Embedded Systems Course

Laboratory Practical 2: MPLAB SIM & PICkit3

TEAM:	Two
DURATION:	3 hours
DOC REF:	PRAC2

1. Introduction

This practical involves writing a simple program for the PIC microcontroller and getting it to run on the PICkit3 evaluation kit**.**

Make sure that your Pitkit3 has both the programmer device, its red USB cable, and the PICkit board.

Please read the following points first before continuing:

Important: connect the PICkit3 development board to the PICkit3 programmer first, BEFORE connecting the USB cable to your computer's USB port. You can happily *remove* the PIKkit3 programmer from the development board while the programmer is connected to the computer (but this will obviously cause the board to shut down unless it has a supplementary power source).

Make sure that the resources for the PICkit3 have been installed on your workstation. Go to start → All Programs → Microchip → MPLAB IDE v8.36.

IMPORTANT: HAND IN code file for this PRAC – see page 7

Some useful resources:

- Information about the PICKit3 from microchip: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en538340
- The academically licensed lite version of the MPLAB C compiler is on forge.ee in the \EEE3074W\Software directory see MPLAB-C18-Lite-v3_36.zip. You could try using the CCS compiler (which comes with the DVD) as an alternative.

1.1. Rules

The rules for this practical are as follows:

- Arrive at the lab at any time during the assigned practical times. If you have booked a seat, you have preference to getting a CSB337 and workstation. You are *encouraged* to arrive when the session starts so that you will have enough time to complete the practical assignment. You can also work on this prac in your own time outside of the assigned lab sessions, however help during such times is not guaranteed.
- No PASS sheet is required for this prac; you just need to submit your code using the Vula assignment for this practical.
- You may be asked from time to time to demonstrate or explain aspects of what you have done, to ensure that both partners in groups of two are involved, and working on, the assignment. If you work in the assigned lab slot, this may be done during that slot; otherwise you may be called on at a later stage to demonstrate aspects of the prac if the lecturer requires it.
- You may work in teams of two, or individually.
- Use class notes and textbooks if you want to. You can also make use of the web; but if you do so, you are expected to cite references for material used (except for material provided on connect / the textbook). Such citations should be in the code, report, or PASS (whichever is applicable).

1.2. Code Handin

Either zip or tar and gzip your Prac2 directory. If you worked on only one C file, you can submit just that file (make sure it includes your name and, if applicable, your lab partners name).

2. Activities

This section explains what you need to do in order to complete this prac.

2.1. Connecting up and creating a project

Start by connecting up the PICKit3 to the USB port using the red USB cable.

Your computer may indicate it found new hardware and is installing drivers for the PICKit3.

It should show that the PICKit3 device was installed.

If there's a problem adding the device, you may need to call the tutor over to give you administrator access to add the driver.

Let's proceed with creating a new project and a simple program…

1. Start by creating a project directory somewhere on your drive. You could call it something like M:\EEE3074W\Prac2.
2. Loading the MPLab IDE (see Figure 1 on right).

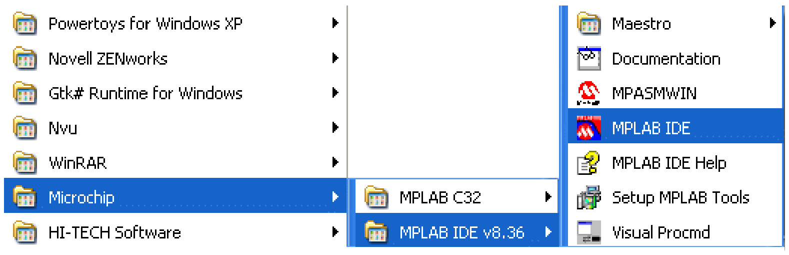
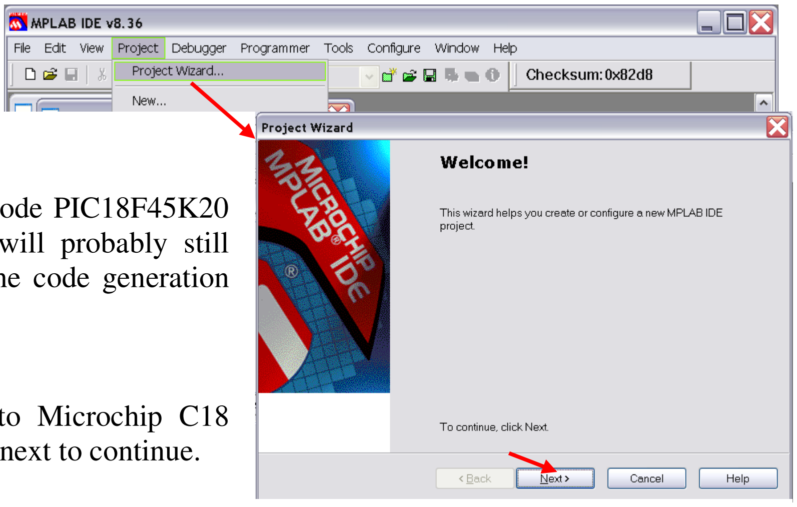


Figure 1: Starting MPLab IDE

3. From the "Project…" menu, select "Project Wizard". (See Figure 2)
4. Click Next to continue…

5. Select the microcontroller product code PIC18F45K20 (note that the default PIC18F420 will probably still work, but it would limit some of the code generation options). Click Next to continue.

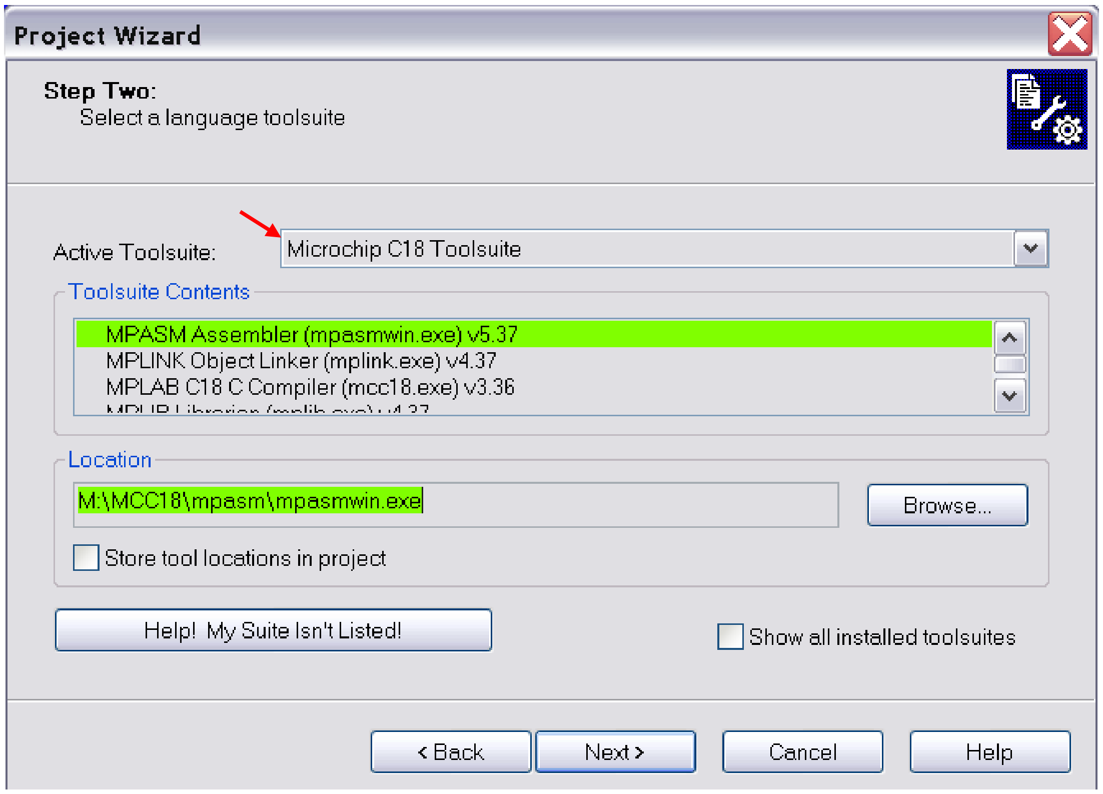
6. Change the *active toolset* to use to Microchip C18 Toolsuite. See Figure 3 below. Click next to continue.



This wizard helps you create or configure a new MPLAB IDE project.

To continue, click Next.

Figure 2: Starting Project Wizard



Active Toolsuite: Microchip C18 Toolsuite

MPASM Assembler (mpasmwin.exe) v5.37
MPLINK Object Linker (mplink.exe) v4.37
MPLAB C18 C Compiler (mcc18.exe) v3.36
MPLIB Librarian (mplib.exe) v4.37

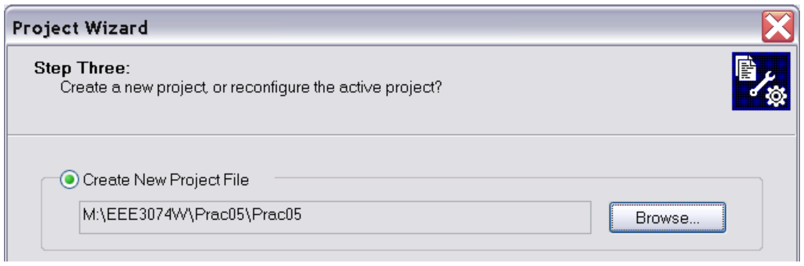
M:\MCC18\mpasm\mpasmwin.exe

☐ Store tool locations in project

☐ Show all installed toolsuites

Figure 3: Change the active toolset to CCS C Computer.

7. Specify a new project folder (i.e., select the folder that you created in step 1). Click Next to proceed.


Project Wizard

Step Three:
Create a new project, or reconfigure the active project?

(●) Create New Project File
M:\EEE3074W\Prac05\Prac05

Figure 4: Specify project

8. Don’t add any existing files to the project at this stage. Just press Next to skip adding files.
9. The final page of the wizard shows a summary of the settings. Press Finish to create the project and to save the new workspace.

2.2. Simple simulated program

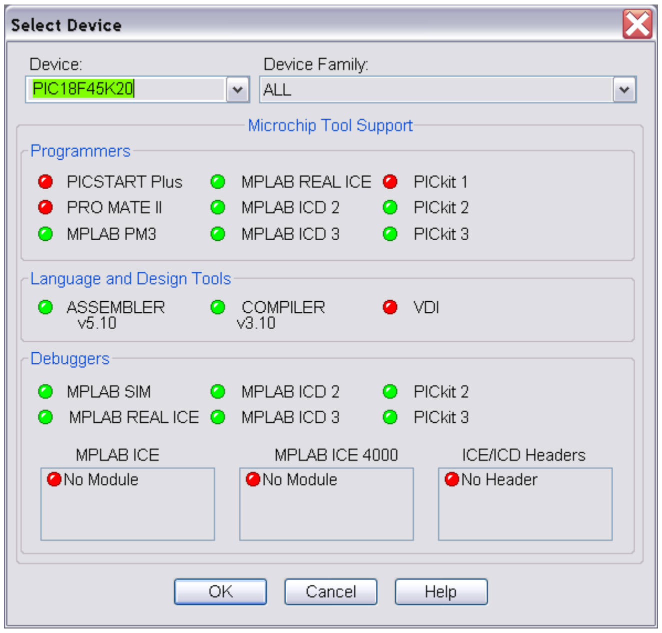
Now it’s time to create a barebones C program. Do this by creating a new file (i.e., File->New) and write the obvious starting point (you could copy and paste the text below). Please put your names and student numbers in the comment at the top.

```
/*
Prac2
-----
This is a simple program for the PIC18.
Authors:   <Your   Names   and
Student Numbers>
*/

#include <stdio.h>

int debug_stage = 0;
 // indicates how far this are
int x; // a global

void main (void)
{
  debug_stage = 1;
  printf("HELLO!\n\r");
  x = 0;
  debug_stage = 2;
  while (1) x = x + 1;
}
```


Select Device

Device: PIC18F45K20
Device Family: ALL

Microchip Tool Support

Programmers

🔴 PICSTART Plus	🟢 MPLAB REAL ICE	🔴 PICkit 1
🔴 PRO MATE II	🟢 MPLAB ICD 2	🟢 PICkit 2
🟢 MPLAB PM3	🟢 MPLAB ICD 3	🟢 PICkit 3

Language and Design Tools

🟢 ASSEMBLER v5.10	🟢 COMPILER v3.10	🔴 VDI

Debuggers

🟢 MPLAB SIM	🟢 MPLAB ICD 2	🟢 PICkit 2
🟢 MPLAB REAL ICE	🟢 MPLAB ICD 3	🟢 PICkit 3

MPLAB ICE	MPLAB ICE 4000	ICE/ICD Headers
🔴 No Module	🔴 No Module	🔴 No Header

Figure 5: Select Device dialog should look like this.

When you save the file, click the checkbox at the bottom of the save prompt to add the new file to your project. (If you forget to add the file to the project you can use Project → Add Files to Project).

Wait! You’re not quite ready to build yet… you need to first set the configuration is correctly specified. Do the check below for the micro we’re using as follows:

1. Select menu option Configure → Select Device. It should all be fine, but make sure that it looks similar to Figure 5.
2. Select menu option Configure → Configuration Bits. Ensure that checkbox for Configuration Bits set in code (i.e., it must be unchecked).
3. Disable the Watch Dog timer (at address 30003 bit value 1F). To do this, click the string “Watchdog

Timer”; click the string Enabled (you may need to use the scrollbar to scroll right to these options); this causes a list of two options to pop up. Choose the Disabled option. You can ignore the timeout setting for the watchdog.

4. Just close the configuration bits dialog and it will save the changes.
5. Save the workspace.
6. Before trying to get the physical hardware to work, it’s a good idea to try the simulator, which makes it much easier to test and diagnose regular C problems (once it comes to using peripherals properly, you don’t have much option besides using the physical chip). Select the simulator using Debugger → Select Tool → MPLAB Sim as shown in Figure 6.

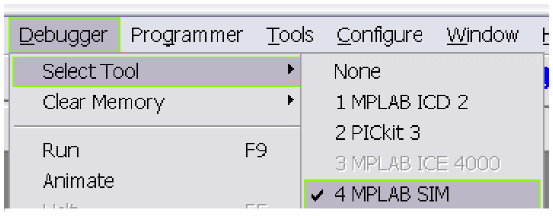
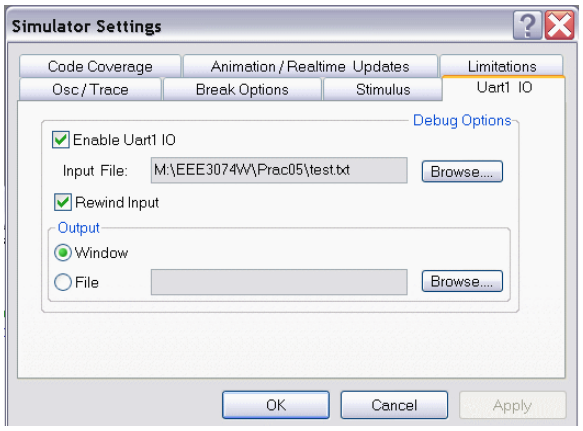


Figure 6: Select simulator to start with.

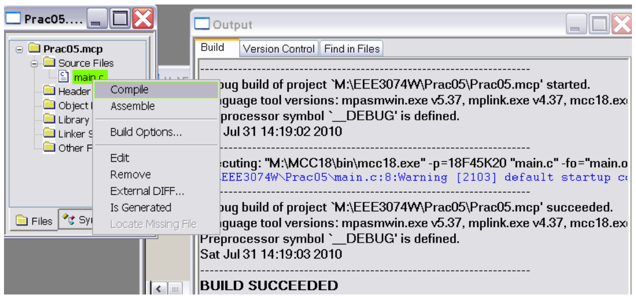
7. You need to change the debugger settings to get your program to run. Select Debugger → Settings. Select the tab UART1 IO, and set an input file and set output to appear on a window. See Figure 7. Click OK to save the changes.



✓ Enable Uart1 IO
Input File: M:\EEE3074W\Prac05\test.txt
✓ Rewind Input
Output: [● Window] ○ File

Figure 7: Debugger settings for MPLAB Sim.

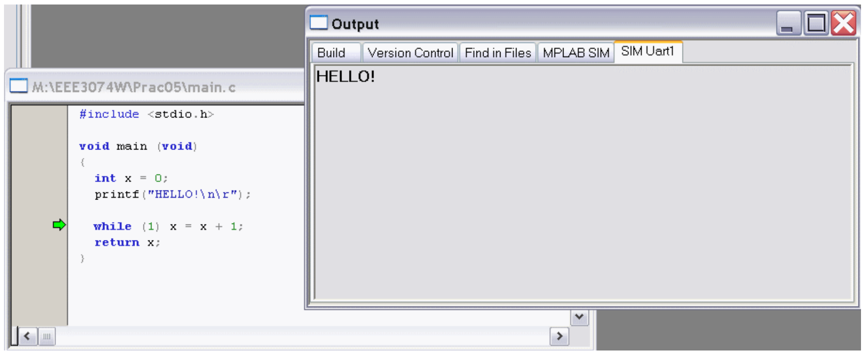
8. Now compile the new code file you created. Do this by right-clicking the main.c file as shown in Figure 8. This should involve the compiler and the output dialog should show eventually show a success message as illustrated.
9. If the file compiled successfully, make the project. Select Project → Make. Again, if it all works successfully a BUILD SUCCESS message should be visible.



```
----------------------------------------------------------------------
bug build of project `M:\EEE3074W\Prac05\Prac05.mcp' started.
guage tool versions: mpasmwin.exe v5.37, mplink.exe v4.37, mcc18.ex
processor symbol `__DEBUG' is defined.
 Jul 31 14:19:02 2010
----------------------------------------------------------------------
cuting: "M:\MCC18\bin\mcc18.exe" -p=18F45K20 "main.c" -fo="main.o
EEE3074W\Prac05\main.c:8:Warning [2103] default startup c
----------------------------------------------------------------------
bug build of project `M:\EEE3074W\Prac05\Prac05.mcp' succeeded.
guage tool versions: mpasmwin.exe v5.37, mplink.exe v4.37, mcc18.ex
Preprocessor symbol `__DEBUG' is defined.
Sat Jul 31 14:19:03 2010
----------------------------------------------------------------------
BUILD SUCCEEDED
```

Figure 8: Compile the file.

10. Now you should be ready to run the program. Press the blue RUN button on the toolbar. As you may expect, you’ll get some simple text output as shown in Figure 9.
11. Press PAUSE (the parallel blue bars on the toolbar at the top of the window) to pause the program. This will cause the editor window to shown a green arrow indicating which line the program stopped on.



```c
#include <stdio.h>

void main (void)
{
  int x = 0;
  printf("HELLO!\n\r");

  while (1) x = x + 1;
  return x;
}
```

```
HELLO!
```

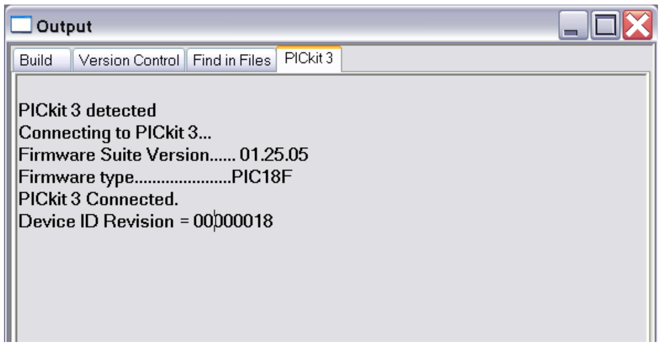
Figure 9: Running the program in the simulator.

12. Take a look at the x and debug_stage global variables in the program. You can do this using View → Watch. In the Watch window, next to Add Symbol, type in x and then press the Add Symbol button. The variable x will now be added, and its value shown (as a 16-bit hex value). Note that debug_stage can be used instead of printf, because printf is ignored when running on the actual hardware (see next page).
13. End the debugging session by pressing the Reset button in the toolbar (it’s the second last button, next to the red B).
14. Remove the entries in the watch window.

2.3. Flashing LEDs program

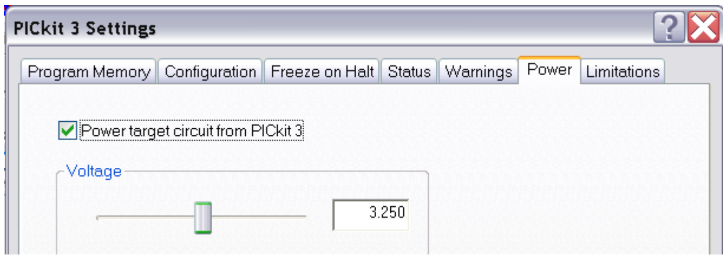
Now, let's run it on the hardware…

1. Change the debugger to use to PICKit3. In MPLAB use Debugger → Select Tool and choose PICKit3.
2. It should indicate that the device was found successfully. It may need new firmware may need to be added. If so, allow this to be done (i.e., click OK). See Figure 10.
3. An error message may indicate that "you must connect to a target device", or that the target is not found, or that it does not have power.
4. Go into the Debugger → Settings. Change to the Power tab. Select the checkbox "Power target circuit from PICKit3". Select 3.25V. Press Apply. Figure 11 shows what the screen should look like. If you need to do this step. Then go to the "Status" tab and select "Refresh Voltage"; it should now show between 3.1V and 3.5V.
5. Once the debugger has been connected properly, you should see a response similar to Figure 10.
6. Push the *Erase Flash Device* button to clear the flash memory. (see Figure 12).
7. Next press the *Program* flash button to write the program. At this stage, you may be asked to change configuration settings; select OK to do so. You should get the message "Programming... Programming/Verify complete" shown in the output window. You may need to select Debugger → Reconnect and then Debugger → Program again if it's the first time you're using the debugger for this run of MPLAB.
8. Press the blue RUN button.
9. Your program will start, but you won't see anything. Click on Pause. It will just to the while(1) line. Go to the watch window and add debug_stage.



```
PICkit 3 detected
Connecting to PICkit 3...
Firmware Suite Version...... 01.25.05
Firmware type.....................PIC18F
PICkit 3 Connected.
Device ID Revision = 00000018
```

Figure 10: Response to selecting PICKit3 as debugger



Power target circuit from PICkit 3 [✓]

Voltage: 3.250

Figure 11: Power setup.

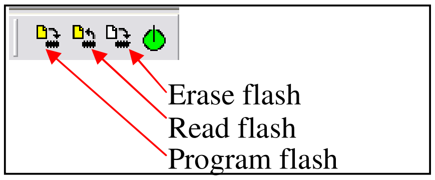


Figure 12: Flash programming buttons

Now, that you know how to use a simple program and the debugger, let's get the LEDs to flash…

1. Close your workspace (but you don't need to exit MPLab).
2. Download the provided program Hello1.zip. Extract it into your working directory.
3. Open the Workspace called Hello1.mcw.

   You may be asked if you want to reset the programmer and upload the different program. Press OK.

   In may ask you to specify where the compiler and linker is located. It should shown two choices; select the choices on C:\ drive (assuming that your computer has the compilers installed on C:\).

   You may get *error messages* about things not being found. Fix this by right-clicking on the file concerned (it shown 'file missing' in the files window) and then select "locate missing file". On the dialog that pops up, find the file – it is probably either in the folder you just extracted, or on the C: drive in the C:\MCC18\lib directory or M:\MCC18\h or one of the others (you could use the Windows Find tool to locate the file).

Note that this workspace is configured to use the PICkit3 in *programmer mode* instead of *debug mode* (leave it that way – it seems to give fewer errors; but it does work in debug mode as well).

A copy of the main.c file used in the Hello1 workspace is shown on page 8 in case you want to recreate the workspace and go through the setup wizard again.

4. Now Rebuild the project (to make sure it will build for you).
5. Again it may complain of missing files. You'll need to change the library paths. This can be found in Project → Built Options → Project. Select the Directories tab. Select the "Library Search Path" drop-down option. Click on directory shown (probably M:\MCC18\lib) and change it to C:\MCC18\lib or wherever your MCC18 tools are installed. Try Rebuild all again if you changed the library paths.
6. With luck it should now compile successfully.
7. Now run the program on the target hardware by selecting Programmer → Program.
8. If it's all worked successfully, you should see LED6 and LED7 alternating between on and off.

2.4. Final task – making a change to Hello1

Now for the final task, where you do a change to main.c. Use the Hello1 workspace that you probably still have open.

Start by backing up your Hello1 project. Do this by making a zip file archive of the entire Hello1 folder. Call it something like Hello1_backup1.zip. This can allow you to revert to an earlier working version quickly.

All you need to do is modify the main.c file to achieve the following requirements:

- Change the Hello1 program so that LEDs 0 to 6 count up in binary from 1 to 127 (i.e., all 6 LEDs on) and then counts back down to 0 (all LEDs off), and then repeats. The cycle should repeat about every half minute.

All you need to do is submit main.c file using Vula. Note that if you've for any reason decided to make any changes to other project settings you can zip the entire Hello1 folder and hand it in.

TOTAL TIME ESTIMATE: 180 minutes

A. Appendix

```
//******************************************************************************
// This is an adapted version of Microchip's PICKit3 Lesson 1
// titled "PIC18F46K20 Starter Kit Lesson 1 - Hello LED".
// *******************************************************************
// This file is used in EEE3074W Prac2.
// Please refer to the Prac2 tutorial for things to do.
// *******************************************************************
// The baseline version of this program swaps between turning on
// LED 7 and LED 6 on/off on the demo board.
// *******************************************************************

/** C O N F I G U R A T I O N   B I T S *****************************/
#pragma config FOSC = INTIO67
#pragma config WDTEN = OFF, LVP = OFF, MCLRE = OFF

/** I N C L U D E S **************************************************/
#include "p18f45k20.h"

/** DECLARATIONS AND FUNCTIONS ***************************************/

void wait (void)
// Cause a short delay
{
  int i,j = 0;
  for (i = 0; i<5000; i++) j++;
}

void main (void)
// The entry point to the program
{
    // Configure tristate PIO D
    TRISD = 0b00000001;      // PORTD bit 1 to 7 to as output (0); bit 0 as input (1)

      LATDbits.LATD1 = 1;          // Set LAT registers to turn on 2nd LED
    wait();
    LATDbits.LATD1 = 0;      // Set LAT registers to turn off 2nd LED

    // swap between LED6 and LED7
      while (1) {
      wait();
      LATDbits.LATD7 = 0;          // Clear LAT register bit 7 to turn off LED
      LATDbits.LATD6 = 1;          // Set LAT registers to turn on LEDs
      wait();
      LATDbits.LATD7 = 1;          // Clear LAT register bit 7 to turn on LED
      LATDbits.LATD6 = 0;          // Set LAT registers to turn on LEDs
      }
}
```