Laboratory 2 – ADC and Temperature Measurement

Purpose:

The purpose of this laboratory exercise is to learn to use the ADC in the PIC24FJ128GA010 and the temperature sensor on the Explorer-16 Development Board.

Background:

The ADC in the PIC24FJ128GA010 has 16-multiplexed channel which allow 10-bit conversion in various automatic and manual modes of conversion.

Prelab 1:

The registers to be initialized are listed below. You may want to write the binary pattern for each register below the figure of the register. Transfer these binary patterns to the Prelab document.

AD1CON1 should be configures as:

1. A/D Converter module is disabled (de-energized)
2. Continue module operation in idle mode
3. Data output format is integer (0000 00dd dddd dddd)
4. Internal counter ends sampling and starts conversion (auto convert)
5. Sampling begins when SAMP bit is set.

Upper Byte:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
ADON	--	ADSIDL	--	--	--	FORM1	FORM0
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
SSRC2	SSRC1	SSRC0	--	--	ASAM	SAMP	DONE
bit 7							bit 0

bit 15 **ADON:** A/D Operating Mode bit
`1` = A/D converter module is operating
`0` = A/D converter is off
bit 14 **Unimplemented:** Read as '`0`'
bit 13 **ADSIDL:** Stop in Idle Mode bit
`1` = Discontinue module operation when device enters Idle mode
`0` = Continue module operation in Idle mode
bit 12-10 **Unimplemented:** Read as '`0`'
bit 9-8 **FORM1:FORM0:** Data Output Format bits
`11` = Signed fractional (`sddd dddd dd00 0000`)
`10` = Fractional (`dddd dddd dd00 0000`)
`01` = Signed integer (`ssss sssd dddd dddd`)
`00` = Integer (`0000 00dd dddd dddd`)
bit 7-5 **SSRC2:SSRC0:** Conversion Trigger Source Select bits

111 = Internal counter ends sampling and starts conversion (auto-convert)
110 = Reserved
10x = Reserved
011 = Reserved
010 = Timer3 compare ends sampling and starts conversion
001 = Active transition on INT0 pin ends sampling and starts conversion
000 = Clearing SAMP bit ends sampling and starts conversion

bit 4-3 **Unimplemented:** Read as '0'

bit 2 **ASAM:** A/D Sample Auto-Start bit
1 = Sampling begins immediately after last conversion completes. SAMP bit is auto-set.
0 = Sampling begins when SAMP bit is set

bit 1 **SAMP:** A/D Sample Enable bit
1 = A/D sample/hold amplifier is sampling input
0 = A/D sample/hold amplifier is holding

bit 0 **DONE:** A/D Conversion Status bit
1 = A/D conversion is done
0 = A/D conversion is NOT done

AD1CON2 should be configured as:

1. $V_{R+} = AV_{DD}$ and $V_{R-} = AV_{SS}$
2. Do not scan inputs
3. Buffer configured as one 16-word buffer (ADC1BUF0 to ADC1BUFF)
4. Always uses MUX A input Multiplexer settings.

Upper Byte:							
R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	U-0	U-0
VCFG2	VCFG1	VCGF0	r	--	CSCNA	--	--
bit 15							bit 8

Lower Byte:							
R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUFS		SMPI3	SMPI2	SMPI1	SMPI0	BUFM	ALTS
bit 7							bit 0

bit 15-13 **VCFG2:VCFG0:** Voltage Reference Configuration bits:

VCFG2:VCFG0	V_{R*}	V_{R-}
000	AV_{DD}	AV_{SS}
001	External V_{REF+} pin	AV_{SS}
010	AV_{DD}	External V_{REF-} pin
011	External V_{REF+} pin	External V_{REF-} pin
1xx	AV_{DD}	AV_{SS}

bit 12 **Reserved:** User should write '0' to this location

bit 11 **Unimplemented:** Read as '0'

bit 10 **CSCNA:** Scan Input Selections for CH0+ S/H Input for MUX A Input Multiplexer Setting bit
1 = Scan inputs
0 = Do not scan inputs

bit 9-8 **Unimplemented:** Read as '0'

bit 7 **BUFS:** Buffer Fill Status bit (Valid only when BUFM = 1)
1 = A/D is currently filling buffer 08-0F, user should access data in 00-07
0 = A/D is currently filling buffer 00-07, user should access data in 08-0F

bit 6 **Unimplemented:** Read as '0'

bit 5-2 **SMPI3:SMPI0:** Sample/Convert Sequences Per Interrupt Selection bits
`1111` = Interrupts at the completion of conversion for each 16th sample/convert sequence
`1110` = Interrupts at the completion of conversion for each 15th sample/convert sequence
`.....`
`0001` = Interrupts at the completion of conversion for each 2nd sample/convert sequence
`0000` = Interrupts at the completion of conversion for each sample/convert sequence
bit 1 **BUFM:** Buffer Mode Select bit
`1` = Buffer configured as two 8-word buffers (ADC1BUFn<15:8> and ADC1BUFn<7:0>)
`0` = Buffer configured as one 16-word buffer (ADC1BUFn<15:0>)
bit 0 **ALTS:** Alternate Input Sample Mode Select bit
`1` = Uses MUX A input multiplexer settings for first sample, then alternates between MUX B and MUX A input multiplexer settings for all subsequent samples
`0` = Always use MUX A input multiplexer settings

AD1CON3 should be configured as:

1. Clock derived from system clock
2. Auto-sampling time set to 31 T_{AD} (31*10*125 ns = 387.5 μs)
3. A/D conversion clock set to 5 T_{CYC} or 10 T_{OSC} (10*125 ns = 1250 ns)

Upper Byte:							
R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADRC	--	--	SAMC4	SAMC3	SAMC2	SAMC1	SAMC0
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADCS7	ADCS6	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0
bit 7							bit 0

bit 15 **ADRC:** A/D Conversion Clock Source bit
`1` = A/D internal RC clock
`0` = Clock derived from system clock
bit 14-13 **Unimplemented:** Read as '0'
bit 12-8 **SAMC4:SAMC0:** Auto-Sample Time bits
`11111` = 31 TAD
`.....`
`00001` = 1 TAD
`00000` = 0 TAD (not recommended)
bit 7-0 **ADCS7:ADCS0:** A/D Conversion Clock Select bits
`11111111` = 128 • TCY
`......`
`00000001` = TCY
`00000000` = TCY/2

AD1CHS should be configured with initChannel assigned to CH0SA3:CH0SA0.

Upper Byte:							
R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
CH0NB				CH0SB3	CH0SB2	CH0SB1	CH0SB0
bit 15							bit 8

Lower Byte:							
R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
CH0NA				CH0SA3	CH0SA2	CH0SA1	CH0SA0
bit 7							bit 0

bit 15 **CH0NB:** Channel 0 Negative Input Select for MUX B Multiplexer Setting bit
1 = Channel 0 negative input is AN1
0 = Channel 0 negative input is VRbit
14-12 **Unimplemented:** Read as '0'
bit 11-8 **CH0SB3:CH0SB0:** Channel 0 Positive Input Select for MUX B Multiplexer Setting bits
1111 = Channel 0 positive input is AN15
1110 = Channel 0 positive input is AN14
.....
0001 = Channel 0 positive input is AN1
0000 = Channel 0 positive input is AN0
bit 7 **CH0NA:** Channel 0 Negative Input Select for MUX A Multiplexer Setting bit
1 = Channel 0 negative input is AN1
0 = Channel 0 negative input is VRbit
6-4 **Unimplemented:** Read as '0'
bit 3-0 **CH0SA3:CH0SA0:** Channel 0 Positive Input Select for MUX A Multiplexer Setting bits
1111 = Channel 0 positive input is AN15
1110 = Channel 0 positive input is AN14
.....
0001 = Channel 0 positive input is AN1
0000 = Channel 0 positive input is AN0

Analog inputs should be selected using the code:

```
TRISBbits.TRISB3 = 1;       // RB3/AN3 as digital input
TRISBbits.TRISB4 = 1;       // RB4/AN4 as digital input
TRISBbits.TRISB5 = 1;       // RB5/AN5 as digital input
AD1PCFGbits.PCFG3 = 0;      // RB3/AN3 as analog mode
AD1PCFGbits.PCFG4 = 0;      // RB4/AN4 as analog mode
AD1PCFGbits.PCFG5 = 0;      // RB5/AN5 as analog mode
```

. From the PIC24FJ128GA Family Data Sheet the following is obtained.

9.2 Configuring Analog Port Pins

The use of the AD1PCFG and TRIS registers control the operation of the A/D port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bit set (input). If the TRIS bit is cleared (output), the digital output level (VOH or VOL) will be converted.

When reading the PORT register, all pins configured as analog input channels will read as cleared (a low level).

Pins configured as digital inputs will not convert an analog input. Analog levels on any pin that is defined as a digital input (including the ANx pins) may cause the input buffer to consume current that exceeds the device specifications.

To turn on the ADC at end of conversion used the statement:

```
AD1CON1bits.ADON=1;
```

Task 1:

Write a routine called initADC that takes one augments and returns no data and add to peripherals.c. Add the following declaration to peripherals.h. The declaration should be:

```
void initADC(unsigned int initChannel);
```

where initiChannel specifies which channel is selected for sampling. The function should follow the step given below in the sequence given:

1. Initialize AD1CON1 with ADC de-energized using Prelab1 values.
2. Initialize AD1CON2 with ADC using Prelab1 values
3. Initialize AD1CON3 with ADC using Prelab1 values.
4. Select initial channel using initChannel
5. Ignore all scan select channels
6. Set AN3, AN4 and AN5 as analog inputs
7. **Enable** A/D converter (energize).

Write a routine called getADC that takes one augments and returns one data and add to peripherals.c. Add the following declaration to peripherals.h. The declaration should be:

```
unsigned int getADC(unsigned int channel);
```

where channel specifies which channel is selected for sampling and the returned value is the converted voltage. The function should follow the steps given below in the sequence given:

1. Select channel using input argument.
2. Start sample by setting AD1CON1bits.SAMP.
3. Wait for AD1CON1bits.DONE to be set.
4. Clear AD1CON1bits.DONE
5. Return the value ADC1BUF0.

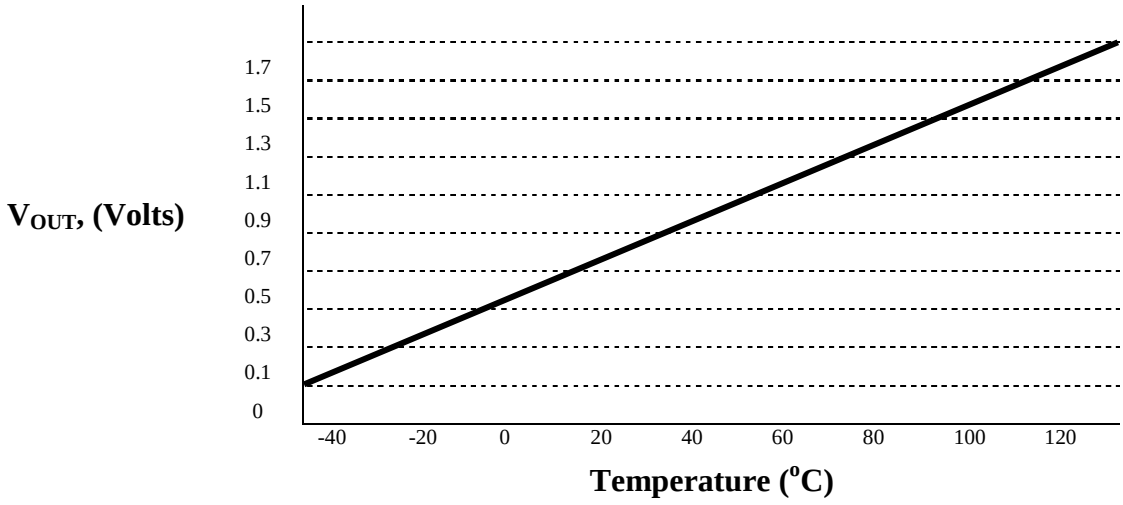
Write a main function that initializes the ADC using initADC(4) and in the endless loop gets the ADC results using getADC( ). Don't forget to add _CONFIG1 and _CONFIG2 macros.

Compile and debug this code. Run a MPLAB simulation with Register Injection for ADC1BUF0 from a file called ADC.txt. Place the hex values of 0042, 0063, and others in ADC.txt. Verify that the code and simulations work by watching the value returned by getADC function.

Prelab 2:

The temperature sensor on the Explore-16 Development Board is the TC1047A. This sensor has the following transfer characteristic.

V_{OUT} = (10mV/^{o}C)(Temperature ^{o}C) + 500 mV



In the units of volts V_{OUT} for the TC1047A is given by;

$$V_{OUT} = 0.01T + 0.5,$$

where T is the temperature in ^{o}C. Therefore the temperature in ^{o}C is given by:

$$T = 100V_{OUT} - 50.$$

The scaled voltage derived from the PIC25FJ128GA010's ADC is given by:

$$V_{SCALED} = \frac{3.3N}{1024},$$

where N is the value obtained from the ADC. Therefore the temperature in ^{o}C as a function of the ADC value is given by:

$$T = \frac{330N - 51200}{1024}.$$

Sometimes the temperature in degrees Fahrenheit, ^{o}F, is desired. The conversion equation is given by:

$$F = \frac{9}{5}T + 32,$$

where T is the temperature in ^{o}C and F is the temperature in ^{o}F

Create a spreadsheet with the following 6 columns:

1. N as whole number from 32 to 312
2. Voltage output of TC1047A to nearest mV
3. Temperature in ^{o}C to nearest degree C as a function of N
4. Temperature in pervious column in binary [use DEC2BIN(value,bits)]
5. Temperature in ^{o}F to nearest degree F as a function of N
6. Temperature in pervious column in binary [use DEC2BIN(value,bits)]

You will find repeated column labels helpful at the top of each page. Print and attach to PreLab document.

Task 2:

Since V_{OUT} can range from 0.1 to 1.75 for the TC1047A, the value of N can range from 32 to 544. Therefore the term 330N has a maximum value of 314,160. The calculations must be performed using long data type.

Write a main function that does the following initialization:

1. Initialize the ADC with channel AN4 set as the initial channel using initADC(0x0004).
2. Initalize the buttons so S4 press can be detected using initButtons(0x0001).
3. Initialize the bargraoph using initBargraph( ).

Write an endless loop for the main function that does the following and then repeats:

1. Read the ADC's channel AN4 using getADC(0x0004).
2. Convert this reading to a binary representation of the temperature in ^{o}C[1].
3. Display temperature in ^{o}C on bargraph using setBargraph(…).
4. Wait for one second using msDelay(1000).

Don't forget to add _CONFIG1 and _CONFIG2 macros. Compile, debug and program the PIC24FJ128GA010 on the Explorer-16 Development Board. Verify that the program is working by measuring the room temperature and your body (finger tip) temperature. Please do not attempt to slide the Explored-16 Development Board under your tongue.

Demonstrate that the task is working to the laboratory assistant or instructor and obtain their signature in the box below.

[1] Watch associativity and precedence

_____ _____ Signature Date

Task 3:

Modify the endless loop in the main function for Task 2 so that the temperature in ^{0}C and ^{0}F are calculated. . Change the endless loop to the following and repeat:

1. Read the adc's channel AN4 using getADC(0x0004).
2. If S4 is pressed
    a. Convert the reading to a binary representation of the temperature in ^{o}C.
    b. Display temperature in ^{o}C on bargraph using setBargraph(…).
3. Else
    a. Convert the reading to a binary representation of the temperature in ^{o}F[2].
    b. Display temperature in ^{o}F on bargraph using setBargraph(…).
4. Wait for one second using msDelay(1000).

Task 4:

Perform the following cleanup steps:

1. Select Programmer | Select Programmer | MPLAB ICD2
2. Select Programmer | Erase Part
3. Copy the all code folders to your WebDrive, your e-mail account as an attachment or flash driver. These files will be erased off the hard drive early tomorrow morning. You **will need** some of this code for future laboratory exercises
4. Inventory desk

Report Format:

This laboratory experience requires an informal report. There also will be a 15 question SpringBoard online quiz to be completed before next Tuesdays lecture. Even though you did the exercises as a group of two or three, the quiz will be individual. Your score on the quiz will be taken as 50% of your score for this laboratory exercise.

The contents of the informal report will be:

1. Title page
2. Signed Prelab Document
3. Prelab Spreadsheet
4. Peripherals.h listing

[2] Watch associativity and precedence

5. initADC( ) and getADC( ) listing in peripherals.c
6. Task 1 listing with main( )
7. Task 2 listing with main( )
8. Task 3 listing with main( )

This Page intentionally left blank.

ADC & Temperature Measurement

Name 1: _____
Please Print

Name 2: _____
Please Print

Name 3: _____
Please Print

Submitted On: ___ Sept 2008

Laboratory Section: _____
Please Print

Check List:

√	Description	Score
	Signed Prelab Document	
	Prelab Spreadsheet	
	Peripherals.h listing	
	initADC( ) and getADC( ) listing in peripherals.c	
	Task 1 listing with main( )	
	Task 2 listing with main( )	
	Task 3 listing with main( )	
	Total	

The work presented in this report is solely the work of the authors. Presenting the work of others is plagiarism and will be penalized by failure of the course for the slightest infraction.

This Page intentionally left blank.

ADC & Temperature Measurement Prelab

Name 1: _____

Name 2: _____

Name 3: _____

Please Print

Determine what the binary pattern for each of the following registers should be for the discussion under the section Prelab 1.

AD1CON1

Bit 15															**Bit 0**

AD1CON2

Bit 15															**Bit 0**

AD1CON3

Bit 15															**Bit 0**

AD1CHS

Bit 15															**Bit 0**

AD1PCFG

Bit 15															**Bit 0**

Attachment:

Spreadsheet discussed in Prelab 2 section.