INSTRUCTIONS FOR SETTING UP ROACH SOFTWARE IN MPLAB-X

These instructions are provided to help you get started on the first lab to get up and running in MPLAB-X using our Uno32 boards with the roaches. Note that if you want to set up your own environment on your laptop, you will need to make some changes (contact the TAs or Instructor for further instructions).

1. First select the MPLAB X IDE and open it.

2. Create a new project. A project is a collection of files and settings that you need to build an application (such as the code for a mobile robot’s microcontroller).
   a. Click on create new project (the icon with the green + sign in the toolbar). This will open the new project dialog window.
   b. Now we choose the type of project. Select a Microchip Embedded -> Standalone Project. Click “Next.”
c. In this screen, we select the microcontroller that we want to run our code on. We want the PIC32MX320F128H, which is the chip that the Uno32 boards use. Click “Next.”

d. This screen, we choose a hardware tool for loading code onto the microcontroller. Select the PICKit3, even though we won’t be using it (your PIC32 has firmware for loading code directly from the USB serial connection). It should have a green light next to it. If it doesn’t you have the wrong PIC32 chosen in the previous screen.

e. Click next.

f. In this screen, we choose a compiler. Select XC32 (v1.20), which is the compiler provided by the microcontroller’s manufacturer. Click “Next.”
g. Give your project a name that aptly describes what you are doing (such as Lab0_RoachTest) and choose a location for the project. We strongly recommend that you put the project in your Dropbox folder, as this will prevent you from losing your work once you log out. Click “Finish.”

h. You should see your empty project in the “Projects.”

i. If you entered something incorrectly, or if you want to change something, you can do so in the Project Properties window, accessed from the toolbar dropdown menu:
3. You now should have a new project. This project is empty, so you need to populate it with some code. Let’s start with a main file.
   
a. In the “Projects” window, right click on Source Files -> New -> Other...

b. Choose “Microchip Embedded” and XC16 main16C file (We’re not using the XC16 compiler, but this will generate the correct file anyway). Click “Next.”
c. Name it descriptively (like RoachTestHarness, BeaconDetector, LineFollowerFSM) and click finish. The file should load into the editor, and appear in the project window. Do the same to create any headers if you need them (for lab0 you won’t need to, but might want to).

4. You’ll also want to add libraries – code that other people have written, and you want to make use of. Every project you do should have the BOARD and serial libraries, and lab0 will use the roach library. Let’s add those now.
   a. First, we need to tell the compiler where to look for the include files (the ones that end in .h) for the libraries we want. Go to the “Project Properties” window using the drop-down menu in the toolbar:

   ![Project Properties screenshot]

   b. Choose the XC32-gcc options and in the Options Categories, choose Preprocessing and messages. Click on the include directories. All of the libraries we provide in this course are in C:\CMPE118, so add . and then “C:\CMPE118\include” to the “Include directories” path and press OK.

   ![Include directories screenshot]

   c. Now the compiler knows about the CMPE118 directory, but we also need to tell MPLABX which specific files we want. Do this by right clicking on the header files folder and selecting “add existing item.”
d. Select the files you want. For lab 0, select BOARD.h, searial.h, pwm.h, and roach.h files.

NOTE: This is different then the below picture. IMPORTANT: If the files you’re adding are in a folder you don’t expect to move (such as the CMPE118 directory), make sure that “Store Path As” is set to “Absolute”. If the files you’re adding are expected to move (like if they’re in your group Dropbox), make sure you set “Store Path As” to “Relative.”

Click select.

e. You should now see the files you added in the “Project” window.

5. One last step before we can compile:

In order for the code to work with the ds30 Bootloader, step outside of MPLABX for a moment and copy the file “procdefs.ld” from the CMPE118 directory into your own project directory. You do not need to add it to the project, just copy it into the project directory.
At this point you can compile the project with no errors (use the small hammer up on toolbar to compile. We’ll get to this later.

6. Make sure that your main file has the following #includes:

   a. #include <xc.h>
   b. #include <stdio.h>
   c. #include "serial.h"
   d. #include "roach.h"

7. Now we’ll add the source files for the CMPE118 Roach libraries so that you can easily play with them. Right click on source files and again, Add Existing Item… and choose from the src folder: BOARD.c, AD.c, pwm.c, roach.c, and serial.c again make sure the Store path as radio button is Absolute:

![Select Item](image)

8. In your main function, make sure to call both BOARD_Init() and Roach_Init() to enable all of the subsystems. Here you can use Control-space to autocomplete and show you documentation of the function. You main.c file should now look something like the following image (this shows the documentation feature for the Roach_Init() function, and is available for most functions.)

![Main Function](image)
9. You are now ready to begin playing with the roach. Take a look at roach.h to see what functions are already defined for you, same with timers. For the serial port, use printf and getchar for communication to the USB port. Note that you can control-click on the header file to open it.

10. Make main look like this:

```c
int main(void) {
    BOARD_Init();
    Roach_Init();
    printf("nHello World!");
```

```c
    while (1) {
        printf("nLight level: %4d Bumpers: 0x%x", LightLevel(), ReadBumpers());
        while (!IsTransmitEmpty()); // bad, this is blocking code
    }
    return 0;
}
```

11. And now let’s compile by clicking on the small hammer, and load it onto a roach ( ):
You have now managed to create a new project, have it do something useful, and set up the IDE so that everything matches the processor and environment we are going to use in the class. Note that you could have also selected the Simulator, rather than the PICkit3, which would allow you to run the simulator (useful for debugging code). We’ll cover that at a later date in the class; this is just to get you up and running.

The ChipKit Uno32 boards from Digilent we use are nominally Arduino compatible boards. However, we don’t use Arduino around here, rather, we program in straight C. To do this, we’ve reflashed these boards with a custom bootloader which will load what we compile in the IDE over the USB port.

12. To do this, we’ll launch our ds30Loader, double click on the desktop icon for ds30 Loader GUI, and plug in your roach usb into the white usb extender cable. You should see a red LED light up on the roach when plugged in. There are a few settings we’ll need to change on the ds30 to match what we need:

13. First, select your hex file by choosing the … next the the Hex-file and navigating to your project directory, choose \dist -> default -> production and select the .hex file. Next change the device to PIC32MX from the pulldown menu, and it should auto-select 320F128H. Change the baud rate to 115200, and select the USB Serial Port (it will only show the active ones), and check the Write Flash box:
14. In the menu, choose View -> Advanced Mode, and select the Reset tab, make sure the radio button for **DTR** is selected, and **Reset time[ms]** is set to 1:

15. Lastly, in the terminal tab, make sure the baud rate is at 115,200. and the Switch after write box is checked (this will turn on the serial port after flashing):

16. Verify that your setting are correct; check for the boot loader by clicking on the **Check for bl** button. You should get a response back:

```
Resetting device...ok
Searching for bl.
Found PIC32MX320F128H fw ver. 5.0.2
```
17. Now you can program your roach by clicking on the Write button, and it will immediately switch to the serial window as soon as it finished flashing the device.

18. Modify your code, reflash the device by closing the serial port and hitting write once again.

19. Good luck and happy programming.