Scheduling, concurrency, timers, and clocks
Lab 3
Reminder to fill out the form with your project preferences by 11pm tonight (you should be able to edit your responses if you’ve already submitted)
Today

Where we’ve been:
Peripherals, embedded programming and CPU, memory models

Where we’re going:
Time - clocks, timers, watchdogs
Brief introduction to scheduling (execution time, concurrency)
Time stories
What are some issues we see with keeping time?
Issues with keeping time
Break
Keeping track of time: system clocks
Or “oscillators”

Basis of control of a CPU - instructions happen on “edges” of a clock (why?)
Counting time

Most basic way to keep track of time on a CPU: \# of clock ticks

On an 8MHz CPU: 8 million clock ticks = 1 second

What is the largest unit of time we can keep track of in 32 bits on an 8MHz clock?
How do we keep track of longer time periods?
Timers

Keep track of time by incrementing every \( n \) clock ticks

On MCUs: hardware support

Often called something like TC (timer/counter) peripheral

*Prescale* the clock (divide it by 2, 4, 8...) and increment on the clock ticks
Uses for timers

- Count to a specific number of clock ticks and generate an interrupt (you will do this in lab!)
- Set a flag whenever rollover happens and use this as a low-overhead way to measure time
Timer rollover math

Rising edge

Falling edge

period $T$

$f = 1/T$
Quantization margins

Keeping track of fractional seconds (say every $2^{-15}$ seconds.. how?)
Clock drift

Imagine 32.768 kHz clock (common oscillator frequency - the SAM D1 has them too!)

0.001% drift rate (0.00001 seconds/second)

Drift during a day:

Drift during a year:
Other sources of error

Atomic operations

Time being stored in 16 bits on an 8-bit cpu?

Overflow (as we saw in the stories!)
Watchdog timers

Special timer peripheral that counts down to 0 on a clock that can’t be powered off

Can be reset by writing a value to a special register

If reaches 0, resets (or shuts down) entire system

Idea is to detect system hang
Petting the watchdog
Rules for watchdog timers

When to pet
How to pet
Where to pet
How would you pet the watchdog for a multitasked system (one where each one of n tasks has to complete in order for the system to be considered live)
watchdog
Break
Brief introduction to concurrency and scheduling

Want multiple tasks to run, more or less independently of each other

How do we schedule these tasks so that they all fulfill their deadlines?

How do we accurately measure how long each task takes?

We will talk more about deadlines when we talk about real-time systems
Concurrency

Multiple tasks doing different computations
  May or may not share resources
  May or may not interrupt each other

In embedded systems: interrupts are a big source of concurrency challenges
What approaches/mechanisms have you seen for resource sharing between tasks (processes/threads) in other computing classes or contexts?
Resource sharing
Scheduling

Decide when CPU runs what task so that deadlines are met

**Soft**: correctness “degrades” if deadlines aren’t met

vs

**Hard**: correctness fails if deadlines aren’t met

**Preemptive**: task can interrupt lower-priority task

vs

**Non-preemptive**: tasks can’t interrupt each other

**Dynamic**: done at run-time

vs

**Static**: done at compile-time
Scheduling periodic tasks

$n$ tasks each with a given period and worst case execution time

Assume task’s deadline is its period

Assume independence and 0 cost context switching

Can we schedule this so that all tasks meet their deadlines?

More on this in real-time lecture
What are the challenges in statically computing worst-case execution time?
Worst-case time
Other approaches

Time it dynamically

- Using special debug registers
- Approximate with timer/counter (you will be asked about this in lab 4)

**Issues?**

Hybrid (dynamically measure short paths and statically add it up)

- Many tools on the market do this
Summary

- Time is tricky and leads to bugs
- Computers measure time using clocks/oscillators
- In embedded, can use timer/counter for measuring time
- Watchdogs are specialized timer/counters to detect hangs
- Measuring execution time is an important application in scheduling