ECE 3740 Systems Engineering Principles
Project 1
MPLAB, TCP/IP Stack, and MX7cK Development Board

Fall 2023

Introduction

The Systems Engineering Principles Project 1 (SEP Project 1) is implemented by the series P1, P1, and P3. The SEP Project 1 has two main objectives: the first objective is to gain experience in using the MPLAB X IDE and the MX7cK board. You will also develop C code for controlling the LEDs through the Push Buttons on the MX7cK board. Finally, you will also develop C code to receive commands to control the LEDs and monitor the Push Button switches from a telnet session running on the Host computer.

P1.1: Part 1 of the Project 1 (P1.1) requires that students perform the following:

1. Install MPLAB® X IDE, MPLAB XC32 Compiler, and PIC32 Legacy Peripheral Libraries.
2. Gain familiarity with using MPLAB® X to create, edit, build, debug, and run C programs.
3. Install PIC TCP/IP Stack software.
4. Run networking tools and programs.
5. Practice Configuration Management.
6. Perform refactorization by modularization.
7. Write test plan and procedure documents.

P1.1 measures the following attributes:

ET.3: Engineering use of tools
- Demonstrate installing MPLAB X IDE, XC32 compiler, peripheral libraries, and Microchip's TCP/IP stack
- Demonstrate the use of MPLAB X IDE for developing C language code for the PIC32MX795F512L on the MX7cK board (compiling, programming, running, and debugging projects)
- Demonstrate the use of TCP/IP networking tools, such as ping.

DE.3: Design
- Design a test plan for testing proper operation of the stack software.

CS.3: Communication skills
- Demonstrate clear and accurate explanation of the project questions.
- Write test plan and procedure to test the stack operation.

The following Steps 1 - 24 are detailed instructions for performing P1.1. Students should follow them exactly:

1. Download MPLAB X IDE (Integrated Development Environment) from: http://www.microchip.com/
   a. Download the latest version of "MPLAB® X IDE" (In 2022, it was v4.50), and install it on your workstation. Follow the installation directions.
   b. You don't need to install the Integrated Programming Environment (IPE); so, deselect the box in the Select Programs dialog.
   c. You need to install the 32-bit compiler and you don't need the 8 and 16 bit compilers.
   d. The final dialog box lists some documentation which may be helpful.
   e. Near the end of the installation, a dialog will ask you to install the XC compiler. Ensure that the "XC compilers …" box is checked, and then click "Finish." You don't need to check the other two boxes.

f. Your browser should take you to page: http://www.microchip.com/mplab/compilers
g. Locate and download and install the latest version of the MPLAB XC32 Compiler (In 2022, it was v4.10).
h. Check the box to agree to the disclaimer.
i. Note the default location where the compiler will be installed.
j. Use default selections, except, additionally, check box "Add xc32 to the PATH environment variable".
k. When the installation completes, locate the MPLAB X **IDE** and pin the application to your taskbar.
l. After you have installed the MPLAB XC32 Compiler, click on the "Go to Downloads Archive" link, and locate the peripheral libraries. Download and install the plibs by clicking on the download icon of the "PIC32 Legacy Peripheral Libraries – Windows" link (near bottom of page). Choose the defaults for the installation, and ensure the libraries are installed under the Same xc32 compiler folder /xc32/v(version number), which you noted in Step (i) above.

2. On Jumper Block J3 (See Fig. 1), AKA Power Select, connect the jumper across the DBG (debug) position. This selects powering the MX7cK board using the computer's USB port, which is connected to the board via the "J15 DEBUG" USB connector (Page 5 of 34 [1]).
3. Connect the MX7cK board to the computer by connecting one side of the USB cable to the computer and the other side to the "J15 DEBUG" USB connector on the MX7cK board. **NOTE: When connecting the USB cable to the MX7cK board, grasp and hold the USB connector of the board with one hand and then connect the cable to the USB connector. The USB connector on the board is glued on the board, and continued re-connection of the USB cable to the connector without adequate support may cause the connector to dislodge from the board.** Switch the board ON (Fig. 1).

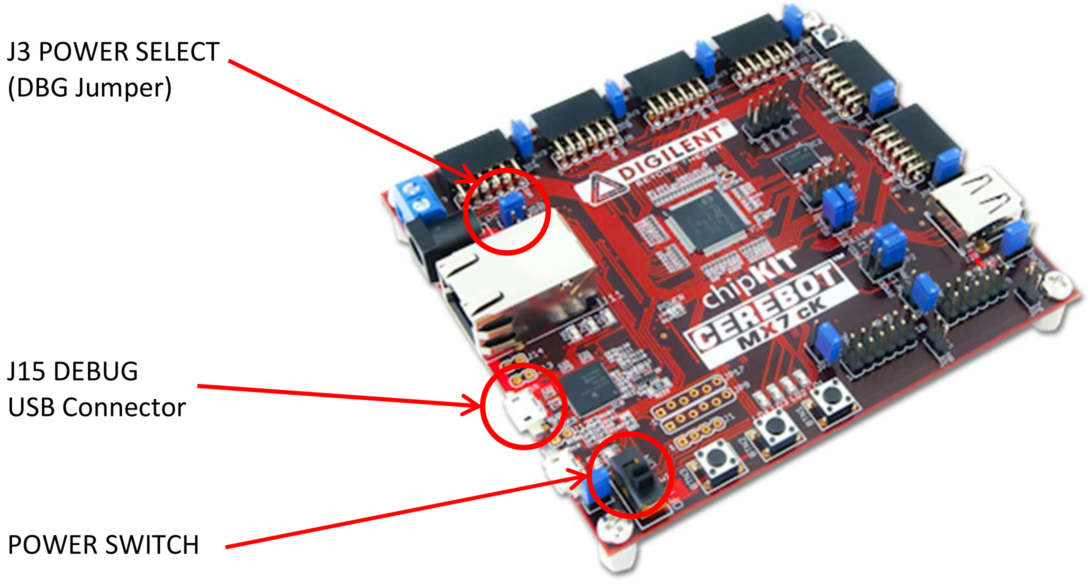


Fig. 1. Power selector and switch (from [1]).

STARTING MPLAB

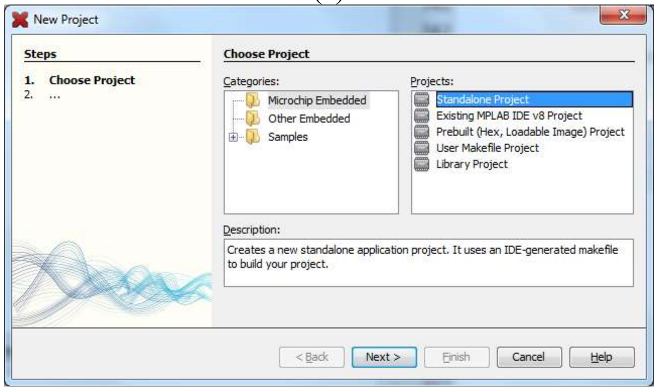
4. On your computer, start MPLAB by clicking on "MPLAB X IDE" of your task bar.

CREATING A NEW PROJECT

5. Create a new project
   a. File->New Project, and make selections as shown in Fig. 2a; click Next
   b. Make selections as shown in Fig. 2b; click Next
   c. Make selection as shown in Fig. 2c (Your board's SN (back of board) will be identified); click Next
   d. Make selection as shown in Fig. 2d (Choose your compiler that you installed; click Next

e. Make selections as shown in Fig. 2e; click Finish.

(a)


New Project

Steps
1. Choose Project
2. ...

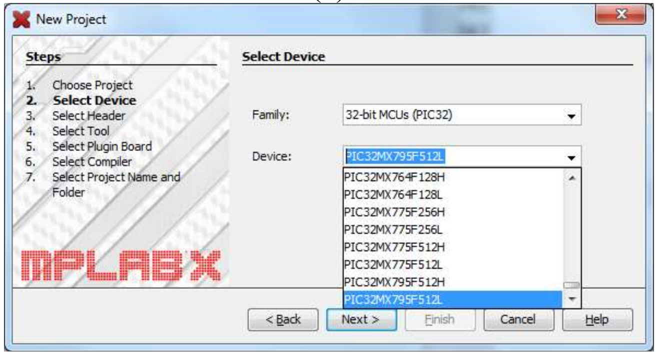
Choose Project

Categories:
- Microchip Embedded
- Other Embedded
- Samples

Projects:
- Standalone Project
- Existing MPLAB IDE v8 Project
- Prebuilt (Hex, Loadable Image) Project
- User Makefile Project
- Library Project

Description:
Creates a new standalone application project. It uses an IDE-generated makefile to build your project.

(b)


New Project

Steps
1. Choose Project
2. Select Device
3. Select Header
4. Select Tool
5. Select Plugin Board
6. Select Compiler
7. Select Project Name and Folder

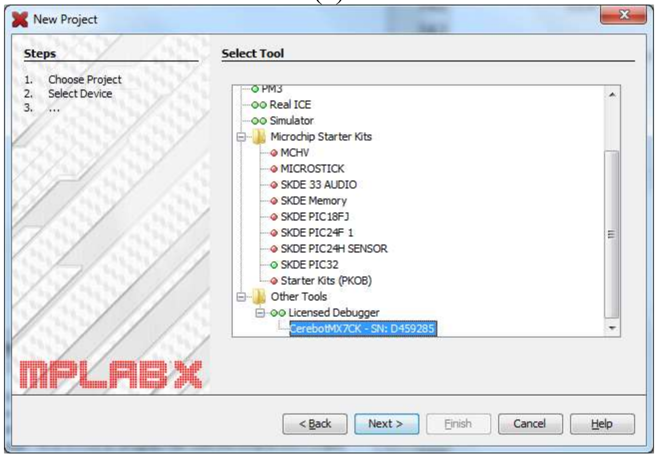
Select Device

Family: 32-bit MCUs (PIC32)

Device: PIC32MX795F512L

- PIC32MX764F128H
- PIC32MX764F128L
- PIC32MX775F256H
- PIC32MX775F256L
- PIC32MX775F512H
- PIC32MX775F512L
- PIC32MX795F512H
- PIC32MX795F512L

(c)

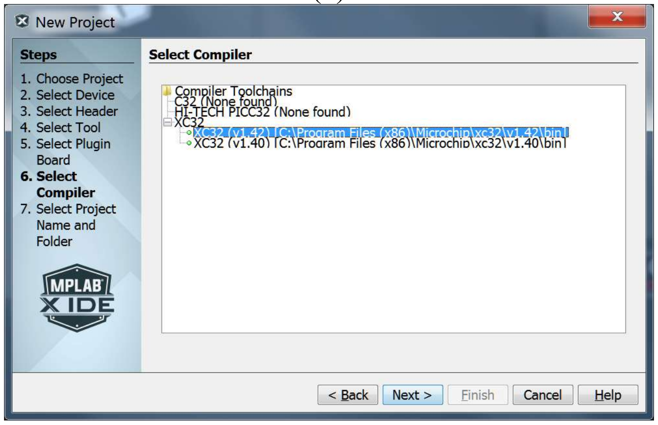

New Project

Steps
1. Choose Project
2. Select Device
3. ...

Select Tool

- PM3
- Real ICE
- Simulator
- Microchip Starter Kits
  - MCHV
  - MICROSTICK
  - SKDE 33 AUDIO
  - SKDE Memory
  - SKDE PIC18FJ
  - SKDE PIC24F 1
  - SKDE PIC24H SENSOR
  - SKDE PIC32
  - Starter Kits (PKOB)
- Other Tools
  - Licensed Debugger
    - CerebotMX7CK - SN: D459285

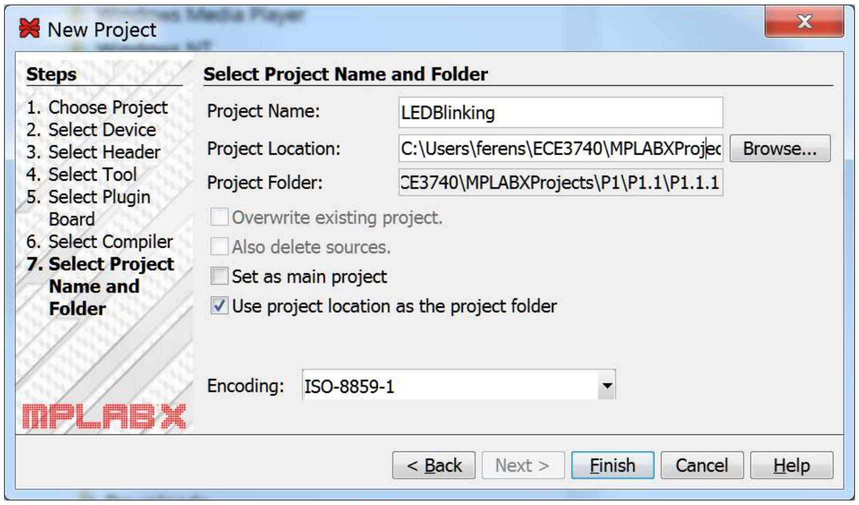
(d)


New Project

Steps
1. Choose Project
2. Select Device
3. Select Header
4. Select Tool
5. Select Plugin Board
6. Select Compiler
7. Select Project Name and Folder

Select Compiler

- Compiler Toolchains
  - C32 (None found)
  - HI-TECH PICC32 (None found)
  - XC32
    - XC32 (v1.42) [C:\Program Files (x86)\Microchip\xc32\v1.42\bin]
    - XC32 (v1.40) [C:\Program Files (x86)\Microchip\xc32\v1.40\bin]


New Project

Steps
1. Choose Project
2. Select Device
3. Select Header
4. Select Tool
5. Select Plugin Board
6. Select Compiler
7. Select Project Name and Folder

Select Project Name and Folder

Project Name: LEDBlinking

Project Location: C:\Users\ferens\ECE3740\MPLABXProjec

Project Folder: CE3740\MPLABXProjects\P1\P1.1\P1.1.1

☐ Overwrite existing project.

☐ Also delete sources.

☐ Set as main project

☑ Use project location as the project folder

Encoding: ISO-8859-1

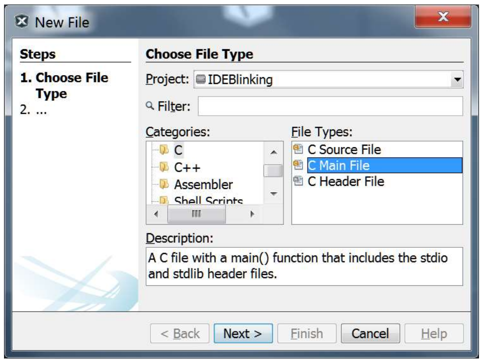
(e)

Fig. 2. Project creation, screens (a) – (e). Project Location and Folder is C:\Users\YourName\ECE3740\MPLABXProjects\P1\P1.1\P1.1.1

CREATING SOURCE FILES

6. Create a new C source file: Right click on "Source Files" and select New->Other->Categories->C->C Main File…"; Make selections as shown in Fig. 3a, and click Next. Make selection as shown in Fig. 3b; Click Finish.

7. Copy the contents of the main.c file (included on umlearn) and paste it to the new file main.c in your project (i.e., replace the current contents).

(a) 

1. Choose File Type
2. ...

Choose File Type

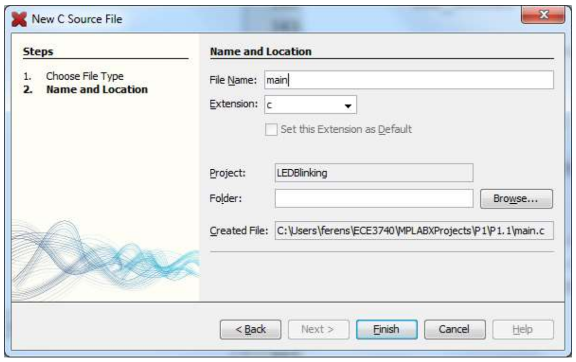
Project: IDEBlinking

Filter:

Categories: C, C++, Assembler, Shell Scripts

File Types: C Source File, C Main File, C Header File

Description: A C file with a main() function that includes the stdio and stdlib header files.

(b) 

1. Choose File Type
2. Name and Location

Name and Location

File Name: main

Extension: c

Set this Extension as Default

Project: LEDBlinking

Folder:

Created File: C:\Users\ferens\ECE3740\MPLABXProjects\P1\P1.1\main.c

Fig. 3. Creating a new C source file.

SELECTING DEBUGGER MODE

For the ability to debug programs, placing breakpoints, stepping through statements, and watching variables, the debugger mode of operation is selected via the **Debug menu** on the top menu bar of MPLAB X IDE. We will be using the Debug menu.

SELECTING PROGRAMMER MODE

For the ability to upload the program into non-volatile memory, and to enable running the program upon switching power on the board, the programmer mode of operation is selected via the **Run menu.**

SELECTING THE DEBUGGER

8. Right-click on your project (IDEBlinking) in the project window and select Properties;
9. In the properties box you should see the S/N of the licensed debugger; check this number with the number located on the back of your board.
10. Click and highlight it and then select Apply, followed by Ok.

COMPILING, PROGRAMMING, RUNNING, AND DEBUGGING

11. Make the project by selecting Debug->Debug Project. This will compile and build the project, download the executable onto the board, and launch the Debugger and run the program. To select these functions individually, select the menu items under Debug->Discrete Debugger operation. Wait for the download to complete successfully. You should see the message "running" and LEDs on the board rotating.
    a. Note that any errors or warnings are displayed in the Output window. Clicking on the error in the Output window navigates to the line in error in the source file.
    b. Note that many warnings of the form "#warning The PLIB functions and macros in this file will be removed from the MPLAB XC32 C/C++ Compiler in future release". To suppress these warnings, first stop debugging by clicking on the Debug->Finish Debugger Session. Then add "_SUPPRESS_PLIB_WARNING" to the preprocessor definitions in the project settings. To do this, right click on LEDBlinking in the project window, and select "Properties". Then select "xc32-gcc" in the Categories window. Then select "Preprocessing and …" in the "Options categories" drop down box. Then click on box to the right of "Preprocessor macros", and then enter the string _SUPPRESS_PLIB_WARNING.
    c. Note also that you may get warnings of the form "Warning: configport argument for OpenADC10 only works for Port B". This warning is just Microchip trying to stop us from using their library with ADC pins on Port A, which we don't do, so we can just ignore this warning. To remove it, add the predefined macro "_DISABLE_OPENADC10_CONFIGPORT_WARNING" to your project's configuration options (use the same method as to suppress the plib warning above). Click on OK.
    d. Now build the project again and notice the warning messages are not displayed (Debug->Discrete Debugger Operation->Build for Debugging).

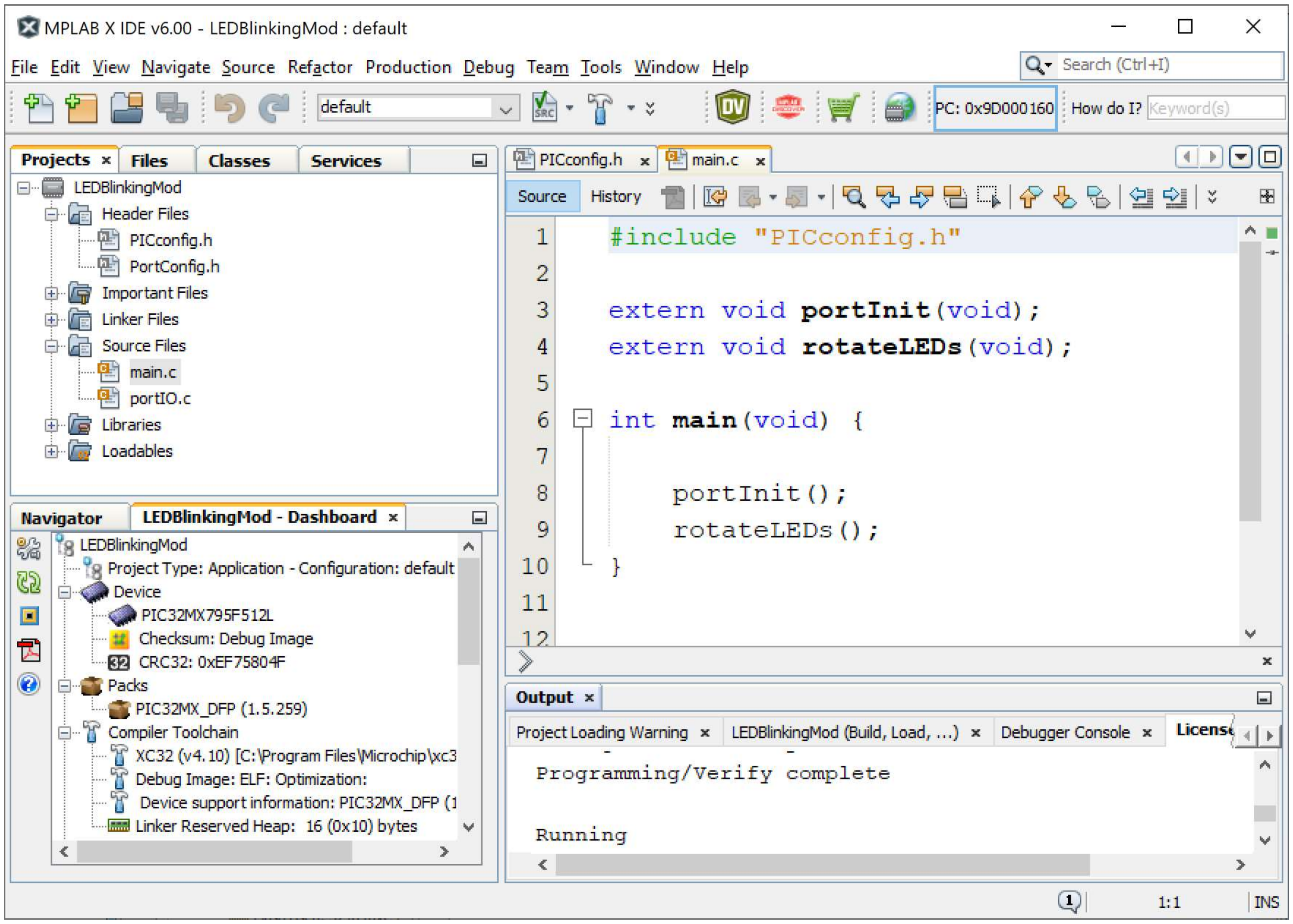
12. You may place breakpoints at strategic locations in the code.
    a. Halt (Finish Debugger Session) the LEDBlinking program. Place a breakpoint at line "PORTSetBits(LED2…);" Note: a breakpoint may be set by right clicking at the desired line and then selecting "New Data Breakpoint", or by single clicking in the left margin (beside the desired line number).
    b. Run the program (Debug->Debug Project). When the program stops at line "PORTSetBits(LED2…);", view the PORTG register. (Note the PORTG register (along with the other registers) can be viewed by selecting Window->PIC Memory Views->Peripherals. Click the magnifying glass icon and search for PORTG. Note that bit 12 of PORTG is clear, due to the statement at line "PORTClearBits(LED1…);".
    c. Step through (over) line "PORTSetBits(LED2…);" by selecting Debug->Step Over (or by pressing F8). Note that PORTG bit 13 becomes set and the LED2 is switched on. Try stepping over other statements and viewing the PORTG register and the LEDs.
    d. Note that stepping over the delay at line "for (i = 0; i < 1000000; i++);" would take a long time, since the debugger stops at each of the 1,000,000 iterations of the for loop. Instead, right click on the desired line to stop, and select "run to cursor."
    e. You may also step into functions. Try selecting Debug->Step Into. Note that the file in which the function you want to step into must be opened before you can step into that function.

CONFIGURATION MANAGEMENT

This section shows how to create new versions of the program without destroying the old version, as well as how to manage different versions of the same project. To demonstrate configuration management, we will create a new version of the LED Blinking project by *refactoring* the code. The main.c file can be organized in an *object oriented* way by creating individual source files to hold the configuration bits, defines, function prototypes, etc. ***All items that pertain to the same concept are gathered and stored in a single file. Different files for different concepts are created for each concept.***

MODULARIZATION (REFACTORING)

13. The first step is to quit MPLAB.
    a. Then create a new folder C:\Users\ferens\ECE3740\MPLABXProjects\P1\P1.1\P1.1.1\v1, and place all the contents of the LED Blinking project \P1.1.1\ into folder \P1.1.1\v1.
    b. Then create another folder C:\Users\ferens\ECE3740\MPLABXProjects\P1\P1.1\P1.1.1\v2. This is where you will create the next version by refactoring the code.
    c. Start MPLAB X, create a new project in the v2 folder. Call the new project "LEDBlinkingMod."
    d. Using the OS, copy the main.c file from the v1 folder to the v2 folder.
    e. Include this v2\main.c file in your new project: Right click on Source Files, select "Add existing item," and then select the main.c file. Note very carefully: this should be the v2\main.c file.
    f. Do the modularization (refactoring) as suggested and shown by Fig. 4. You need to figure out what should be placed in the suggested files.



```
#include "PICconfig.h"

extern void portInit(void);
extern void rotateLEDs(void);

int main(void) {

    portInit();
    rotateLEDs();
}
```

```
LEDBlinkingMod
Project Type: Application - Configuration: default
Device
PIC32MX795F512L
Checksum: Debug Image
CRC32: 0xEF75804F
Packs
PIC32MX_DFP (1.5.259)
Compiler Toolchain
XC32 (v4.10) [C:\Program Files\Microchip\xc3
Debug Image: ELF: Optimization:
Device support information: PIC32MX_DFP (1
Linker Reserved Heap:  16 (0x10) bytes
```

```
Programming/Verify complete

Running
```

Fig. 4. Workspace after modularization.

QUESTIONS

14. Where are the macros TRISGbits.TRISG12, LATGbits.LATG12, etc defined? Hint: press Ctrl and click on the xc.h file in the statement "#include <xc.h>". This should open xc.h. Analyze xc.h and find another include file and open it. Analyze this file and find another include file and open it. In this final file, find the string TRISGBits. Once found, give the memory address of the Port G Tristate register (i.e., TRISGBits).
15. Explain what happens when you accidentally duplicate the configuration bits settings in multiple places? For instance copy the configuration bits settings and paste them into main.c and portio.c. Now build (Debug) the project. What is wrong?
16. Submit a test plan and procedure for your modularized LED Blinking program, and the answers to Questions 14-15; submit to umlearn (TA) for marking.

TCP/IP STACK INSTALLATION AND UPDATE

17. Unzip the Microchip's TCP/IP Stack from the Stack.zip file (double click on Stack.zip) included in ECE3740P1v1.zip, and save the unzipped folder TCPIPStack to folder C:\Users\ferens\ECE3740\MPLABXProjects\TCPIPStackOrg.
18. Copy the TCPIPStack folder to C:\Users\ferens\ECE3740\MPLABXProjects\P1\P1.1\P1.1.2\v1\.
19. Note that we made two copies, the original and the current working version in the v1 folder. We do this because if anything goes wrong with the current working version, we have a backup in TCPIPStackOrg.
20. Quit MPLAB X. Then Start MPLAB X.
21. Open the TCP/IP demo application by Selecting File->Open Project, and browsing to folder "C:\Users\ferens\ECE3740\MPLABXProjects\P1\P1.1\P1.1.2\v1\TCPIPStack\TCPIP\Demo App" and then selecting the XC32-PIC32_ETH_SK_ETH795.mcp project file. (Close any other projects by right clicking their name and selecting "Close".)

22. As explained in [1], all devices on an Ethernet network must have a unique address. This unique address is called a MAC address, and, for this board, it is printed on a sticker attached to the bottom of the board. In addition, all devices on an Ethernet should have a default IP address and a host name. The MAC and IP default addresses are configured in the file TCPIP ETH795.h of the XC32-PIC32_ETH_SK_ETH795.mcp project. We suggest to use the host name of mxXX, and to use the default IP address of 192.168.1.XX, where XX=LSB (Least Significant Byte) of the MAC address of the board you are programming. For example, for the board which has MAC address = 00-18-3E-01-11-79 we make the following changes to the TCPIP ETH795.h file: (note that the "79" is a Hexadecimal number, which is 121 in decimal.)

```
#define MY_DEFAULT_HOST_NAME                 "mx79"

#define MY_DEFAULT_MAC_BYTE1   (0x00)        // MAC address printed on back of board
#define MY_DEFAULT_MAC_BYTE2   (0x18)        //
#define MY_DEFAULT_MAC_BYTE3   (0x3E)        //
#define MY_DEFAULT_MAC_BYTE4   (0x01)        //
#define MY_DEFAULT_MAC_BYTE5   (0x11)        //
#define MY_DEFAULT_MAC_BYTE6   (0x79)        //

#define MY_DEFAULT_IP_ADDR_BYTE1             (192ul)
#define MY_DEFAULT_IP_ADDR_BYTE2             (168ul)
#define MY_DEFAULT_IP_ADDR_BYTE3             (1ul)
#define MY_DEFAULT_IP_ADDR_BYTE4             (0x79ul)

#define MY_DEFAULT_MASK_BYTE1                         (255ul)
#define MY_DEFAULT_MASK_BYTE2                         (255ul)
#define MY_DEFAULT_MASK_BYTE3                         (255ul)
#define MY_DEFAULT_MASK_BYTE4                         (0ul)

#define MY_DEFAULT_GATE_BYTE1                         (192ul)
#define MY_DEFAULT_GATE_BYTE2                         (168ul)
#define MY_DEFAULT_GATE_BYTE3                         (1ul)
#define MY_DEFAULT_GATE_BYTE4                         (121ul)

#define MY_DEFAULT_PRIMARY_DNS_BYTE1         (192ul)
#define MY_DEFAULT_PRIMARY_DNS_BYTE2         (168ul)
#define MY_DEFAULT_PRIMARY_DNS_BYTE3         (1ul)
#define MY_DEFAULT_PRIMARY_DNS_BYTE4         (121ul)
```

23. Setting up a simple network: one host computer and one MX7cK board
    a. In the TCPIP ETH795.h file, select only the ICMP_SERVER, DHCP_SERVER, and ANNOUNCE modules. This can be done by commenting out all other modules except for the ICMP, DHCP_SERVER, and ANNOUNCE modules. In particular, in the "Application Level Module Selection" section of the file TCPIP ETH795.h, comment out all modules except for the following:
    ```
    #define STACK_USE_ICMP_SERVER      // Ping transmission capability
    #define STACK_USE_DHCP_SERVER      // Single host DHCP server
    #define STACK_USE_ANNOUNCE         // Microchip Embedded Ethernet Device Discoverer
    ```
    b. Ensure that the host's TCP-IP v4 configuration is set to "obtain an IP address automatically."
    c. Connect an Ethernet cable between the Host's Ethernet port and the MX7cK's Ethernet port. Note: normally, a "cross-over" Ethernet cable must be used to connect two devices directly. However, modern hosts can detect when another device is connected through a "straight" Ethernet cable.
    d. The current version of this complier XC32 v 1.40 changed a flag for the linker (ld). The previous flag for the linker was "mperhiperal-libs". To correct this, remove the "Additional options". To do this, right click on the project name in the project window, and select "Properties". Then select "xc32-ld" in the Categories window. Then remove the text in the "Additional options" text box.

   e. Build the project, program the board, and run the program: Debug->Debug Project.
   f. Start a command console on the Host: Start->cmd. Ping the MX7cK board by running the ping command with the IP address noted above. For example, if the IP address of the Mx7cK board is 192.168.1.121, then run: “ping 192.168.1.121 in the command window. You should see replies such as “Reply from 192.168.1.121: bytes=32 time<1ms TTL=100.

24. Submit a test plan and procedure for this TCP/IP Stack portion of P1.1.2 to umlearn (TA) for marking.

P1.2: Part 2 of the Project 1 (P1.2) requires that students perform the following:

1. Design C software, which runs on the MX7cK board, to control the LEDs through the Push Button (PB) switches on the MX7cK board. The software should monitor the PB switches. If a switch is held down, then the software should light the corresponding LED. When the switch is released the corresponding LED should be switched off.
2. Gain familiarity with using MPLAB® X to create, edit, build, debug, and run C programs.
3. Practice Configuration Management.
4. Write test plan and procedure documents.

P1.2 measures the following attributes:

ET.3: Engineering use of tools

- Demonstrate the use of MPLAB X IDE for developing C language code for the PIC32MX795F512L on the MX7cK board (compiling, programming, running, and debugging projects)

PA.3: Analysis

- Demonstrate an analysis of given C code to determine how to control the LEDs through the PBs.

DE.3: Design

- Demonstrate creating C code to control the LEDs through the PBs.
- Design a test plan for testing control the LEDs through the PBs.

CS.3: Communication skills

- Demonstrate clear and accurate explanation of the project questions.
- Write test plan and procedure documents

P1.3: Part 3 of the Project 1 (P1.3) requires that students perform the following:

1. Design a console control (telnet) of the LEDs and Push Buttons. Create a new MPLAB project that extends P.1.2. Use the "TCP Template Server" lecture slides as a template of how to build your own TCP server application for control/monitor of the LEDs and the Push Buttons. Use telnet on the Host to connect to your TCP code, as discussed in class. Submit your requirements document, state diagram(s) of your design, and test plan and procedure document to umlearn (TA) for marking.
2. Gain familiarity with using MPLAB® X to create, edit, build, debug, and run C programs.
3. Run networking tools and programs.
4. Practice Configuration Management.
5. Write and submit a test plan and procedure document for P1.3.

P1.1 measures the following attributes:

ET.3: Engineering use of tools
- Demonstrate the use of MPLAB X IDE for developing C language code for the PIC32MX795F512L on the MX7cK board (compiling, programming, running, and debugging projects)
- Demonstrate the use of TCP/IP networking tools, such as ping and telnet.

PA.3: Analysis
- Demonstrate an analysis of given C server code to determine how to create an LED server.

DE.3: Design
- Demonstrate writing C code to implement an LED server on the MX7cK board.
- Design a test plan for testing proper operation of the LED server.

CS.3: Communication skills
- Demonstrate clear and accurate explanation of the project questions and design of the LED server.
- Write test plan and procedure document to test the console control of the LED server.

References

[1] Digilent Inc., "Cerebott MX7cK™ Board Reference Manual," 3 January 2012. [Online]. Available: http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,396,986&Prod=CEREBOT-MX7CK. [Accessed 4 September 2012].

[2] SMSC, "SMSC - Product Information," 23 August 2012. [Online]. Available: http://www.smsc.com/media/Downloads_Public/Data_Sheets/8720a.pdf. [Accessed 5 September 2012].

[3] Digilent Inc., "Cerebott MX7cK™ Board Schematics," 27 August 2011. [Online]. Available: http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,396,986&Prod=CEREBOT-MX7CK. [Accessed 5 September 2012].