

Low Power Lab

Description: The purpose of this lab is to show some of the low power operating modes of the M16C Tiny Family. This lab will demonstrate placing the device into a low power operating mode where the MCU is in Wait mode and wakes up once every second to update a real-time clock value then goes back to a low power state. The lab will also look at some common problems when entering and exiting low power mode. Finally, the lab exercise will conclude with an examination of the flexibility of the clock configuration of the processor when using low power modes.

Objectives

1. Show the low-power modes of the Renesas M16C/Tiny MCUs.
2. Perform actual measurements to validate hardware manual data
3. Show some common problems when entering and exiting a low-power mode.
4. Examine potential power robbing settings in a typical design.

Lab Materials:

Please verify you have the following materials at your lab station.

- Laptop PC with Renesas tools installed
- RSK/29 Demo Platform
- Multimeter and probes
- This lab sheet

Skill Level: Basic understanding of MCU operating modes. Experience in basic bench top measuring equipment recommended. Knowledge of operating the HEW IDE and Renesas tools recommended

Time to Complete Lab: 50 minutes

Lab Sections

- | | |
|---|--|
| <div style="display: flex; align-items: center;"> <div style="background-color: #006633; color: white; border-radius: 50%; width: 30px; height: 30px; display: flex; align-items: center; justify-content: center; margin-right: 10px;">1</div> <div> <h3>Simple Low Power Demonstration</h3> </div> </div> | <p>Time to complete task: 10 minutes</p> |
| <div style="display: flex; align-items: center;"> <div style="background-color: #ff0000; color: white; border-radius: 50%; width: 30px; height: 30px; display: flex; align-items: center; justify-content: center; margin-right: 10px;">2</div> <div> <h3>Using HEW with the Low Power Demonstration</h3> </div> </div> | <p>Time to complete task: 10 minutes</p> |
| <div style="display: flex; align-items: center;"> <div style="background-color: #0000ff; color: white; border-radius: 50%; width: 30px; height: 30px; display: flex; align-items: center; justify-content: center; margin-right: 10px;">3</div> <div> <h3>Some Common Low Power Issues</h3> </div> </div> | <p>Time to complete task: 10 minutes</p> |
| <div style="display: flex; align-items: center;"> <div style="background-color: #006633; color: white; border-radius: 50%; width: 30px; height: 30px; display: flex; align-items: center; justify-content: center; margin-right: 10px;">4</div> <div> <h3>Some Other Low Power Numbers</h3> </div> </div> | <p>Time to complete task: 10 minutes</p> |

1


Title of Lab Section 1

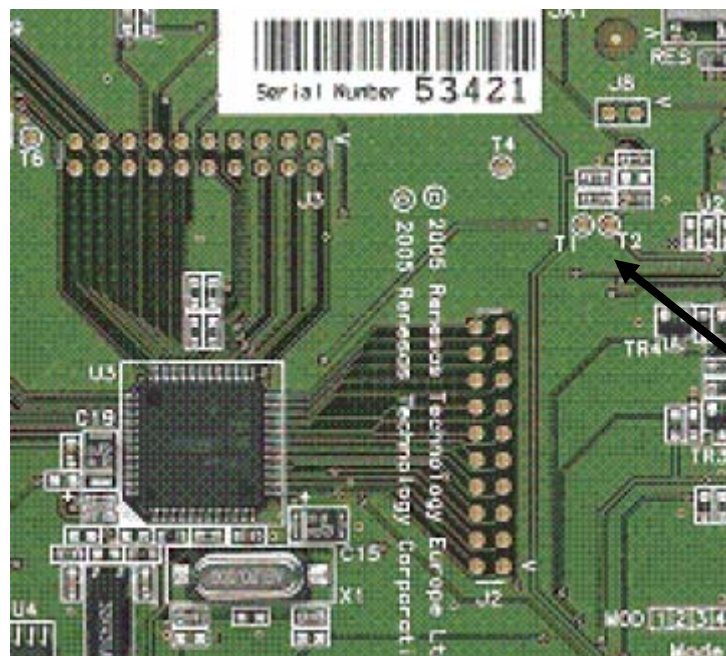
Time to complete task: 10 minutes

Overview:

This section will show a short demo of WAIT mode using the M16C/29 RSK platform. As a part of this lab section, the effect of floating I/O ports on the Icc current will be shown.

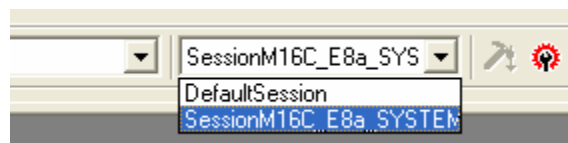
Procedural Steps

1. Start HEW by clicking on the HEW icon on the Windows desktop.
2. Using File->Open Workspace from the HEW menubar, load the "**Low_Power_1A.hws**" project from the **C:\WorkSpace\Low_Power\Low_Power_1A**.
3. Build the project and verify there are no errors using **Build->Build All** from the HEW menubar. Alternatively, you can use the  button on the HEW toolbar.
4. Remove the Icc measurement jumper and place the multimeter across the pins (the Icc jumper is between T1 and T2 on the RSK29). See the reference picture below for approximate location of the jumper. Set the multimeter for a 20mA scale DC current reading.

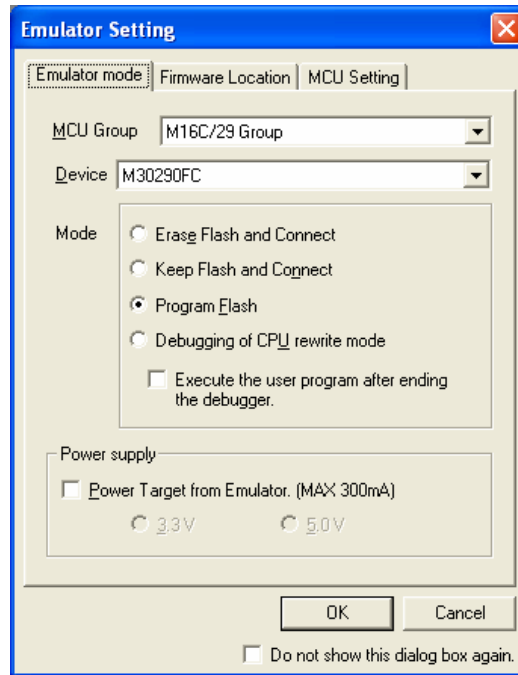


T1-T2 Jumper

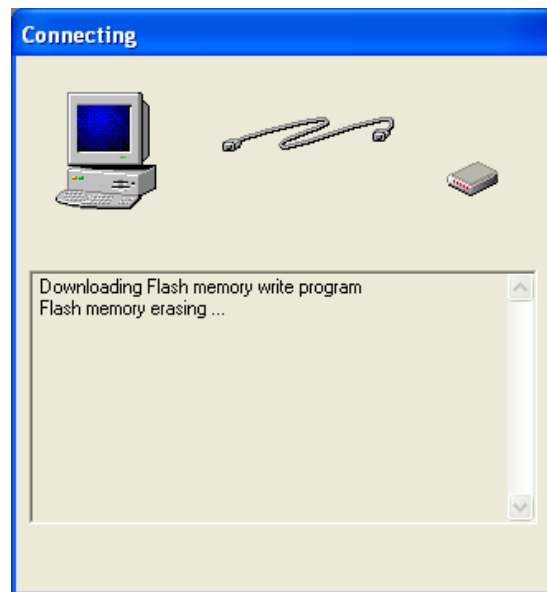
5. Returning to the HEW environment, now program the Low_Power_1A.mot file into the board. From the HEW toolbar, select the SessionM16C_E8a_System session from the session dialog.



- Once the session starts, a dialog box will appear. In the Dialog box, select the radio button 'Program Flash'. No other selections are necessary for this lab section. Select 'OK' at the bottom of the dialog box.

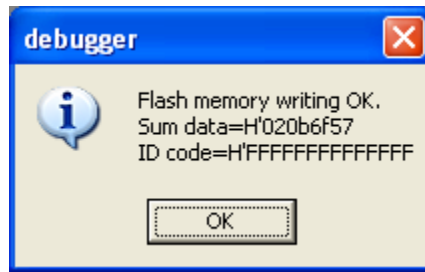


The debugger will now connect to the target board.

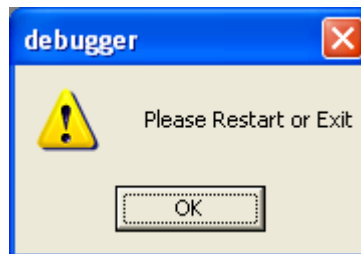



- To download the .mot file to the target, select Debug->Download Modules->All Modules from the HEW toolbar. Alternatively, you can double-click on the Low_Power_1A.x30 file in the left-hand pane of the HEW environment.

8. A dialog box similar to below will appear at the end of the programming process. Click ok.



9. Another dialog box will appear asking you to exit the debugger. Click ok.



10. Disconnect the E8a from the target board by selecting Debug->Disconnect from the HEW menubar. Alternatively, you can click on the  icon on the HEW toolbar.
11. Remove the E8a connector from the RSK29 board. The program should now be running. The LED0 should be blinking. If it is not, press and release the RESET button on the RSK29 board.

Questions:

- 1: What is Icc current shown on the meter?
 2: Is the current steady?

12. Press and release SW2. The MCU will go to low power mode; the current on the meter should drop to a very low value and the LEDs will go out. (depending on the meter you may just see 0 current)
13. Press SW1, this will place the device back in a RUN mode. The LED1 should be lit and LED0 will be blinking again.

Questions:

- 3: What is Icc current shown on the meter?
 4: Is the current acting the same as in step 12?

2


Second Lab Section Title

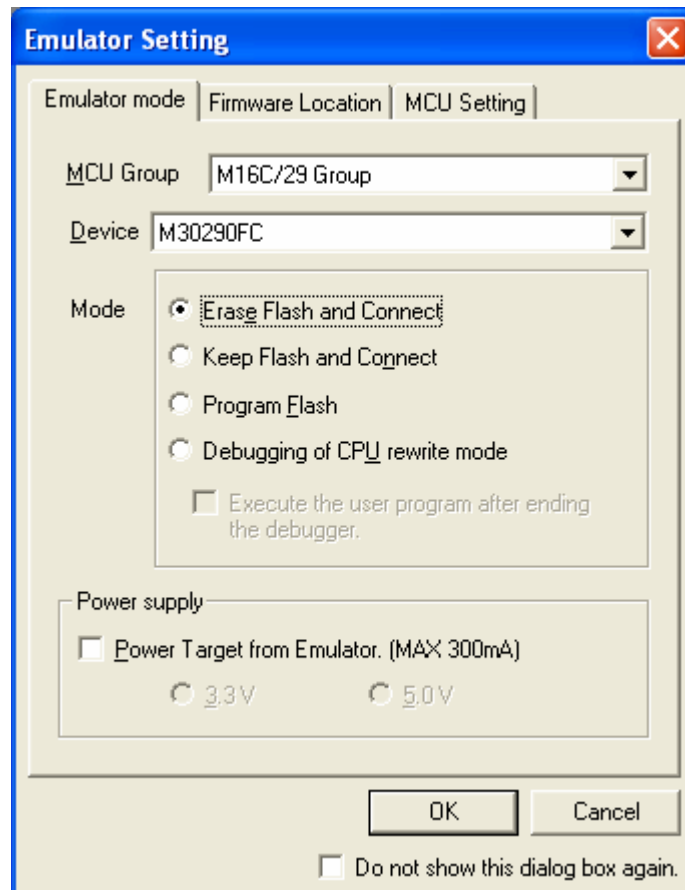
Time to complete task: 10 minutes


Overview:




Using the E8a this section of the lab exercise will show that low power modes can be debugged real-time.

Procedural Steps:

1. Reconnect the E8a debugger to the RSK29 board. The program execution will stop once the connector is plugged in. This is because the insertion of the E8a connector places the processor into RESET.
2. Connect to the debug session using Debug->Connect from the HEW menubar. Alternatively, you can choose the  icon from the HEW toolbar.
3. Once the connection is established, choose the "Erase Flash and Connect" radio button in the dialog box. No other settings will need to be modified. HEW will now connect to the target.



4. To download the .mot file to the target, select Debug->Download Modules->All Modules from the HEW menubar. Alternatively, you can double-click on the Low_Power_1A.x30 file in the left-hand pane of the HEW environment.
5. The debugger will now reset the target MCU and the debugger window will be at the start of code execution.
6. Run the program using Debug->Reset Go from the HEW menubar, or alternately using the  icon from the HEW toolbar.




7. Press and release SW2. The MCU will go to low power mode; the current on the meter should drop to a very low value and the LEDs will go out. (depending on the meter you may just see 0 current)
8. Press SW1, this will place the device back in a RUN mode. LED1 should be lit and LED0 will be blinking again.
9. Stop the debugger by clicking the  icon on the toolbar. Alternately, you can use Debug->Stop from the menubar. The program will stop normally
10. Start the debugger again from this point using the  icon from the toolbar. Alternately, you can use Debug->Go from the menubar. You can press SW1 and SW2 and the program will toggle between the low power and full speed modes
11. Press and release SW2 so the device is in the low power mode. LED0 should stop blinking.
12. Stop the debugger by clicking the  on the toolbar. Debugging will stop at the instruction immediately after the WAIT command in the HEW window. It will look as follows:

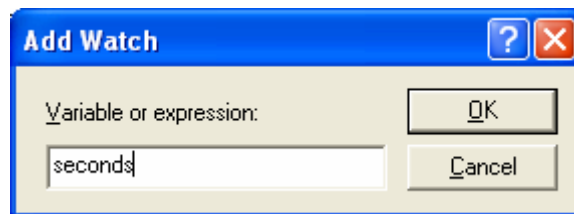
```


while (mode == LOW_PWR) { /*goto WAIT mode and loop back into WAIT mode after
                           timer interrupt service routine returns */
_asm ("wait");
_asm ("nop"); /*put nop's in to make sure queue is clear */
_asm ("nop");
_asm ("nop");
_asm ("nop");

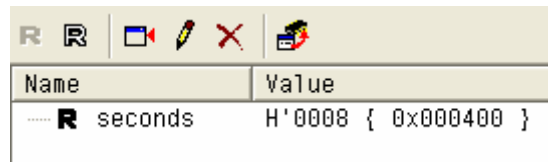
```

The yellow line and arrow indicate where the processor stopped executing code.

13. Reset the debugger by clicking the  icon on the HEW toolbar. Alternately, you can use Debug->Reset from the HEW menubar.
14. Open a C Watch window by using View->Symbol->Watch from the HEW toolbar. Alternately, you can click the  icon from the HEW toolbar. A watch window will appear in the HEW environment.
15. Add a watch variable to the Watch window by clicking the  icon in the watch window.
16. An 'Add Watch' window will appear. Enter a variable named **seconds** into the dialog box and click 'OK'. The variable will appear in the watch window.



17. Click on the 'Auto Update' icon  in the Watch window. The 'R' symbol will turn black in the watch window. This will auto-update all variable values in the window while the code is running.





18. Run the program. You should see that the variable is incrementing.
19. Press and release SW2. The MCU will go to low power mode; the current on the meter should drop to a very low value and the LEDs will go out. (depending on the meter you may just see 0 current)

Questions:


5: Is the variable still incrementing?

6: Is the current steady?

Although the device is in low power mode, with the auto-update feature enabled, the debugger wakes the device, and thus exits low power mode briefly, to update the variable. This causes the change in current. The device then returns to low power mode.

20. Press SW1, this will place the device back in a RUN mode. LED1 should be lit and LED0 will be blinking again.
21. Stop the debugger by clicking the  icon on the toolbar. Alternately, you can use Debug->Stop from the menubar.
22. Click on the 'Delete Auto Update' icon  in the Watch window. The 'R' symbol will turn white in the watch window. This will turn off auto-update all variable values in the window while the code is running.



23. Start the debugger again from this point using the  icon from the toolbar. Alternately, you can use Debug->Go from the menubar. Note the last value of seconds in the watch window and then push SW2 on the RSK. Wait a little while (20-30 seconds) then press SW1 to wake the device again.
24. Stop the debugger. The watch window will refresh and you will see that the Seconds variable has been counting.

Questions:

5: How much time will elapse before the Seconds variable overflows ?

6: How can you quickly check that the seconds_overflow variable gets incremented properly?


3 Questions

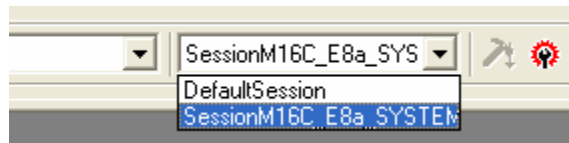
Time to complete task: 10 minutes

Overview:

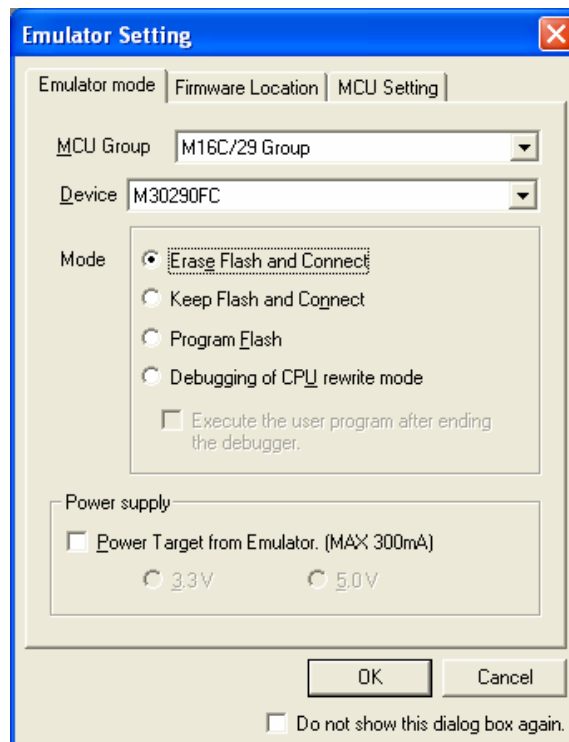
This lab section will show the effects of floating I/O ports.

Procedural Steps:


1. Using File->Open Workspace from the HEW toolbar, load the "**Low_Power_1B.hws**" project from the **C:\Workspace\Low_Power\Low_Power_1B** directory. This project is basically the same project as the "low_power_lab1a" with a few issues
2. Build the project and verify there are no errors using **Build->Build All** from the HEW menubar. Alternatively, you can use the  button on the HEW toolbar.
3. Now connect the debugger to the board. From the HEW toolbar, select the SessionM16C_E8A_System session from the session dialog.

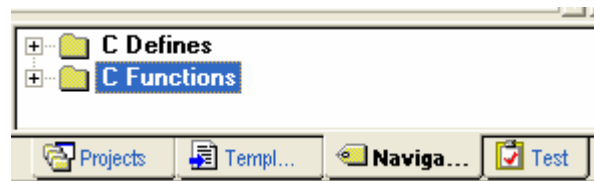


4. Once the connection is established, choose the "Erase Flash and Connect" radio button in the dialog box. No other settings will need to be modified. HEW will now connect to the target.

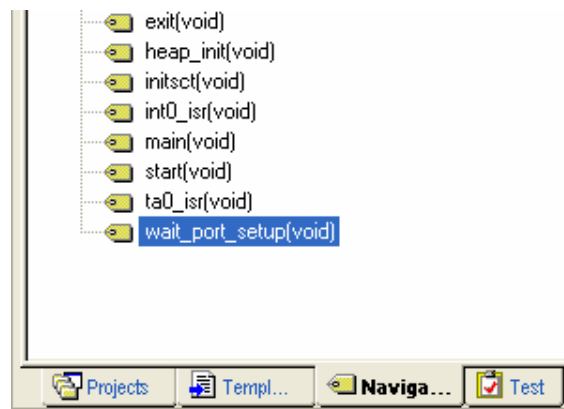


The debugger will now reset the target MCU and the debugger window will be at the start of code execution.

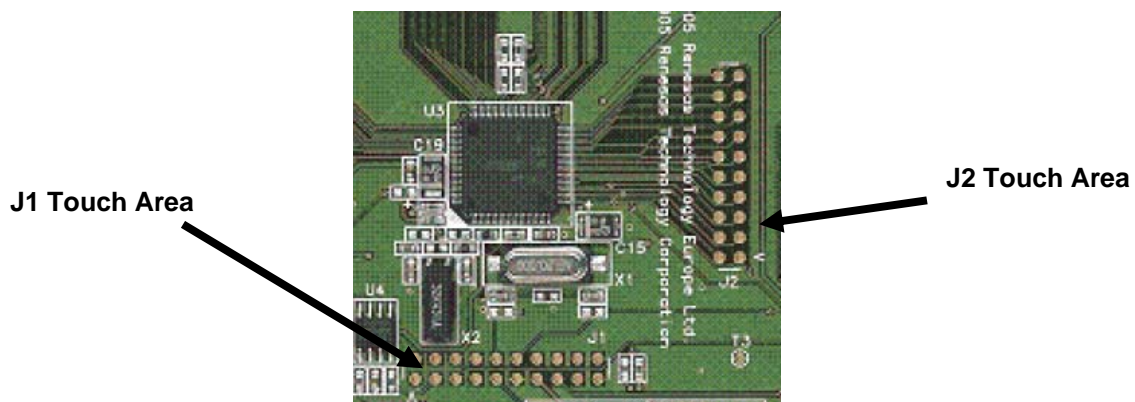
5. To download the .mot file to the target, select Debug->Download Modules->All Modules from the HEW toolbar. Alternatively, you can double-click on the Low_Power_1A.x30 file in the left-hand pane of the HEW environment.
6. The debugger will now reset the target MCU and the debugger window will be at the start of code execution. Run the program using Debug->Reset Go from the HEW menubar, or alternately using the  icon from the HEW toolbar
7. Press SW2 - You should notice that the current does not go to the same low value as it did
8. Find the function **wait_port_setup(void)** routine in the code. This can be done by clicking on the 'Navigation' tab in the workspace pane of the HEW environment.



9. Expand the **C Functions** list to show a list of all functions in the current workspace. Double click on the function name to directly view it in the main HEW window.





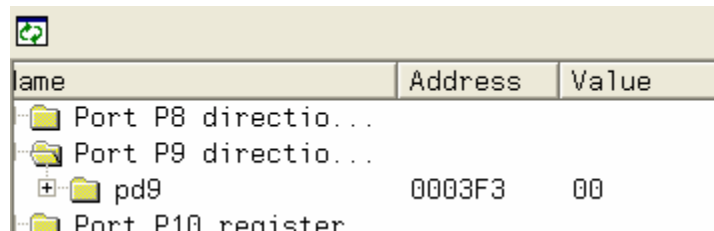
10. Notice that the GPIO ports are all driven as outputs and low unless they are connected to a pull-up or pull-down as shown in the code comments.
11. Place a finger on the areas shown below for J1 and then on J2.



Questions:

- 5: What happens when the J1 area is touched? How about the J2 area?
- 6: When you remove your finger from the J1 area, what happens to the current on the meter?

12. P9 of the MCU is floating (the pins associated with the touch area of J2); once you remove your finger from the connector you should see the current gradually rising again. In the code P9 is set as an output and low. It appears that these pins are floating.
13. Press SW1, this will place the device back in a RUN mode. LED1 should be lit and LED0 will be blinking again.
14. Stop the debugger by clicking the  icon on the toolbar. Alternately, you can use Debug->Stop from the menubar.
15. You can check the status of the port9 data direction register by accessing the MCU IO window by clicking View->CPU->IO. Alternately, you can click the  icon from the HEW toolbar. Note, you may need to resize the window by dragging the top of the I/O window pane.
16. Scroll thru the list of registers until you see the Port P9 Direction register. Expand the tab and use the slider bar until you can see the value of the P9 direction register. Notice that it is not the value being written to the register as shown in the code.



Name	Address	Value
Port P8 direction...		
Port P9 direction...		
pd9	0003F3	00
Port P10 register		



17. Fix the code by adding the following line of code to the project


```
prc2 = 1;
```

between these two lines of code in the **wait_port_setup** routine

```
p9=0x00;
pd9 = 0xff;
```

Port 9 direction register is protected from inadvertent accesses by a protection bit '**prc2**'. This protection bit must be set to modify the direction register. This bit automatically clears after the next write to memory space so do not set breakpoints or try to step through this command.

18. Rebuild the the project and verify there are no errors using **Build->Build All** from the HEW menubar. Alternately, you can use the  button on the HEW toolbar.
19. The debugger will now reset the target MCU and the debugger window will be at the start of code execution. Run the program using Debug->Go from the HEW menubar, or alternately using the  icon from the HEW toolbar

20. The current should now drop to the proper level when SW2 is pressed. Again, touching the J1 area should cause no current fluctuation on the meter.
21. Push SW1 so the device is in RUN mode. Notice the RUN current is lower than before and the LED0 is not blinking as fast.
22. Stop the debugger by clicking the  icon on the toolbar. Alternately, you can use Debug->Stop from the menubar.
23. Find the **int0_isr(void)** function in the **Low_Power_1B.c** source file. Notice the code re-starts the main oscillator and switches to the main clock, why is it running slower?



```
void int0_isr (void){
    unsigned char delay_cntr = 0x20;
    p7_2 = 0;          // turn green led on;
    cm05=0;           // turn on main clock
    while (delay_cntr >0) delay_cntr--;           // let main oscillator stabilize
    cm07 = 0;         // main clock as cpu clock
    mode = RUN;
}
```

Renesas M16C MCUs, after resuming from WAIT mode, automatically divide the processor clock by 8. To return the processor clock to full speed, simply add the following line:

```
cm06 = 0;
```

between these two lines of code in the **int0_isr**

```
cm07 = 0;
mode = RUN;
```

24. Rebuild the the project and verify there are no errors using **Build->Build All** from the HEW menubar. Alternatively, you can use the  button on the HEW toolbar.
25. The debugger will now reset the target MCU and the debugger window will be at the start of code execution. Run the program using Debug->Go from the HEW menubar, or alternately using the  icon from the HEW toolbar. This will return the blink rate of LED0 to the original frequency of that in the Section1 lab code.

4

Some other Guidelines


Time to complete task: 10 minutes

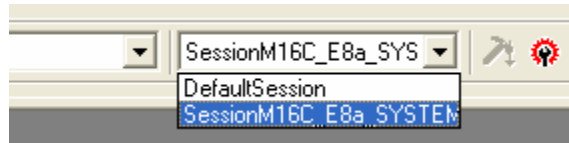
Overview:

This lab section will show the lcc difference in leaving the main processor clock off or on and running the device from the sub clock. Additionally we will turn off the peripheral clock in WAIT mode.

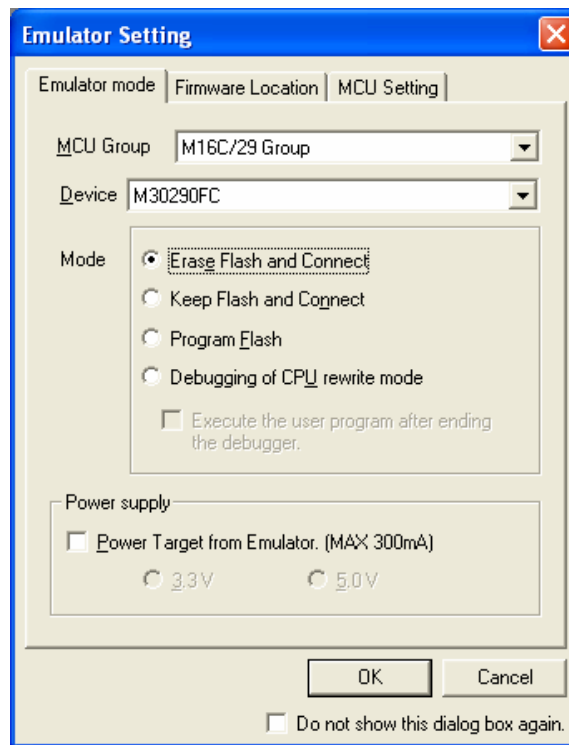
Procedural Steps:

1. Using File->Open Workspace from the HEW toolbar, load the "**Low_Power_2.hws**" project from the **C:\WorkSpace\Low_Power\Low_Power_2**. This project is basically the same project as "low_power_lab1a" but the code has been modifies so it does not restart the main clock after leaving low power mode


- Build the project and verify there are no errors using **Build->Build All** from the HEW menubar. Alternatively, you can use the  button on the HEW toolbar.
- Now connect the debugger to the board. From the HEW toolbar, select the SessionM16C_E8A_System session from the session dialog.



- Once the connection is established, choose the “Erase Flash and Connect” radio button in the dialog box. No other settings will need to be modified. HEW will now connect to the target.




The debugger will now reset the target MCU and the debugger window will be at the start of code execution.

- To download the .mot file to the target, select **Debug->Download Modules->All Modules** from the HEW toolbar. Alternatively, you can double-click on the Low_Power_1A.x30 file in the left-hand pane of the HEW environment.
- The debugger will now reset the target MCU and the debugger window will be at the start of code execution. Run the program using **Debug-> Go** from the HEW menubar, or alternately using the  icon from the HEW toolbar
- Press and release SW2. The MCU will go to low power mode; the current on the meter should drop to a very low value and the LEDs will go out. (depending on the meter you may just see 0 current)
- Press SW1, this will place the device back in a RUN mode. The LED1 should be lit and LED0 will be blinking very slowly.

Questions:

3: What is lcc current shown on the meter?

The current differential between this and other sections of this lab is because the processor remains on the 32kHz sub clock.

9. Stop the debugger by clicking the  icon on the toolbar. Alternately, you can use Debug->Stop from the menubar.
10. Next mode we will put the device into WAIT mode with the main clock still oscillating, In the **wait_mode2.c** source file section of code shown below

```

/* setup clocks for wait conditions */
        cm03= 0;        //low drive on sub-clock
//        cm02 = 1;        // turn off peripheral clock in WAIT mode
        cm07=1;        // sub-clock as main clock
        cm05=1;        // shut off main clock



```

modify the code so it is as shown below

```


/* setup clocks for wait conditions */
        cm03= 0;        //low drive on sub-clock
//        cm02 = 1;        // turn off peripheral clock in WAIT mode
//        cm07=1;        // sub-clock as main clock
//        cm05=1;        // shut off main clock

```

11. Rebuild the the project and verify there are no errors using **Build->Build All** from the HEW menubar. Alternately, you can use the  button on the HEW toolbar.
12. The debugger will now reset the target MCU and the debugger window will be at the start of code execution. Run the program using Debug->Go from the HEW menubar, or alternately using the  icon from the HEW toolbar.
13. Press and release SW2. The MCU will go to low power mode.

Questions:

3: What is lcc current shown on the meter?

14. Press SW1, this will place the device back in RUN mode. The LED1 should be lit and LED0 will be blinking very slowly.
15. Stop the debugger by clicking the  icon on the toolbar. Alternately, you can use Debug->Stop from the menubar.

16. The last mode we will look at will put the device into WAIT mode with the main clock and the peripheral clocks turned off, In the MAIN function find the section of code shown below

```

/* setup clocks for wait conditions */
    cm03= 0;      //low drive on sub-clock
//    cm02 = 1;    // turn off peripheral clock in WAIT mode
    cm07=1;      // sub-clock as main clock
    cm05=1;      // shut off main clock



```

modify the code so it is as shown below

```


/* setup clocks for wait conditions */
    cm03= 0;      //low drive on sub-clock
    cm02 = 1;    // turn off peripheral clock in WAIT mode
    cm07=1;      // sub-clock as main clock
    cm05=1;      // shut off main clock

```

17. Rebuild the the project and verify there are no errors using **Build->Build All** from the HEW menubar. Alternatively, you can use the  button on the HEW toolbar.
18. The debugger will now reset the target MCU and the debugger window will be at the start of code execution. Run the program using Debug->Go from the HEW menubar, or alternately using the  icon from the HEW toolbar.
19. Press and release SW2. The MCU will go to low power mode.

Questions:

3: What is lcc current shown on the meter?

20. Press SW1, this will place the device back in RUN mode. The LED1 should be lit and LED0 will be blinking very slowly.
21. Stop the debugger by clicking the  icon on the toolbar. Alternately, you can use Debug->Stop from the menubar.

Answer Page

Section 1: Break the answers into their appropriate sections

1.1.) Repeat the question from the section

1.2.) Lines are a pain but necessary

Section 2: A few other items

2.1) Tables can be placed on the answer sheet where appropriate

Data	To	Fill	In

2.2) Make sure you leave enough room for someone to put in the answer
