

CENG 336

Introduction to Embedded Systems Development

Spring 2021-2022

Take Home Exam 1 - Individual Work

Version 1.0

Due date: 6 April 2022, Wednesday, 23:55

Submission: **via ODTUClass**

Objective and Logistics

The purpose of this assignment is to familiarize you with **basic I/O operations** on **MPLAB X IDE simulation environment**.

This is not a group assignment, you must solve it **individually**. Any clarifications and revisions to the assignment will be posted to **ODTUClass**. You should ask your questions in the discussion forum dedicated for this assignment in ODTUClass.

Scenario

Imagine that you are playing a game with your friends where each player gets a playtime in each turn depending on their actions and level points. For this game, you need a calculator and a countdown clock to keep track of time. Since you are taking CENG336 this term and really enthusiastic about it, you tell your friends that you can implement a playtime calculator and countdown clock to use when you are playing the game. Hence, your task in this assignment is to do just that. You will calculate the playtime for a player using the information from input buttons, and count down until their playtime is over using the output LEDs.

PORTB and PORTC will take level point and action type as inputs respectively. Value configurations will be shown by lighting the LEDs assigned to the ports. After the value configurations and calculation, LEDs assigned to PORTD will be used to show the playtime countdown. Port changes and value configurations will be done by pressing and releasing assigned buttons.

Specifications

Five ports will be used in this assignment: **PORTA**, **PORTB**, **PORTC**, **PORTD**, and **PORTE**. LEDs are connected to the following pins: **RB[0-3]**, **RC[0-1]**, and **RD[0-7]**. The push buttons that are used for value assignments and port changes are **RA4** and **RE4**.

Note: The naming of pins is $R + \text{"PORT NAME"} + \text{"BIT ORDER"}$. R represents *PORT*. Here, ports names are *A*, *B*, *C*, *D* and *E*. And the range of the *BIT ORDER* is between 0 and 7.

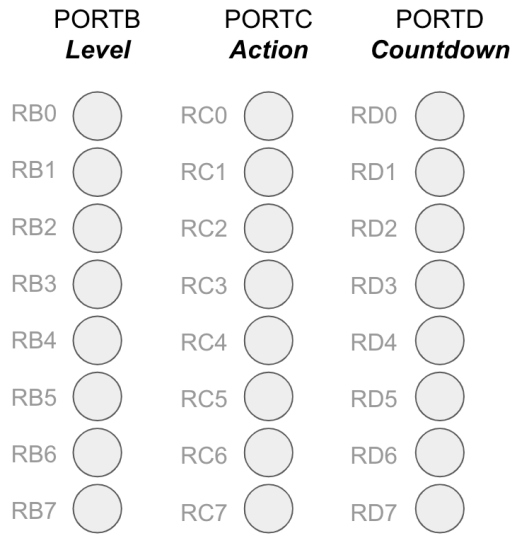


Figure 1: Ports and the values they represent, with pin names shown.

At the start of the program, you must turn on all the LEDs that are used in this assignment, **RB[0-3]**, **RC[0-1]**, and **RD[0-7]**, (14 pins in total) for **1 second**. (Refer to Coding Rules and Hints sections for time delay implementation.)

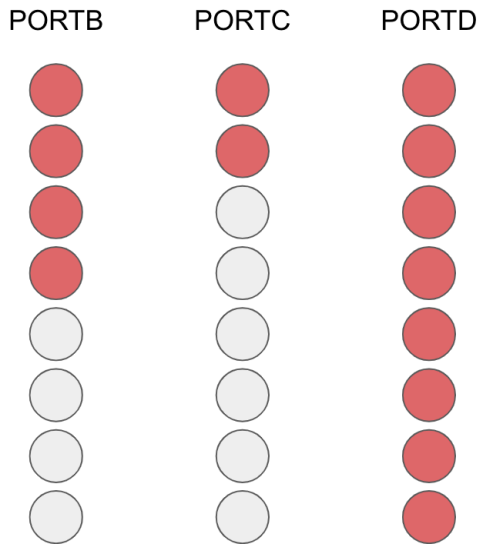


Figure 2: Turning on all LEDs that are used at the very beginning of the program.

When the 1 second period is done, your program must be ready and waiting for port selection input in the **default configuration** shown in Figure 3. **As default, level point will be 1 and action type will be "attack"**, meaning that while waiting for port selection input, instead of turning off all the LEDs, only **RB0** and **RC0** LEDs should be kept on.

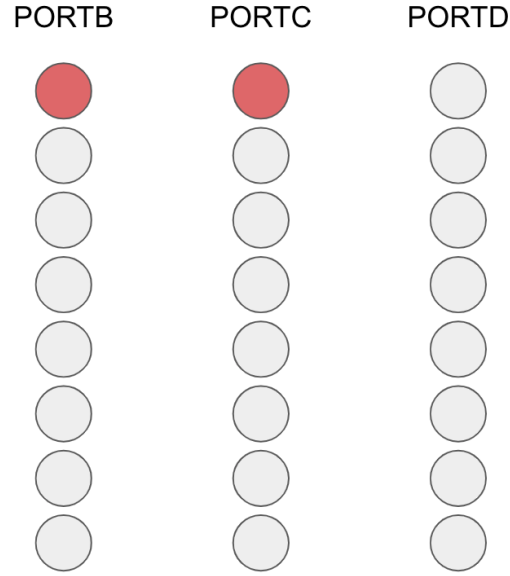


Figure 3: Program waiting for input at the **default configuration**.

RE4 button will be pressed and released for port selection. First press and release of the button will select PORTB for value configuration, second will select PORTC for value configuration and third will select PORTD to display the countdown. When the ports PORTB and PORTC is selected, **RA4** button will be used to **configure values**. Each press of RA4 button will increase the value configured in a port. The LEDs assigned to the port will turn on respectively to the configured value in that port.

For PORTB, the configured value signifies level points of a player. **Level points can take values from 1 to 4**, this means that each time the RA4 button is pressed and released for value configuration, the LEDs on PORTB will turn on respectively from RB0 to RB3. Another press and release of RA4 will set the level point back to 1. If RA4 is never pressed and released, the value will remain as the default configuration.

For PORTC, the configured value signifies action type of a player. **There are two types of actions: "attack" or "defend", which are represented by LEDs RC0 and RC1 respectively**. Each time the RA4 button is pressed and released for configuring the action type, the LEDs on PORTC will switch between RC0 and RC1. This means that the first time RA4 is pressed and released, RC0 will turn off and RC1 will be turned on. The second time RA4 is pressed and released RC1 will be turned off and RC0 will be turned on. These states will repeat with each press and release of RA4. If RA4 is never pressed and released, the value will remain as the default configuration where RC0 is turned on and action type is "attack".

For PORTB and PORTC, LEDs of the port that is currently selected should blink with 500ms intervals to indicate the selection. Meaning that, if RE4 button is pressed and PORTB (or PORTC) is selected, for 500ms the value configured for PORTB (or PORTC) should be visible via LEDs, and for next 500ms all LEDs in PORTB (or PORTC) should be turned off. These actions should repeat until another port is selected. **If PORTD is selected or no port is selected yet, LEDs should not blink.**

When PORTD is selected, countdown value will be calculated from level points and action type, and the calculated value will be shown by turning on the LEDs in PORTD. **After each 500ms, the LED with the highest value will be turned off to show the countdown.** When the countdown is finished,

Table 1: Countdown value calculations

Level point	Action type and value	Countdown value and number of LEDs
1	Attack - 1	500 ms - 1 LED
1	Defend - 2	1 sec - 2 LEDs
2	Attack - 1	1 sec - 2 LEDs
2	Defend - 2	2 sec - 4 LEDs
3	Attack - 1	1.5 sec - 3 LEDs
3	Defend - 2	3 sec - 6 LEDs
4	Attack - 1	2 sec - 4 LEDs
4	Defend - 2	4 sec - 8 LEDs

first all LEDs in PORTD will be turned off, then all ports will go back to the default configuration shown in Figure 3 to wait for port selection.

Calculation of playtime will be done by multiplying the level points with time assigned to an action type. For the attack action, each player will have 500ms per their level point. For defend action, each player will have 1 second per their level point. **Since the counting down operation is done with 500ms intervals, you will turn on 1 LED per level point for attack action, and 2 LEDs per level point for defend action to start the countdown.** Details for calculation is shown in Table 1, countdown process is shown step-by-step in Figure 8.

Example execution

After all the LEDs are turned on for 1 second the program waits for port selection at the default configuration shown in Figure 3.

- RE4 is pressed and released once to select PORTB. After port selection, the program is visually identical to the default configuration, but PORTB LEDs are blinking with 500ms intervals as shown in Figure 4.

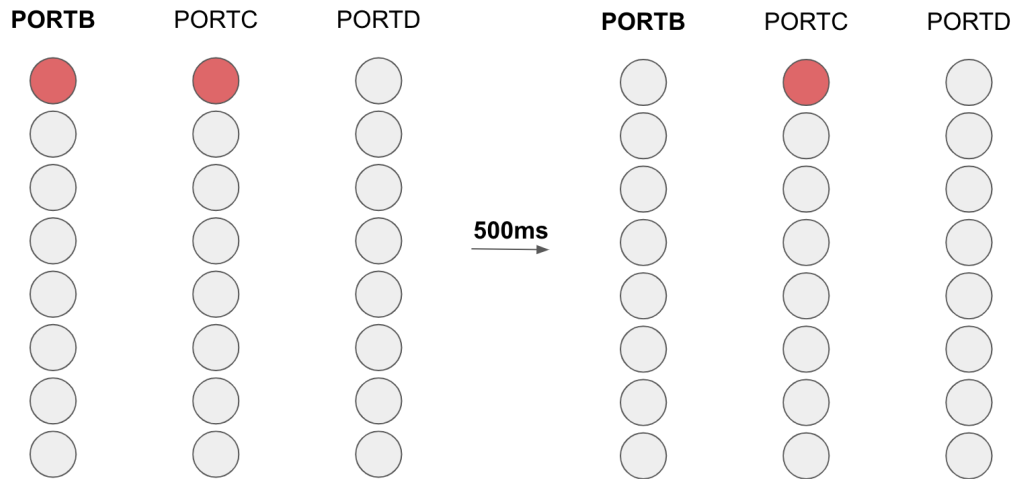


Figure 4: PORTB selected and blinking.

- RA4 is pressed and released again to configure the level point as 2 in PORTB. After the configuration, the LEDs are shown in Figure 5. Note that blinking should continue as long as PORTB is selected, meaning that even the value is updated, LEDs should continue to blink in 500ms intervals.

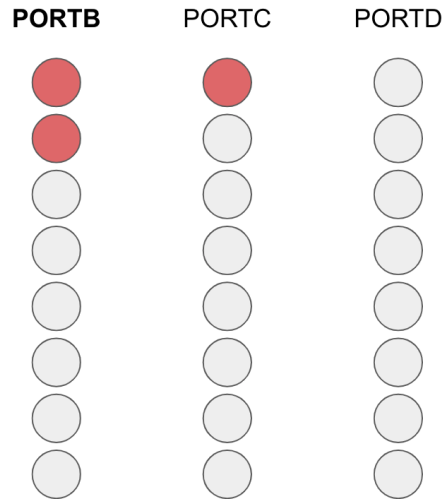


Figure 5: PORTB configured to level point 2.

- RE4 is pressed and released again to select PORTC. Selecting PORTC should stop the blinking in PORTB, and PORTC should start to blink.
- RA4 is pressed and released to configure action type as "defend" in PORTC. After the configuration, the LEDs are shown in Figure 6. Note that blinking should continue as long as PORTC is selected, meaning that even the value is updated, LEDs should continue to blink in 500ms intervals.

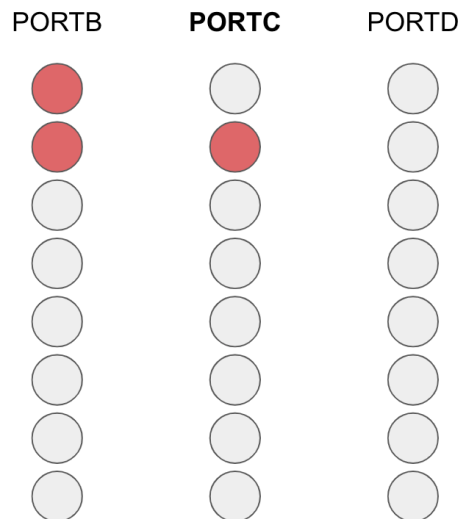


Figure 6: PORTC configured to "defend" action.

- RE4 is pressed and released again to select PORTD. This triggers the calculation of playtime and stops the blinking. Values for PORTB, PORTC and PORTD should be fully visible. With level point 2 and "defend" action, the calculation is done as $2 \times 1\text{second} = 4 \times 500\text{ms}$ and 4 LEDs are turned on as shown in Figure 7.

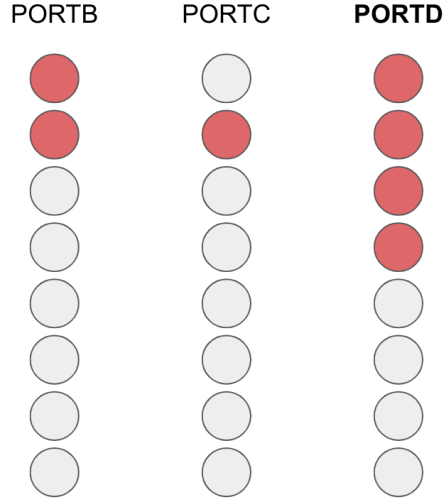


Figure 7: PORTD is selected, playtime is calculated, and countdown is started.

- After each 500ms passed, the LED with the highest order in PORTD will be turned off. The countdown process is shown in Figure 8. At the end of the countdown, last stage of the countdown will be visible for another 500ms and then the program will go back to the default configuration and wait for port selection.

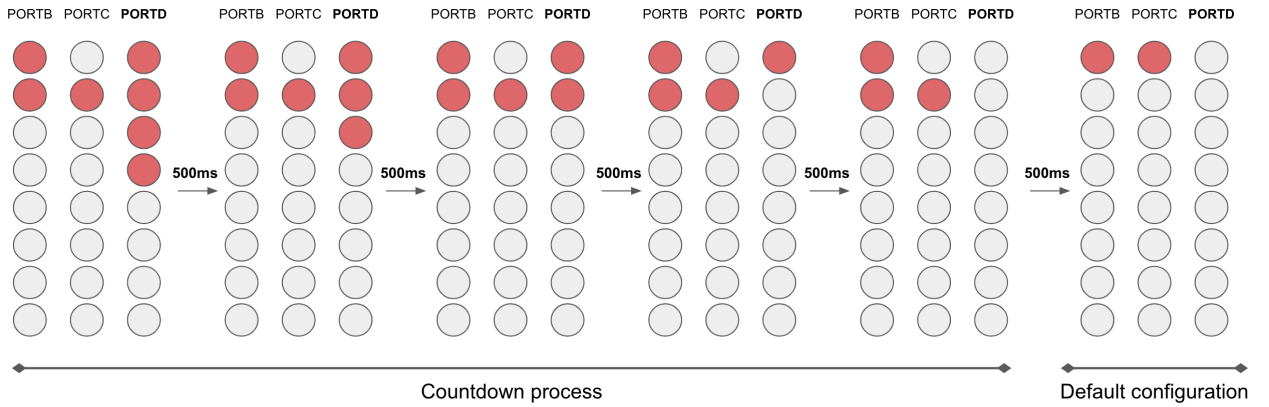
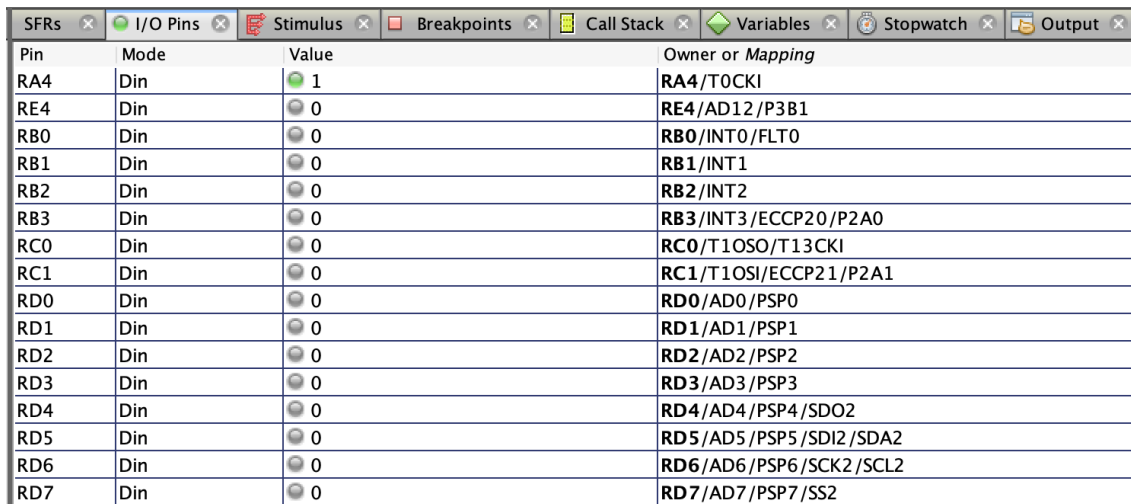


Figure 8: Countdown process and return to default configuration.

Implementation

You are expected to complete the assignment in **MPLAB X IDE Simulator** environment. You can track the status of the LEDs and buttons by using **RB0, RB1, RB2, RB3, RC0, RC1, RD0, RD1, RD2, RD3, RD4, RD5, RD6, RD7, RA4** and **RE4** pins in the simulator.

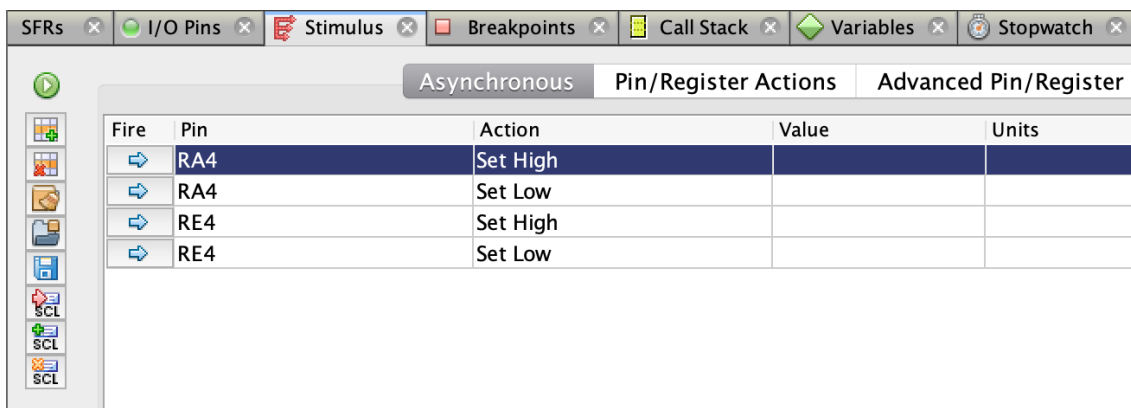
To see the output of the pins, you can use **I/O Pins** window that is available from **Window → Simulator** menu. A screen shot from the I/O pins menu of the simulator is given in Figure 9, you can see the states of pins that represent buttons and LEDs.



Pin	Mode	Value	Owner or Mapping
RA4	Din	1	RA4/T0CKI
RE4	Din	0	RE4/AD12/P3B1
RB0	Din	0	RB0/INT0/FLT0
RB1	Din	0	RB1/INT1
RB2	Din	0	RB2/INT2
RB3	Din	0	RB3/INT3/ECCP20/P2A0
RC0	Din	0	RC0/T1OSO/T13CKI
RC1	Din	0	RC1/T1OSI/ECCP21/P2A1
RD0	Din	0	RD0/AD0/PSP0
RD1	Din	0	RD1/AD1/PSP1
RD2	Din	0	RD2/AD2/PSP2
RD3	Din	0	RD3/AD3/PSP3
RD4	Din	0	RD4/AD4/PSP4/SDO2
RD5	Din	0	RD5/AD5/PSP5/SDI2/SDA2
RD6	Din	0	RD6/AD6/PSP6/SCK2/SCL2
RD7	Din	0	RD7/AD7/PSP7/SS2

Figure 9: I/O Pins window

To simulate the press and release action of the push buttons, you can use **Stimulus** window which is also available from **Window → Simulator** menu. A screen shot from the Stimulus menu of the simulator is given in Figure 10. It can be seen that by clicking the "Fire" buttons, voltage level of the buttons can be controlled as High or Low. If you add the corresponding pin to the I/O Pins menu, you can see the state change in the pin caused by your action.



Asynchronous				
Pin/Register Actions				
Advanced Pin/Register				
Fire	Pin	Action	Value	Units
→	RA4	Set High		
→	RA4	Set Low		
→	RE4	Set High		
→	RE4	Set Low		

Figure 10: Stimulus window

Debug Tests

Some of the application behavior will be blackbox checked by debugger scripts along with glassbox checking for any errors that the debugger scripts might miss. Since the debugger scripts cannot know when to stop execution and check for values by themselves, you will need to call dummy subroutines in certain locations of execution (in the form `<label>;nop; return;`, a bare-bones file to write your code is already provided to you with these dummy subroutines). These are:

- **ms1000_passed**: should be called *once* after you initialize the configuration values and have turned on all leds used in the assignment for 1 ± 0.1 sec.
- **ms500_passed**: should be called every 0.5 ± 0.1 sec after **ms1000_passed** is called. This period should not be above/lower than the tolerance limits (0.1 sec) by program/button states. **Your program is not allowed to freeze/be unresponsive to buttons after ms1000_passed is called.**

The debugger scripts are provided to you. Here are the instructions to set them up in your local machine/ineks:

1. You will require **python3.6** or higher and **make** in your local machine in the **PATH** environment variable. Command line form of MPLAB debugger(**mdb**) should be in your **PATH** environment variable.
2. Create your project in the MPLAB X IDE the usual way. Your code should be in a single assembly file, directly under the project directory, named **the_1_1234567.s** where 1234567 represents your seven-digit student number. A template file is provided to you.
3. Put the **tests** directory inside your project directory (MPLAB names it **projectname.X** unless you explicitly change it).
4. Within tests, add **run_tests.sh** your project directory name and student number to the appropriate locations.
5. **cd** into tests and run **run_tests.sh**.

You are free to inspect and modify the scripts to further enable/disable tests to your liking. Most of the commands in **mdb** scripts are self-explanatory, but the documentation might also help you.

Coding Rules

- You will code your program using PIC assembly language.
- Your program should be written for **PIC18F8722** working at **40 MHz**.
- When the push button is pressed, then RA4/RE4 pin goes high, and when it is released, the RA4/RE4 pin goes low.
- **Actions must take place after the press and release of a button**, only pressing the button is not enough to take action.
- When you are writing your codes, please look at the relevant lecture notes and recitation documents. In particular, you are expected to apply the **round robin approach**, so your program will execute in an infinite loop, and there will be **your own tasks for your states** in this loop at the main scope.

- Be precise for constructing time delays (intervals). **You should use loops for implementing the time delays, you MUST NOT use Timers in this assignment.** You may need to count the number of instructions for implementing time delays. For the expected delays, +/- 100 ms deviation is acceptable. You can measure the time by using the **stopwatch** and **breakpoints**, so please look at the Hints section.
- **Your code should not busy wait for button releases.** In other words, when the selected port is blinking, pressing and holding a button should not stop the blinking, and your program must be responsive.
- If no port is selected yet, press and release of RA4 should not cause any change.
- If PORTD is selected and the countdown has started, press and release of RE4 or RA4 must be ignored.
- When the countdown starts, blinking must stop and values for PORTB and PORTC must be visible.

Submission and Grading

You should submit your code as **a single file** named as `the_1.1234567.s` through ODTUClass, where 1234567 represents your seven-digit student number.

Your codes will be evaluated on the MPLAB X IDE simulation environment in department computers. The debug script provided to you will also be used to check if your program obeys the timing requirements.

Hints

- There are plenty of resources available to you in ODTUClass. To refer to instructions and specifications of PIC18F8722, you should use the PIC18F8722 Datasheet. Since you will be using MPLAB X IDE simulation environment for this assignment, Recitation 1 documents can also be a great set up guide. If you set up a local environment, you should use MPLAB X IDE version 5.45 and XC8 Compiler version 2.30, both are available in the downloads archive. If you cannot set up a local environment, you can use the department labs.
- **Stopwatch** tool of the simulator can be used to measure time spent by a code segment, by adding **breakpoints** to the start and end of that segment. You can find this window from **Window** → **Debugging** menu. It is important to configure your clock speed to 10 Mhz from **Project properties** → **Simulator** → **Instruction Frequency** before starting the simulator for the time values to be correct. **Project Properties** window can be reached by right clicking the project name in **Projects** panel. You can refer to Recitation 1 documents for detailed explanations and screenshots.
- You can also see the states of your ports, variables and pins by using the Logical Analyzer, SFR and Variables windows. They can be found under the **Window** menu in **Simulator**, **Debugging**, **Target Memory Views** sections respectively.