.w IROP2M ;             Clear multiplier high word
L$002 bit.w #1,IROP1 ;          Test actual bit
jz L$01 ;                       If 0: do nothing
add.w Result,htX100_lsw ;       If 1: Add multiplier to Result
addc.w IROP2M,htX100_msw ;
L$01 rla.w Result ;             Multiplier X 2
rlc.w IROP2M ;
rrc.w IROP1 ;                   Next bit to test
jnz L$002 ;                     If bit in carry : finished
ret
;*****************************************************************************
Divide ;        Subroutine for 32/16 bits division
        ;       inputs 32bit htX100_msw & htX100_lsw and #06 16bit, output DIGIT 16bit
;*****************************************************************************
clr.w DIGITS ;                  Clear buffer to hold new Result
mov.w #17,IRBT ;                Initialize loop counter
div1 cmp.w #06,htX100_msw ;     Compare divisor with dividend high word
jlo div2 ;                      If less : jump to div2
sub.w #06,htX100_msw ;          Subtract 6 from high word
div2 rlc.w DIGITS ;             Rotate result left through carry 1 bit
jc div4 ;                       If carry set: finished
dec.w IRBT ;                    Decrement bit counter
jz div3 ;                       If counter = 0 : finished
```

```
rla.w htX100_lsw ;          Dividend X 2
rlc.w htX100_msw ;
jnc div1 ;                  If carry not set jump to step div1
sub.w #06,htX100_msw ;      Subtract 6 from high word
setc ;                      Set carry
jmp div2 ;                  Jump to repeat
div3 clrc ;                 Clear carry
div4 ret ;                  Return from subroutine
;*******************************************************************************
Hex2bcd ;     Subroutine for converting 16bit hexadecimal value to BCD value
          ;     input in DIGITS 16bit hexadecimal, output in DIGITS 16bit BCD
;*******************************************************************************
mov #16,r9 ;                R9 no of bits
clr r8 ;                    Clear R8
clr r7 ;                    Clear R7
L$1 rla DIGITS ;            Rotate left arithmetic DIGITS
dadd r7,r7 ;                Add source and carry decimally
dadd r8,r8 ;                to destination
dec r9 ;                    Decrement bit counter
jnz L$1 ;                   Is 16 bits over ?
mov r7,DIGITS ;             Result in DIGITS
ret ;                       Return from subroutine
;*******************************************************************************
COMMON INTVEC ;         MSP430x41x Interrupt vectors
;*******************************************************************************
ORG BASICTIMER_VECTOR
BT_VEC DW BT_ISR ;      Basic Timer Vector
ORG TIMERA1_VECTOR ; Timer_AX Vector
TIMA_VEC DW TAX_ISR ;
ORG RESET_VECTOR
RESET_VEC DW RESET ;  POR, ext. Reset, Watchdog
;*******************************************************************************
END
;*******************************************************************************
```

RESULT

The program was successfully compiled and linked in 'IAR –KICKSTART WORKBENCH'. But at the time of downloading the program into the MSP430 device the C-SPY was not showing the Source Code and we were not sure whether the program downloaded into the RAM memory or not. Also, due to non-availability of ultra sonic transducers (transmitter and receiver) we could not simulate the module completely.

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 integrated analog Comparator_A, the 16-bit Timer_A with hardware capture/compare registers, the Basic Timer1, and the LCD driver peripherals simplify this ultrasonic distance measurement application design and provides a system-in-a-chip solution. The average current consumed by the application is 1.3 mA during a 15-inch distance measurement. This includes the quiescent current of LDO U2, operational amplifier U3, and CMOS hex inverter U4. The operational amplifier alone has a quiescent current of 1 mA and the remainder of the circuit current consumption is 300 µA. The LED draws 5 mA while it is on. The MSP430 draws an average current of 2.1 µA with the LCD continuously active. This is made possible by taking advantage of the ultralow-current features of the MSP430. The MSP430 sleeps in LPM3 most of the time and the CPU resources used by this application are only 5.6%.

Since the speed of sound is temperature dependent, the measured reading will be less accurate at temperatures other than room temperature. A simple thermistor-based temperature measurement and distance compensation could be employed in this application to allow the system to measure accurately over a wide range of temperatures. The measured distance and temperature data could also be stored in the flash memory if required. Adding additional receiver gain stages and using a multiplexed LCD to read out as many digits as required could increase the range.

REFERENCES

1. MSP430x41x Mixed Signal Microcontroller data sheet SLAS340
2. MSP430x4xx Family User's Guide, SLAU056
3. MSP430 Family Mixed-Signal Microcontrollers, application report SLAA024
4. TPS770xx Ultra Low-Power LDO Linear Regulators, data sheet SLVS210
5. TLV277x Family of High-Slew-Rate Operational Amplifiers, data sheet SLOS209
6. CD4049UB, CMOS Hex Inverting Buffer/Converter, data sheet SCHS046A