Lab #5 - Watchdog timer

Leonardo Fusser (1946695)

Objectives:

- Install and setup a Watch Dog timer in a RTOS environment.
- Use persistent variables.

Material: Explorer16/32 and PIC32MX795F512L.

To hand in:

These sheets including answers to all questions on Teams.
C code on GitHub.
Relevant screenshots. Must include explanations.

Lab Work

GitHub:

- Make sure you are logged into GitHub. Copy-paste the following URL to accept an invitation for the lab.

https://classroom.github.com/a/dE1qW0CF

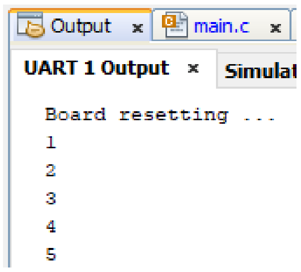
You will get a private new repository for lab1. You will push your project to this repo at the end of this lab.

https://github.com/Embedded-OS-Vanier/watchdog_timer-your_name

Part1: Up counter task

- Enable vTask1. Set its priority to 1.
- In vTask1 you must print out a 100 mS period counter to the UART1 console. The counter counts up from 0 and increments by 1 every 100mS.
- LED9 (RA6) must toggle at 100mS.

  See screenshot below:



```
Board resetting ...
1
2
3
4
5
```

Part2: Configuring the WDT

- Make sure your simulator's Fcyc is set at 40 MHz.
- Configure the watchdog timer for a 512 mS delay. The board must print out a message to C_UART1 whenever it resets (e.g., `reset`).

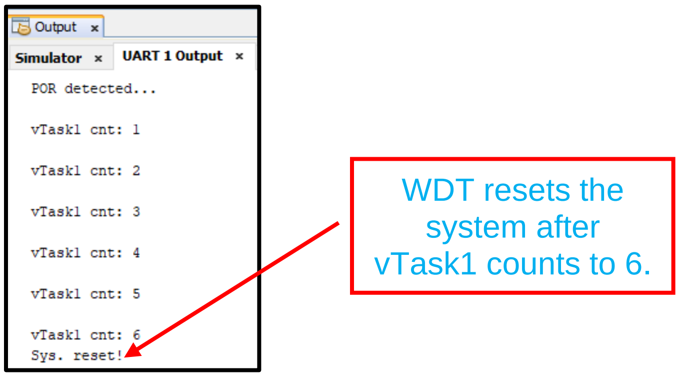
Program the board and test your code. What do you observe?

- ➢ vTask1 starts to count up, but after around 6 counts, the system resets itself. This behaviour repeats over and over.

  This is because the WDT never gets kicked. In our case, if the WDT hasn't been kicked within 512mS, it will reset the system.

You MUST provide a screenshot of the C_UART1 output tab.

- ➢ Refer to screenshot on next page.



```
POR detected...

vTask1 cnt: 1

vTask1 cnt: 2

vTask1 cnt: 3

vTask1 cnt: 4

vTask1 cnt: 5

vTask1 cnt: 6
Sys. reset!
```

Figure 1. Screenshot above shows vTask1 counting up but system resets after around 6 counts. WDT is not getting kicked which is causing this issue.

Give the value of the RCON register:

- The value of the RCON register has the WDTO (WDT time-out flag) set.

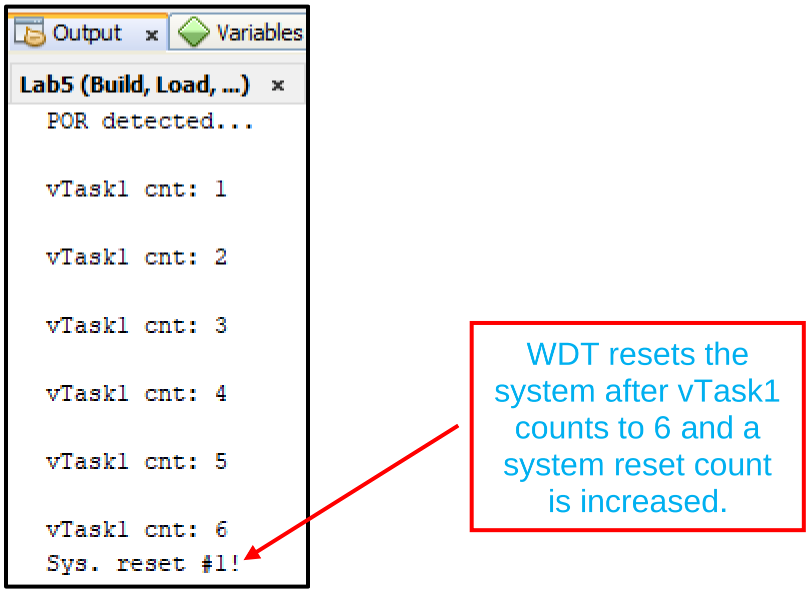
Explain the value.

- This is because when the WDT resets the system, the WDT time-out flag in the RCON register is set. The user has to manually clear the flag once it is set.

- Modify the code so a reset counter `resetCount` must count the number of resets. Again, the board must print out a message to C_UART1(e.g., `reset no.3`).

- Test your code. You MUST provide a screenshot of the UART1 output tab.

  - Refer to screenshot on next page.



```
POR detected...

vTask1 cnt: 1

vTask1 cnt: 2

vTask1 cnt: 3

vTask1 cnt: 4

vTask1 cnt: 5

vTask1 cnt: 6
Sys. reset #1!
```

WDT resets the system after vTask1 counts to 6 and a system reset count is increased.

Figure 2. Screenshot above shows added feature where every time the system is reset, a count is increased. This count keeps counting up every time the system is reset by the WDT.

When you do a "POR Reset" by clicking the reset the blue reset icon –

Explain what happens:

- When a POR occurs, a message is displayed indicating a POR has occurred, and the program starts back from the very beginning. This is similar to how a WDT resets the system.

Does it clear `resetCount` or does it continue to count up without clearing it?

- When a POR has occurred, the system reset count gets cleared. This is because it is deliberately cleared in software when the POR occurs.

Give the value of the RCON register after a “POR Reset”:

- The value of the RCON register has the POR (power-on reset flag) set.

Explain the value:

- Similar to how the WDTO flag is set when the WDT resets the system, a similar behaviour occurs when a POR happens. Instead of the WDTO flag being set, the power-on reset flag in the RCON register is set instead.

POR reset:

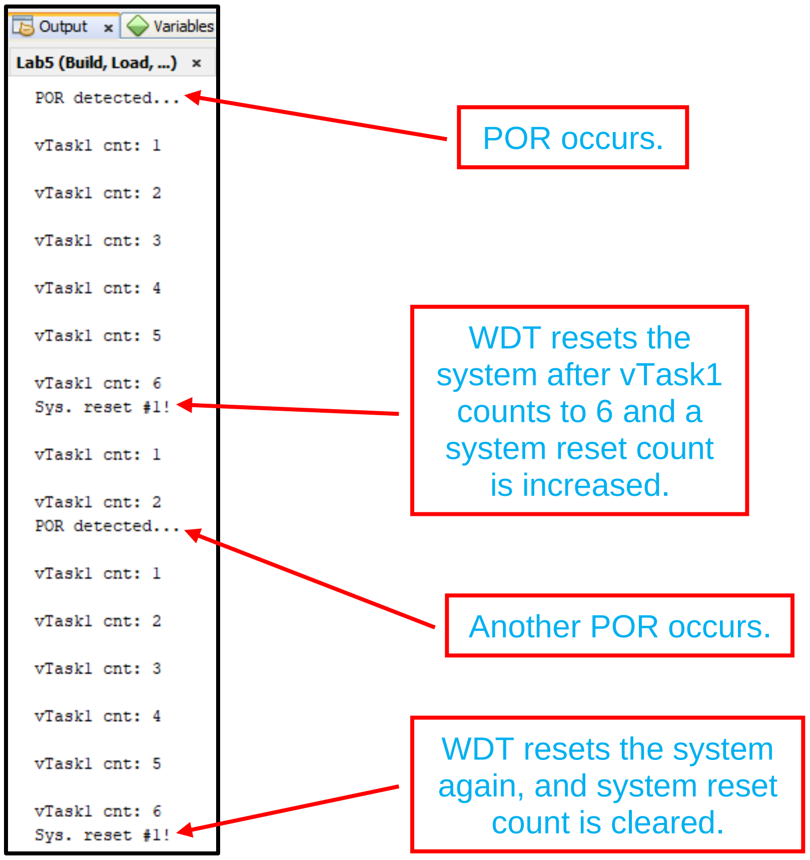
Improve your code so to clear your `resetCount` variable only on an “POR Reset”.

Explain what happens:

- When a POR occurs, the system reset count is reset back to zero. Until another POR occurs, the system reset count keeps increasing every time the WDT resets the system.

You MUST provide a screenshot of the UART1 output tab.

- Refer to screenshot on next page.


Output
Variables
Lab5 (Build, Load, ...)
POR occurs.
WDT resets the system after vTask1 counts to 6 and a system reset count is increased.
Another POR occurs.
WDT resets the system again, and system reset count is cleared.

```
POR detected...

vTask1 cnt: 1

vTask1 cnt: 2

vTask1 cnt: 3

vTask1 cnt: 4

vTask1 cnt: 5

vTask1 cnt: 6
Sys. reset #1!

vTask1 cnt: 1

vTask1 cnt: 2
POR detected...

vTask1 cnt: 1

vTask1 cnt: 2

vTask1 cnt: 3

vTask1 cnt: 4

vTask1 cnt: 5

vTask1 cnt: 6
Sys. reset #1!
```

Figure 3. Screenshot above shows when a POR occurs, the system reset count goes back to zero. Until another POR occurs, the system reset count keeps increasing every time the WDT resets the system.

Kick the dog

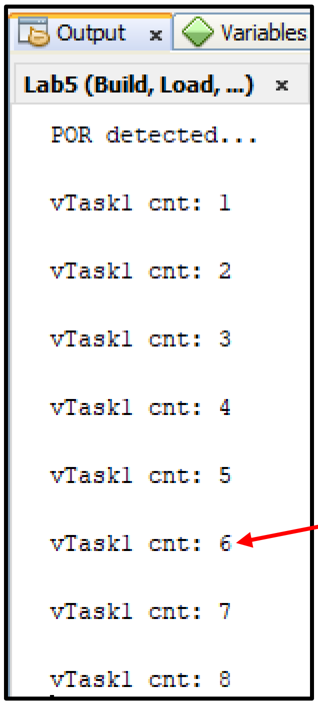
Now kick the dog in the IDLE HOOK task. Explain what happens:

- When the WDT gets kicked in the idle hook function, the system no longer gets reset. This is expected because the WDT knows now not to reset the system.

  With the WDT getting kicked in the idle hook function, the only time it can reset the system is if the system CPU is being hogged.

You MUST provide a screenshot of the UART1 output tab.

- Refer to screenshot on next page.



```
POR detected...

vTask1 cnt: 1

vTask1 cnt: 2

vTask1 cnt: 3

vTask1 cnt: 4

vTask1 cnt: 5

vTask1 cnt: 6

vTask1 cnt: 7

vTask1 cnt: 8
```

WDT no longer resets the system after vTask1 counts to 6.

Figure 4. Screenshot above shows that when the WDT is getting kicked in the idle hook function, the system does not get reset anymore. This is also because vTask1 does not hog the CPU, so it will give slack time to the system to run the idle hook function.

Part 3: Hogging the processor

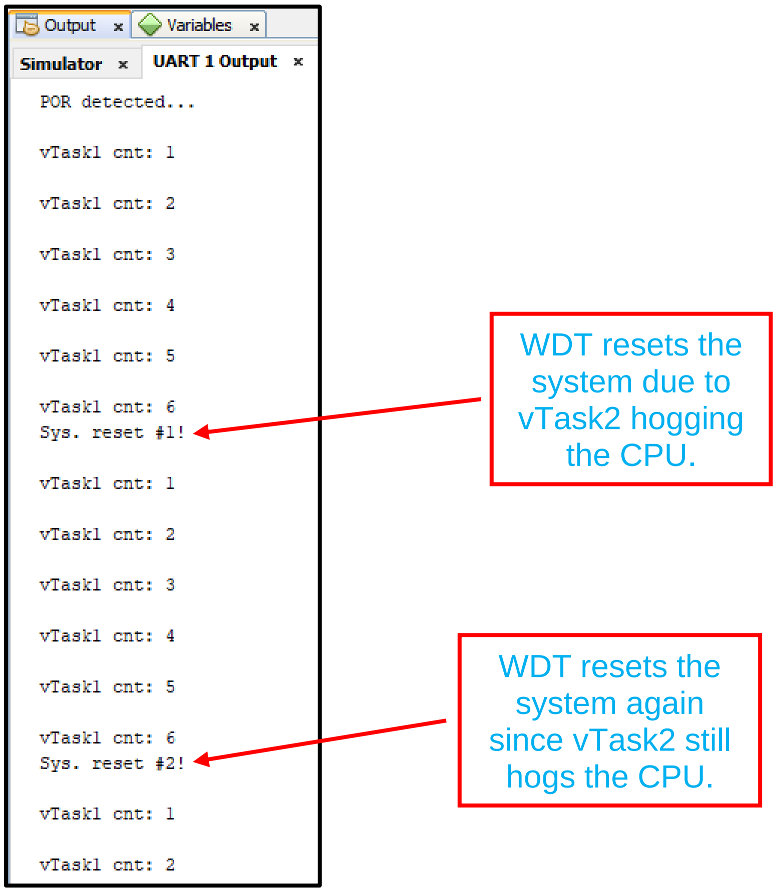
- Enable vTask2. Set its priority to 1.
- Inside this task add a blocking `while(1)` loop that will hog the CPU and give it no slack.

Explain what happens:

➢ The same behaviour as when the WDT was not getting kicked is occurring again here.

This is because the WDT is indeed not getting kicked anymore. When the CPU is hogged, the idle hook function cannot run and thus the WDT cannot be kicked. Therefore, since vTask2 is hogging the CPU, the WDT will reset the system after some time.

You MUST provide a screenshot of the UART1 output tab.


Output
Variables
Simulator
UART 1 Output
WDT resets the system due to vTask2 hogging the CPU.
WDT resets the system again since vTask2 still hogs the CPU.

```
POR detected...

vTask1 cnt: 1

vTask1 cnt: 2

vTask1 cnt: 3

vTask1 cnt: 4

vTask1 cnt: 5

vTask1 cnt: 6
Sys. reset #1!

vTask1 cnt: 1

vTask1 cnt: 2

vTask1 cnt: 3

vTask1 cnt: 4

vTask1 cnt: 5

vTask1 cnt: 6
Sys. reset #2!

vTask1 cnt: 1

vTask1 cnt: 2
```

Figure 5. Screenshot above shows that when vTask2 runs, it will hog the system, which in turn will force the WDT to reset the system after some time. The system reset count increases by 1 each time this occurs.

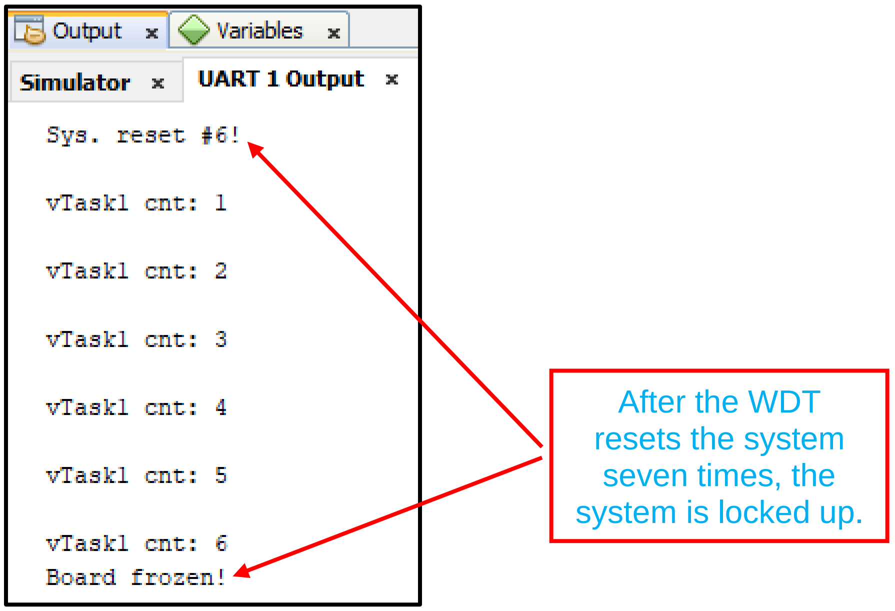
- Improve your code by freezing the CPU after seven (7) WDT resets. You must display a message accordingly:

```
Board frozen
```

Explain what happens:

➢ Essentially, with this new feature added, the system reset count will wait until seven system resets have occurred. Once occurred, the system will lock up and the message "Board frozen!" will show on the console.

You MUST provide a screenshot of the UART1 output tab.



```
Sys. reset #6!

vTask1 cnt: 1

vTask1 cnt: 2

vTask1 cnt: 3

vTask1 cnt: 4

vTask1 cnt: 5

vTask1 cnt: 6
Board frozen!
```

After the WDT resets the system seven times, the system is locked up.

Figure 6. Screenshot above shows that the system will keep running and resetting until seven system resets have occurred. Once occurred, the system will be locked up and the message "Board frozen!" will appear.

Part 4: Test the target

Now implement the code on the target:

- The UART1 console is replaced by the LCD.
- Set the watchdog timer to about 4 seconds.
- Task2 must hang and hog the processor.

To reset the board, you MUST press the MCLR button on the board.

Resetting with MCLR will trigger RCONbits.EXTR rather then RCONbits.POR.
You must improve your code accordingly.

Approval You must give a **remote** demo to the teacher when ready.

Upload to GitHub.

After lab questions:

Q1- Explain why we use watchdogs in embedded systems. Give examples.

- Watchdogs are used in embedded systems to make them a little more self-reliant; less user intervention needed when something goes wrong.

  We would want a watchdog to protect against hogging tasks, hung queue receptions, or anything else that may want to hog the CPU and cause trouble to normal program execution.

Q2- After implementing vTask2 task, explain why the board resets even though the watchdog was kicked in the Idle Hook task.

- As explained briefly above, even though the WDT kicking was implemented in the idle hook function, it never gets kicked because the idle hook function never runs!

  As soon as a resource hogs the CPU, the idle hook function will not be able to run. In turn, the WDT will never get kicked which will cause the system to get reset after some time.

Q3- Explain the persistent attribute. How does it affect the variable?

- The persistent attribute protects the system reset count variable by making it survive any type of system reset (POR, BOR, EXTR, WDTO, etc…). This is why whenever the WDT resets the system, the system reset count does not get reset to zero.

  In our case, the only time it gets cleared is because a POR has occurred, since we had deliberately reset the system reset count to zero when it does occur.