LAB 1:
INTRODUCTION TO THE COMPUTER SYSTEM

Overview

Starting in the year 2001, the EE757 lab will be illustrating the use of embedded microcontrollers. We will specifically use a microcontroller based on the powerful TMS320 DSP chip (the TMS320LF2407). Microcontrollers have become very powerful during the last few years. Indeed, the microcontroller board used in the lab (the eZdsp.LF2407) has the following features:

- TMS320LF2407TM Digital Signal Processor
- 64K words onboard program/data RAM
- 32K words on-chip Flash memory
- 7.3728 MHz CPU Clock
- 5-Volt only operation
- TI Code Composer tools driver
- 41 Digital I/O pins
- 10-bit 16 ADCs (0-3.3V range)

In this first day of the laboratory you will be introduced to the microcontroller board and some of the EE757 laboratory interface software and hardware. You will:

- Learn how to write programs in C using the Code Compose Version 4-12 editor and compiler.

- Learn how to access the digital ports.

- Read from the A/D and write to the D/A converters.

- Learn how to use the timers.

- Learn how to use interrupt routines.

- Use *Matlab.*

- Save data for post experiment plotting.

Preparation

1. Obtain and study a book on programming in C.

2. Read Appendix A, which describes the software environment.

3. You may not assume that any files left on the hard disk will remain there from week to week. You should either have a floppy disk to store all lab work between sessions or save it in your own EE account, accessible through the network.

4. Refer to your notes on *Matlab* from EE755. We do have Matlab on the PC's and will use this package at times.

5. Read the information in Appendix B on the peripherals.

6. Read task descriptions and become familiar with software listings provided.

Lab Description and Operation

There is no required lab write up for this task. You are to demonstrate the various results at each stage to the lab TA as you progress. The routines you generate here may come in handy later, so remember to save them.

This lab consists of the following series of very straight forward exercises.

Getting In and Out of Code Compose and Opening Projects

To start, turn on the computer. The operating system will be loaded by default.

We will begin by using Code Composer to enter and run the simple program shown in Figure 1. To get into the Code Composer environment click on CC. All your project files should be under

C:\tic2xx\myprojects

And should have the

.mak

extension.
In this first Task, you will generate a sinusoidal signal and save it into an m-file and then plot it using Matlab.

Follow the steps below and try to understand the steps well, as you will need them in future Tasks and Labs.

1. Connect the eZdsp™ to the PC parallel port using the cable provided.

2. Connect the power supply to the eZdsp™. Verify that the red power lights are on.

3. From the desktop, launch Code Composer. You should see the debugger window open.

4. Open the project "Program Listing1". From the pull-down menu bar, go to Project → Open

5. Browse to the project file directory:
   C:\tic2xx\myprojects\progrsm_listing1\listing1.mak
   and click Open.

6. Under the "Files" window, expand the project tree.

7. Build the project by going to the pull-down menu:
   Project → Rebuild All.
   The output window status should read:
   "Build complete, 0 Errors, 0 Warnings"
   when complete.

8. Download the generated machine code to the board. From the pull-down menu go to
   File → Load Program.
   Select "figure1.out" from the dialog box in the directory
   C:\tic2xx\myprojects\program_listing1\
   and click on "Open".

Now you are ready to run the compiled and linked code using Code Composer.

9. From the pull-down menu bar, go to
   Debug → Reset DSP
   and after that, go to
   Debug → Run.
   Due to the "while" loop in the C code, the program will be running continuously.

In order to practice other debugging tools, try to use watch windows and breakpoints as explained in Appendix A.

10. To stop the program, from the pull-down menu bar, go to

Debug → Halt.

If you want to restart the program, do not forget to Reset DSP before running the processor again.

11. To save data into a file, from the pull-down menu bar, go to

    File → Data → Save.

    Write data.m into the "File name" section of the dialog box and select *.dat(float) in "Save as type" section. Finally, click the "Save" button.

12. The new box appears after clicking "Save" button. The array name used in the C source file needs to be written into The "Address" section of this box to store data. In our case, it will be "Sin_Ar". The "Length" section should contain the number of words that is occupied by the Sin_Ar array in the memory. Because floating data type occupies 2 words in the memory of TMS320LF2407 DSP and Sin_Ar array is a float array with the size of 1000, it is obvious that 2000 should be entered into "Length" section. Clicking "OK" button causes Sin_Ar to be saved into the data.m file. The data in that file is then ready-to-use for further processing in Matlab.

13. To plot the saved data in Matlab, open the "data.m" file in Matlab editor. The first line in "data.m" is put by the Code Composer for file information. Therefore, that line should be deleted for plotting purpose. After deletion, write onto the top line of the file

    " data = ["

    and onto the bottom line of the line

    "]; ".

    The use of this addition is to convert the data stored in the file into a vector format. Finally, use

    "plot(A)"

    the m-file of Matlab to plot the data.

    In your output plot, you should see a sine wave.

```
/*****************************************************************
* Filename: listing1.c                                            *
*                                                                  *
*******************************************************************/
/*** Address Definitions ***/

#include        "f2407_c.h"     /* LF2407 register definitions */
#include              <math.h>
#define Pi 3.14
#define f 100

/****************************** MAIN ROUTINE **************************/
void main(void)
{
        float k = 0.0;
        unsigned int i = 0;
        float Sin_Ar[1000];

        *SCSR1 = 0x0005;
/*              0000 0000 1000 1101
 bit 15         0:      reserved
 bit 14         0:      CLKOUT = CPUCLK
 bit 13-12      00:     IDLE1 selected for low-power mode
 bit 11-9       000:    PLL x4 mode
 bit 8          0:      reserved
 bit 7          0:      1 = enable ADC module clock (0)
 bit 6          0:      1 = enable SCI module clock (0)
 bit 5          0:      1 = enable SPI module clock (0)
 bit 4          0:      1 = enable CAN module clock (0)
 bit 3          0:      1 = enable EVB module clock
 bit 2          1:      1 = enable EVA module clock
 bit 1          0:      reserved
 bit 0          1:      clear the ILLADR bit
*/
   *SCSR2 = (*SCSR2 | 0x000B) & 0x000F;
/*
 bit 15-6       0's:    reserved
 bit 5          0:      do NOT clear the WD OVERRIDE bit
 bit 4          0:      XMIF_HI-Z, 0=normal mode, 1=Hi-Z'd
 bit 3          1:      disable the boot ROM, enable the FLASH
 bit 2      no change   MP/MC* bit reflects state of MP/MC* pin
 bit 1-0       11:      11 = SARAM mapped to prog and data
*/
/*** Disable the watchdog timer ***/
    *WDCR  = 0x00E8;
/*
 bits 15-8      0's:     reserved
 bit 7          1:       clear WD flag
 bit 6          1:       disable the dog
 bit 5-3        101:     must be written as 101
 bit 2-0        000:     WDCLK divider = 1
*/
/*** Setup external memory interface for LF2407 ***/
    WSGR = 0x0040;
/*
 bit 15-11      0's:    reserved
 bit 10-9       00:     bus visibility off
 bit 8-6        001:    1 wait-state for I/O space
 bit 5-3        000:    0 wait-state for data space
 bit 2-0        000:    0 wait state for program space
*/
        for (i=0;i<1000;i++)
        {
                Sin_Ar[i] = sin(2*Pi*f*k);
                k+=0.1;
        }
        while(1)
        {}
}
```

Program Listing 1

Accessing the Digital I/O

Program Listing 2 demonstrates accessing the digital I/O ports.

The eZdsp™ board has six parallel ports. These ports can be accessed using their 16-bit data direction registers. The first 8 bits set the direction of the port pins and the remaining 8 bits are the voltage levels at the pins.

*PXDATDIR = 0x0FYF where X is one of A, B, C, D, E, F, and Y is don't care bits:
The C code above sets bits 0-3 as outputs and writes 1 (3.3V) to them and sets bits 4-7 as inputs.

Further information about accessing the digital I/O can be found in Program Listing 2.

To test your programs, use the logic probes to verify that the digital output lines actually assume the values you enter on the computer. Also, connect jumper wires from the digital output port to the digital input port in order to implement a loop-back configuration. You should then see whatever value you apply to the output port appearing at the input port.

```
/*******************************************************************
* Filename: listing2.c                                             *
*                                                                   *
*******************************************************************/
/*** Address Definitions ***/

#include        "f2407_c.h"     /* LF2407 register definitions */

/****************************** MAIN ROUTINE **************************/
void main(void)
{
   unsigned int input = 0, data = 0;
   /*** Configure the System Control and Status registers ***/
   *SCSR1 = 0x0005;

/*             0000 0000 1000 1101
 bit 15        0:      reserved
 bit 14        0:      CLKOUT = CPUCLK
 bit 13-12     00:     IDLE1 selected for low-power mode
 bit 11-9      000:    PLL x4 mode
 bit 8         0:      reserved
 bit 7         0:      1 = enable ADC module clock (0)
 bit 6         0:      1 = enable SCI module clock (0)
 bit 5         0:      1 = enable SPI module clock (0)
 bit 4         0:      1 = enable CAN module clock (0)
 bit 3         0:      1 = enable EVB module clock
 bit 2         1:      1 = enable EVA module clock
 bit 1         0:      reserved
 bit 0         1:      clear the ILLADR bit
*/

    *SCSR2 = (*SCSR2 | 0x000B) & 0x000F;
/*
 bit 15-6      0's:    reserved
 bit 5         0:      do NOT clear the WD OVERRIDE bit
 bit 4         0:      XMIF_HI-Z, 0=normal mode, 1=Hi-Z'd
 bit 3         1:      disable the boot ROM, enable the FLASH
 bit 2     no change   MP/MC* bit reflects state of MP/MC* pin
 bit 1-0      11:      11 = SARAM mapped to prog and data
*/


/*** Disable the watchdog timer ***/

    *WDCR  = 0x00E8;

/*
 bits 15-8     0's:     reserved
 bit 7         1:       clear WD flag
 bit 6         1:       disable the dog
 bit 5-3       101:     must be written as 101
 bit 2-0       000:     WDCLK divider = 1
*/


/*** Setup external memory interface for LF2407 EVM ***/

    WSGR = 0x0040;

/*
 bit 15-11     0's:    reserved
 bit 10-9      00:     bus visibility off
 bit 8-6       001:    1 wait-state for I/O space
 bit 5-3       000:    0 wait-state for data space
 bit 2-0       000:    0 wait state for program space
*/


/*** Setup shared I/O pins ***/

    *MCRA = 0x0000;                     /* group A pins */
```

```
/*            0000 0000 0000 0000
 bit 15        0:      0=IOPB7,     1=TCLKINA
 bit 14        0:      0=IOPB6,     1=TDIRA
 bit 13        0:      0=IOPB5,     1=T2PWM/T2CMP
 bit 12        0:      0=IOPB4,     1=T1PWM/T1CMP
 bit 11        0:      0=IOPB3,     1=PWM6
 bit 10        0:      0=IOPB2,     1=PWM5
 bit 9         0:      0=IOPB1,     1=PWM4
 bit 8         0:      0=IOPB0,     1=PWM3
 bit 7         0:      0=IOPA7,     1=PWM2
 bit 6         0:      0=IOPA6,     1=PWM1
 bit 5         0:      0=IOPA5,     1=CAP3
 bit 4         0:      0=IOPA4,     1=CAP2/QEP2
 bit 3         0:      0=IOPA3,     1=CAP1/QEP1
 bit 2         0:      0=IOPA2,     1=XINT1
 bit 1         0:      0=IOPA1,     1=SCIRXD
 bit 0         0:      0=IOPA0,     1=SCITXD
*/

    *MCRB = 0xFE00;                     /* group B pins */

/*
 bit 15        1:      0=reserved,  1=TMS2 (always write as 1)
 bit 14        1:      0=reserved,  1=TMS  (always write as 1)
 bit 13        1:      0=reserved,  1=TD0  (always write as 1)
 bit 12        1:      0=reserved,  1=TDI  (always write as 1)
 bit 11        1:      0=reserved,  1=TCK  (always write as 1)
 bit 10        1:      0=reserved,  1=EMU1 (always write as 1)
 bit 9         1:      0=reserved,  1=EMU0 (always write as 1)
 bit 8         0:      0=IOPD0,     1=XINT2/ADCSOC
 bit 7         0:      0=IOPC7,     1=CANRX
 bit 6         0:      0=IOPC6,     1=CANTX
 bit 5         0:      0=IOPC5,     1=SPISTE
 bit 4         0:      0=IOPC4,     1=SPICLK
 bit 3         0:      0=IOPC3,     1=SPISOMI
 bit 2         0:      0=IOPC2,     1=SPISIMO
 bit 1         0:      0=IOPC1,     1=BIO*
 bit 0         0:      0=IOPC0,     1=W/R*
*/

    *MCRC = 0x0000;                     /* group C pins */

/*          0000 0000 0000 1010
 bit 15        0:      reserved
 bit 14        0:      0=IOPF6,     1=IOPF6
 bit 13        0:      0=IOPF5,     1=TCLKINB
 bit 12        0:      0=IOPF4,     1=TDIRB
 bit 11        0:      0=IOPF3,     1=T4PWM/T4CMP
 bit 10        0:      0=IOPF2,     1=T3PWM/T3CMP
 bit 9         0:      0=IOPF1,     1=CAP6
 bit 8         0:      0=IOPF0,     1=CAP5/QEP4
 bit 7         0:      0=IOPE7,     1=CAP4/QEP3
 bit 6         0:      0=IOPE6,     1=PWM12
 bit 5         0:      0=IOPE5,     1=PWM11
 bit 4         0:      0=IOPE4,     1=PWM10
 bit 3         0:      0=IOPE3,     1=PWM9
 bit 2         0:      0=IOPE2,     1=PWM8
 bit 1         0:      0=IOPE1,     1=PWM7
 bit 0         0:      0=IOPE0,     1=CLKOUT
*/


/*** Configure IOPA0 pin as an output and the other IOPA's as input***/

    *PBDATDIR = 0xF0F0;
/*          1111 0000 1111 0000
 bit 15        1:      0=IOPB7 as input,  1=IOPB7 as output
 bit 14        1:      0=IOPB6 as input,  1=IOPB6 as output
 bit 13        1:      0=IOPB5 as input,  1=IOPB5 as output
 bit 12        1:      0=IOPB4 as input,  1=IOPB4 as output
```

```
 bit 11        0:      0=IOPB3 as input,  1=IOPB3 as output
 bit 10        0:      0=IOPB2 as input,  1=IOPB2 as output
 bit 9         0:      0=IOPB1 as input,  1=IOPB1 as output
 bit 8         0:      0=IOPB0 as input,  1=IOPB0 as output
 bit 7         1:      0=IOPB7 output low,1=IOPB7 output high
 bit 6         1:      0=IOPB6 output low,1=IOPB6 output high
 bit 5         1:      0=IOPB5 output low,1=IOPB5 output high
 bit 4         1:      0=IOPB4 output low,1=IOPB4 output high
 bit 3         0:      0=IOPB3 output low,1=IOPB3 output high
 bit 2         0:      0=IOPB2 output low,1=IOPB2 output high
 bit 1         0:      0=IOPB1 output low,1=IOPB1 output high
 bit 0         0:      0=IOPB0 output low,1=IOPB0 output high

Further information can be found in 240x Peripheral Guide Section 5
*/
 *PFDATDIR = 0x0000;
/*          0000 0000 0000 0000
 bit 15        x:      reserved
 bit 14        x:      reserved
 bit 13        0:      0=IOPF5 as input,  1=IOPF5 as output
 bit 12        0:      0=IOPF4 as input,  1=IOPF4 as output
 bit 11        0:      0=IOPF3 as input,  1=IOPF3 as output
 bit 10        0:      0=IOPF2 as input,  1=IOPF2 as output
 bit 9         0:      0=IOPF1 as input,  1=IOPF1 as output
 bit 8         0:      0=IOPF0 as input,  1=IOPF0 as output
 bit 7         x:      reserved
 bit 6         x:      reserved
 bit 5         0:      0=IOPF5 output low,1=IOPF5 output high
 bit 4         0:      0=IOPF4 output low,1=IOPF4 output high
 bit 3         0:      0=IOPF3 output low,1=IOPF3 output high
 bit 2         0:      0=IOPF2 output low,1=IOPF2 output high
 bit 1         0:      0=IOPF1 output low,1=IOPF1 output high
 bit 0         0:      0=IOPF0 output low,1=IOPF0 output high

Further information can be found in 240x Peripheral Guide Section 5
*/
 data = 0xFFF0;      /* change the y's at this line: data = 0xFFy0;
                                                y is the 4-bit number you will see at the
output  */

 /* Add a watch window and insert the variable "input"
       You can change the value of data using Edit->Edit Variable */

 /* Make sure that the following connections are done: PB7 -> PF4

          PB6 -> PF3

          PB5 -> PF2

          PB4 -> PF1  */

/*** Proceed with main routine ***/

    while(1)
    {
        *PBDATDIR &= data;
       input = *PFDATDIR & 0x001E;    /* reads the value at IOPF1-IOPF4 pins */
           input = input >>1;
           while(1)
           {}

     }


} /* end of main() */
```

Program Listing 2

Analog I/O

Now we want to use the Microcontroller A/D and D/A converters. The eZdsp board is basically a signal processing board, so to provide the full functionality expected from a Microcontroller, a D/A converter has been added.

The eZdsp™ board has 4 10-bit A/D converters in the range of 0-3.3V. However, there is no D/A converter on the eZdsp™ board. Therefore, an external D/A converter chip, DAC 714P, has been added to the motherboard. The driver program for the D/A converter called *D2A(int vdig)* is given in Program Listing 3.

In order to supply the desired voltage to the output of the D/A converter, the following code should be executed:

vdig = (int) (vout(32767.5/10.0));*
D2A(vdig)

Since the D/A converter is a bipolar IC with 16-bit resolution and ±10V output range, the desired output should be scaled by $\frac{2^{16}-1}{2} \cdot \frac{1}{10}$.

Furthermore, the routines used to operate the A/D converters are also given in Program Listing 3.

Connect the D/A output to the input of A/D Channel 0 input and run the program given in Program Listing 3.

For this channel, the input range is ±10V. Therefore, the output of D/A should be in this range.

Do the same operation for the A/D Channel 1 input. Make sure you don't apply negative voltages to the Channel 1, because for Channel 1, the input range is only 0-10V.

```
/*******************************************************************
* Filename: listing3.c                                             *
*                                                                   *
*******************************************************************/

/*** Address Definitions ***/

#include        "f2407_c.h"     /* LF2407 register definitions */
#include              "variables.h"

unsigned int timer2_per = 2293;
int vdig=0;
float vout=0.0;
float A_in0 = 0.0, A_in1 = 0.0;
void D2A(int vdig);

/****************************** MAIN ROUTINE ***************************/
void main(void)
{
   /*** Configure the System Control and Status registers ***/
   *SCSR1 = 0x0085;

/*             0000 0000 1000 1101
 bit 15        0:      reserved
 bit 14        0:      CLKOUT = CPUCLK
 bit 13-12     00:     IDLE1 selected for low-power mode
 bit 11-9      000:    PLL x4 mode
 bit 8         0:      reserved
 bit 7         1:      1 = enable ADC module clock (0)
 bit 6         0:      1 = enable SCI module clock (0)
 bit 5         0:      1 = enable SPI module clock (0)
 bit 4         0:      1 = enable CAN module clock (0)
 bit 3         1:      1 = enable EVB module clock
 bit 2         1:      1 = enable EVA module clock
 bit 1         0:      reserved
 bit 0         1:      clear the ILLADR bit
*/

    *SCSR2 = (*SCSR2 | 0x000B) & 0x000F;
/*
 bit 15-6      0's:    reserved
 bit 5         0:      do NOT clear the WD OVERRIDE bit
 bit 4         0:      XMIF_HI-Z, 0=normal mode, 1=Hi-Z'd
 bit 3         1:      disable the boot ROM, enable the FLASH
 bit 2     no change   MP/MC* bit reflects state of MP/MC* pin
 bit 1-0      11:      11 = SARAM mapped to prog and data
*/

/*** Disable the watchdog timer ***/

    *WDCR  = 0x00E8;

/*
 bits 15-8     0's:     reserved
 bit 7         1:       clear WD flag
 bit 6         1:       disable the dog
 bit 5-3       101:     must be written as 101
 bit 2-0       000:     WDCLK divider = 1
*/

/*** Setup external memory interface for LF2407 EVM ***/

    WSGR = 0x0040;
/*
 bit 15-11     0's:    reserved
 bit 10-9      00:     bus visibility off
 bit 8-6       001:    1 wait-state for I/O space
 bit 5-3       000:    0 wait-state for data space
 bit 2-0       000:    0 wait state for program space
*/
```

```
/*** Setup shared I/O pins ***/
    *MCRA = 0x0000;                     /* group A pins */
/*            0000 0000 0010 8000
 bit 15        0:      0=IOPB7,     1=TCLKINA
 bit 14        0:      0=IOPB6,     1=TDIRA
 bit 13        0:      0=IOPB5,     1=T2PWM/T2CMP
 bit 12        0:      0=IOPB4,     1=T1PWM/T1CMP
 bit 11        0:      0=IOPB3,     1=PWM6
 bit 10        0:      0=IOPB2,     1=PWM5
 bit 9         0:      0=IOPB1,     1=PWM4
 bit 8         0:      0=IOPB0,     1=PWM3
 bit 7         0:      0=IOPA7,     1=PWM2
 bit 6         1:      0=IOPA6,     1=PWM1
 bit 5         0:      0=IOPA5,     1=CAP3
 bit 4         0:      0=IOPA4,     1=CAP2/QEP2
 bit 3         1:      0=IOPA3,     1=CAP1/QEP1
 bit 2         0:      0=IOPA2,     1=XINT1
 bit 1         0:      0=IOPA1,     1=SCIRXD
 bit 0         0:      0=IOPA0,     1=SCITXD
*/

    *MCRB = 0xFE00;                     /* group B pins */
/*
 bit 15        1:      0=reserved,  1=TMS2 (always write as 1)
 bit 14        1:      0=reserved,  1=TMS  (always write as 1)
 bit 13        1:      0=reserved,  1=TD0  (always write as 1)
 bit 12        1:      0=reserved,  1=TDI  (always write as 1)
 bit 11        1:      0=reserved,  1=TCK  (always write as 1)
 bit 10        1:      0=reserved,  1=EMU1 (always write as 1)
 bit 9         1:      0=reserved,  1=EMU0 (always write as 1)
 bit 8         0:      0=IOPD0,     1=XINT2/ADCSOC
 bit 7         0:      0=IOPC7,     1=CANRX
 bit 6         0:      0=IOPC6,     1=CANTX
 bit 5         0:      0=IOPC5,     1=SPISTE
 bit 4         0:      0=IOPC4,     1=SPICLK
 bit 3         0:      0=IOPC3,     1=SPISOMI
 bit 2         0:      0=IOPC2,     1=SPISIMO
 bit 1         0:      0=IOPC1,     1=BIO*
 bit 0         0:      0=IOPC0,     1=W/R*
*/

    *MCRC = 0x0000;                     /* group C pins */
/*          0000 0000 0000 1010
 bit 15        0:      reserved
 bit 14        0:      0=IOPF6,     1=IOPF6
 bit 13        0:      0=IOPF5,     1=TCLKINB
 bit 12        0:      0=IOPF4,     1=TDIRB
 bit 11        0:      0=IOPF3,     1=T4PWM/T4CMP
 bit 10        0:      0=IOPF2,     1=T3PWM/T3CMP
 bit 9         0:      0=IOPF1,     1=CAP6
 bit 8         0:      0=IOPF0,     1=CAP5/QEP4
 bit 7         0:      0=IOPE7,     1=CAP4/QEP3
 bit 6         0:      0=IOPE6,     1=PWM12
 bit 5         0:      0=IOPE5,     1=PWM11
 bit 4         0:      0=IOPE4,     1=PWM10
 bit 3         0:      0=IOPE3,     1=PWM9
 bit 2         0:      0=IOPE2,     1=PWM8
 bit 1         0:      0=IOPE1,     1=PWM7
 bit 0         0:      0=IOPE0,     1=CLKOUT
*/

 *PCDATDIR = *PCDATDIR | 0xFF00;     /* C0,C1,C2,C4,C5 are used as outputs */
 /* PC0 ->      A0
        PC1     ->      A1
        PC2     ->      SDI
        PC4 ->  CLK
        PC5     ->      CLR */

/*** Configure Devices and Initialize Variables ***/
        timer_config();
```

```
/*** Setup the core interrupts ***/
    *IMR = 0x0000;                      /* clear the IMR register */
    *IFR = 0x003F;                      /* clear interrupt flags */
    *IMR = 0x0001;                      /* enable desired core interrupts */

       /* enable timer 1,2  */
    *T2CON = *T2CON | 0x0040 ;
    asm(" CLRC INTM");   /* enable global interrupts */

    /* Change the value of vout by going to Edit->Edit Variable
    /* Add watch window and insert the variable observer */
    vout = 4.75;

/*** Proceed with main routine ***/
    while(1)
    {
      vdig = (int) (vout*(32767.0/10.0));
      D2A(vdig);
    }


} /* end of main() */


void D2A(int vdig)
{
       int i=0;
    int data = 0;
       /* drives the D2A chip */
       data = vdig;
    *PCDATDIR |= (CLK | A0 | A1) ;
    *PCDATDIR |= CLR;

/* 16 bit data is sent to D2A chip serially */
    for(i=0;i<16;i++) {
       if ( (data & 0x8000) )
              *PCDATDIR |= SDI;
       else *PCDATDIR &=~SDI;
       data = data << 1;
       *PCDATDIR &= ~A0;
              *PCDATDIR &= ~CLK;
              *PCDATDIR |= CLK;
     }

     *PCDATDIR |=A0;
     *PCDATDIR &=~A1;
     *PCDATDIR &= ~CLK;
        *PCDATDIR |= CLK;
}

unsigned int res1 = 0, res2 = 0;
interrupt void ADC_isr(void)
{ /*A2D interrupt*/
       res1 = *RESULT0;
       res1 = res1>>6;
       res2 = *RESULT1;
       res2 = res2>>6;
       A_in0 = ((float)res1/1023.0)*3.34;
       A_in0 = 6.0*(A_in0-1.661);         /* The input range of A/D Channel0 is 0-3.3V.
To increase this range to -10V-+10V, external circuitry is designed. To go back to the
exact measurement, this operation is necessary */
       A_in1 = ((float)res2/1023.0)*3.34;
       A_in1 = 3.0*A_in1;                 /* The input range of A/D Channel1 is 0-3.3V.
To increase this range to 0-+10V, external circuitry is designed. To go back to the exact
measurement, this operation is necessary */

   /*clear flags*/
       *ADCTRL2 |= 0x0202;
}
```

Program Listing 3

Using Timed I/O

In this task a periodic interrupt is generated and in the interrupt service routine (ISR), the signal level of an I/O pin is toggled. Timer 2 is used to adjust the sampling period of the ISR. The clock frequency of the timer and the value of the timer period register determine the period of the ISR. The maximum value of the clock frequency of the timer is 29.4 MHz. This value can be divided into 2,4,8,16,32,64,128 using T2CON. Then, the desired interrupt period can be obtained by setting timer period register value. The Timer 2 periodic interrupt uses the int3 core interrupt. To enable it, a 1 is written to the corresponding bits of IMR, IFR and event manager registers.

In Program Listing 4, the period of ISR is set to 200 Hz. The clock frequency is divided into 128 (see paragraph above.) In the interrupt service routine, the parallel port A bit 1 is toggled. Therefore, if a scope is connected to the PA1, a 100 Hz square wave should be observed.

```
/*******************************************************************
* Filename: listing4.c                                             *
*                                                                   *
*******************************************************************/
/*** Address Definitions ***/

#include        "f2407_c.h"     /* LF2407 register definitions */

unsigned int timer2_per = 1148;
/****************************** MAIN ROUTINE **************************/
void main(void)
{
   /*** Configure the System Control and Status registers ***/
    *SCSR1 = 0x0005;
/*             0000 0000 1000 1101
 bit 15        0:      reserved
 bit 14        0:      CLKOUT = CPUCLK
 bit 13-12     00:     IDLE1 selected for low-power mode
 bit 11-9      000:    PLL x4 mode
 bit 8         0:      reserved
 bit 7         0:      1 = enable ADC module clock (0)
 bit 6         0:      1 = enable SCI module clock (0)
 bit 5         0:      1 = enable SPI module clock (0)
 bit 4         0:      1 = enable CAN module clock (0)
 bit 3         0:      1 = enable EVB module clock
 bit 2         1:      1 = enable EVA module clock
 bit 1         0:      reserved
 bit 0         1:      clear the ILLADR bit
*/
    *SCSR2 = (*SCSR2 | 0x000B) & 0x000F;
/*
 bit 15-6      0's:    reserved
 bit 5         0:      do NOT clear the WD OVERRIDE bit
 bit 4         0:      XMIF_HI-Z, 0=normal mode, 1=Hi-Z'd
 bit 3         1:      disable the boot ROM, enable the FLASH
 bit 2     no change   MP/MC* bit reflects state of MP/MC* pin
 bit 1-0      11:      11 = SARAM mapped to prog and data
*/
/*** Disable the watchdog timer ***/

    *WDCR  = 0x00E8;
/*
 bits 15-8     0's:     reserved
 bit 7         1:       clear WD flag
 bit 6         1:       disable the dog
 bit 5-3       101:     must be written as 101
 bit 2-0       000:     WDCLK divider = 1
*/

/*** Setup external memory interface for LF2407 EVM ***/

    WSGR = 0x0040;
/*
 bit 15-11     0's:    reserved
 bit 10-9      00:     bus visibility off
 bit 8-6       001:    1 wait-state for I/O space
 bit 5-3       000:    0 wait-state for data space
 bit 2-0       000:    0 wait state for program space
*/

/*** Setup shared I/O pins ***/

    *MCRA = 0x0048;                     /* group A pins */
/*            0000 0000 0010 8000
 bit 15        0:      0=IOPB7,     1=TCLKINA
 bit 14        0:      0=IOPB6,     1=TDIRA
 bit 13        0:      0=IOPB5,     1=T2PWM/T2CMP
 bit 12        0:      0=IOPB4,     1=T1PWM/T1CMP
 bit 11        0:      0=IOPB3,     1=PWM6
 bit 10        0:      0=IOPB2,     1=PWM5
 bit 9         0:      0=IOPB1,     1=PWM4
```

```
 bit 8         0:      0=IOPB0,     1=PWM3
 bit 7         0:      0=IOPA7,     1=PWM2
 bit 6         1:      0=IOPA6,     1=PWM1
 bit 5         0:      0=IOPA5,     1=CAP3
 bit 4         0:      0=IOPA4,     1=CAP2/QEP2
 bit 3         1:      0=IOPA3,     1=CAP1/QEP1
 bit 2         0:      0=IOPA2,     1=XINT1
 bit 1         0:      0=IOPA1,     1=SCIRXD
 bit 0         0:      0=IOPA0,     1=SCITXD
*/
    *MCRB = 0xFE00;                     /* group B pins */
/*
 bit 15        1:      0=reserved,  1=TMS2 (always write as 1)
 bit 14        1:      0=reserved,  1=TMS  (always write as 1)
 bit 13        1:      0=reserved,  1=TD0  (always write as 1)
 bit 12        1:      0=reserved,  1=TDI  (always write as 1)
 bit 11        1:      0=reserved,  1=TCK  (always write as 1)
 bit 10        1:      0=reserved,  1=EMU1 (always write as 1)
 bit 9         1:      0=reserved,  1=EMU0 (always write as 1)
 bit 8         0:      0=IOPD0,     1=XINT2/ADCSOC
 bit 7         0:      0=IOPC7,     1=CANRX
 bit 6         0:      0=IOPC6,     1=CANTX
 bit 5         0:      0=IOPC5,     1=SPISTE
 bit 4         0:      0=IOPC4,     1=SPICLK
 bit 3         0:      0=IOPC3,     1=SPISOMI
 bit 2         0:      0=IOPC2,     1=SPISIMO
 bit 1         0:      0=IOPC1,     1=BIO*
 bit 0         0:      0=IOPC0,     1=W/R*
*/
    *MCRC = 0x0000;                     /* group C pins */
/*          0000 0000 0000 1010
 bit 15        0:      reserved
 bit 14        0:      0=IOPF6,     1=IOPF6
 bit 13        0:      0=IOPF5,     1=TCLKINB
 bit 12        0:      0=IOPF4,     1=TDIRB
 bit 11        0:      0=IOPF3,     1=T4PWM/T4CMP
 bit 10        0:      0=IOPF2,     1=T3PWM/T3CMP
 bit 9         0:      0=IOPF1,     1=CAP6
 bit 8         0:      0=IOPF0,     1=CAP5/QEP4
 bit 7         0:      0=IOPE7,     1=CAP4/QEP3
 bit 6         0:      0=IOPE6,     1=PWM12
 bit 5         0:      0=IOPE5,     1=PWM11
 bit 4         0:      0=IOPE4,     1=PWM10
 bit 3         0:      0=IOPE3,     1=PWM9
 bit 2         0:      0=IOPE2,     1=PWM8
 bit 1         0:      0=IOPE1,     1=PWM7
 bit 0         0:      0=IOPE0,     1=CLKOUT
*/

/*** Configure IOPA0 pin as an output and the other IOPA's as input***/
    *PADATDIR = 0x0202;
/*          0000 0010 0000 0000
 bit 15        0:      0=IOPA7 as input,  1=IOPA7 as output
 bit 14        0:      0=IOPA6 as input,  1=IOPA6 as output
 bit 13        0:      0=IOPA5 as input,  1=IOPA5 as output
 bit 12        0:      0=IOPA4 as input,  1=IOPA4 as output
 bit 11        0:      0=IOPA3 as input,  1=IOPA3 as output
 bit 10        0:      0=IOPA2 as input,  1=IOPA2 as output
 bit 9         1:      0=IOPA1 as input,  1=IOPA1 as output
 bit 8         0:      0=IOPA0 as input,  1=IOPA0 as output
 bit 7         0:      0=IOPA7 output low,1=IOPA7 output high
 bit 6         0:      0=IOPA6 output low,1=IOPA6 output high
 bit 5         0:      0=IOPA5 output low,1=IOPA5 output high
 bit 4         0:      0=IOPA4 output low,1=IOPA4 output high
 bit 3         0:      0=IOPA3 output low,1=IOPA3 output high
 bit 2         0:      0=IOPA2 output low,1=IOPA2 output high
 bit 1         0:      0=IOPA1 output low,1=IOPA1 output high
 bit 0         0:      0=IOPA0 output low,1=IOPA0 output high
*/

/*** Configure Devices and Initialize Variables ***/
```

```
       timer_config();

/*** Setup the core interrupts ***/
    *IMR = 0x0000;                      /* clear the IMR register */
    *IFR = 0x003F;                      /* clear interrupt flags */
/*
 bits 15-6 Reserved.
 bit 5 INT6    0: No INT6 interrupt is pending 1: At least one INT6 interrupt is pending.
 Write a 1 to this bit to clear it to 0 and clear the interrupt request
 bit 4 INT5
 bit 3 INT4
 bit 2 INT3
 bit 1 INT2
 bit 0 INT1

Further information can be found in 240x Peripheral Guide Section 2
*/

    *IMR = 0x0004;                      /* enable desired core interrupts */
/*
 bits 15-6 Reserved.
 bit 5 INT6.   0: Level INT6 is masked 1: Level INT6 is unmasked
 bit 4 INT5.
 bit 3 INT4.
 bit 2 INT3.
 bit 1 INT2.
 bit 0 INT1.

Further information can be found in 240x Peripheral Guide Section 2
*/


/*** Setup the event manager interrupts ***/
    *EVAIMRA = 0x0000;                  /* enable desired EVA group A interrupts */
/*
 bits 15-11 Reserved.
 bit 10 T1OFINT (GP timer 1 overflow interrupt) ENABLE  0: Disable 1: Enable
 bit 9 T1UFINT (GP timer 1 underflow interrupt) ENABLE
 bit 8 T1CINT (GP timer 1 compare interrupt) ENABLE
 bit 7 T1PINT (GP timer 1 period interrupt) ENABLE
 bits 6-4 Reserved.
 bit 3 CMP3INT (Compare 3 interrupt) ENABLE
 bit 2 CMP2INT (Compare 2 interrupt) ENABLE
 bit 1 CMP1INT (Compare 1 interrupt) ENABLE
 bit 0 PDPINTA (Power drive protection interrupt) ENABLE.

Further information can be found in 240x Peripheral Guide Section 6
*/

    *EVAIMRB = 0x0001;                  /* enable desired EVA group B interrupt (periodic
interrupt)*/
/*
 bits 15-4 Reserved.
 bit 3 T2OFINT (GP timer 2 overflow interrupt) ENABLE   0: Disable 1: Enable
 bit 2 T2UFINT (GP timer 2 underflow interrupt) ENABLE
 bit 1 T2CINT (GP timer 2 compare interrupt) ENABLE
 bit 0 T2PINT (GP timer 2 period interrupt) ENABLE

Further information can be found in 240x Peripheral Guide Section 6
*/

    *EVAIFRB |= 0x0001;
/*
 bits 15-4 Reserved.
 bit 3   T2OFINT FLAG.
 Read:   0 Flag is reset
                1 Flag is set
 Write:  0 No effect
                1 Resets flag
 bit 2   T2UFINT FLAG.
 Read:   0 Flag is reset
```

```
                1 Flag is set
 Write:  0 No effect
                1 Resets flag
 bit 1   T2CINT FLAG.
 Read:   0 Flag is reset
                1 Flag is set
 Write:  0 No effect
                1 Resets flag
 bit 0   T2PINT FLAG.
 Read:   0 Flag is reset
                1 Flag is set
 Write:  0 No effect
                1 Resets flag

Further information can be found in 240x Peripheral Guide Section 6
*/

        /* enable timer2 */
        *T2CON = *T2CON | 0x0040 ;
    asm(" CLRC INTM");   /* enable global interrupts */

/*** Proceed with main routine ***/
    while(1)
    { }

} /* end of main() */

int flag = 0;
interrupt void timer2_isr(void)
{

        if(flag == 1)
        {
          *PADATDIR = *PADATDIR & 0xFFFD;  /* sets IOPA1 pin to 0 */
          flag = 0;
        }
        else
        {
          *PADATDIR = *PADATDIR | 0x0002; /* sets IOPA1 pin to 1 */
          flag = 1;
        }
        *EVAIFRB |= 0x0001;      /* clear T2PINT flag */
}
```

Program Listing 4

PWM and the Capture Operation

Program Listing 5 demonstrates the capture operation and generation of pulse width modulated (PWM) signals using TMS320LF2407 DSP.

In many applications it is necessary to detect the level change of the signals. The most straightforward way to do is polling the desired pins. However, the processor cannot do other operations during polling. Therefore, it is not an efficient method. TMS320LF2407 DSP offers a neat way to overcome this problem. The chip has six capture inputs. The processor can sense the level changes at these inputs and cause interrupts and/or do some desired tasks automatically, without stopping its current operation. The settings of the capture units are done by CAPCONA/B and CAPFIFOA/B. The settings are ready in Program Listing 5; you do not have to modify them. Detailed information about them is given in Appendix B.

TMS320LF2407 DSP has special circuitry to generate PWM signals. It uses a timer, timer count value, timer period and timer compare registers to generate PWM signal. It is strongly recommended that you refer to Appendix B to understand how the PWM is generated. In general the following steps should be performed to generate a PWM signal:

1. Determine the timer to be used (in program listing 5, timer 1 is used).

2. Determine the frequency of the PWM signal.

3. Set the clock frequency of the timer to a desired value using TXCON (As explained before Program Listing 4).

4. Calculate the desired timer period using clock frequency (from step 3) and frequency of the PWM signal (from step 2) . Write this value to TxPR.

5. Change the contents of COMCONA and ACTRA registers according to your needs.

6. Determine the desired duty cycle and write the result into CMPR register.

7. Enable the selected timer.

8. Attach the scope to the 3rd pin of port 111 (PWM1 of EzDSP board) to observe the PWM signal. Connect the PWM signal to the CAP1-QEP1 pin and observe the vale of the X_position variable on the Watch Window.

9. Run the program. Change the value of pwm_on to change the duty cycle.

The settings are preset in program listing 5; you do not have to modify them for this lab.

```
/*******************************************************************
* Filename: listing5.c                                             *
*                                                                   *
*******************************************************************/
/*** Address Definitions ***/

#include        "f2407_c.h"     /* LF2407 register definitions */

unsigned int timer1_per = 383;
unsigned int pwm_on = 100;
unsigned int x_position=0;
/****************************** MAIN ROUTINE **************************/
void main(void)
{
   /*** Configure the System Control and Status registers ***/
    *SCSR1 = 0x0005;
/*             0000 0000 1000 1101
 bit 15        0:      reserved
 bit 14        0:      CLKOUT = CPUCLK
 bit 13-12     00:     IDLE1 selected for low-power mode
 bit 11-9      000:    PLL x4 mode
 bit 8         0:      reserved
 bit 7         0:      1 = enable ADC module clock (0)
 bit 6         0:      1 = enable SCI module clock (0)
 bit 5         0:      1 = enable SPI module clock (0)
 bit 4         0:      1 = enable CAN module clock (0)
 bit 3         0:      1 = enable EVB module clock
 bit 2         1:      1 = enable EVA module clock
 bit 1         0:      reserved
 bit 0         1:      clear the ILLADR bit
*/
    *SCSR2 = (*SCSR2 | 0x000B) & 0x000F;
/*
 bit 15-6      0's:    reserved
 bit 5         0:      do NOT clear the WD OVERRIDE bit
 bit 4         0:      XMIF_HI-Z, 0=normal mode, 1=Hi-Z'd
 bit 3         1:      disable the boot ROM, enable the FLASH
 bit 2     no change   MP/MC* bit reflects state of MP/MC* pin
 bit 1-0      11:      11 = SARAM mapped to prog and data
*/
/*** Disable the watchdog timer ***/

    *WDCR  = 0x00E8;
/*
 bits 15-8     0's:     reserved
 bit 7         1:       clear WD flag
 bit 6         1:       disable the dog
 bit 5-3       101:     must be written as 101
 bit 2-0       000:     WDCLK divider = 1
*/

/*** Setup external memory interface for LF2407 EVM ***/

    WSGR = 0x0040;
/*
 bit 15-11     0's:    reserved
 bit 10-9      00:     bus visibility off
 bit 8-6       001:    1 wait-state for I/O space
 bit 5-3       000:    0 wait-state for data space
 bit 2-0       000:    0 wait state for program space
*/
/*** Setup shared I/O pins ***/

    *MCRA = 0x0048;                     /* group A pins */
/*            0000 0000 0100 1000
 bit 15        0:      0=IOPB7,     1=TCLKINA
 bit 14        0:      0=IOPB6,     1=TDIRA
 bit 13        0:      0=IOPB5,     1=T2PWM/T2CMP
 bit 12        0:      0=IOPB4,     1=T1PWM/T1CMP
 bit 11        0:      0=IOPB3,     1=PWM6
 bit 10        0:      0=IOPB2,     1=PWM5
```

```
 bit 9         0:      0=IOPB1,     1=PWM4
 bit 8         0:      0=IOPB0,     1=PWM3
 bit 7         0:      0=IOPA7,     1=PWM2
 bit 6         1:      0=IOPA6,     1=PWM1
 bit 5         0:      0=IOPA5,     1=CAP3
 bit 4         0:      0=IOPA4,     1=CAP2/QEP2
 bit 3         1:      0=IOPA3,     1=CAP1/QEP1
 bit 2         0:      0=IOPA2,     1=XINT1
 bit 1         0:      0=IOPA1,     1=SCIRXD
 bit 0         0:      0=IOPA0,     1=SCITXD
*/
    *MCRB = 0xFE00;                     /* group B pins */
/*
 bit 15        1:      0=reserved,  1=TMS2 (always write as 1)
 bit 14        1:      0=reserved,  1=TMS  (always write as 1)
 bit 13        1:      0=reserved,  1=TD0  (always write as 1)
 bit 12        1:      0=reserved,  1=TDI  (always write as 1)
 bit 11        1:      0=reserved,  1=TCK  (always write as 1)
 bit 10        1:      0=reserved,  1=EMU1 (always write as 1)
 bit 9         1:      0=reserved,  1=EMU0 (always write as 1)
 bit 8         0:      0=IOPD0,     1=XINT2/ADCSOC
 bit 7         0:      0=IOPC7,     1=CANRX
 bit 6         0:      0=IOPC6,     1=CANTX
 bit 5         0:      0=IOPC5,     1=SPISTE
 bit 4         0:      0=IOPC4,     1=SPICLK
 bit 3         0:      0=IOPC3,     1=SPISOMI
 bit 2         0:      0=IOPC2,     1=SPISIMO
 bit 1         0:      0=IOPC1,     1=BIO*
 bit 0         0:      0=IOPC0,     1=W/R*
*/
    *MCRC = 0x0000;                     /* group C pins */
/*          0000 0000 0000 1010
 bit 15        0:      reserved
 bit 14        0:      0=IOPF6,     1=IOPF6
 bit 13        0:      0=IOPF5,     1=TCLKINB
 bit 12        0:      0=IOPF4,     1=TDIRB
 bit 11        0:      0=IOPF3,     1=T4PWM/T4CMP
 bit 10        0:      0=IOPF2,     1=T3PWM/T3CMP
 bit 9         0:      0=IOPF1,     1=CAP6
 bit 8         0:      0=IOPF0,     1=CAP5/QEP4
 bit 7         0:      0=IOPE7,     1=CAP4/QEP3
 bit 6         0:      0=IOPE6,     1=PWM12
 bit 5         0:      0=IOPE5,     1=PWM11
 bit 4         0:      0=IOPE4,     1=PWM10
 bit 3         0:      0=IOPE3,     1=PWM9
 bit 2         0:      0=IOPE2,     1=PWM8
 bit 1         0:      0=IOPE1,     1=PWM7
 bit 0         0:      0=IOPE0,     1=CLKOUT
*/

/*** Configure Devices and Initialize Variables ***/
       timer_config();

/*** Setup the core interrupts ***/
    *IMR = 0x0000;                      /* clear the IMR register */
    *IFR = 0x003F;                      /* clear interrupt flags */
/*
 bits 15-6 Reserved.
 bit 5 INT6    0: No INT6 interrupt is pending 1: At least one INT6 interrupt is pending.
 Write a 1 to this bit to clear it to 0 and clear the interrupt request
 bit 4 INT5
 bit 3 INT4
 bit 2 INT3
 bit 1 INT2
 bit 0 INT1

Further information can be found in 240x Peripheral Guide Section 2
*/

    *IMR = 0x0008;                      /* enable desired core interrupts */
/*
```

```
 bits 15-6 Reserved.
 bit 5 INT6.   0: Level INT6 is masked 1: Level INT6 is unmasked
 bit 4 INT5.
 bit 3 INT4.
 bit 2 INT3.
 bit 1 INT2.
 bit 0 INT1.

Further information can be found in 240x Peripheral Guide Section 2
*/


/*** Setup the event manager interrupts ***/
    *EVAIMRA = 0x0000;                  /* enable desired EVA group A interrupts */
/*
 bits 15-11 Reserved.
 bit 10 T1OFINT (GP timer 1 overflow interrupt) ENABLE  0: Disable 1: Enable
 bit 9 T1UFINT (GP timer 1 underflow interrupt) ENABLE
 bit 8 T1CINT (GP timer 1 compare interrupt) ENABLE
 bit 7 T1PINT (GP timer 1 period interrupt) ENABLE
 bits 6-4 Reserved.
 bit 3 CMP3INT (Compare 3 interrupt) ENABLE
 bit 2 CMP2INT (Compare 2 interrupt) ENABLE
 bit 1 CMP1INT (Compare 1 interrupt) ENABLE
 bit 0 PDPINTA (Power drive protection interrupt) ENABLE.

Further information can be found in 240x Peripheral Guide Section 6
*/


    *EVAIMRC = 0x0001;                  /* enable desired EVA group C interrupts */
/*
 bits 15-3 Reserved.
 bit 2 CAP3INT (Capture 3 interrupt) ENABLE   0: Disable 1: Enable
 bit 1 CAP2INT (Capture 2 interrupt) ENABLE
 bit 0 CAP1INT (Capture 1 interrupt) ENABLE

Further information can be found in 240x Peripheral Guide Section 6
*/

  /* clear all EVA group interrupts */
    *EVAIFRA |= 0x0080;
/*
 bits 15-11 Reserved.
 bit 10  T1OFINT FLAG.
 Read:   0 Flag is reset
         1 Flag is set
 Write:  0 No effect
                1 Resets flag
 bit 9   T1UFINT FLAG.
 Read:   0 Flag is reset
                1 Flag is set
 Write:  0 No effect
                1 Resets flag
 bit 8   T1CINT FLAG.
 Read:   0 Flag is reset
                1 Flag is set
 Write:  0 No effect
                1 Resets flag
 bit 7   T1PINT FLAG.
 Read:   0 Flag is reset
         1 Flag is set
 Write:  0 No effect
            1 Resets flag
 bits 6-4 Reserved.
 bit 3   CMP3INT FLAG.
 Read:   0 Flag is reset
                1 Flag is set
 Write:  0 No effect
                1 Resets flag
 bit 2   CMP2INT FLAG.
```

```
 Read:   0 Flag is reset
                1 Flag is set
 Write:  0 No effect
                1 Resets flag
 bit 1   CMP1INT FLAG.
 Read:   0 Flag is reset
                1 Flag is set
 Write:  0 No effect
                1 Resets flag
 bit 0   PDPINTA FLAG.
 Read:   0 Flag is reset
                1 Flag is set
 Write:  0 No effect
                1 Resets flag

Further information can be found in 240x Peripheral Guide Section 6
*/


    *EVAIFRC |= 0x0001;
/*
 bits 15-3 Reserved.
 bit 2   CAP3INT FLAG.
 Read:   0 Flag is reset
                1 Flag is set
 Write:  0 No effect
                1 Resets flag
 bit 1   CAP2INT FLAG.
 Read:   0 Flag is reset
                1 Flag is set
 Write:  0 No effect
                1 Resets flag
 bit 0   CAP1INT FLAG.
 Read:   0 Flag is reset
                1 Flag is set
 Write:  0 No effect
                1 Resets flag

Further information can be found in 240x Peripheral Guide Section 6
*/

        /* enable timer 1,2  */
        *T1CON = *T1CON | 0x0040 ;
    asm(" CLRC INTM");   /* enable global interrupts */

/*** Proceed with main routine ***/
    while(1)
    { }

} /* end of main() */

interrupt void cap_isr(void){

        x_position++;
        *EVAIFRC |= 0x0001;      /* clear T2PINT flag */
}
```

Program Listing 5