Laboratory 1 – MPLAB & C-30 Tutorial

Purpose:

The purpose of this laboratory exercise is to become familiar with the Microchip MPLAB Integrated Development Environment (IDE), Microchip MPLAB In-Circuit Debugger-2, and Microchip C-30 compiler.

Configuration Attributes:

Each PIC has configuration words that affect the overall operation of the device. These bits may be set in the Configure | Configuration Bits… menu or the __CONFIG1( ) AND __CONFIG2( ) macro. In this course we will use the __CONFIG1( ) and __CONFIG2( ) macro. See Chapter 23 SPECTIAL FEATURES in the PIC24FJ128GA Family Data Sheet for detail on the affect of each bit. Table 1 is a summary of definitions used to set and clear each bit.

Table 1 PIC24FJ128GA010 Configuration Macros in Microchip C-30

Configuration 1	Option	Configuration 2	OPTION
JTAG disable	JTAGEN_OFF	Two-Speed Start-up Disabled	IESO_OFF
JTAG enable	JTAGEN_OFF	Two-Speed Start-up Enabled	IESO_ON
Code protect enabled	GCP_ON	Fast RC oscillator	FNOSC_FRC
Code protect disable	GCP_OFF	Fast RC Oscillator with divide and PLL	FNOSC_FRCPLL
Write protect enable	GWRP_ON	Primary oscillator (XT,HS.EC)	FNOSC_PRI
Write protect disabled	GWRP_OFF	Primary oscillator (XT, HS, EC) with PLL	FNOSC_PRIPLL
Background debug enabled	BKBUG_ON	Secondary oscillator	FNOSC_SOSC
Background debug disabled	BKBUG_OFF	Low power RC oscillator	FNOSC_LPRC
Clip-on emulation enabled	COE_ON	Fast RC oscillator with divide	FNOSC_DIV
Clip-on emulation disabled	COE_OFF	Clock switch and monitor enabled	FCKSM_CSECME
EMUC/EMUD share PGC1/PGD1	ICS_PGx1	Clock switch enabled	FCKSM_CSECMD
EMUC/EMUD share PGC2/PGD2	ICS_PGx2	Clock switch and monitor disabled	FCKSM_CSDCMD
Watchdog timer disabled	FWDTEN_OFF	OSCO to RC15	OSCIOFNC_ON
Watchdog timer enabled	FWDTEN_ON	OSCO or Fosc/2	OSCIOFNC_OFF
Windowed WDT enabled	WINDIS_ON	External clock	POSCMOD_EC
Windowed WDT disabled	WINDIS_OFF	XT oscillator	POSCMOD_XT
WDT prescale 1:32	FWPSA_PR32	HS oscillator	POSCMOS_HS
WDT prescale 1:126	FWPSA_PR128	Primary disabled	POSCMOD_NONE
WDT postscale 1:1	WDTPS_PS1		
WDT postscale 1:2	WDTPS_PS2		
WDT postscale 1:4	WDTPS_PS4		
WDT postscale 1:8	WDTPS_PS8		
WDT postscale 1:16	WDTPS_PS16		
WDT postscale 1:32	WDTPS_PS32		
WDT postscale 1:64	WDTPS_PS64		
WDT postscale 1:128	WDTPS_PS128		
WDT postscale 1:256	WDTPS_PS256		
WDT postscale 1:512	WDTPS_PS512		
WDT postscale 1:1,024	WDTPS_PS1024		
WDT postscale 1:2,048	WDTPS_PS2048		
WDT postscale 1:4,096	WDTPS_PS4096		
WDT postscale 1:8,192	WDTPS_PS0192		
WDT postscale 1:16,384	WDTPS_PS16384		
WDT postscale 1:32,768	WDTPS_PS32768		

To affect each bit or flag bitwise-AND it with the other flags or bits in the configuration word. As an example to disable the watchdog time and enable the clip-on emulator, the macro would be __CONFIG1(FWDTEN_OFF&COE_ON). Note the macro begins with two underscores. The macro does not allow the line to be ended with a semicolon. For this semester laboratories the configuration given below will be used:

```
_CONFIG1(JTAGEN_OFF & GCP_OFF & GWRP_OFF & BKBUG_OFF & COE_OFF & FWDTEN_OFF)
```

```
_CONFIG2(FCKSM_CSDCMD & OSCIOFNC_ON & POSCMOD_HS & FNOSC_PRI)
```

Task 1:

Watch the instructor's presentation for tutorial I. Perform the following steps when told to do so by the laboratory instructor

1. Create new folder for Tutorial 1
2. Create project for PIC24FJ128G in created folder using C-30
3. Enter the code for Tut01.C and save in new folder
4. Add Tut01.c to project if not already added
5. Add C:\Program Files\Microchip\MPLAB C30\support\PIC24F\gld\P24FJ128GA010.gld linker script
6. Select Project | Build Configuration | Debug
7. Select Project | Build Options … | Project | MPLAB C30 | General | Generate debugging information
8. Build under Projects menu
9. Program PIC24FJ128GA010 on Explorer-16 Development Board under Debugger menu
10. Select View | Watch
11. Add SFRs PORTA, PORTD, TRISA, and TRISD and the symbol `display`
12. Single Step Program and observe the watch window and the bargraph
13. Recompile with 0L in for loop set to 1001L
14. Reprogram
15. Run Program and observe the bargraph while exercising S3 (reset) and S4 (increment). With no switches pressed the bargraph should decrement.

The Tut0l.C code is given below:

```
#include <p24fj128ga010.h>

//Configuration words for Explorer-16 when using ICD-2
_CONFIG1(JTAGEN_OFF & GCP_OFF & GWRP_OFF & BKBUG_OFF & COE_OFF & FWDTEN_OFF)
_CONFIG2(FCKSM_CSDCMD & OSCIOFNC_ON & POSCMOD_HS & FNOSC_PRI)

#define SCALE 308L

int main(void){
unsigned long i;
unsigned char display=0;
//initialization
        PORTA=0x0000;           //clear port A
        TRISA=0xFF00;           //set PORTA <7:0> to output
        TRISD=0xFFFF;           //set PORTD to input
//endless loop
again:
        Nop();                                  //breakpoint access point
        PORTA=(unsigned int) display; //sent count to display
        for(i=0L*SCALE;i>0;i--) Nop();          //delay 0 milliseconds
        if(PORTDbits.RD13 == 0)
                display=display+1;              //if S4 pressed increment display counter
        else if(PORTDbits.RD6 == 0)
                display=0;                      //else if S3 pressed reset display counter
        else
                display=display-1;              //else decrement display counter

        goto again;                             //one more time
        }
```

Task 2:

With embedded systems it is a very good idea to modularize your code. This allows one to develop "libraries" of interfacing code that may be reused in future code. Also one should include copious comments to code while writing said code. The comment will help when using the code in the future. Also often the comments are correct but the code has logical errors. The comments will help you and others find the logical errors.

Watch the instructor's presentation for tutorial 2. Perform the following steps when told to do so by the laboratory instructor:

1. Create new folder for Tutorial 2
2. Create project for PIC24FJ128G in created folder using C-30
3. Enter the code for Tut02.C, peripherals.c and peripherals.h and save in created folder
4. Add Tut02.c, peripherals.c and peripherals.h to project if not already added
5. Add C:\Program Files\Microchip\MPLAB C30\support\PIC24F\gld\P24FJ128GA010.gld linker script
6. Select Project | Build Configuration | Debug
7. Select Project | Build Options … | Project | MPLAB C30 | General | Generate debugging information
8. Build All under Projects menu
9. Select ICD-2 under Debugger menu
10. Program PIC24FJ128GA010 on Explorer-16 Development Board under Debugger menu
11. Run the program and observe behavior of bargraph while exercising the S4 and S3
12. Select MPLAB SIM under Debugger menu
13. Open the stopwatch under debugger window and the select Debugger | Settings … | OSC/Trace |Processor Frequency 8 MHz
14. Measure the execution time for each function in peripheral.c.
15. Log the execution times in units of time and cycles in a spread sheet

The Tut02.C is given below:

```
#include <p24fj128ga010.h>
#include "peripherals.h"

//Configuration words for Explorer-16 when using ICD-2
_CONFIG1(JTAGEN_OFF & GCP_OFF & GWRP_OFF & BKBUG_OFF & COE_OFF & FWDTEN_OFF)
_CONFIG2(FCKSM_CSDCMD & OSCIOFNC_ON & POSCMOD_HS & FNOSC_PRI)

int main(void){
       unsigned char display=0;
//initialization
       initButtons(0x0009);
       initBargraph();
//endless loop
again:
       Nop();                            //breakpoint access point
       setBargraph(display);             //sent count to display
       msDelay(1000);                    //delay approximately 1 second
       if(getButton(0x0001))
              display=display+1;         //if S4 pressed increment display
counter
       else if(getButton(0x0008))
              display=0;                 //else if RD6 presse3d reset display
counter
       else
              display=display-1;         //else decrement display counter
       goto again;                       //one more time
       }
```

The peripherials.C code is given below

```
       #include <p24fj128ga010.h>
```

```
#include "peripherals.h"

void initBargraph(void){
       PORTA=0x0000;       //clear port A
       TRISA=0xFF80;       //set PORTA <6:0> to output
                           //NOTE: RA7 shared with SW5 button
       }

void initButtons(unsigned int mask){
       if(mask&0x0008)
              TRISDbits.TRISD6=1; //switch S3 as input
       if(mask&0x0004)
              TRISDbits.TRISD7=1; //switch S6 as input
       if(mask&0x0002)
              TRISAbits.TRISA7=1; //switch S5 as input
                                  //NOTE: RA7 shared with
                                  //D10 in bargarph
       if(mask&0x0001)
              TRISDbits.TRISD13=1; //switch S4 as input
       }

unsigned int getButton(unsigned int mask){
       unsigned int button;
       switch(mask){
              case 0x0008: //read switch S3 (1 = pressed)
                                  button=!PORTDbits.RD6;
                                  break;
              case 0x0004: //read switch S6 (1 = pressed)
                                  button=!PORTDbits.RD7;
                                  break;
              case 0x0002: //read switch S5 (1 = pressed)
                           //NOTE: RA7 shared with D10 of bargraph
                                  TRISAbits.TRISA7=1;
                                  button=!PORTAbits.RA7;
                                  break;
              case 0x0001: //read switch S4 (1 = pressed)
                                  button=!PORTDbits.RD13;
                                  break;
              default:
                                  button=0;
              }
       return (button);
       }

void setBargraph(unsigned int display){
              //NOTE: RA7 shared with SW5 button
              TRISAbits.TRISA7=0;
              PORTA=0x00FF&display;      //transfer to bargraph
              }

void msDelay(unsigned int ms){
              unsigned long i;
              for(i=(unsigned long)(ms+1)*SCALE;i>0;i-=1) Nop();
              }
```

The code for peripheral.h is given below:

```
#ifndef PERIPHERALS_H
#define PERIPHERALS_H

#define SCALE 308L
```

```
/* ******************************************
  Initialize the bargraph.
  Note S5 used for both bargraph and buttons.
****************************************** */
void initBargraph(void);

/* ******************************************
  Initialize selected buttons using mask:
    0b1000  S3 or RD6,
    0b0100  S6 or RD7,
    0b0010  S5 or RA7,
    0b0001  S4 or RD13.
  Note S5 used for both bargraph and buttons.
***************************************** */
void initButtons(unsigned int mask);

/* *****************************************
  Get a single button with 1 = closed and
  0 = open, using mask:
    0b1000  S3 or RD6,
    0b0100  S6 or RD7,
    0b0010  S5 or RA7,
    0b0001  S4 or RD13.
  Note if more than one button selected, all
  assumed open.
****************************************** */
unsigned int getButton(unsigned int mask);

/* ******************************************
  Display given character in binary on bargraph
****************************************** */
void setBargraph(unsigned int display);

/* *****************************************
  Delay specified milliseconds when using 8
  MHz clock oscillator
***************************************** */
void msDelay(unsigned int ms);
#endif
```

Task 3:

Perform the following cleanup steps:

1. Select Programmer | Select Programmer | MPLAB ICD2
2. Select Programmer | Erase Part
3. Copy the Tutorial 1 and Tutorial 2 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 the first laboratory exercise.

The contents of the informal report will be:

1. Title page
2. Tutorial 1 listings
3. Tutorial 2 listings
4. Spread sheet with execution time information from task 2.

This page intentionally left blank.

MPLAB IDE & C-30 Tutorial

Name 1: ____________________
Please Print

Name 2: ____________________
Please Print

Name 3: ____________________
Please Print

Submitted On: ___ Sept 2008

Laboratory Section: ____________
Please Print

Check List:

√	Description	Score
	Task 1 Listings	
	Task 2 Listings	
	Execution Time Spread Sheet	
	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.