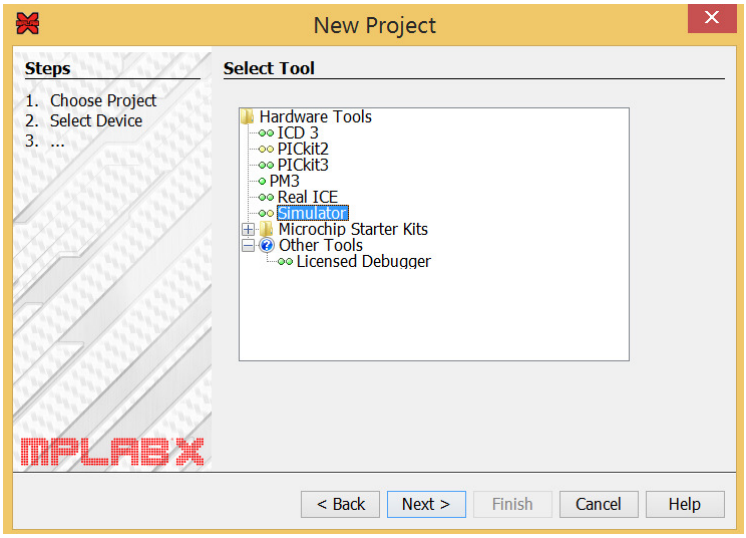
Basic MPLABX simulation

Introduction

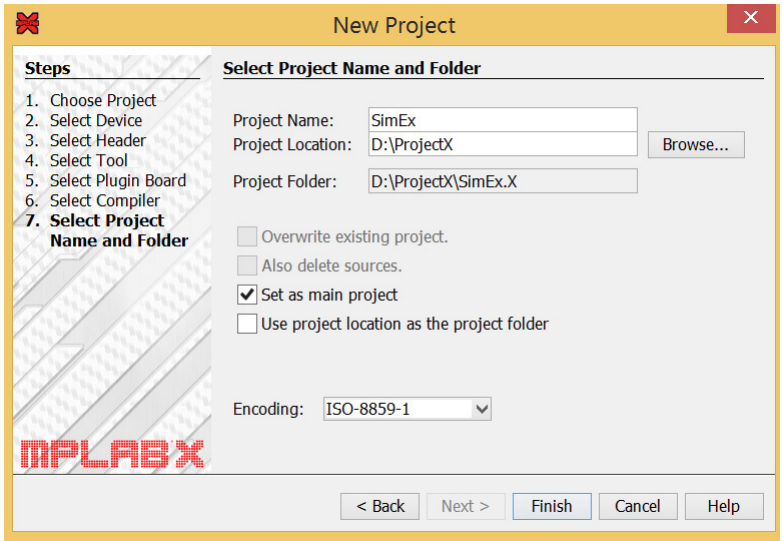
1. To check if your PIC18 program is written correctly, the best way is to test it with some hardware – PIC18 board + IO circuits. What if you only have the MPLABX installed on your PC, but the hardware is not available or not ready? Well, you can perform simulation. This guide shows you just the basics. Once you have learnt the basics, you can watch youtube videos to learn more, learn online or explore the MPLABX simulation / debugging features yourself.

Creating a project & writing the C program

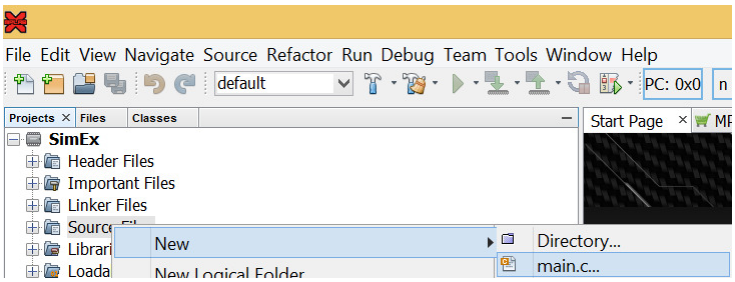
2. Launch "MPLAB-X" and create a project as usual. However, select "Simulator" in the "Select Tool" step:



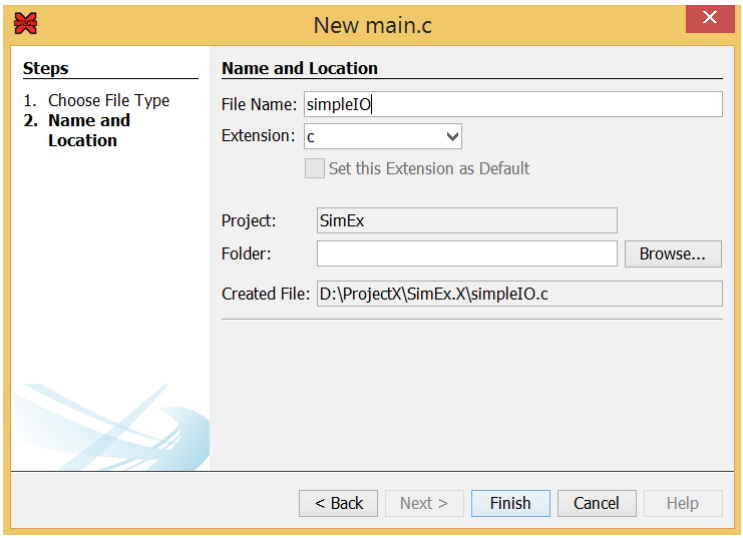
3. Name the project as "SimEx" (short for Simulation Exercise) in the "Select Project Name and Folder" step:



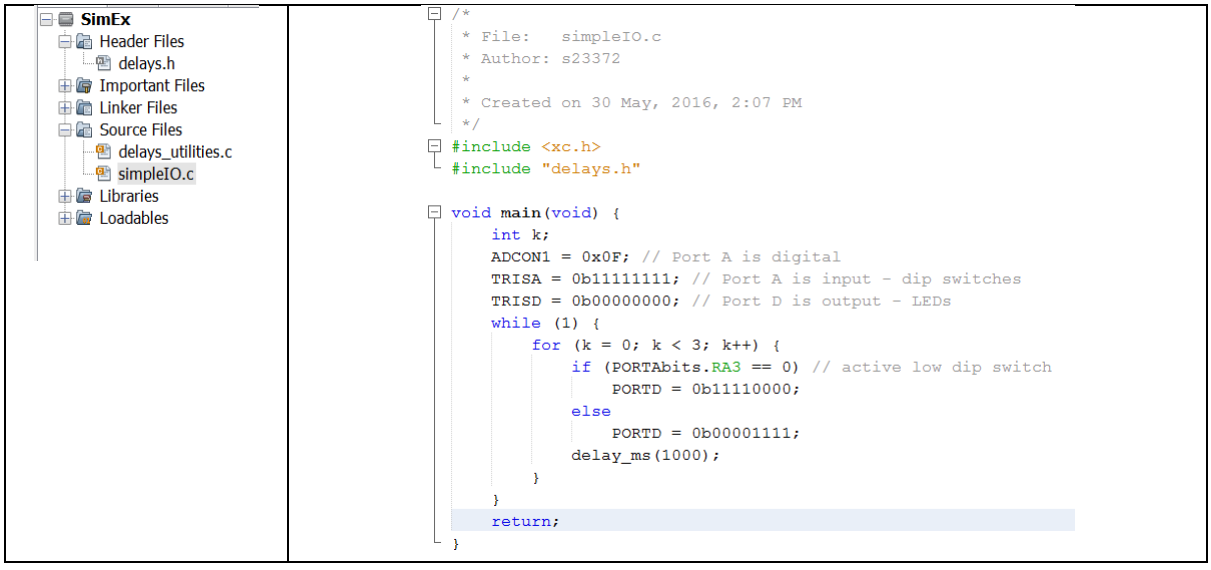
4. Right click on “Source File”, then select “New -> main.c…”:



5. In the “New main.c” window, use “simpleIO” as the “File Name”:



6. Change “simpleIO.c” to the codes shown below, and add “delays.h” to “Header Files” and “delays_utilities.c” to “Source Files”, as usual:



7. PORTA is first made a digital input port, and PORTD an output port. Then, the for-loop is executed 3 times, with the variable k changing from 0 to 1 to 2. In each iteration, RA3 is checked. If it is equal to 0, PORTD will be set to 0xF0. Otherwise, PORTD will be set to 0x0F. A 1-second delay follows that. (Note: Because of the while(1) loop, the for-loop is strictly redundant, but has been included to show how a variable changes value during simulation.)

For the benefit of those who wants to “copy & paste”, this is the C program used:

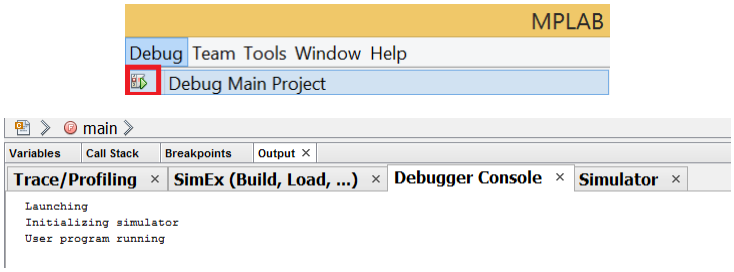
```
/*
 * File:   simpleIO.c
 * Author: s23372
 *
 * Created on 30 May, 2016, 2:07 PM
 */
#include <xc.h>
#include "delays.h"

void main(void) {
   int k;
   ADCON1 = 0x0F; // Port A is digital
   TRISA = 0b11111111; // Port A is input - dip switches
   TRISD = 0b00000000; // Port D is output - LEDs
   while (1) {
      for (k = 0; k < 3; k++) {
         if (PORTAbits.RA3 == 0) // active low dip switch
            PORTD = 0b11110000;
         else
            PORTD = 0b00001111;
         delay_ms(1000);
      }
   }
   return;
}
```

8. Make sure the project can be built successfully before continuing.

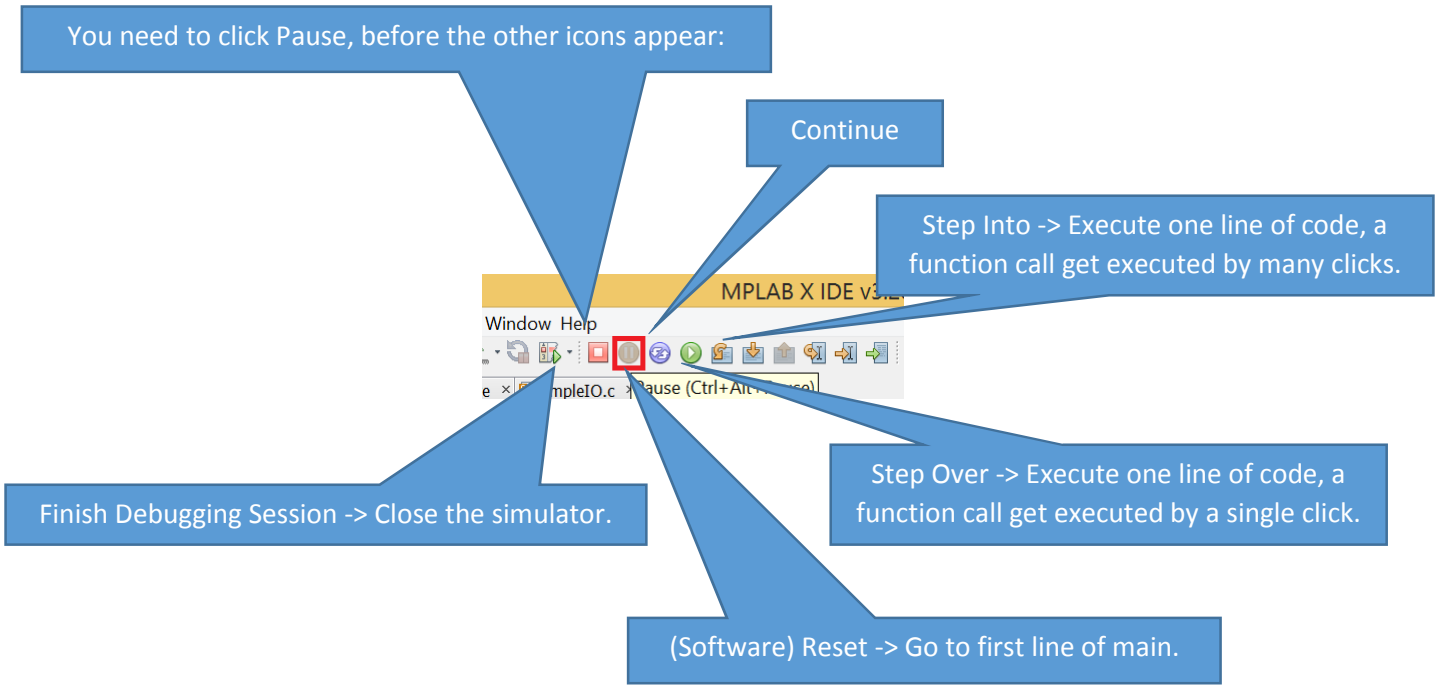
The simulator icons

9. Click “Debug -> Debug Main Project” to start the simulator:



10. This causes the simulator to start running continuously. So click “Pause” (see next paragraph), and the other simulator icons (discussed below) will appear.

11. The simulation icons (at the top) are these:



12. When you click “Step Into”, one line of code gets executed. When you click “Step Over”, one line of code also gets executed. However, if the line is a function call such as delay_ms (1000); (- which is equivalent to many lines of code! -) “Step Over” will get it done in one click. But “Step Into” will bring you into the lines of code inside the function, and you may need to click “Step Out”, if you don’t want to spend the rest of the afternoon there!

13. There are other icons, but we will leave them out for now.

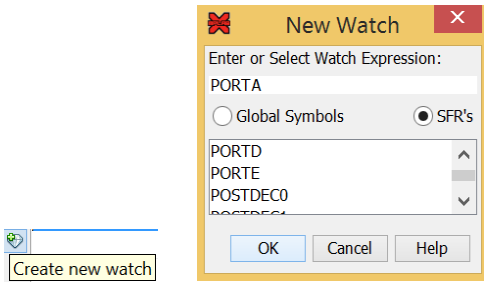
14. If your fingers feel itchy now, go ahead and click the “Reset" icon once, followed by the “Step Into / Step Out / Step Over” icons as many times as you like. But you won’t be able to apply stimulus / input (to PORTA) or observe the output (at PORTD) until you do the next 2 things (which we will do next):
    a. Adding variables to “watch”.
    b. Adding “stimulus” to be fired at appropriate time.

Creating Watches

15. We will now add the following to the “watch list”: PORTA (our input port), PORTD (our output port) and local variable k. The steps are given next.

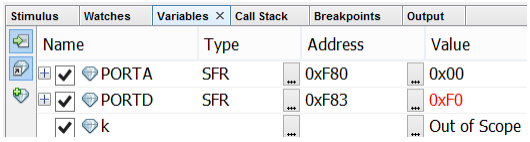
16. First (at the bottom of the screen), click on “Watches” tab.

17. Click on the “Create new watch” icon. In the “New Watch” window which pops up, type PORTA and click OK. (Alternatively, click SFR’s, select PORTA from the list and click OK. SFR stands for Special Function Register.)



18. Next, add PORTD.

19. Finally, add k. Ignore the “Out of Scope” warning for now. (Note that k is a local variable in main....)



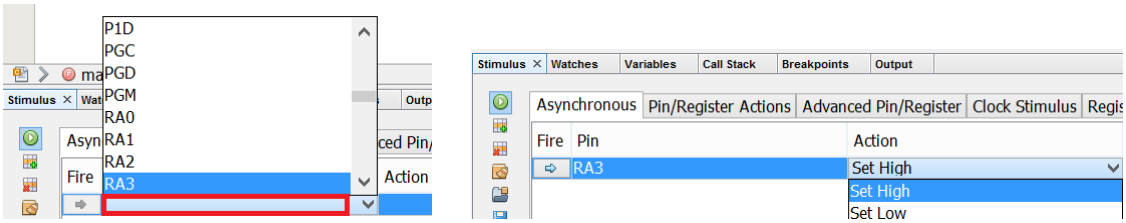
20. We will be watching how the “Values” of these items change when simulation is run, step by step.

Creating Stimulus

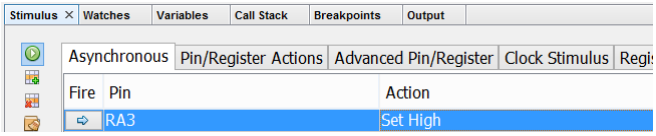
21. We will now create the following stimulus: “Set A High”, “Set A Low”. The steps are given next.

22. First (at the bottom of the screen), click on “Stimulus” tab (and then “Asynchronous” tab if not selected by default).

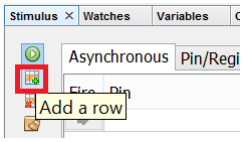
23. Then, click the box next to the “Fire ⇨” and select “RA3”. Next, select “Set High” as the “Action”:



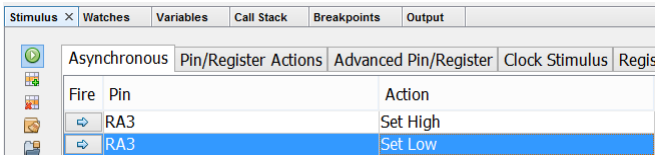
24. What this means is, when the “Fire” icon / ⇨ next to RA3 is clicked, RA3 will be set high.



25. A new row of stimulus can be added by clicking the “Add a row” icon:



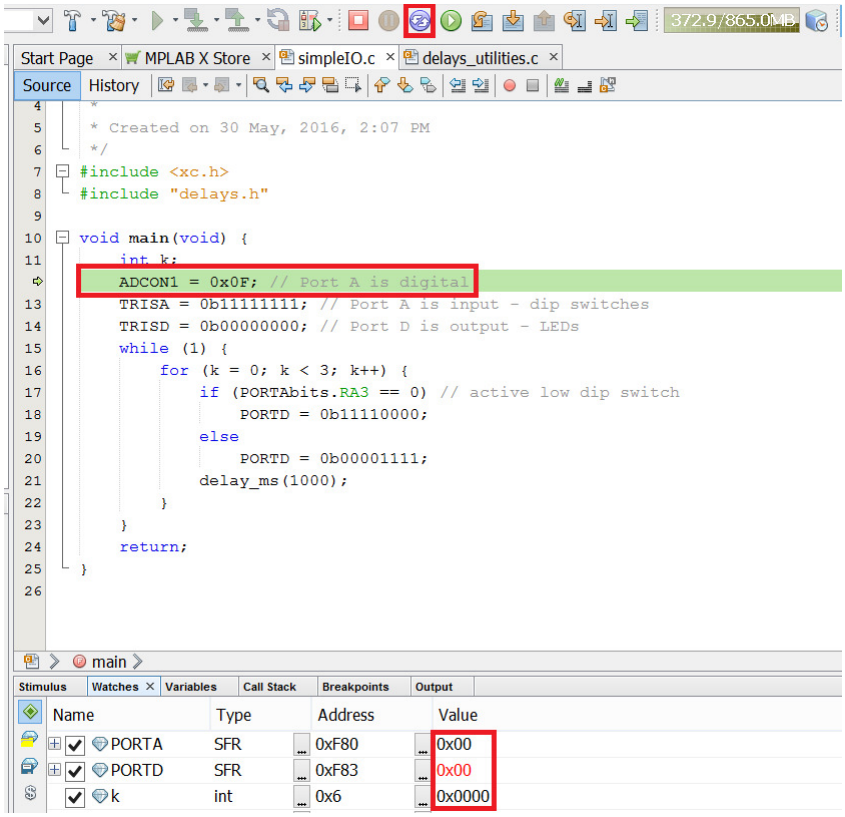
26. In a similar way, create a Set A Low stimulus, so that you have these at the end:



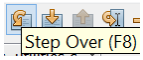
27. Other than “Set High” and “Set Low”, there are other choices, but we will not cover them here. Similarly, beside “Asynchronous” stimulus, there are other stimulus, which will not be covered here.

Let the show begin…

28. Click on the "Reset" icon, you should see the first line of code in main (ADCON1 = 0x0F;) highlighted in light green, ready to be executed. PORTA, PORTD and k assume the initial values of zeros.



29. Click on "Step Over" icon (or press F8) to see how the program flows, paying attending to which of the if-else branch gets executed, and what happens to PORTD (in the Watches) when that happens. (As you step, change in value will be highlighted in red.)



When "Step Over" is clicked _____ times from Reset, PORTD changes to _____, because the _____ (if or else) branch is chosen, since RA3 == _____.

30. Continue clicking "Step Over" and note these:
    - ✓ Stepping over delay_ms (1000); takes considerably more time than other lines.
    - ✓ If you press "Reset" half way, the value of k (a variable stored in memory) will be not reset to 0. If you are not comfortable with that, you may want to write int k = 0; instead.

31. Continue "Stepping Over", but this time, "Fire" the Set (RA3) High or Set (RA3) Low stimulus before executing the if-else statement.

    ⇨ RA3 Set High: If RA3 == 1, the _____ (if or else) branch will be chosen, and

    PORTD will change to _____.

    ⇨ RA3 Set Low: If RA3 == 0, the _____ (if or else) branch will be chosen, and

    PORTD will change to _____.

32. You have probably noticed that the value of PORTA (especially the RA3 bit) only changes value after you click "Step Over". It does not change after you click "Fire".

That's all for now!

33. In the simple simulation exercise,
    a. You have created a project and written a C program.
    b. You have learnt what the simulation icons (Stop, Pause, Reset, Continue, Step Over, Step Into, Step Out) will do when clicked.
    c. You have learnt to create Watches.
    d. You have learnt to create Stimulus.
    e. You have performed simulation using a - d above and recorded down some observations:
        i. How "firing" a stimulus at RA3 will cause a different if-else branch to be taken, thereby changing the PORTD value.
        ii. How Stepping Over a delay function will take longer than other line of code.
        iii. How an uninitialized variable will assume a random value upon reset.

34. In future, if you want to check the behaviour of your program when the hardware is not available or not ready, you can always resort to simulation. Of course, simulation has its many limits.