CHAPTER 5

ANALOG INTERFACING:

Analog to Digital and Digital to Analog Conversions

3.1 Example Programs

This chapter discusses the ADC (Analog to Digital Converter) module of the TMS320F2407. The EVM has an on-board Digital to Analog converter module, which will be discussed here too.

Example Program 1:

Configure Channels 0-15 but only read channel 0 for the result. No interrupt involved.

The complete code is given below. All register settings are commented in details.

```
;*********************************************************************
; File Name:    ch5_e1.asm
; Target System: C240x Evaluation Board
; Description:   This program initializes the ADC module of the 2407
;           and does a conversion of all the analog input channels.
;           The results of the conversion are available in the
;           RESULTSn register, which can be accessed by the user
;           application. The ADC operates as one 16-state sequencer.
;           The conversions are stopped once the sequencer reaches
;           EOS (End of sequence). No interrupts. When conversion
;           sequence ends, the flag bit in ADCTRL2 is set. In this
;           program, only Channel 0 result is loaded into a memory
;           location "RESULT".
;*********************************************************************
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Global symbol declarations
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        .def START,PHANTOM,GISR1,GISR2,GISR3,GISR4,GISR5,GISR6

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Address definitions
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        .include f2407.h

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Uninitialized global variable definitions
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        .bss    GPR0,1      ;general purpose variable
       .bss   RESULT,1     ;for A/D conversion result
;=====================================================================
```

```
; M A I N   C O D E  - starts here
;=====================================================================
       .text
START:
       NOP
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Configure the System Control and Status Registers
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        LDP     #DP_PF1         ;set data page

        SPLK    #0000000011111101b, SCSR1
*                |||||||||||||||
*                FEDCBA9876543210
* 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
* bit 6         1:      1 = enable SCI module clock
* bit 5         1:      1 = enable SPI module clock
* bit 4         1:      1 = enable CAN module clock
* 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

        LACC    SCSR2                   ;ACC = SCSR2 register
        OR      #0000000000001011b      ;OR in bits to be set
        AND     #0000000000001111b      ;AND out bits to be cleared
*                |||||||||||||||
*                FEDCBA9876543210
* 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 the state of the MP/MC* pin
* bit 1-0      11:      11 = SARAM mapped to prog and data (default)

        SACL    SCSR2                   ;store to SCSR2 register

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Other setup
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Setup the core interrupts
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        LDP     #0h                     ;set data page
        SPLK    #0h,IMR                 ;clear the IMR register
        SPLK    #111111b,IFR           ;clear any pending core interrupts
        SPLK    #000000b,IMR            ;enable INT2 interrupts

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Setup the event manager interrupts
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        LDP     #DP_EVA                 ;set data page
        SPLK    #0FFFFh, EVAIFRA       ;clear all EVA group A interrupts
        SPLK    #0FFFFh, EVAIFRB       ;clear all EVA group B interrupts
        SPLK    #0FFFFh, EVAIFRC       ;clear all EVA group C interrupts
```

```
        SPLK    #00000h, EVAIMRA ;enabled desired EVA group A interrupts
        SPLK    #00000h, EVAIMRB ;enabled desired EVA group B interrupts
        SPLK    #00000h, EVAIMRC ;enabled desired EVA group C interrupts

        LDP     #DP_EVB                 ;set data page
        SPLK    #0FFFFh, EVBIFRA      ;clear all EVB group A interrupts
        SPLK    #0FFFFh, EVBIFRB      ;clear all EVB group B interrupts
        SPLK    #0FFFFh, EVBIFRC      ;clear all EVB group C interrupts
        SPLK    #00000h, EVBIMRA ;enabled desired EVB group A interrupts
        SPLK    #00000h, EVBIMRB ;enabled desired EVB group B interrupts
        SPLK    #00000h, EVBIMRC ;enabled desired EVB group C interrupts

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Enable global interrupts
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        CLRC    INTM                    ;enable global interrupts

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Disable the watchdog timer
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        LDP     #DP_PF1                 ;set data page

        SPLK    #0000000011101000b, WDCR
*                |||||||||||||||
*                FEDCBA9876543210
* 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
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        LDP     #GPR0     ;set current data page to
                           ;the data page of variable GPR0

        SPLK    #0000000001000000b, GPR0
*                |||||||||||||||
*                FEDCBA9876543210
* 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

        OUT     GPR0, WSGR

;-----------------------------------------------------------------------
;       RESET SECTION - BEGINS
;-----------------------------------------------------------------------
        LDP    #DP_EVA                  ;set data page
       SPLK    #0, GPTCONA
       SPLK    #0, T1CON
       SPLK    #0, T2CON

       SPLK    #0, COMCONA
       SPLK    #0, ACTRA
       SPLK    #0, DBTCONA
```

```
       SPLK   #0, CAPCONA
;-----------------------------------------------------------------------
;      RESET SECTION - ENDS
;-----------------------------------------------------------------------

* Initialize ADC registers
       LDP  #DP_PF2
       SPLK   #0100000000000000b,ADCTRL1 ; Reset ADC module
       NOP
       SPLK   #0011000000010000b,ADCTRL1 ; Take ADC out of reset
          ;     ||||||||||||||||
          ;     5432109876543210
          ; 15 - RSVD | 14 - Reset(1) | 13,12 - Soft & Free
          ; 11,10,9,8 - Acq.prescalers | 7 - Clock prescaler
          ; 6 - Start/stop mode (0) | 5 - Int.priority (Hi.0)
          ; 4 - Seq.casc (0 - dual)

* Setup a maximum of 16 conversions
       SPLK   #15, MAX_CONV        ; Setup for 16 conversions
* Program the conversion sequence. This is the sequence of channels that
* will be used for the 16 conversions.
       SPLK   #03210h, CHSELSEQ1      ; Convert Channels 0,1,2,3
       SPLK   #07654h, CHSELSEQ2      ; Convert Channels 4,5,6,7
       SPLK   #0BA98h, CHSELSEQ3      ; Convert Channels 8,9,10,11
       SPLK   #0FEDCh, CHSELSEQ4      ; Convert Channels 12,13,14,15
       SPLK   #0010000000000000b,ADCTRL2   ; Start the conversions
            ;  ||||||||||||||||
            ;  5432109876543210
       NOP
       NOP
       NOP
       NOP
CHK_EOS1:
       BIT  ADCTRL2, BIT12     ; Wait for SEQ1 Busy bit to clear
                               ; BIT12 is loaded into TC bit of ST1
       BCND   CHK_EOS1, TC     ; If TC=1, keep looping.
       LACC   RESULT0          ; Load conversion result of Channel 0
                               ; from Ch0 buffer register into ACC
       CLRC   SXM              ; Reset Bit 10 of status register ST1,
                              ; enabling logic shifts of the accumulator.
       RPT      #5            ; Repeat the following instruction 6 times
       SFR                   ; Shift right (6 times since ADC is 10-bit)
       LDP    #RESULT        ; Load data page for RESULT
       SACL RESULT    ; Store the Ch0 conversion result into a variable
LOOP:  B    LOOP      ; The conversion results are now
                      ; available in the RESULTSn regs.

;=======================================================================
; I S R  -  PHANTOM
; Description: Dummy ISR, used to trap spurious interrupts.
; Modifies: Nothing
;=======================================================================
PHANTOM RET
GISR1  RET
GISR2  RET
GISR3  RET
GISR4  RET
GISR5  RET
GISR6  RET
```

Example Program 2:

Configure Channels 0-15 but only read channel 0 for the result. Use low priority ADC interrupt to read the converted value.

Programming with Real-Time Monitor feature involves more supporting files besides what we used before: f2407.h, vector.asm, main.asm, and 2407.cmd. The new files are:

rtvector.asm – used to replace the previous vector.asm;

sys_init.asm – the environmental setting part of the code in the old main.asm is moved to this file. This file is called by the new main.asm.

c200mnrt.i – header file for real-time monitor macros

c200mnrt.asm – code defining real-time monitor macros

The complete main.asm is given below. All register settings are commented in details. The real-time related settings can be used as a template to be implanted to any other programs.

```
;*******************************************************************
; File Name:    ch5_e2.asm
; Target System: C240x Evaluation Board
; Description:   This program initializes the ADC module of the 2407
;           and does continuous conversion of all the analog input
;           channels. An interrupt is generated at the end of each
;           conversion sequence. The RESULT0 value must be loaded
;           into the memory location "RESULT" after each sequence in
;           the ISR.
;
;           Real-time monitor is used to support the real-time
;           running mode of Code Composer. ADC result can be
;           observed graphically with the real-time mode and data
;           buffer storage.
;*******************************************************************

;*******************************************************************
;               SYSTEM OPTIONS
;*******************************************************************
real_time         .set   1    ; 1 for real time mode, otherwise set 0
BUFFER         .set 8000h     ; buffer location
BUFFER_size      .set 50      ; buffer size 50
;*******************************************************************

;-------------------------------------------------------------------
; External references
;-------------------------------------------------------------------
```

```
       .include  "f2407.h"
       .global   MON_RT_CNFG

       .ref    SYS_INIT

;---------------------------------------------------------------------
; Local Variable Declarations
;---------------------------------------------------------------------
       .def    GPR0        ;General purpose register.

       .bss    GPR0,1        ;General purpose register.
       .bss    ctr,1              ;counter for background loop
       .bss    RESULT,1      ;store ADC result
       .bss    BUFFER_write_ptr,1 ;data buffer location pointer
       .bss    BUFFER_ctr,1     ;data counter for writing buffer

;=====================================================================
; V E C T O R    T A B L E    ( including RT monitor traps )
;=====================================================================
        .include   "c200mnrt.i"  ;Include conditional assembly options.

       .global  _c_int0,PHANTOM,GISR1,GISR2,GISR3,GISR4,GISR5,GISR6

;=====================================================================
; M A I N   C O D E  - starts here
;=====================================================================
       .text
_c_int0:
       CALL    SYS_INIT               ;DSP initialization

;-----------------------------------------------------------
; Initialise the Real time monitor
;-----------------------------------------------------------
;---Real Time option---------------
      .if (real_time)
       CALL    MON_RT_CNFG        ;For Real-Time
      .endif
;---------------------------------

;-----------------------------------------------------------
; System Interrupt Init.
;-----------------------------------------------------------

;---Real Time option -------------------------------------------------
      .if (real_time)
       SPLK #0000000001100000b,IMR  ;En Int lvl 6 & 7 for A/D and RT
              ;||||||||||!!!!!!!
             ;5432109876543210
      .endif


      .if (real_time != 1)
       SPLK #0000000000100000b,IMR  ;Enable Int 6 only for A/D
             ;||||!!!!!!!!!!!!!
             ;5432109876543210
      .endif

       SPLK #0FFFFh, IFR       ;Clear any pending Ints
;      EINT               ;Enable global Ints
```

```
;-----------------------------------------------------------------------

* Initialize ADC registers
       LDP  #DP_PF2
       SPLK   #0100000000000000b,ADCTRL1 ; Reset ADC module
       NOP
       SPLK   #0000000001110000b,ADCTRL1 ; Take ADC out of reset
         ;     ||||||||||||||||
         ;     5432109876543210
         ; 15 - RSVD | 14 - Reset(1) | 13,12 - Immed. stop
         ; 11,10,9,8 - Acq.prescalers | 7 - Clock prescaler
         ; 6 - Cont.run (1) | 5 - Int.priority (Hi: 0, Lo: 1)
         ; 4 - Seq.casc (0 - dual)

* Setup a maximum of 16 conversions
       SPLK   #15, MAXCONV          ; Setup for 16 conversions
* Program the conversion sequence. This is the sequence of channels that
* will be used for the 16 conversions.
       SPLK   #03210h, CHSELSEQ1      ; Convert Channels 0,1,2,3
       SPLK   #07654h, CHSELSEQ2      ; Convert Channels 4,5,6,7
       SPLK   #0BA98h, CHSELSEQ3      ; Convert Channels 8,9,10,11
       SPLK   #0FEDCh, CHSELSEQ4      ; Convert Channels 12,13,14,15
       SPLK   #0100000000000000b,ADCTRL2   ; Reset ADCTRL2
       SPLK   #0010011000000000b,ADCTRL2   ; Start the conversions
            ;  ||||||||||||||||          Enable SEQ1 interrupt (Mode 1)
            ;  5432109876543210         Clear INT flag for SEQ1


* Initialize A/D data buffer:
       LDP    #BUFFER_write_ptr     ;Initialization for buffer operation
       SPLK #BUFFER, BUFFER_write_ptr   ; Initial value for the pointer
       SPLK #BUFFER_size, BUFFER_ctr    ; Initial value for the counter

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Enable global interrupts
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       EINT          ;Enable all interrupts; clear the INTM to 0.

;=====================================================
;Main system background loop
;=====================================================
       LDP    #ctr
MAIN: SPLK  #7FFFh, ctr    ;An infinite loop is running in background
loop: LACC  ctr          ;doing counter decrementing operation while
       SUB    #1           ;waiting for next ADC interrupt request.
       SACL ctr          ;This background loop can be used to test
       BCND loop,NEQ     ;the real-time running mode
       B    MAIN

;=====================================================================
; I S R  -  GISR6
;
; Description: Store the conversion result to variable "RESULT"
;           and update the data buffer for graphical display
;=====================================================================
GISR6
       ;Context save regs
       MAR    *,AR1  ;AR1 is stack pointer
       MAR    *+      ;skip one position
```

```
       SST    #1, *+  ;save ST1
       SST    #0, *+  ;save ST0
       SACH *+      ;save ACC high
       SACL *    ;save ACC low
;=========================================================
;Start main section of ISR
;=========================================================
       LDP  #DP_PF2
        LACC   RESULT0            ; Load channel ADC0 result
       RPT    #5                  ; Shift right 6 bits
       SFR
       LDP    #RESULT
       SACL RESULT                ; Store result in the variable

       ; Write BUFFER
       LARP 5                      ; Use AR5 to be the current AR
       LAR    AR5, BUFFER_write_ptr  ; Load buffer pointer into AR5
       SACL *+                ; Write ADC result into the buffer location
                              ; pointed by the pointer; Increment AR5
       SAR    AR5, BUFFER_write_ptr   ; Store updated AR5 value
                                      ; back into the pointer
       LACC BUFFER_ctr          ; Load data counter
       SUB    #1                ; Decrement by 1
       BCND ST_CTR,NEQ          ; If ACC >= 0, branch to ST_CTR
                                ;(store counter)
RS_BUF:SPLK #BUFFER, BUFFER_write_ptr  ; If ACC = 0, buffer full,
                                       ; reset the pointer
                                       ;  to the beginning of the buffer
       LACC #BUFFER_size ;Reset the data counter back to the buffer size
ST_CTR:SACL    BUFFER_ctr

       ; Enable ADC
       LDP  #DP_PF2
       SPLK   #0000011000000000b,ADCTRL2   ;
            ;  |||||||||||||||          Enable SEQ1 interrupt (Mode 1)
            ;  5432109876543210         Clear INT flag for SEQ1
;=========================================================
;End main section of ISR
;=========================================================
       ;Context restore regs
       POINT_PG0
       MAR    *, AR1      ;make stack pointer active
       LACL *-     ;Restore Acc low
       ADDH *-     ;Restore Acc high
       LST    #0, *-      ;load ST0
       LST     #1, *-   ;load ST1
       EINT               ;Enable all interrupts; clear the INTM to 0.
       RET

;=========================================================================
; I S R  -  PHANTOM
; Description: Dummy ISR, used to trap spurious interrupts.
;=========================================================================
PHANTOM B PHANTOM
GISR1  RET
GISR2  RET
GISR3  RET
GISR4  RET
GISR5  RET
```

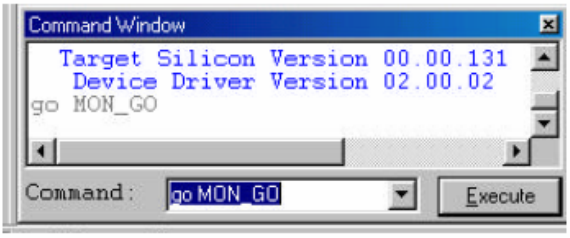
```
;GISR6 RET
```


Basic Settings

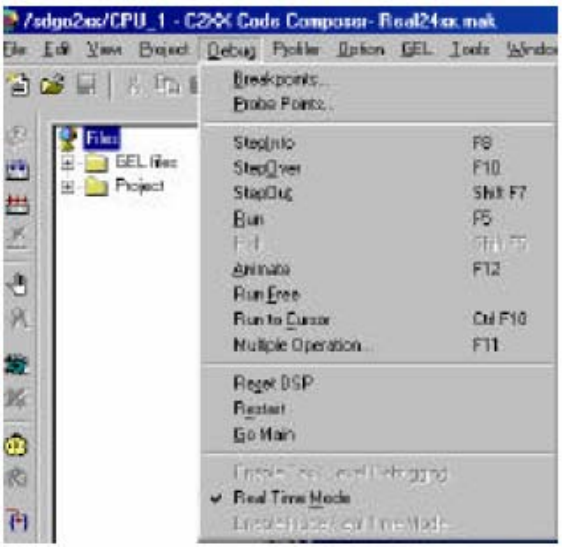
- To open the project *Ch5_e2.mak*, select Menu bar →Project →Open, then select the `.mak` file.
- To build the project (assemble and link code), select Menu bar → Project → Rebuild All

The Output window status should read: Build complete, 0 Errors, 0 Warnings

- To download the tutorial example code to the EVM board, select Menu bar → File →Load Program. Select Ch5_e2.out from the dialog box and click Open.
- To reset the DSP, select Debug →Reset DSP.
- Open the Command window from Code Composer's Tools menu.
- To run code emulation in real-time mode, type go MON_GO in the Command Window and click Execute.



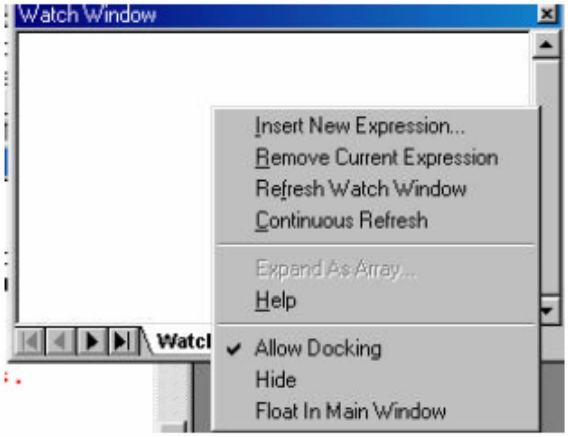
- To select real-time mode, select Debug → Real Time Mode. Then run the software by selecting Debug → Run.



The DSP should now be running in real-time mode.

Adding Watch Windows

- To add Watch windows, select View →Watch Window.
- To declare variables or peripheral registers to watch (note that the cursor should be on Watch window), right-click on Insert New Expression. (the Watch Add Expression box appears).



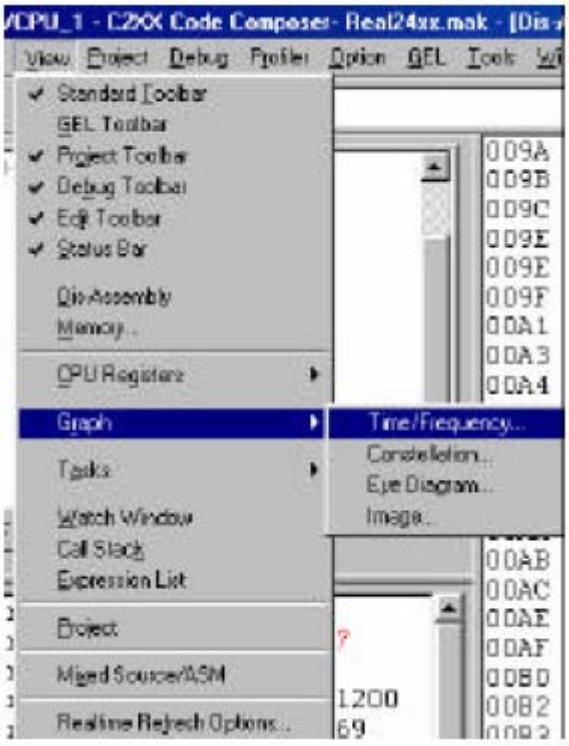
Type *RESULT. Repeat for variables:

• To allow continuous refresh of watch variable, right-click on Continuous refresh (in Watch window)

RESULT should now be updating rapidly.

Adding Graph View

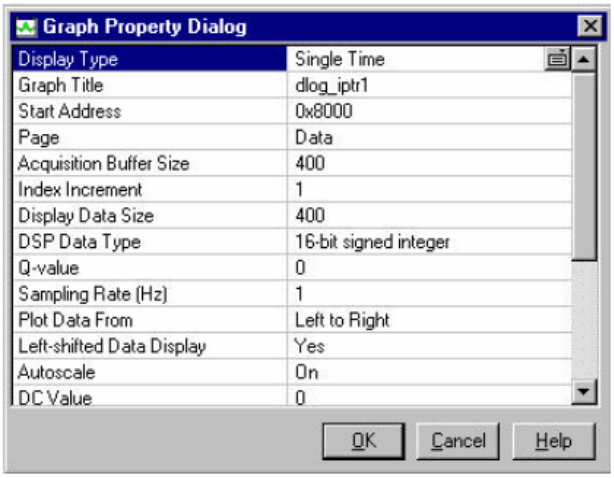
• To add Graph window, select View →Graph → Time/Frequency.



The Graph Property Dialog box will open.

• Modify the following in the Graph Property
Dialog box:

· Start Address →0x8000
· Acquisition Buffer Size →50
· Display Data Size →50
· DSPData Type →16-bit unsigned integer



• Click OK.

• Right-click on Graph Window and click Continuous refresh. The Graph window

should be displaying a sine wave as shown below.

Workspace Example Available

This Real-Time Tutorial for 2407 DSP also provides a workspace example. Open Ch5_e2.wks. To utilize the example workspace:
Open the workspace by selecting File → Workspace →Load Workspace, then select the file from the browser window.

Example Program 3:

Generate a staircase waveform of 50 Hz from Pin DACOUT1 (DAC0). Five steps of 0V, 0.825V, 1.65V, 2.475V, and 3.3V are equal in duration. Use an oscilloscope to observe the output waveform.

The main part of the code is shown below:

```
;*********************************************************************
; File Name:    ch5_e3.asm
; Target System: C240x Evaluation Board
; Description:  This sample program outputs a staircase waveform
;           with steps 0V, 0.825V, 1.65V, 2.475V, and 3.3V of 50Hz
;           at the pin DAC0OUT.
;*********************************************************************

;*********************************************************************
;                SYSTEM OPTIONS
;*********************************************************************
real_time .set 0         ; 1 for real time mode, otherwise set 0
DAC0   .set 0h           ; corresponding to Pin DACOUT1
DACUPDATE .set 0004h
;*********************************************************************


       LDP    #DP_EVA
       SPLK   #3750, T1PR   ;Period=4ms for one stair, so 5 stairs in
total=20ms
                     ;T1PR=30000000/32/250 for 30MHz 2407 DSP
       SPLK   #0055h, GPTCONA
       SPLK   #0h, T2CON

       SPLK   #0080h, EVAIMRA  ;Enable timer 1 period interrupt

       LDP    #CNT
       SPLK   #0, CNT         ;Initialize stair counter

       LDP    #DP_EVA
       SPLK   #1544h, T1CON  ;Continuous up-counting
                     ;Prescaler=32
                     ;Enable timer operation
                     ;Disable timer compare operation
```

```
       CLRC   INTM      ;Enable interrupts

WAIT  B       WAIT

;=====================================================================
; Routine Name: GISR2             Routine Type: ISR
;
; Description:
;=====================================================================
GISR2

       LDP    #CNT
       LACC #DACTBL     ;Set pointer to start of table
       ADD    CNT         ;Point to appropriate value in table
       POINT_PG0
       TBLR DAC0VAL     ;depending on the stair counter CNT
       OUT    DAC0VAL, DAC0
       OUT    DAC0VAL, DACUPDATE ;Trigger the D/A conversion
       LDP    #CNT
       LACC CNT
       SUB    #4
       BCND NXT, NEQ    ;Check if CNT reaches 4
       SACL CNT         ;If CNT=4, reset it to 0
       B    NXT1
NXT    LACC CNT         ;If CNT<4, increment by 1
       ADD    #1
       SACL CNT         ;Store the new CNT value

       EINT             ;Enable all interrupts; clear the INTM to 0.
       RET

PHANTOM B PHANTOM
GISR1  RET
;GISR2 RET
GISR3  RET
GISR4  RET
GISR5  RET
GISR6 RET

DACTBL .word  0000h     ; 0V
       .word  03FFh     ; 0.825V
       .word  07FFh     ; 1.65V
       .word  0BFFh     ; 2.475V
       .word  0FFFh     ; 3.3V
```

LABORATORY EXPERIMENT 4

Objectives

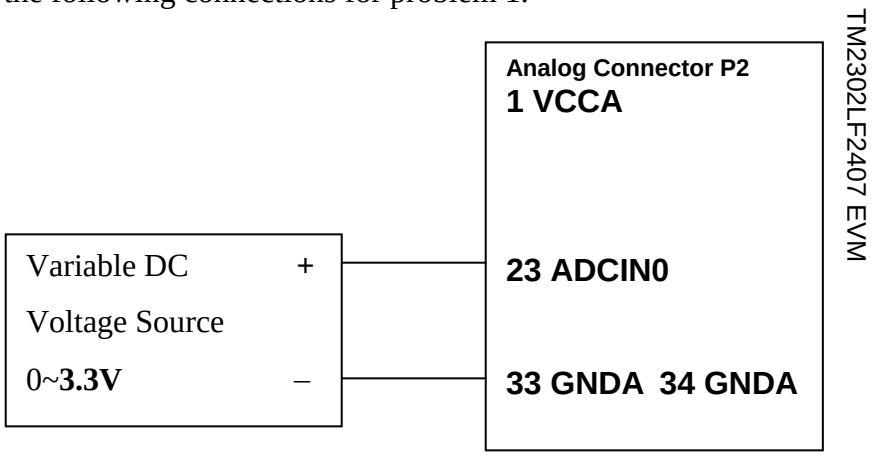
The objective of this lab session is to familiarize the students with the analog to digital converter (ADC) of the TMS320LF2407. In this session, the students will write and test assembly language programs that use TMS320LF2407 ADC to read analog

inputs using different approaches. The student will also generate different analog signals employing the digital to analog converter (DAC) on the EVM. Learn how to use Real-Time Monitor feature of Code Composer and write user code to support the real-time feature.

Procedure

Setup

1. Setup the EVM and PC as discussed in LAB1.
2. Make the following connections for problem 1.



Laboratory Assignments

Problem 1

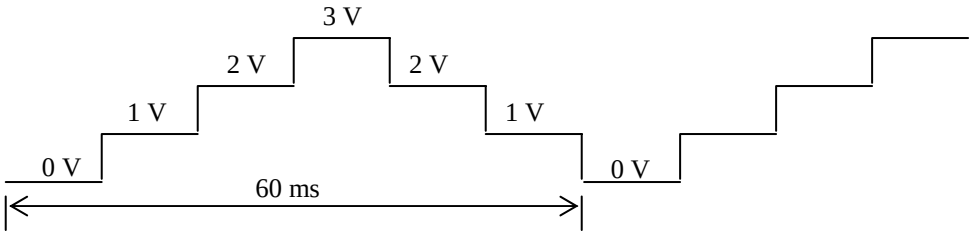
Configure channel 15 of ADC for continuous conversion. No interrupts need to be enabled. When the conversion ends, the FIFO value should be loaded into a memory location "RESULT". Connect the dc source as analog input. Observe the value of RESULT for different values of RESULT in the "Watch" window of the debugger. Tabulate the readings as follows and plot the data in an x-y plane where x-axis is the input in volt and y-axis is the converted decimal number.

Analog Input (volts)	RESULT

0	
0.55	
1.1	
1.65	
2.2	
2.75	
3.3	

Problem 2

Write a program to output a waveform at the DAC2OUT as shown below. Observe it on the oscilloscope.



Problem 3

Write a program that reads an analog voltage at ADCIN15 and outputs a proportional variable duty cycle PWM wave of frequency 1 kHz at the pin T1PWM/T1CMP/IOPB4. Observe the output on oscilloscope for various values of input. Use Real-Time Monitor feature to observe the input voltage in both watch window and graph window. Make connections as below. (Hint: T1PR=7500, prescaler=4; T1CMPR=ADC_RESULT*7, which covers value 0 – 7168<7500; if ADC interrupt used, timer interrupt not necessary.)

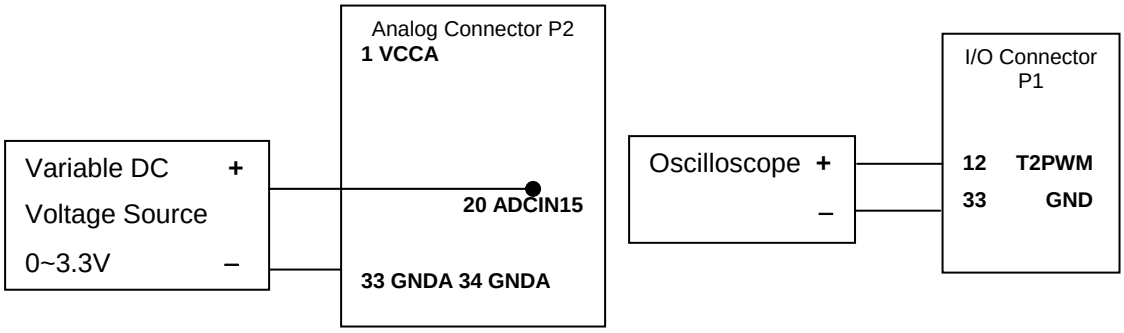


Table 2: P1 I/O

Pin #	Signal	Pin #	Signal
1	VCC, +5 Volts	2	VCC, +5 Volts
3	PWM1/IOPA6	4	PWM2/IOPA7
5	PWM3/IOPB0	6	PWM4/IOPB1
7	PWM5/IOPB2	8	PWM6/IOPB3
9	PWM7/IOPE1	10	PWM8/IOPE2
11	PWM9/IOPE3	12	T1PWM/T1CMP/IOPB4
13	T2PWM/T2CMP/IOPB5	14	T3PWM/T3CMP/IOPF2
15	* TDIRA/IOPB6	16	* TCLKINA/IOPB7
17	GND	18	GND
19	BOOTEN-/XF	20	* BIO/IOPC1
21	* CAP1/QEP1/IOPA3	22	* CAP2/QEP2/IOPA4
23	* CAP3/IOPA5	24	* CAP4/QEP3/IOPE7
25	RESERVED	26	* PDPINTA-
27	SCITXD/IOPA0	28	* SCIRXD/IOPA1
29	* SPISIMO/IOPC2	30	* SPISOMI/IOPC3
31	* SPICLK/IOPC4	32	* SPISTE/IOPC5
33	GND	34	GND

* Signal is interfaced through a quick switch to allow 5 volt tolerant inputs.

Table 3: P2 Analog

Pin #	Signal	Pin #	Signal
1	VCCA, +5V Analog	2	VCCA, +5V Analog
3	TMS2/IOPD7	4	* IOPF6
5	ADCIN2	6	ADCIN3
7	ADCIN4	8	ADCIN5
9	ADCIN6	10	ADCIN7
11	ADCIN8	12	ADCIN9
13	ADCIN10	14	ADCIN11
15	ADCIN12	16	ADCIN13
17	AGND	18	AGND
19	ADCIN14	20	ADCIN15
21	VREFHI	22	VREFLO
23	ADCIN0	24	ADCIN1
25	DACOUT1	26	DACOUT2
27	DACOUT3	28	DACOUT4
29	RESERVED	30	RESERVED
31	RESERVED	32	XINT2-/ADCSOC/IOPD1
33	AGND	34	AGND