What is time?

- Time for earth to make a complete rotation about its axis.
    - A day
- Time need for a cesium-133 atom to perform 9,192,631,770 complete oscillations.
    - A second
- What is oscillations
    - A regular periodic variation in value about a mean.
- Time is relative
    - Related to the observer's clock.

Frequency:

- Number of occurrences within a period of time
    - Usually 1 second
    - E.g. 40 cycles/second = 40 Hz
- A computer's clock has a crystal, that crystal defines its operating frequencies.
    - Number of oscillations/second
    - Operations are performed on each cycle
- We can track time based on clock frequency.
    - Count oscillations (cycles)
    - How do we count?
        - In a register(comp2280)
        - Increment or i/p transition
- How do we know how much time passed?
    - E.g. a 1MHz clock
        - Counted 50,000 cycles 1,000,000 cycles/second , or 1/1,000,000 second/cycle
        - $\frac{1\ second}{1{,}000{,}000\ cycle} * 50{,}000\ cycles = 0.05 second$
- We talk about time in 2 ways
    - Cycle time = time between occurrences
        - 
    - Rate: number of occurrences over a tie period
    - E.g. 10 times a second, 1000 times a second
        - Every millisecond

What is the defining characteristic of a real-time systems?

- They all interact with the physical world
    - As soon as this happens you have to meet time constraints

    - If we don't, then things (whatever we doing) don't work
        - Probably breaks
        - This applies to simple features like button press, video, cars

Real-time Tasks

- Recall standard OS terms:
    - Run-time
    - Throughput
    - Response time
- The defining characteristic of a task in a Real-time System is the deadline
    - This is the time by which execution of task must be completed.
        - Usually relative to start time.
        - E.g. triggered by an event
    - We have 2 timing constraints to see whether or not something happens "correctly"
    - 1) soft
        - deadlines are a guide but aren't the only consideration
        - being tardy is OK
            - completion – deadline
        - trying to make timing register
        - e.g. streaming video
    - 2) hard
        - The task must meet the timing constraint
        - If, on average, it meets the constraints, then its implement is soft.
        - The timing of a task must be deterministic for us to validate that it is a hard implementation.
    - Soft vs hard deadline
        - Hard is hard to guarantee
            - We can verify is a system can recover from missed deadlines
                - How many missed deadlines cause a failure?
                - If it is safe under all conditions, the implementation is good.
                    - We are liable (responsible)
        - The tricky part is validating all timings
        - Most systems are combinations of hard and soft tasks
- 2 types of tasks:
    - Periodic
        - A task repeatedly run at a set of frequency
            - The period
            - E.g. sampling inputs every 100ms
        - Note : the execution time must be less than a period
    - Aperiodic (sporadic)
        - A task is run in response to an event
            - System waits for a event to trigger activity
            - (1) Can be polling based
                - Using a periodic task

                - - OR –
                - (2) can be interrupt driven
            - Can be soft or hard
- Interrupts
    - Life-blood of Real Time Systems
    - The only way to do hard real-time implementation
    - Review:
        - Interrupts stop regular flow (context switch) and jump to an ISR
            - Interrupt Server Routine (ISR) must be defined in the interrupt vector
            - Must enable the interrupt & all interrupts
        - Execution begins as soon as event triggers, regardless of what is happening
            - Ignores scheduling
        - Interrupts have defined priorities
            - Can't be interrupted by lower priorities
            - CPU has the lowest priority(O)
            - Has serious implications:
                - (a) Do you want your ISR to be interrupted?
                    - Modify the priority
                    - Turn off interrupts
                - (b) Do you want regular code interrupted?
                    - Disable interrupts within critical sections
                    - ISRs have access to all data
                        - Mutex issues apply…
                - (c) priority inversion issues must be managed

<u>Architecture</u>:

- When we interact with the physical world we need to have interface h/w
- The way we comm with h/w affects our designs
    - UP(microprocessor) vs UC(microcontroller)
- Microprocessor Based Systems:
    - Microprocessor interact with peripherals via a bus
    - Use memory mapped I/O for comms
    - Peripheral may or may not contain a processor
    - Standard implementation:
        - A single board computer (SBC) with a UP, RAM, HD or flash, (maybe) video, logic controller for bus
        - Design a custom card for our application
            - Logic layer (&UC) for bus & controlling h/w
        - Code is written with standard memory access in C(language):
            - Unsigned char *activate: 0xfffe;
            - *activate = 1;
                - Sending a 1 on the data lines & fffe on the addr lines