Microcomputers 1

Lab 6
Output Port Programming

Concepts:

- Using the HCS12 general-purpose I/O ports
- Get hands-on experience with output Ports: configuring ports as output and sending data to the output devices.
- Get a better understanding of how to create specific delays.

Objectives:

- Use **the HCS12 output ports** connected to the Dragon12+ 7-segment LED displays.
- Download and execute programs on the Dragon12+ board using CodeWarrior.

Assignment:

Part I: Working on the Memory-Mapped I/O devices: Implementing 7- segment LED display using Code Warrior simulation. Seven-segment LED displays are mainly used to display decimal digits and small letters. In the HCS 12 + board, four 7-segment LEDs are connected to the 8 bits in **Port B (= memory location $0001)**. Therefore, the HCS12 microcontroller must send an appropriate value to Port B to display a specific digit on a 7-segment LED.

As shown in Figure 1, Port B's bits (pins) are connected to *all four 7-segment LEDS* and 8 flashing LEDs in the HCS 12+ board. **This means it cannot output two different patterns on two 7-segment displays at the same time**. **Instead, a time-multiplexing technique is used to display multiple digits simultaneously.** **Repeatedly, enable one 7-segment LED display for a short period (ex: 1 ms) and then disable this display to enable the other 7-segment LED**. Port B holds the data you want to display on the enabled 7-segment LED. Within a second, each display is lighted in turn many times. **Due to the persistence of vision**, all digits appear to be lighted at the same time.

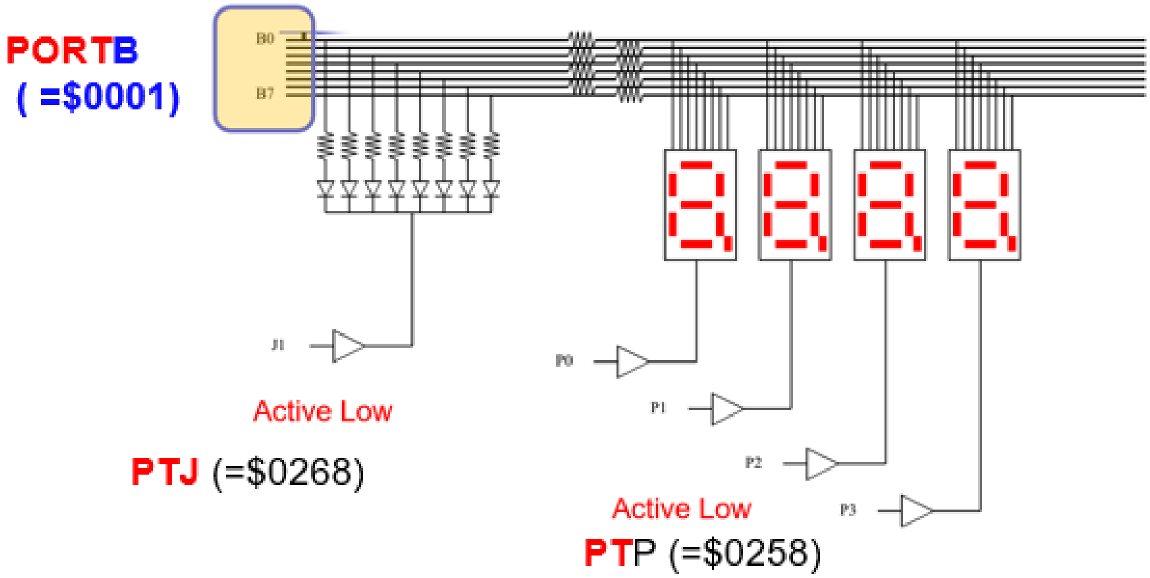


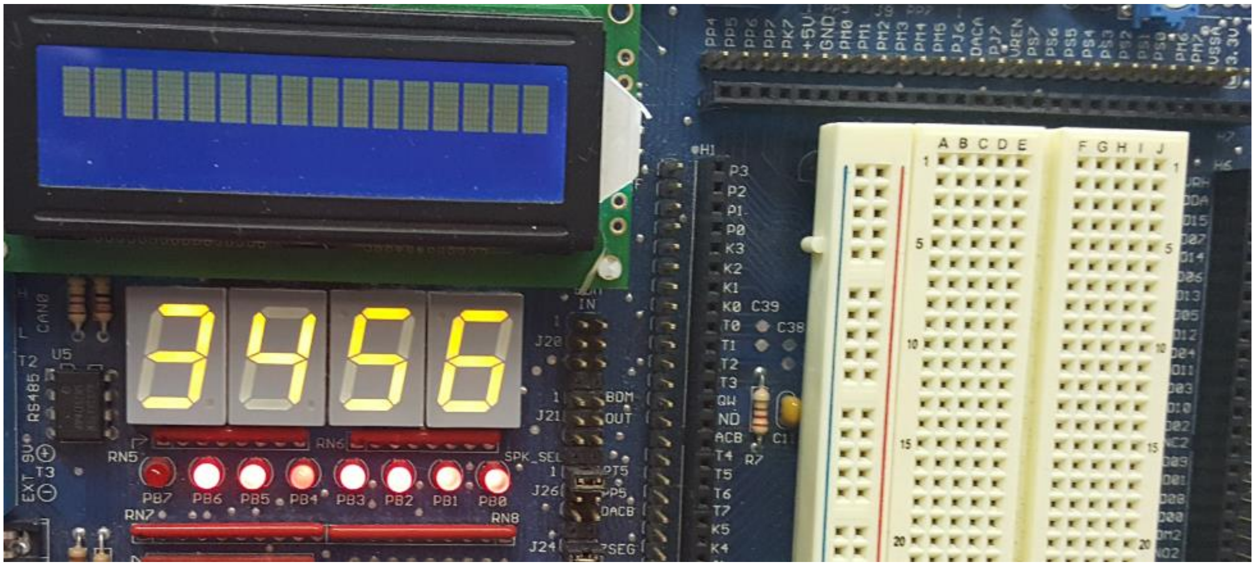
Figure1. Four 7-segment LEDs and eight flashing LEDs are connected to PORT B.

In Lab6, **you need to disable eight flashing LEDs by setting Bit 1 in PORT J to 1**. It is an active low system and disabled the device by setting Bit 1 in Port J to 1. To enable/disable four 7-segment LEDs, **Bit 0 to Bit 3 in PORT P** are used. Only one bit in Bit 0 to Bit 3 in **PORT P** should be set to zero, and the other bits are ones to enable one 7-segment LED display at a time.

The table below shows how to set bit in **PORT B** to display the digit from 0-9.

Decimal digit	Segments							Value to send to port B
	6	5	4	3	2	1	0	
0	0	1	1	1	1	1	1	$3F
1	0	0	0	0	1	1	0	$06
2	1	0	1	1	0	1	1	$5B
3	1	0	0	1	1	1	1	$4F
4	1	1	0	0	1	1	0	$66
5	1	1	0	1	1	0	1	$6D
6	1	1	1	1	1	0	1	$7D
7	0	0	0	0	1	1	1	$07
8	1	1	1	1	1	1	1	$7F
9	1	1	0	1	1	1	1	$6F

Write an assembly program to **display the number 3456 on the four seven-segment displays** in Dragon 12 plus board, as shown below.



Steps involved in Part I are defined as below:

1. Create a new project with the name '**Lab6_part1.mcp**' as you did in the previous labs. Open the main.asm file in the project to write your code. CodeWarrior's main window opens up. Select "Full Chip Simulation" from the dropdown list on the left and select "**Full Chip Simulation**" if not selected yet. Delete the prewritten sample code marked with black background in main.asm provided by the CodeWarrior IDE.

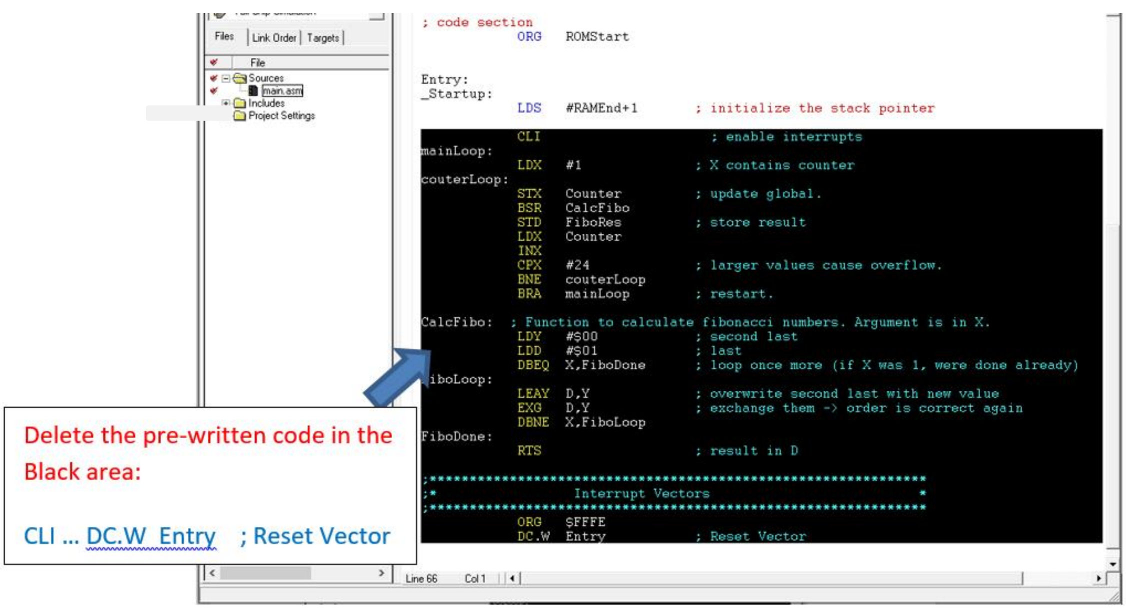


Fig. 2: Main program part.

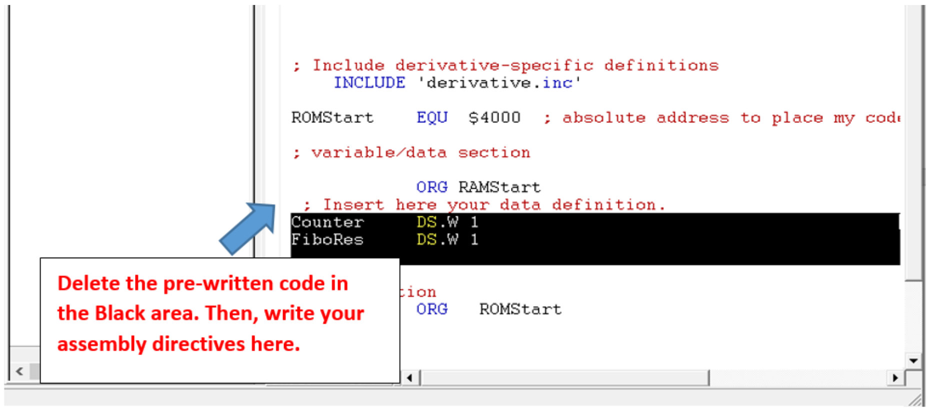


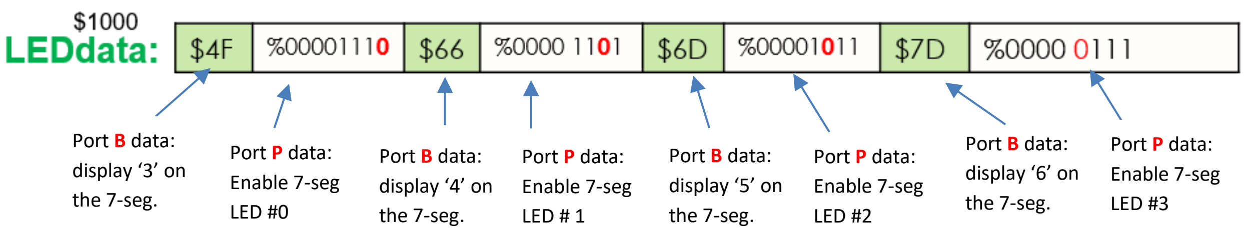
Fig. 3: Data declaration section.

2. Data definition using the assembly directives:

Two sets of data are required to display the digits on the four seven-segment LEDs:

- **PORT B** data: the data that turns on and off each segment to display the number on the 7-segment LED.
- **PORT P** data: the data selects one 7-segment LED by enabling or disabling the four 7-segment LEDs.

You need to place the data definition in the data section shown in Fig 3. Using the assembly directives, define the data



3. I/O port configuration. In HCS12, I/O Ports are **bidirectional ports. Bidirectional ports** have a "**data direction register**" (DDR) that sets the direction of data flow. In lab6, three output ports, PORT B, PORT P, and PORT J, are used. Configure those three ports using **Data Direction Registers (DDRs).**

   Configure PORT B, PORT P, and PORT J as output ports using **Data Direction Registers**.
   - **All 8 bits** in Port B are configured using **DDRB** as output pins.
   - **Bit1** in Port J is configured using **DDRJ** as an output pin.
   - Port P's lower four bits (**Bit3 - Bit0**) are configured using **DDRP** as output pins.

Use the BSET instruction to configure each DDR register for PORT B, PORT J, and PORT P.
Ex) **BSET DDRB, %11111111**

4. Disable the eight flashing LEDs by setting a value in **Bit1 in PORT J**. As shown in Figure 1, it is an active low system. By setting Bit1 in Port J to 1, eight flashing LEDs are disabled.

5. Two loop structures are required to continuously display the output on the four 7-segment LEDs.

FOREVER:

Step 1. Register X is used as an index register. In the beginning, it is assigned the address of the first element in the LED data table.

LOOP:

Step 2. The data for **PORT B** is loaded using an index register X. Remember, PORT B is a memory location, so you need to use the instruction MOVB as below:
MOVB < first operand>, < destination operand>

Step 3. Then, register X is incremented to get the address of the next element in the LED data table.

Step 4. The data for **PORT P** is loaded using an index register X. This enables one of the 7-segment LEDs. For example, if the data for PORT P is \$0D (or %00001101), then LED #1 will be enabled.

Step 5. Then, register X is incremented to get the address of the next element in the LED data table.

Step 6. Delay: Wait 0.01 *seconds*. **Write a subroutine that delays Y ms.** The parameter for the delay subroutine is passing by register Y. Before you call the delay subroutine, load the value you want to delay into register Y. The value for the delay has the *millisecond* (ms) unit and 1000ms =1 second.
Ex: **LDY** #1 ; 1 ms is loaded into register Y.

Step 7. Compare register X with the address of the last element of the LEDdata.

Step 8. If register X (holds the address of the array element) is lower than the address of the last location of the LEDdata, then go to the inner loop labeled as '**LOOP'** to continue to display the next digit on the next LED. If not, then go to the beginning of the outer loop labeled as '**FOREVER'** to start the display from the beginning as below:

BLO LOOP

BRA FOREVER

6. After completing your code, click on the green **debug arrow** on the menu bar to start the CODE Warrior simulator/Debugger.
   a) Add two breakpoints at the beginning of the inner loop, '**LOOP'** and the outer loop, **'FOREVER'**. Breakpoints are set by right-clicking on the instruction and selecting "Set Breakpoint".
   b) In Lab6, **the program counter (PC) in the register window starts at $4000.** The instruction starting at the memory location $4000 is to initialize the stack pointer as below:
      LDS #RAMEnd+1
   c) Run the program from the beginning by clicking on the "Start/Continue" button. Note that when the processor stops at a breakpoint, the instruction that the PC points to has not been executed.

Task1: Component Identification: Identify the components (the peripheral modules on the HCS12 microcontroller) that you need to solve this problem. Please be as specific as possible; e.g., if you need 8 pins of PORT B, mention it here along with their directions. Also, briefly explain what they are used for, e.g., PTJ enables or disables eight flashing LEDs. Also, show any data declaration and initialization that you need for the initialization of your system.

Task2: Draw a flowchart of the main program and subroutine program. The flowchart demonstrates your program logic flow. If you need some help drawing a flowchart, an example is in Fig. 1 in the previous lab5 handout.

Task3: To analyze and validate your code using the CodeWarrior simulator, check the memory locations $0001 (PORT B) and $0258(PORT P) in the memory pane. For each inner loop iteration, write down the values in the memory locations (at $0001 and $0258) in Table 1 and generate the memory screenshots. The sample screenshots of the memory map are provided in Figure 4.

Table 1. Memory map

Iteration	Memory $0001 (PORT B)	Memory $0258 (PORT P)
1		
2		
3		
4		

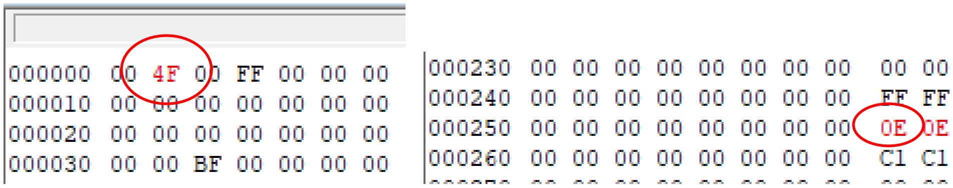


Fig. 4. Screen shots of the Memory map

Task4: Change the data in the data section to display the number **7890 on the four seven-segment LEDs** in the Dragon 12 + board. Repeat step 7 and check the memory locations $0001 (PORT B) and $0258(PORT P). For each inner loop iteration, write down the values in the memory locations (at $0001 and $0258) in Table 2 and **generate the screenshots of the memory**.

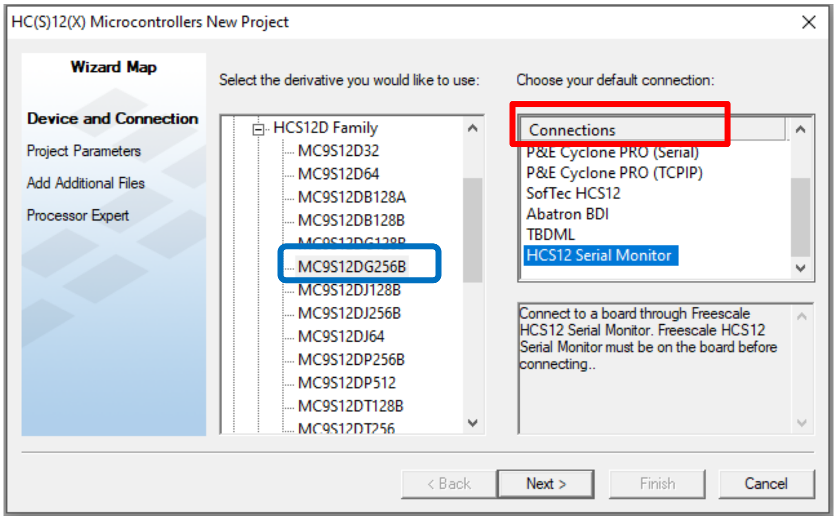
Table 2. Memory map

Iteration	Memory $0001 (PORT B)	Memory $0258 (PORT P)
1		
2		
3		
4		

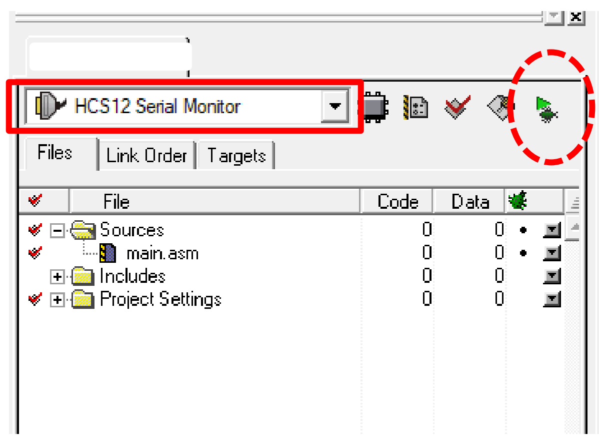
Part II. Download and execute programs on the Dragon12+ board

This Part II is intended to familiarize you with Wytec's **Dragon12+ evaluation board. The Dragon12+ board runs the program that CodeWarrior communicates with over the RS-232 monitor.** This allows the PC to control the Freescale HCS12 in a debugging environment.

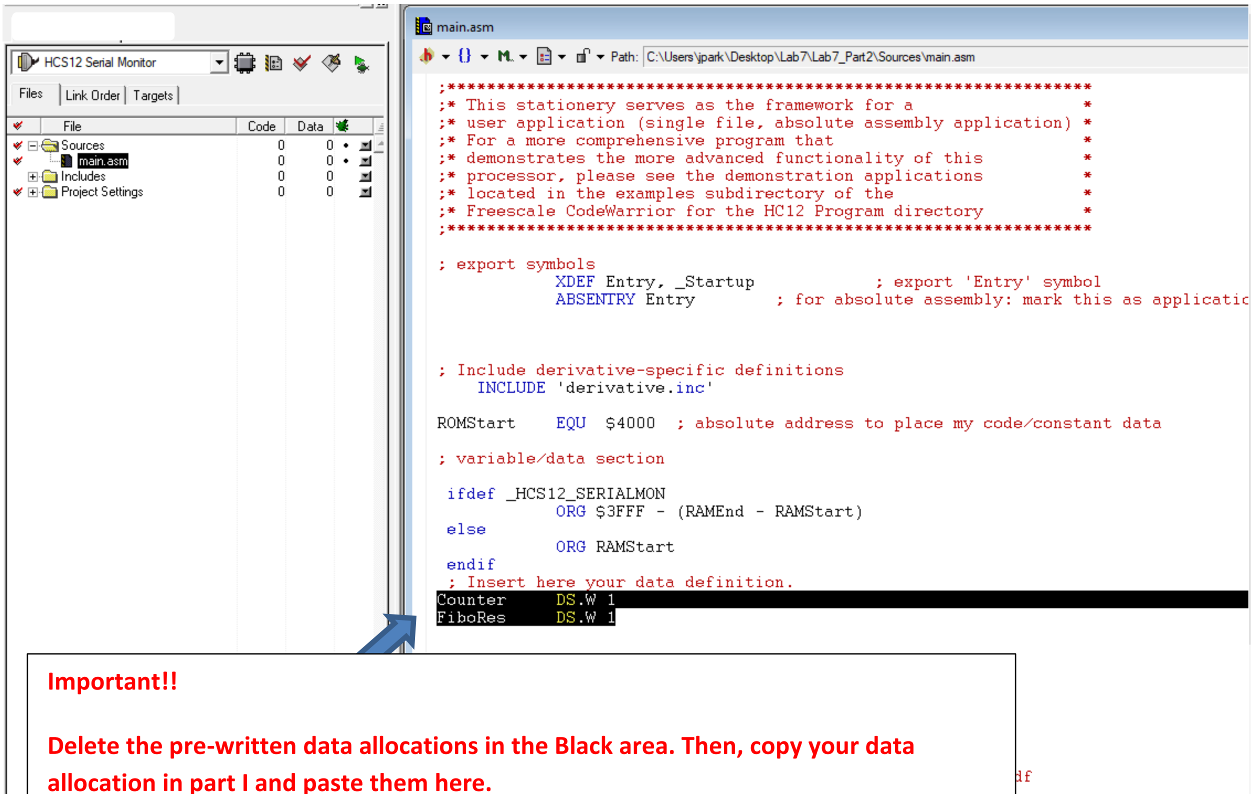
1. Create a new project by clicking the 'Create New Project' button. In the HC(S)12(X) Microcontrollers New Project wizard window, select the same processor ("**HCS12**" →"**HCS12D Family**" → "**MC9S12DG256B**"). In the Connections pane, select "**HCS12 Serial Monitor**", as shown in the below figure. Click Next.



2. In the **Project name** field, type **Lab6_Part2.mcp**. Uncheck all the checkboxes and then check the **Absolute assembly** checkbox. Click on **Finish** (NOT Next!).
3. CodeWarrior main window opens up, as shown below. From the dropdown list on the left, select "**HCS12 Serial Monitor**" if it is not so yet.



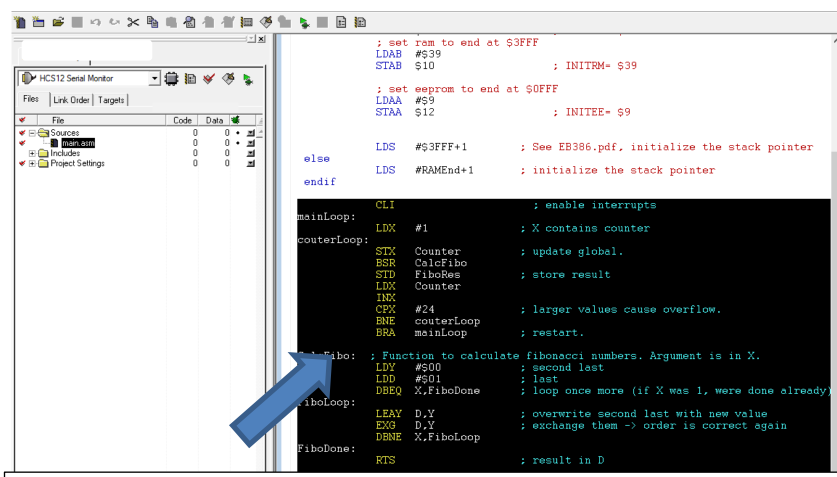
4. Delete the prewritten data allocation marked with the black background. Then, copy your data allocations in Part I and paste them here.



Important!!

Delete the pre-written data allocations in the Black area. Then, copy your data allocation in part I and paste them here.

5. Delete the prewritten sample code marked with the black background in main.asm provided by the CodeWarrior IDE.

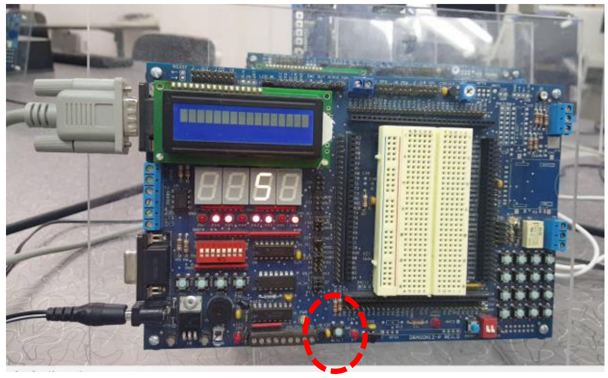


Delete the pre-written code in the Black area: CLI … …….. DC.W Entry ; Reset Vector

Copy your lines of code in Lab6_partI starting from the I/O port configuration to the end of your code and paste them here.

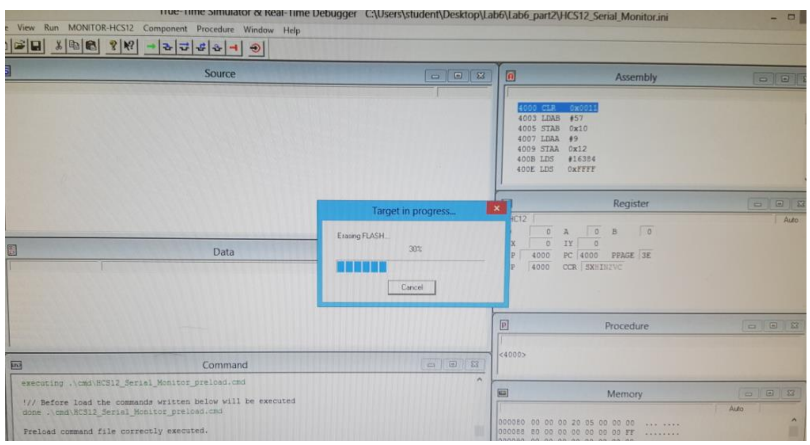
Copy your lines of code in Lab6_partI starting from the I/O port configuration to the end of your code and paste them here: (BSET ….. RTS).

6. **Press the light blue RESET button on the Dragon12+ board** located along the bottom edge in the middle of the board. The 8 red LEDs should light up from right to left, indicating that the CodeWarrior firmware is running on the evaluation board.

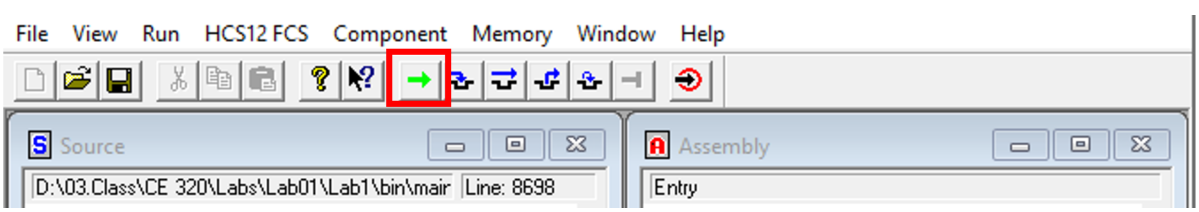


7. Then, click on the "debug" button, shown with a green arrow. This should assemble the program and open a second window. As this second window opens, **CodeWarrior will automatically download the machine code into the Dragon12+ board**.

8. Then, click on the "debug" button, shown with a green arrow. This should assemble the program and open a second window. As this second window opens, **CodeWarrior will automatically download the machine code into the Dragon12+ board**.



9. Run the program by clicking on the "Start/Continue" button.



10. Run your program **three times with different delays, 1ms, 100ms, and 1000ms**. In addition, generate the short video clips showing the four 7-segment LED displays with varying delay values.

What to Submit in Blackboard:

1) **Per each member: 50pts**

   Complete Lab6_WorkBook.doc :

   - **(10 pts) Task1: Component Identification**
   - **(20 pts) Task2: Flowchart**
   - **(10 pts) Task3: Memory-mapped I/O**
   - **(10 pts) Task4 :Memory mapped I/O**

2) **One copy per group : 50 pts**

   - 20 pts: The assembly source code for Part II (main.asm files only)
   - 30 pts: three short video clips that show the four 7-segment LED displays with 1ms, 100ms, and 1000ms delays, respectively.

   - **Pay attention to the file name convention:**

   - **Individual file: Lab6_Student1_Firstnname_Lastname.pdf**
     Ex) Lab6_Smith_Green.pdf

   - **Group file:**
     Source file: main.asm
     Video files:
     1ms _Student1 Lastname_Student2 Lastname.mp4
     100ms _Student1 Lastname_Student2 Lastname.mp4
     1000ms _Student1 Lastname_Student2 Lastname.mp4