Embedded System Interfacing

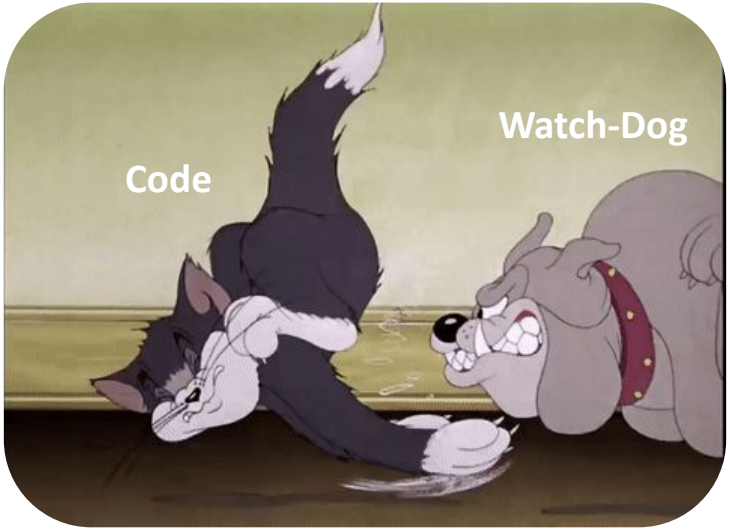
Lecture 12
Watch-Dog Timer

IMT SCHOOL

Introduction

What is Watch-Dog Timer

Most embedded systems need to be self-reliant. It's not usually possible to wait for someone to reboot them if the software hangs. Some embedded designs, such as space probes, are simply not accessible to human operators. If their software ever hangs, such systems are permanently disabled. In other cases, the speed with which a human operator might reset the system would be too slow to meet the uptime requirements of the product.



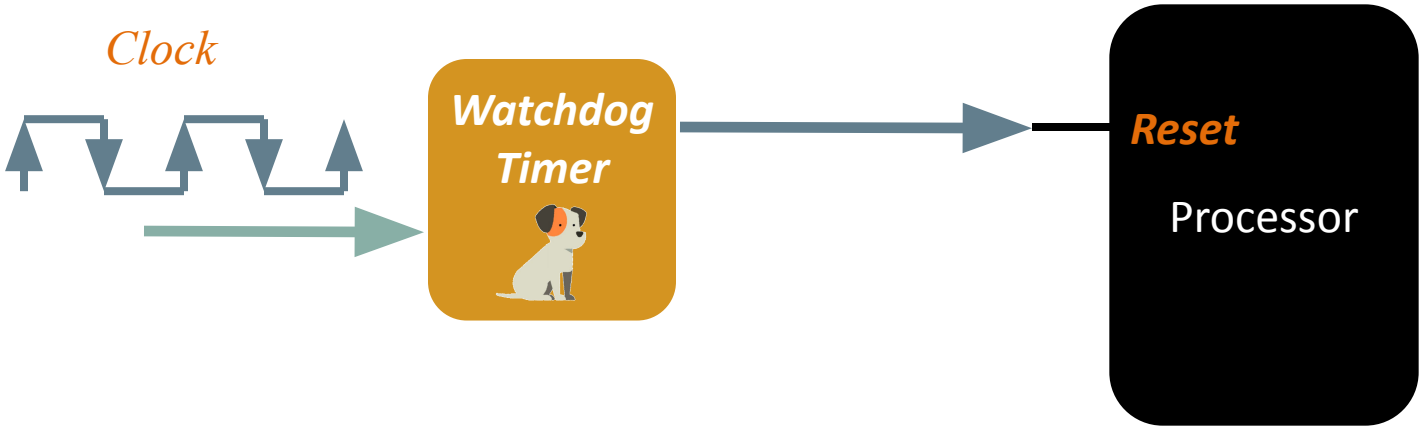
What is Watch-Dog Timer

A watchdog timer is a piece of hardware that can be used to automatically detect software anomalies and reset the processor if any occur. Generally speaking, a watchdog timer is based on a counter that counts down from some initial value to zero. The embedded software selects the counter's initial value and periodically restarts it. If the counter ever reaches zero before the software restarts it, the software is presumed to be malfunctioning and the processor's reset signal is asserted. The processor (and the embedded software it's running) will be restarted as if a human operator had cycled the power.

Note: The Watch-Dog Timer must be restarted before the end of time to avoid the power reset

What is Watch-Dog Timer

The process of restarting the watchdog timer's counter is sometimes called "kicking the dog." The appropriate visual metaphor is that of a man being attacked by a vicious dog. If he keeps kicking the dog, it can't ever bite him. But he must keep kicking the dog at regular intervals to avoid a bite. Similarly, the software must restart the watchdog timer at a regular rate, or risk being restarted.



IMT SCHOOL

Operation

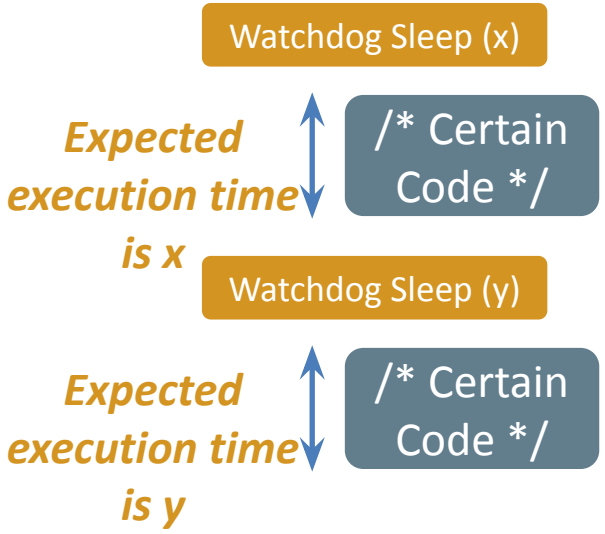
02

Watch-Dog

The watchdog Timeout value is the value that the watchdog time will count before it resets the processor.
Before executing any part of the code, set the timeout period of the watchdog to be the expected execution time for that part, usually the function that sets the timeout is called ***Watchdog_sleep***.

If the code executed in the expected time, then the watchdog would be refreshed again and a new timeout value would be assigned for the next part of the code.

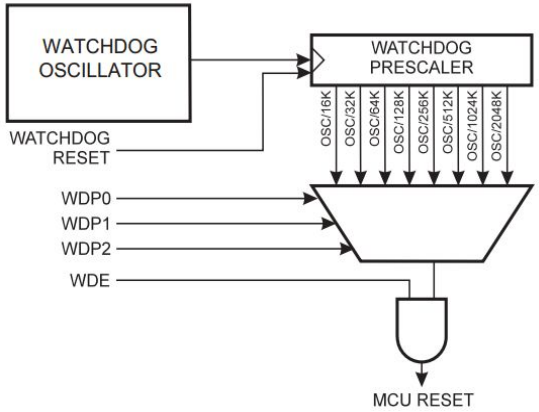
If the code is executed in a time more than expected, then the timeout would be passed and the watchdog would reset the processor.



By this way you will make sure that the processor ***would never halt*** in a code !

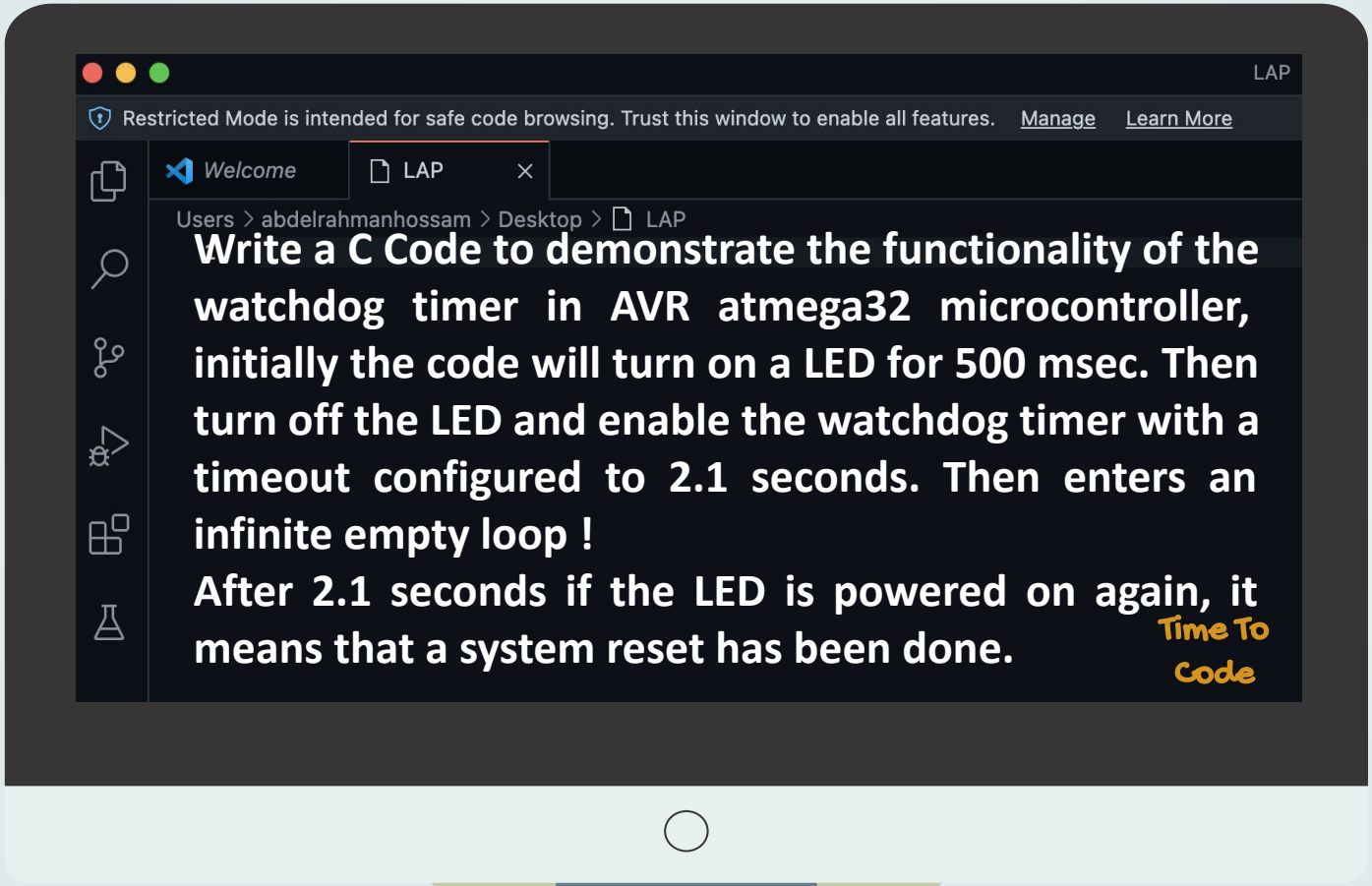
Watch-Dog

- The Watchdog Timer is clocked from a separate On-chip Oscillator which runs at 1MHz.
- To prevent unintentional disabling of the Watchdog, a special turn-off sequence must be followed when the Watchdog is disabled.



WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 3.0V	Typical Time-out at V_{CC} = 5.0V
0	0	0	16K (16,384)	17.1 ms	16.3 ms
0	0	1	32K (32,768)	34.3 ms	32.5 ms
0	1	0	64K (65,536)	68.5 ms	65 ms
0	1	1	128K (131,072)	0.14 s	0.13 s
1	0	0	256K (262,144)	0.27 s	0.26 s
1	0	1	512K (524,288)	0.55 s	0.52 s
1	1	0	1,024K (1,048,576)	1.1 s	1.0 s
1	1	1	2,048K (2,097,152)	2.2 s	2.1 s

LAb 1



IMT SCHOOL

Execution Time

IMT SCHOOL

Execution Time Measurment

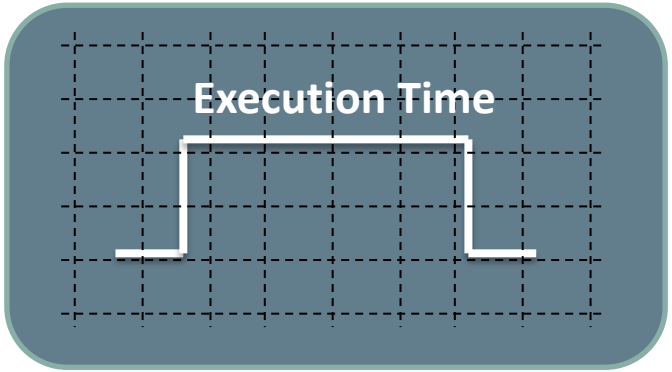
There are many ways to measure the execution time of a certain code, one way is:

By using Oscilloscope

- Set a pin a high before the code to be measured
- Set the pin low after the code to be measured
- Measure the width of the high part which represent the code execution time

```
Set Pin High
/* Certain Code */
Set Pin Low
```

Oscilloscope

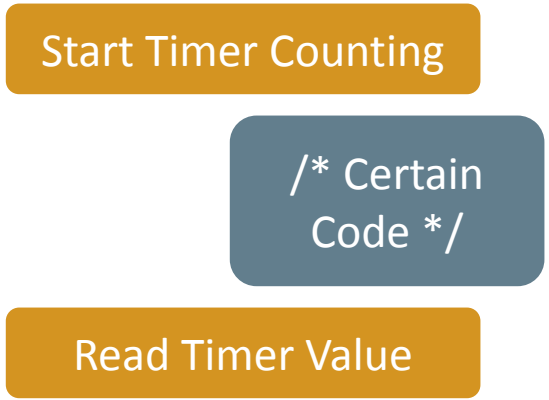


Execution Time Measurment

There are many ways to measure the execution time of a certain code, one way is:

By using Timer

- Start a timer before the code to be measured
- Read the timer value just after the code to be measured
- Calculate the execution time of the code according to the following equation



$$\textbf{\textit{Execution Time}} = Timer\ Value\ x\ Timer\ Tick\ Time$$

Execution Time Measurment

There are many ways to measure the execution time of a certain code, one way is:

By using Listing File

The listing file (.lss) is a generated file by the liker that shows the assembly generated for each line of the code, for each generated assembly line, the listing file writes the number of clock cycles for that line.

Count the number of the clock cycles for the part of the code you need to measure, then multiply this count by the period of the clock you use (period of the crystal oscillator used).

The End

IMT SCHOOL

www.imtschool.com

www.facebook.com/imaketechnologyschool/