ECE3411 – Fall 2015

Lab 5c.

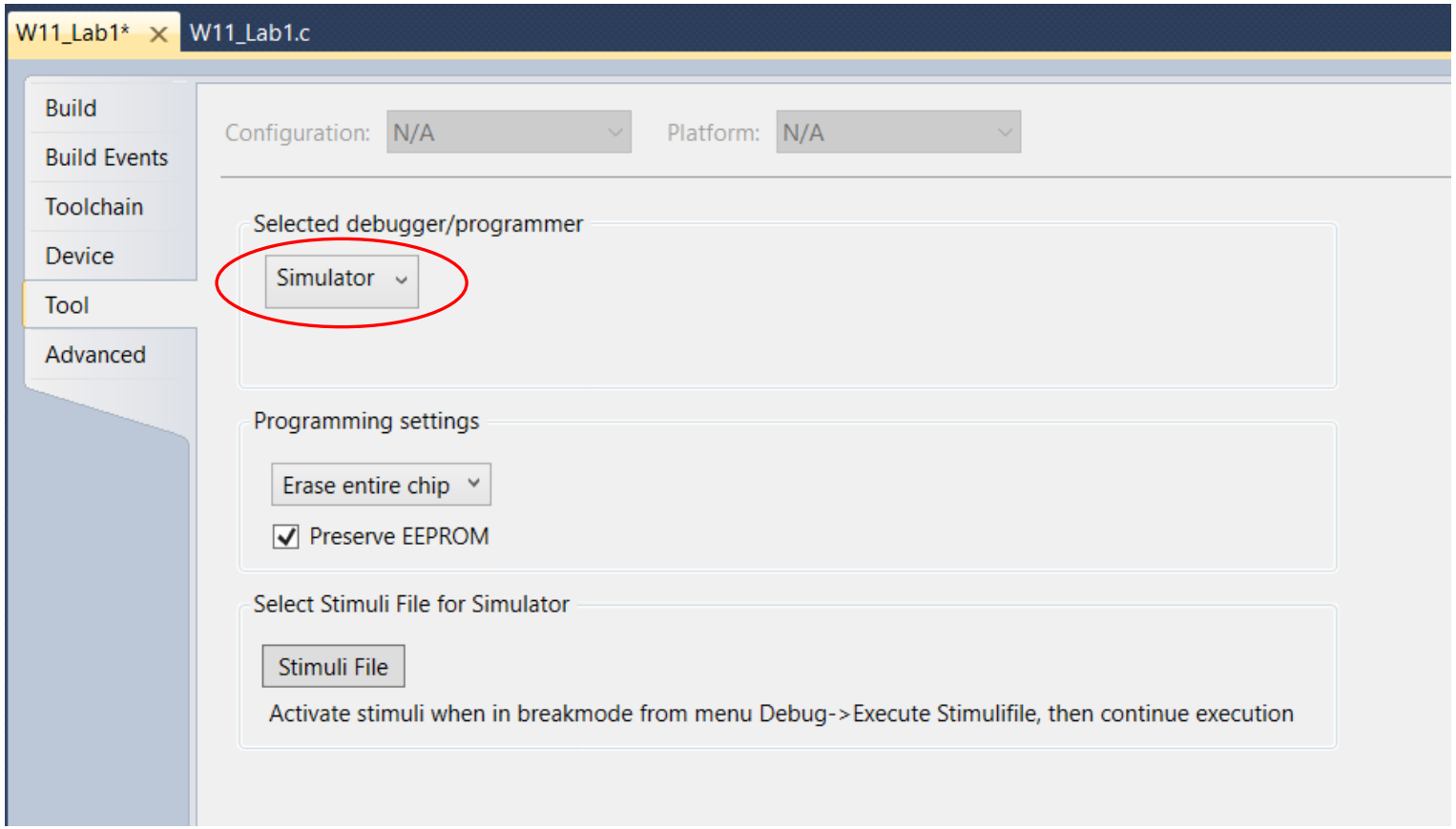
Debugging using Atmel Studio

Marten van Dijk, Syed Kamran Haider
Department of Electrical & Computer Engineering
University of Connecticut
Email: {vandijk, syed.haider}@engr.uconn.edu

UCONN

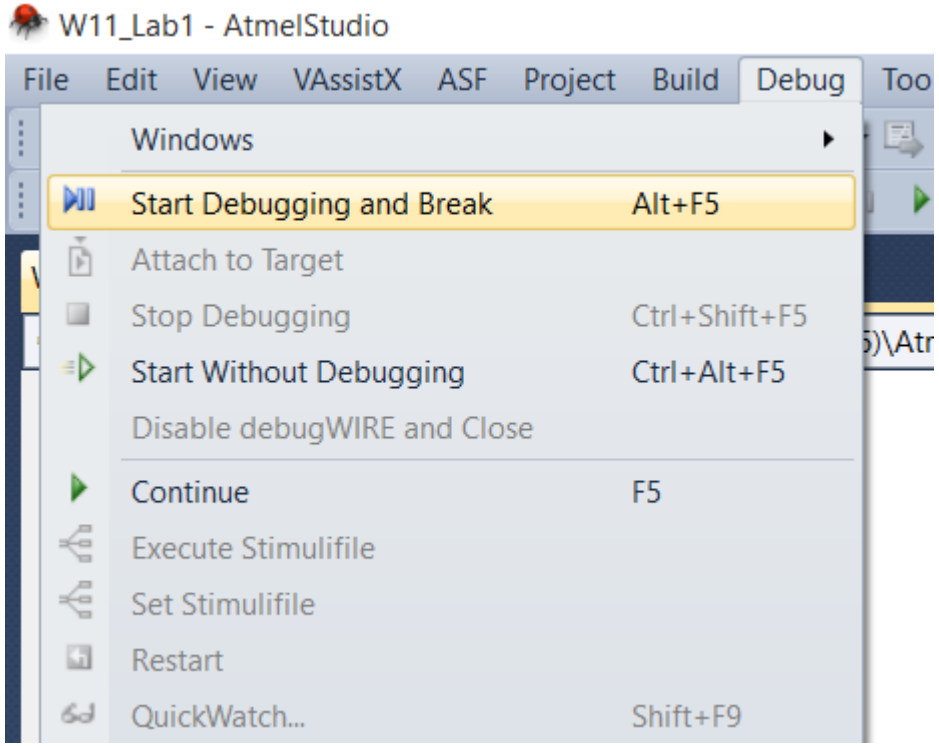
Starting a Debugging Session

- Create a new Atmel Studio project
- Select “Simulator” from the Tool Selection tab



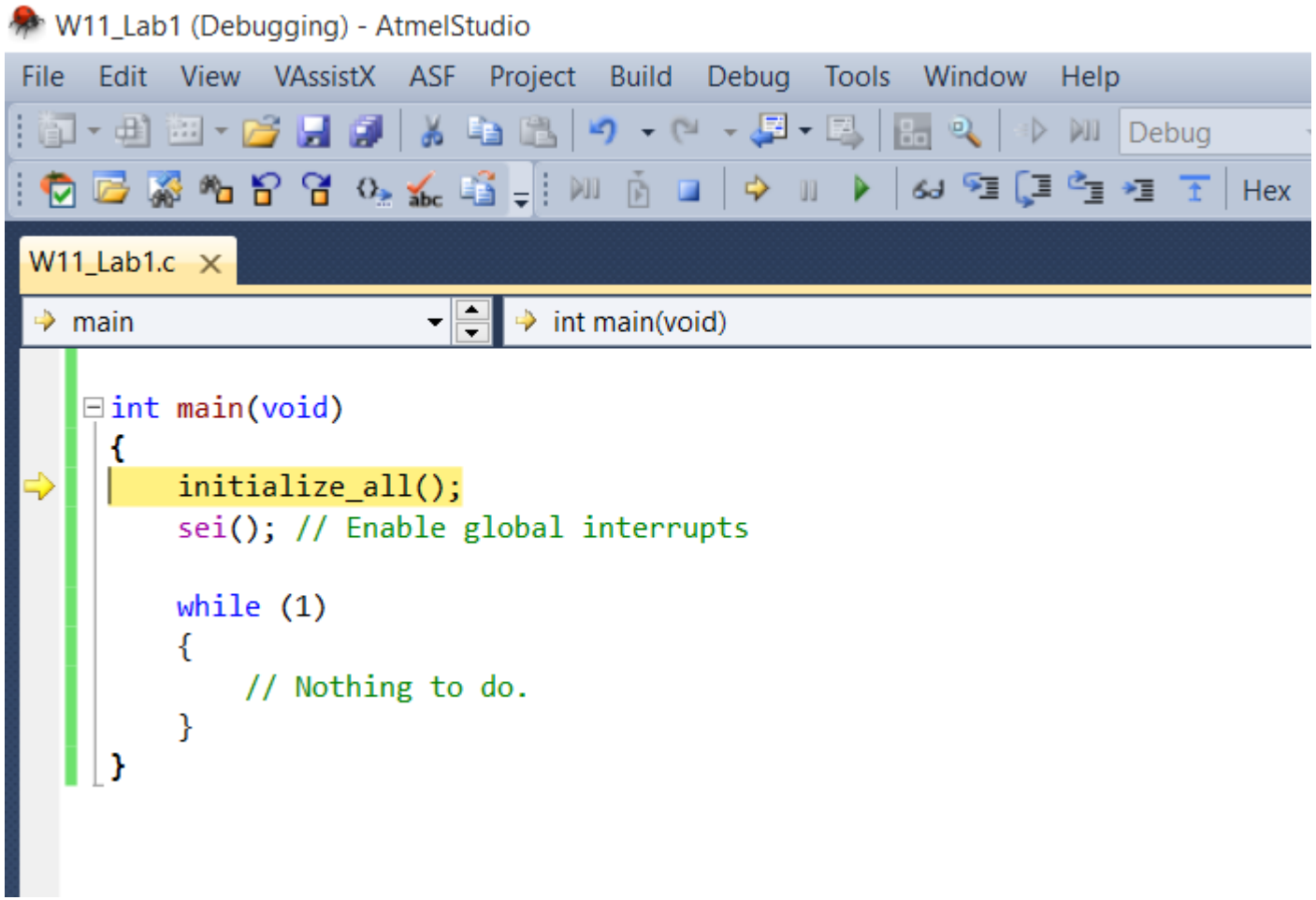
Starting a Debugging Session

- Build the project. (Hit F7)
- From Debug tab, select “Start Debugging and Break”
- The debugger pauses at the start of main.



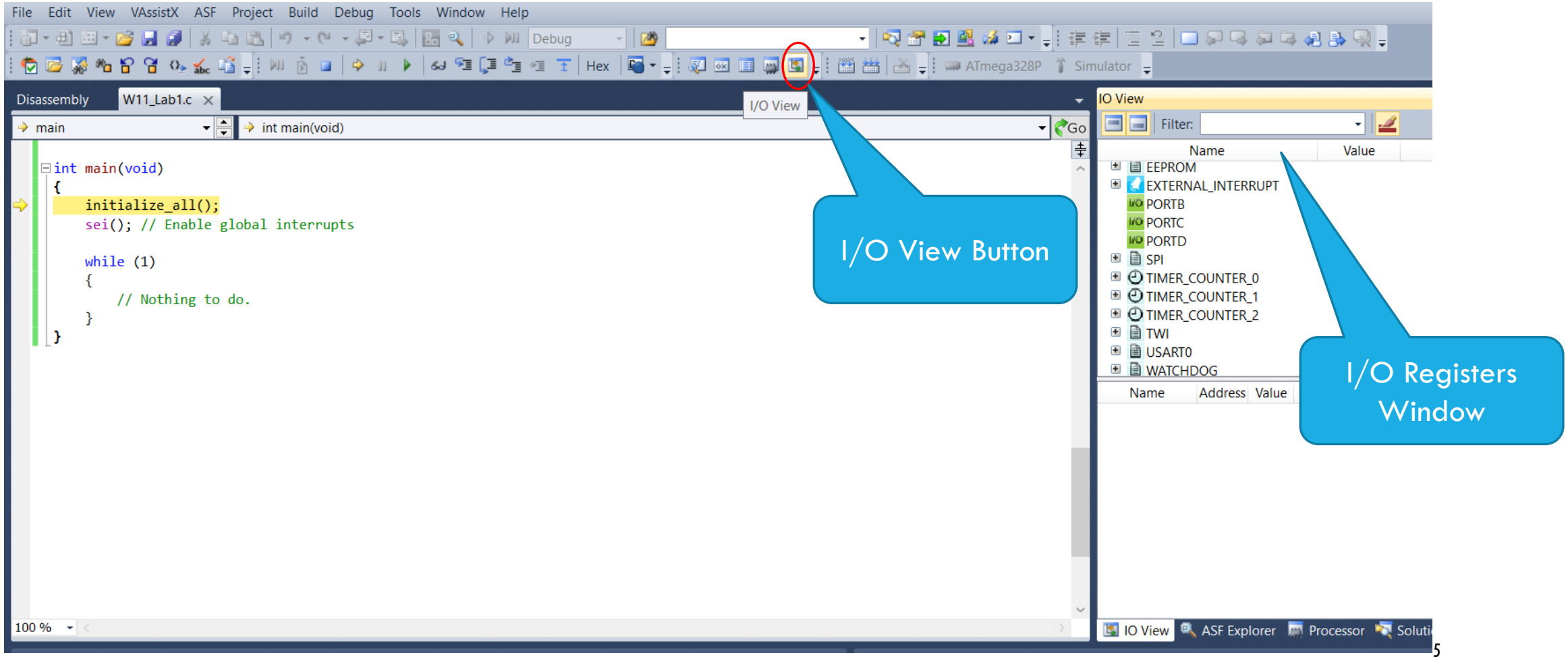
Start of Debugging Session

- The debugger pauses at the start of main.



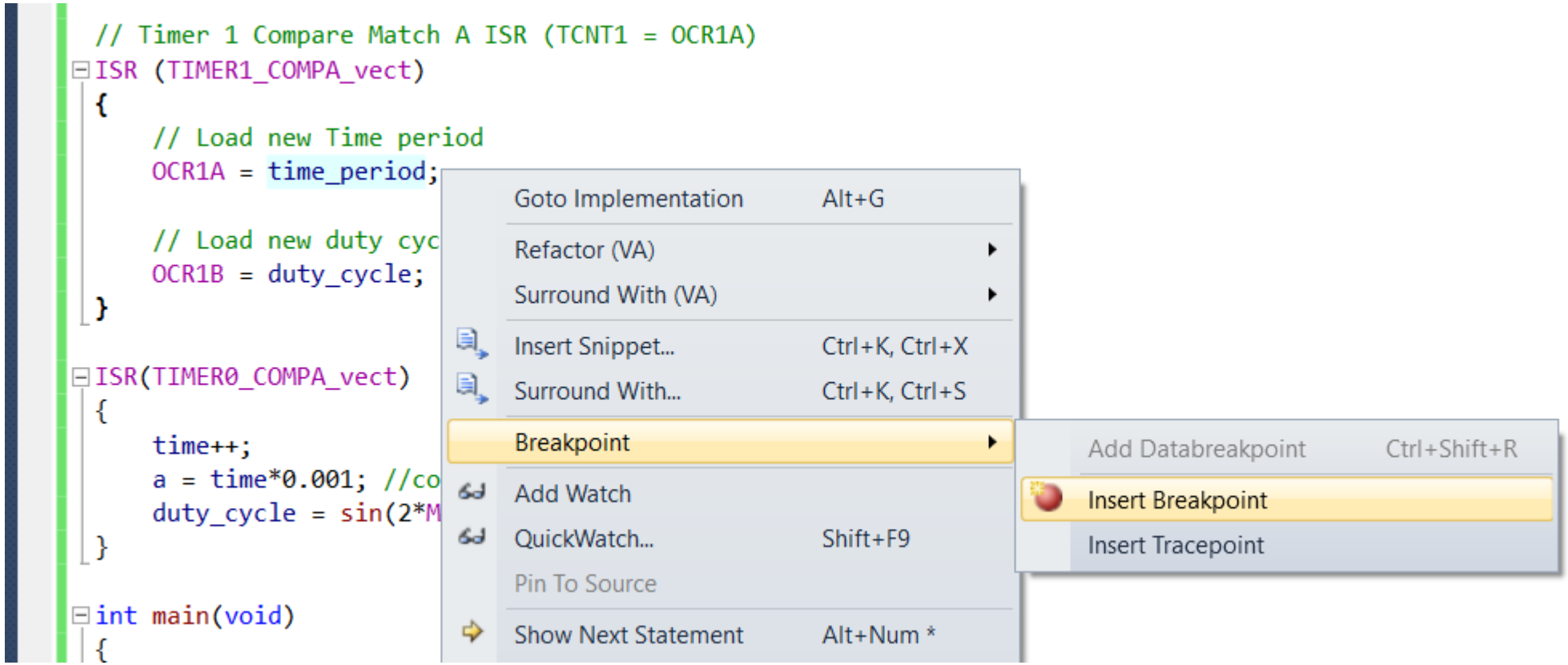
Peripheral Registers in Debugging Session

- Click on I/O view button to see all peripheral registers in an I/O Window



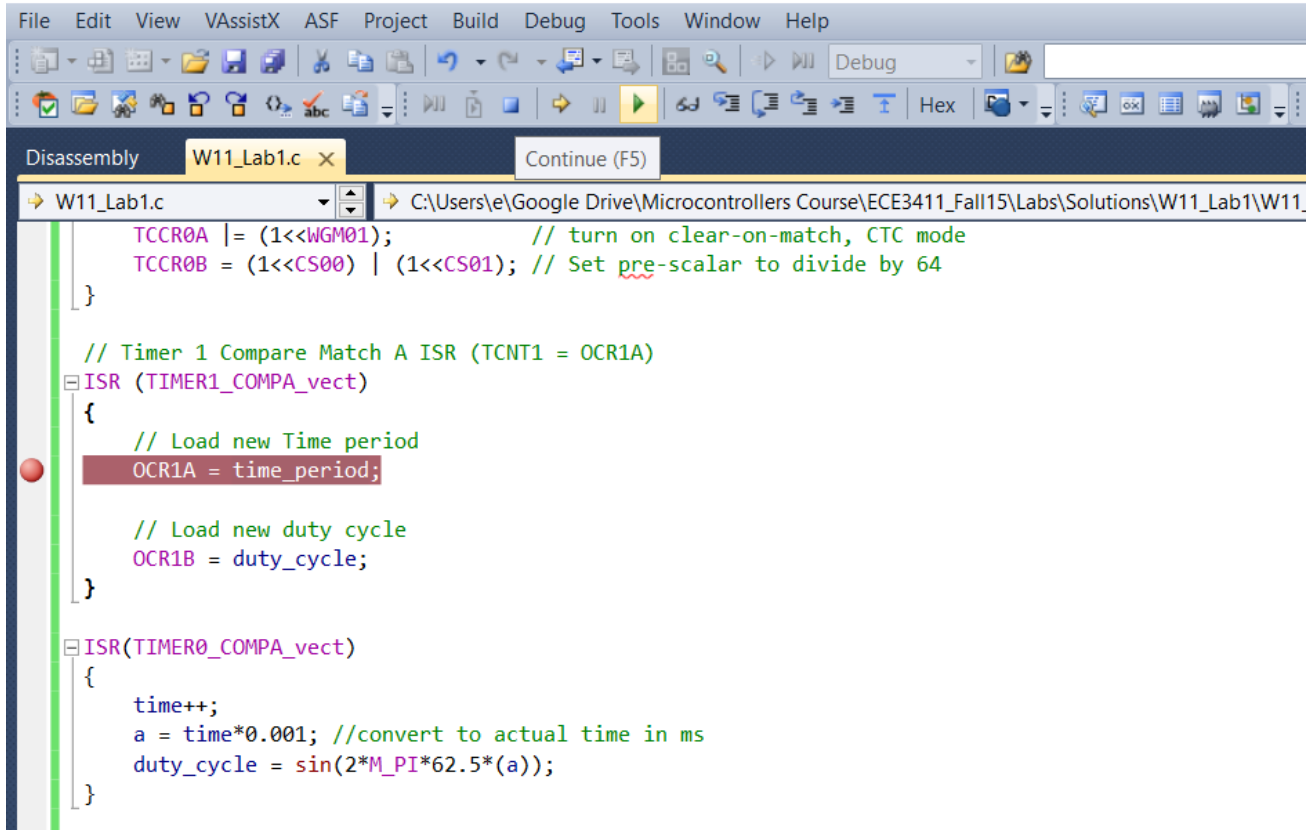
Adding a Breakpoint in Debugging Session

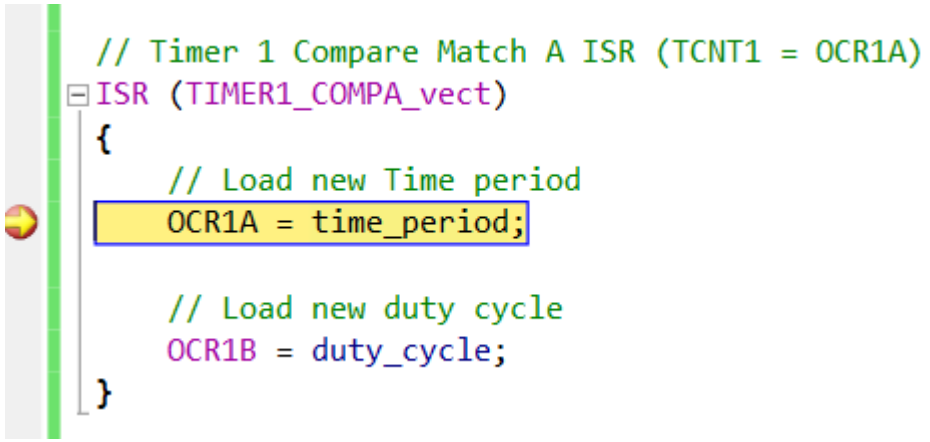
- Select any instruction in the code
- Right Click and insert a Breakpoint as follows



Continue to the next Breakpoint

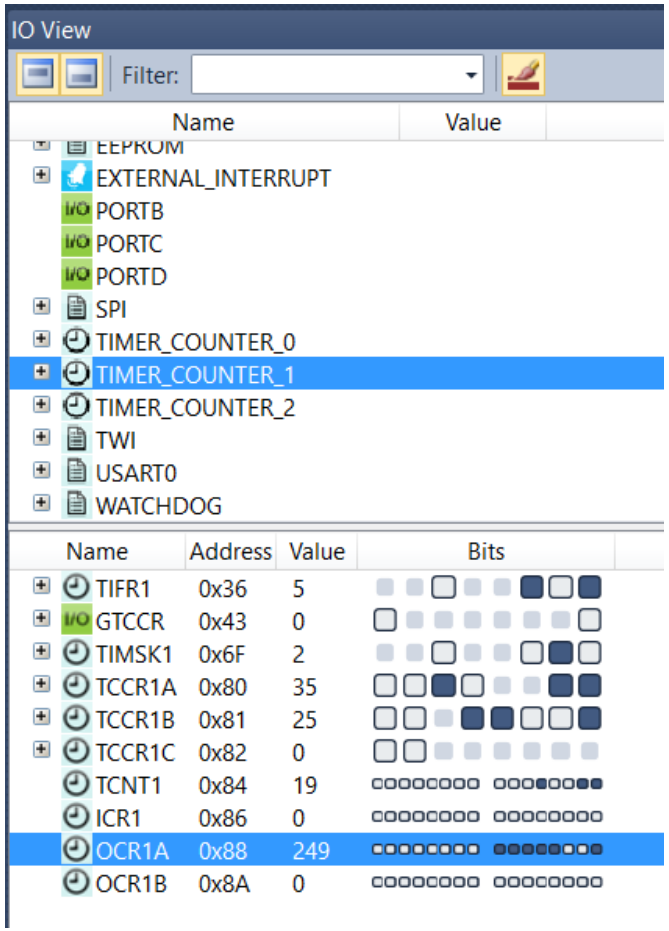
- After inserting a breakpoint, click Continue (F5)
- The program will stop at Breakpoint as shown in the right window.

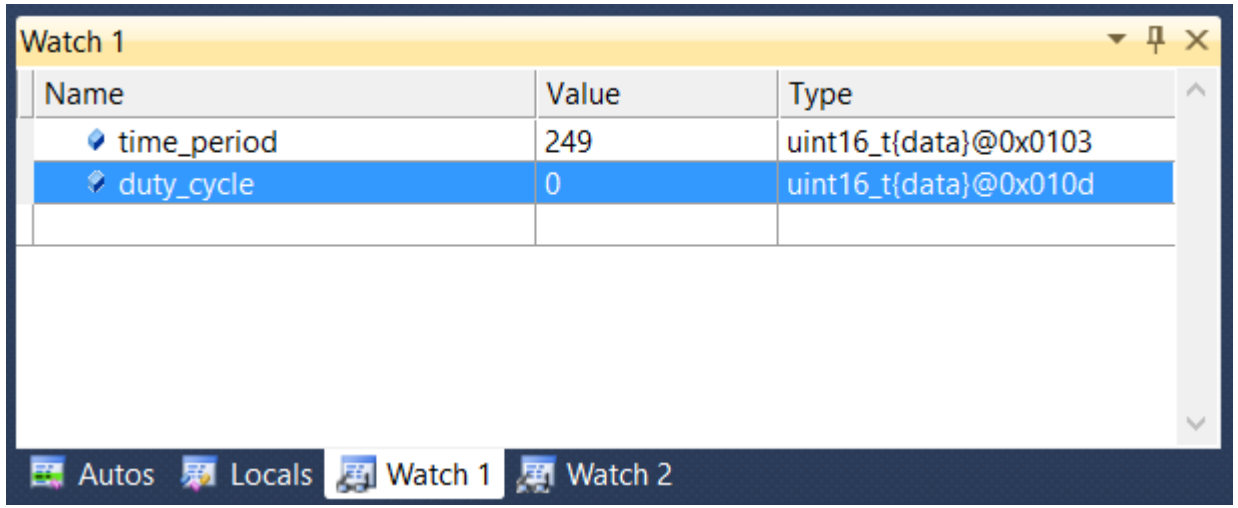




Observing Register/Variable Values at a Breakpoint

- Select particular peripheral and then the register to observe the value. (shown on left)
- Type variable names from your code in Watch Window to monitor their values. (shown on right)





Files for today's Lab Tasks

- Download the zipped file from the link below.

  http://www.piazza.com/class_profile/get_resource/idhg4rqfhcm1uh/igsgo1qx1j86ok

- This file contains three C code files.
  - Task1.c
  - Task2.c
  - Task3.c

Task1,2: Debugging a buggy PWM

The codes in Task1.c and Task2.c generate a 'rectified' 62.5Hz sine waveform using a 64kHz PWM.

The PWM signal is generated at PB2 using Timer1 such that the duty cycle of the PWM is a function of a 62.5Hz sine wave. I.e. for f = 62.5

$$duty\ cycle \propto |\sin(2\pi f t)|$$

- There are some bugs in these codes. Your task is to use Atmel Studio debugger to find the bugs in these codes.

Task3: Debugging a buggy Stopwatch

The code in Task3.c is a buggy implementation of a Stopwatch (1ms resolution) for measuring the total time and the individual lap times of a car racer. The detailed functionality is as follows:

- When SW1 is pressed (i.e. start of the race), Timer1 starts counting the number of milliseconds.
- If SW2 is pressed while the stopwatch is counting (i.e. during the race), it records the current time and prints the time elapsed between this and the previous most recent push. This shows the lap time.
- Finally when SW1 is pressed again (i.e. at the end of the race), the total time and the best (i.e. shortest) lap times are printed on the LCD.
- Timer0 is used to count a debounce delay of 16ms for SW1 and SW2.

Your task is to find the bugs in this code and make it run on your MCU boards!