H63ECH Workshop Introduction

This document describes the steps you should take to get started with the workshop sessions and to familiarize yourself with the PIC16F887 board you will receive.

The board

To receive your board, please go to the technicians room on the 8th floor. It may be worthwhile starting the software section below first – it may take a while to sign out boards to all students.

You should have a “44-pin demo board” with PICKIT 2 programmer. You should also have a USB cable. The manual for the board, “44-pin Demo Board User Guide.pdf”, is included in the LAB_MPLAB_X.zip file on moodle (see below).

The software

You will be using the MPLAB X software from Microchip during the workshop sessions. This can be accessed through the start menu at **Microchip\MPLAB X IDE v3.61**

If you wish to install the same software on your own computer (I would recommend doing this), you will need to download both MPLABX and the XC8 compiler. The download links are

http://www.microchip.com/mplabx   http://www.microchip.com/compilers

Getting Started

The aim of this first workshop session is threefold: to distribute a microcontroller board to you, to ensure that the board is working and to give you some familiarity with the MPLAB IDE. To carry out steps 2 and 3, I am suggesting the use of one of the lessons that Microchip provide with the demo board. These can be found on moodle in the **LAB_MPLAB_X.zip** file.

Testing the board

Plug your board into a USB port on your computer. Extract the zip file to your Z: drive, then start MPLAB X. When it runs it is likely that a dialog box “Invalid jdkhome specified” will appear – select “Yes” to use the default version.

Use the menu **File/Open Project**, then navigate to your Z: drive and select the project directory **A02_Blink**. In the project list in MPLAB, right click on A02_Blink and choose “Set as Main Project”. This means this project is the one that will be built, used for programming and debugging the board using the toolbar buttons at the top of the window. It is not strictly necessary here because you only have a single project open.

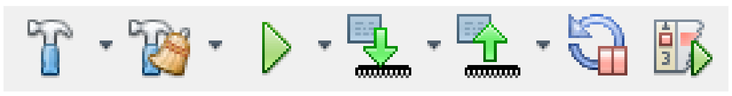
To build the project code into a HEX file ready for programming, click the hammer icon at the top of the window. If your build fails with the error

```
make[1]: *** No rule to make target '.build-conf'.  Stop.
make: *** [.build-impl] Error 2
BUILD FAILED (exit value 2, total time: 3s)
```

…then you need to configure the build tools. To do this, right click on your project and select

Properties from the menu. You should ensure that in the **Compiler Toolchain** box, the selected **mpasm** entry has a green light next to it (or if you are using the C compiler, that the XC8 entry has a green light next to it).

Once your build has completed, you can program the board using the icon that looks like an arrow pointing to a chip.



You may be asked to choose the programmer – look for the PICkit2 in the list. All being well, you should now see a flashing LED on your board. If not, check with me or a demonstrator. Once you have this working you are free to either look at more of the example projects, or to start work on the coursework tasks.

Appendix: Demo board hardware specifics and how they affect firmware development

Any firmware should be written bearing in mind the pin connections on the board being used (or more properly, for the functionality that you wish to implement which will then guide the hardware design you are creating – we are using a premade development board though, so do not have this luxury/requirement). The following connections are present on the demo board you have received:

- No external oscillator; it is necessary to select the internal 4 MHz oscillator using the appropriate configuration word
- The LEDs are connected to **PORTD**. To use them, any firmware needs to configure the pins of this port as outputs (**TRISD**). To display a value you simply need to write to this port (e.g. `movwf PORTD`) or manipulate its bits to switch an LED on (e.g. `bsf PORTD, RD1`) or off (e.g. `bcf PORTD, RD2`). Please make sure that you have JP1 in place to operate the LEDs.
- The button SW1 is connected to the pin **RB0** of **PORTB**, and a pullup resistor is also connected to this pin. This means that if the button is released the pin reads 1. If it is pressed, it reads 0.

You can skip the next instruction if the button is pressed using the following statement:

```
        BTFSC    PORTB, RB0
```

Alternatively skipping the following instruction if the button is released is achieved by

```
        BTFSS    PORTB, RB0
```

Please always double check that you use the required condition – operating **`BTFSC/BTFSS`** correctly can be quite confusing.

To use any mechanical switches reliably, you need to provide some means for debouncing (this is discussed briefly in lesson 6 and we will discuss the issue in a lecture).

The on board potentiometer is connected to the pin **`RA0`** which needs to be configured as an input of the ADC that is available on the microcontroller. We are not going to discuss the ADC operation in detail in class, and I suggest you use the ADC code as it is in the lessons. I provide subroutines to operate the ADC without the need to examine all the possible options for its operation, however you may need to modify or make your own version of the ADC code for use with the final task.