SE3910 – REAL TIME SYSTEMS

Designing Multithreaded Software For the Beaglebone

FOR THE GOOD OF THE ORDER

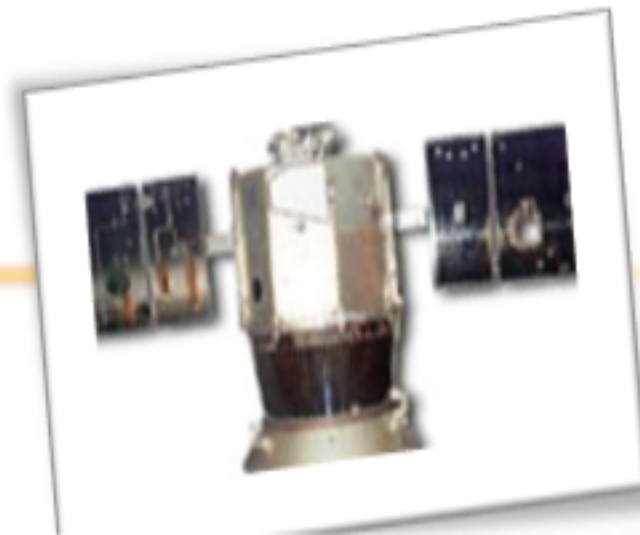
- There is a video online about the lab and GPIO.
  - Watch it. It has a quiz in it.
- There is a video about debugging with gdb
  - Watch it.
  - It !has a quiz in it
- Lab tomorrow
  - Need to come in with circuit design drawn out in advance.

OBJECTIVES

- Explain the purpose for a watchdog timer
  - Finish up from Monday
- Explain how to design a simple multithreaded application using POSIX
  - Review from CS3844
- Explain the concept of conditional compilation
  - How can we use conditional compilation to our benefit

THE CLEMENTINE

- In 1994, a deep space probe, the Clementine, was launched to make observations of the moon and a large asteroid (1620 Geographos).
- After months of operation, a software exception caused a control thruster to fire for 11 minutes, which depleted most of the remaining fuel and caused the probe to rotate at 80 RPM.
- Control was eventually regained, but it was too late to successfully complete the mission.



WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually rescuing a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation from humans and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually rescuing a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation from humans and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and are not accessible to an operator.
- Making a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually rescuing a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

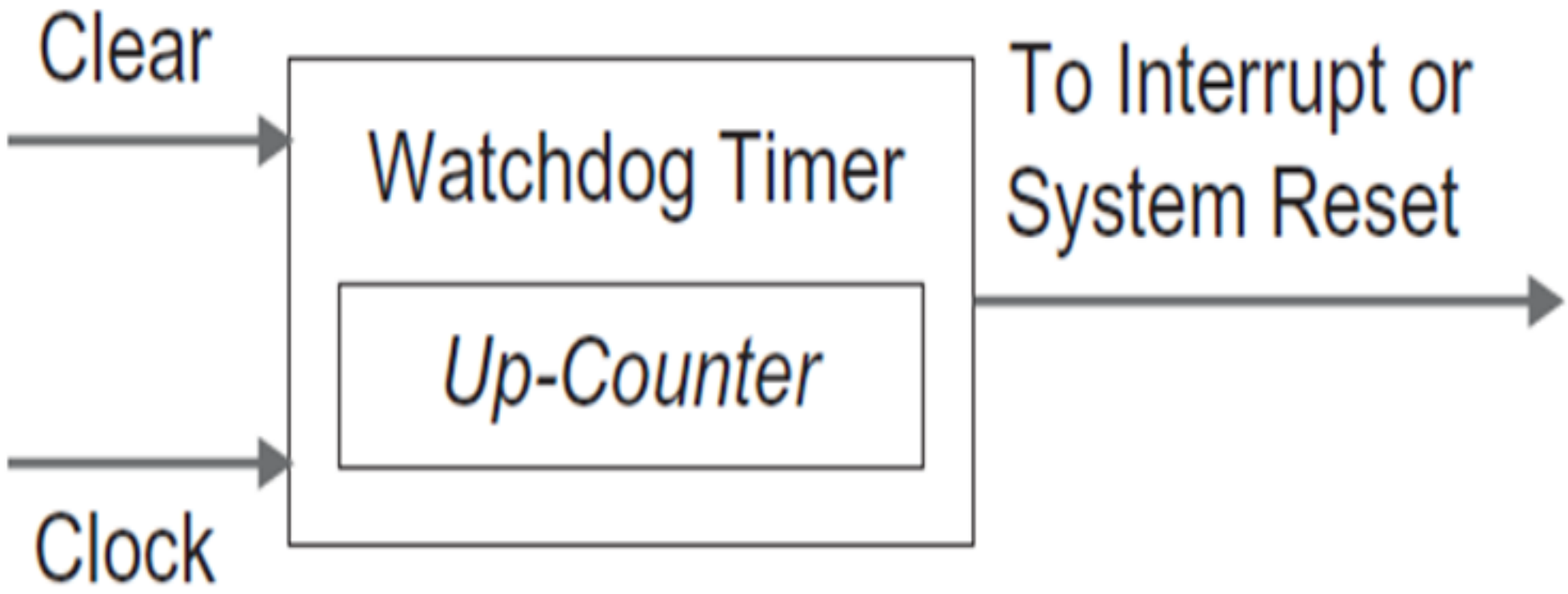
WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation from humans and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

WATCHDOG TIMER STRUCTURE



- How do you start designing a multithreaded application in C using POSIX?

QUESTION

- C programming review from last spring
  - What does #if do?

QUESTION

- Lets make some code multithreaded and use conditional compilation to make it testable and able to be debugged on multiple platforms...

TUTORIAL