CS 466/566 Spring 2024

Miller Lowe

Resource of the Week

Set yourself up for a successful spring semester

Prepare for your classes

- **Manage your time.** Read your syllabus, make note of major assignments, and enter all due dates on your calendar. Create a weekly schedule that you follow. Set reminders for tasks and block off time to get work done. Create a work plan for exams and papers.
- **Eliminate distractions.** Set up a regular workspace. Consider the **library**, if your home environment is not sufficient: vancouver.wsu.edu/library. When studying, turn off cell phone, text or email notifications, and refrain from social media.

Know your academic resources

- **For accommodations** for a disability, visit the Access Center: vancouver.wsu.edu/access
- **For help with writing assignments**, visit the Writing Center: cas.vancouver.wsu.edu/writing-center
- **For help with a math or science course**, visit the Math and Science Skills Center: studentaffairs.vancouver.wsu.edu/mssc
- **For tech support**, connect with the IT HelpDesk: https://www.vancouver.wsu.edu/information-technology

Stay safe

- **Cloth masks, neck gaiters, or bandanas are no longer considered effective face coverings**. To mitigate the spread of omicron, a bi-layer medical-style mask or a KN95 should be worn when indoors at a WSU location or in other areas where masks are required.
- **Consider getting a booster, if eligible**. At this time, WSU is not requiring boosters, but will continue to monitor data and recommendations from public health officials. To find COVID-19 vaccine appointments, visit: http://vaccinelocator.doh.wa.gov
- **Stay home when you are sick.** Stay home until your symptoms start to resolve, and then continue to wear a (high quality) mask until fully recovered.

WASHINGTON STATE UNIVERSITY
VANCOUVER

Mechanics of Course

- Lecture Wednesday — VECS 221 5:45pm-7:25pm
- Lab Friday — VECS 221 5:45pm-8:15pm
- Course Syllabus

Books Used

- No book required, I will pull some material from *"An Embedded Software Primer by David E. Simon"* but not enough to force you to buy the book.
- Extra material as needed from other sources
  - Web links
  - Google search

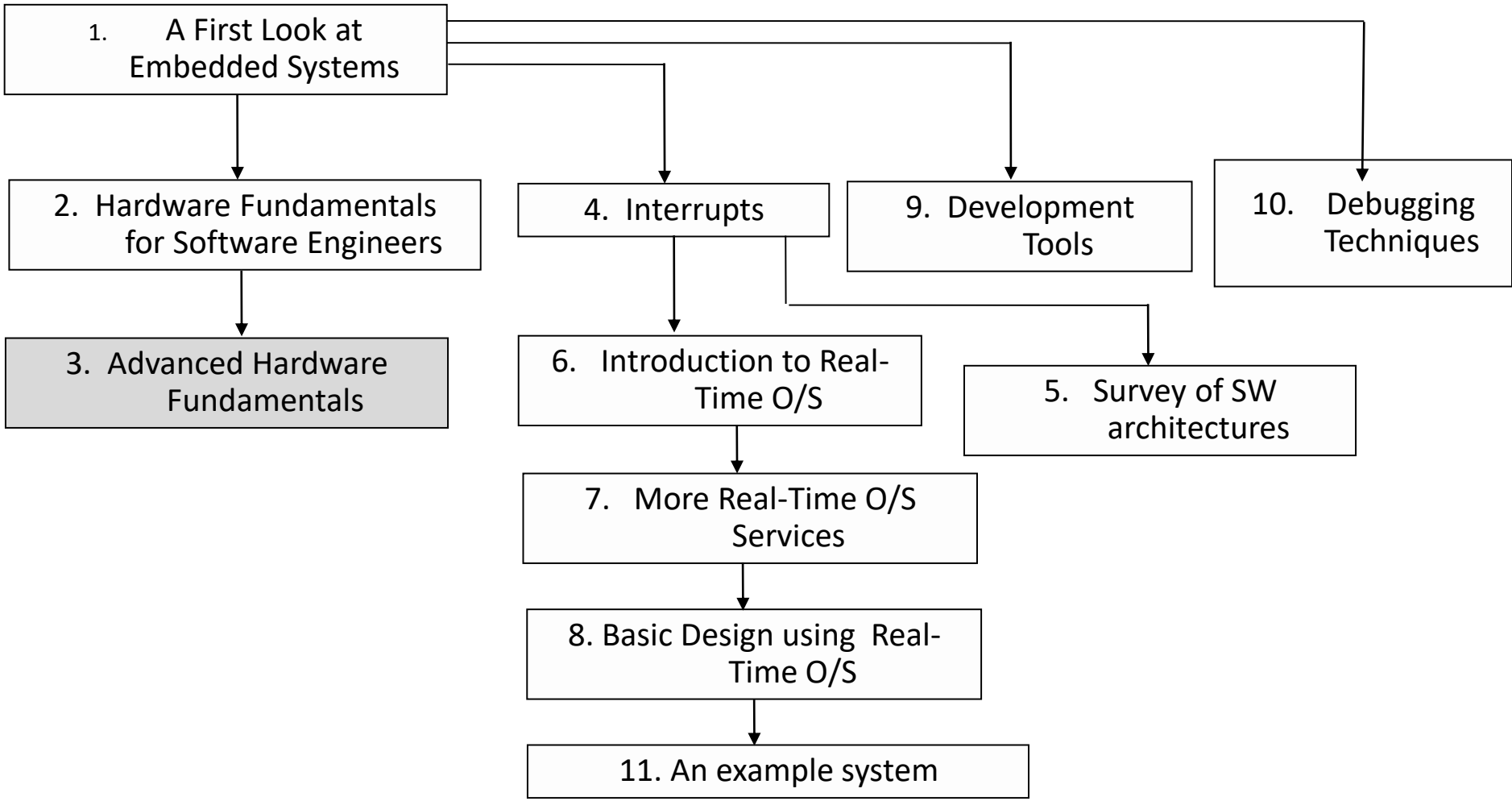
Time Management

- As a student, Time Management is your responsibility.
- As Jr's and Sr's you should know how to focus on the requirements of the course.
- I have failed several smart students because they chose to work on some pretty cool side projects and not handed in the work that is actually used to calculate a grade.
- In the workplace, missing details and deadlines will earn you inspection that you don't want. No manager will support a engineer that cannot get their base workload complete. Left uncorrected it could cost you your job.
- In this class it can cost you a deserved grade

Embedded Systems

- Terminology is not very consistent
- For every word, several subtly different meanings
- Code sizes very from under tiny bytes to many megabytes
- There is always a battle between development velocity and coding elegance.
- Delivery dates are considered cast in stone.

Class Flow


1. A First Look at Embedded Systems
2. Hardware Fundamentals for Software Engineers
3. Advanced Hardware Fundamentals
4. Interrupts
6. Introduction to Real-Time O/S
5. Survey of SW architectures
7. More Real-Time O/S Services
8. Basic Design using Real-Time O/S
11. An example system
9. Development Tools
10. Debugging Techniques

C++

- We will tend to ignore C++ and use C to focus on embedded systems because C++ is more complex and we have enough to learn.
- Embedded C++ is an advanced C++ topic.
- For the nerd in all of us, there can be great mental reward in a solid C++ embedded project but as project size goes down the usefulness of C++ degrades with it. (IMHO)

Rust

- Last semester I had one team working to develop all the programs in rust.
- There is good rust support for the processor that we are using.
- I haven't decided yet but am curious if students are interested in performing a lab in rust.
- I'm no expert having only worked on a single rust development recently but most surveys play it up.

  (show of hands?)

Code Style

- We will jump all over the place as far as variable naming..
- I do not care what you use as long as you:
  - Stay fairly consistent in your use.
  - Emulate that within an existing module.
  - Make the code readable, readable, readable
  - Did I mention to make the code readable?
  - Copy what you find on leveraged modules.
- Reasons for consistent style?
  - Diffs, Others, Contractors, etc…

More Style, Brace Conventions

- Expect an adoption of a consistent style for this class.

- Typical choices are: [My preference, takes more space]
  - Allman style (extended braces)

```
if(a  ==  b)
{
    cout << a;
    b++;
}
```

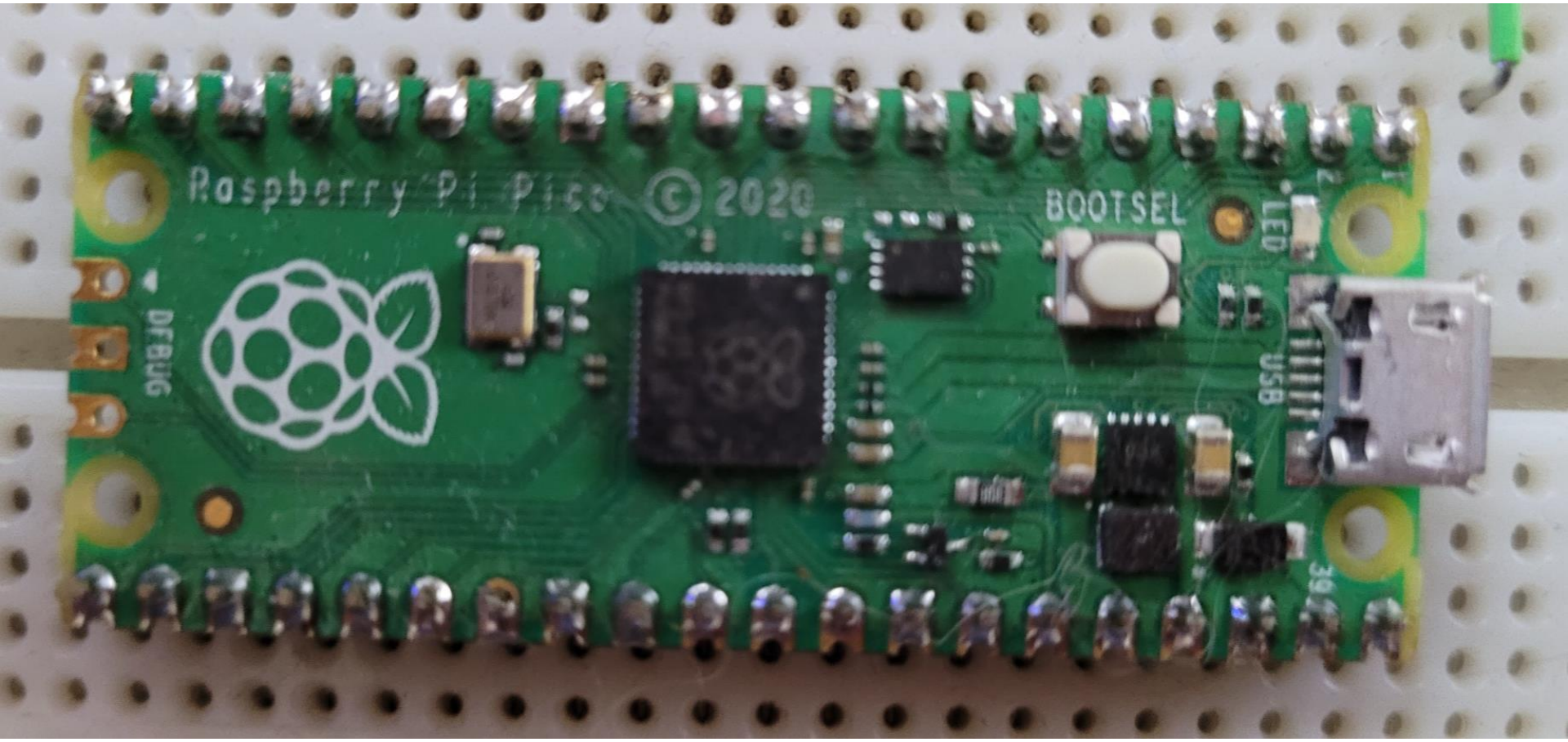
  - One True Brace Style (K & R) (linesaver) [OK, Popular, can get wierd]

```
if(a  ==  b){
    cout << a;
    b++;
}
```

  - WhiteSmith's Style (indented braces) *deprecated, still sneaks into C++ class decl

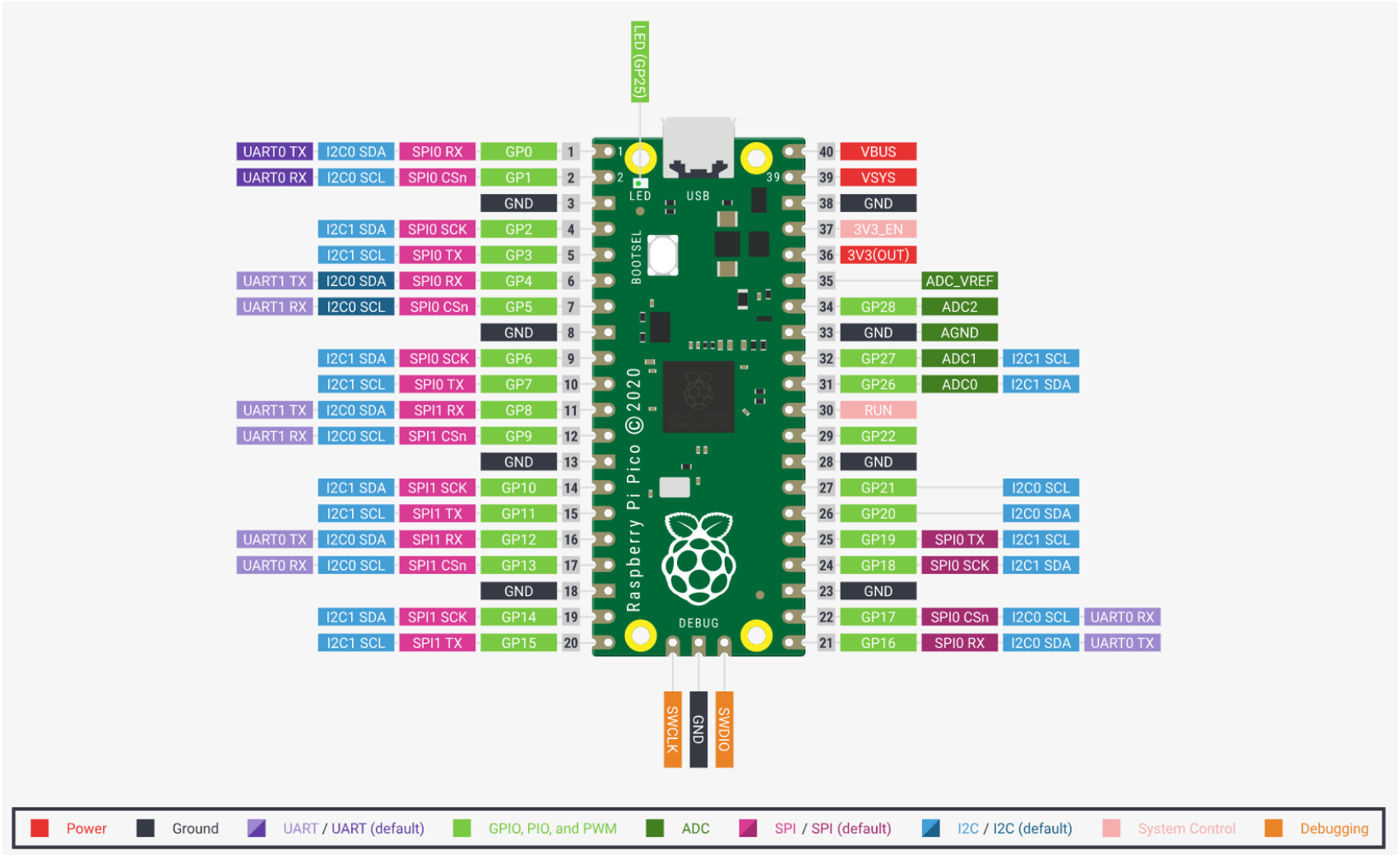
```
if(a  ==  b)
  {
    cout << a;
    b++;
  }
```

Lab Embedded CPU Board



Tiva Pin-Muxing

(This will cause you heartburn, but it's real..)



Hardware Documentation

- Processor Datasheet, Hardware Reference
  - Our Processor, ~5 MB .pdf file, Only 654 Pages (a reduction from last year)
  - By comparison NXP IMX6-Solo is 5789 Pages
  - Learning to Navigate these documents is essential, often I will give contrived exercises that are designed to send you back to the datasheet multiple times.
  - *“When in doubt, go read the datasheet again”*
- pico-datasheet.pdf
  - Gives information the dev-board
- rp2040-datasheet.pdf
  - Information about the processor SOC (chip) itself
- https://developer.arm.com/
  - All the ARM processor information you could possibly hope for

Accessing Registers

- The Arm CORE and Associated Periphials are largely controlled via the use of memory mapped registers.
  - Register Access
  - 1<< Notation
  - bitwise operations
  - Why read/modify/write
- Simple Peripheral Example
  - RP2040 peripheral GPIO or Watchdog.

Hardware Documentation

- Pin-Mux Quick Reference
  - When hunting for pins
  - When hunting for alternate pins

Micro-Quiz #1, Buildup (Queue Terror Music)

- It’s my responsibility to see that if you pass this class you don’t fall flat in an entry-level embedded interview.
- Closed Book, Closed Phone, Closed Computer,
- Get a pencil and a piece of paper

Micro-Quiz #1, Go!

- You are writing ‘C’ code on a 32 bit embedded processor
- There is a 32 bit Read/Write GPIO data register at physical address 0x40040108
- Without effecting other bits in the register, set bit-3 to 1.
  - Hint: LSB is bit-0, a 32 bit value with only bit-3 set is 0x00000008

Micro-Quiz #1, Go!

- You are writing ‘C’ code on a 32 bit embedded processor
- There is a 32 bit Read/Write GPIO data register at physical address 0x40040108
- Without effecting other bits in the register, set bit-3 to 1.
  - Hint: LSB is bit-0, a 32 bit value with only bit-3 set is 0x00000008

```
?  #define GPIO_DATA_REG  (*((volatile uint32_t *) 0x40040108))
?  GPIO_DATA_REG |= (1<<3);
```

Lab 1 Some Information

- What's a register?
- How to locate a register?
- We have an address, How to use it
- Example (From last years TIVA board, Needs Update
  - Want to output to a GPIO

Table 10-6. GPIO Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPIODATA	RW	0x0000.0000	GPIO Data	662
0x400	GPIODIR	RW	0x0000.0000	GPIO Direction	663
0x404	GPIOIS	RW	0x0000.0000	GPIO Interrupt Sense	664
0x408	GPIOIBE	RW	0x0000.0000	GPIO Interrupt Both Edges	665

- GPIO Port A (AHB): 0x4005.8000
- GPIO Port B (APB): 0x4000.5000
- GPIO Port B (AHB): 0x4005.9000
- GPIO Port C (APB): 0x4000.6000
- GPIO Port C (AHB): 0x4005.A000
- GPIO Port D (APB): 0x4000.7000
- GPIO Port D (AHB): 0x4005.B000

Address I want is GPIO Port A GPIO Data Register 0x40058000 + offset(0x000)

C Language Edge-Topics Review

- Basic Compilation and Segments
- LD File
- static and volatile

Lab 1 Introduction

- Bare-Metal Blinky (https://en.wikipedia.org/wiki/Bare_machine)
- Main objective is to get you targeting code, Some concepts are needed to answer the lab questions.
- Do prep-work before you get to lab
- I encourage you to get a development system setup on your personal computer. There limitations on Windows and OSX platforms but most the development chain works.
- We will use Linux in the Lab as our home development. I predominantly use Linux and sometimes Windows.

LAB 1 Process

- I recommend that you get used to looking at the ‘Lab Preparation’ section of the lab and acting on it some before you show up in lab..
- I am a Cruel and Unfair instructor. I have added an issue with this lab that you will need to discover and overcome.
- Get a your Linux login working
- Install the git repo and build the lab-1 code provided
- Perform the modifications outlined in the lab handout. (available on Blackboard and in the git repo)
- I have a strict lab format, *(remember the ‘Cruel and Unfair’ comment above)* If you do not address and satisfy all the sections your grade will represent the omissions.

Recurring Examples of Embedded Systems used in the old book.

- Telegraph
  - Allows a printer with only a high speed serial port to connect to a network
  - Looks like a little plastic box 2 to 3 inches by ½ inch thick
  - Has a pigtail connector for the serial connection and a network connector

Telegraph Requirements:

- On the network, data packets sometimes arrive out of order, data gets lost, and sometimes data arrives twice. Telegraph must make order out of chaos
- Multiple computers on the network may want to print at once. Telegraph must allow one computer to print and somehow hold off the others.
- Network printers must provide status info to any computer on the network that requires it, even if busy printing a job for some other computer.
- Telegraph has to work with different types of printers.
- Telegraph must respond rapidly to certain events, like various kinds of network frames.
- Telegraph must keep track of time. If a computer which is sending a print job crashes, telegraph must be able to give up on that print job and print from another computer on the network.

Telegraph development challenges

- To satisfy the previous requirements, telegraph has an embedded microprocessor. It must have logically correct performance.
- Throughput – telegraph must not be a bottleneck for print data flowing through it.
- Response – telegraph must respond to certain network frames within 200 usec, even if it is doing something else. We will talk about throughput and response and avoid speed…
- Testability – telegraph must deal with everything without human intervention. It is hard to test "everything" and how do you make something happen to see if the code handles it.

Telegraph Development (cont)

- Debugability – telegraph has no screen, no keyboard, no mouse, no speaker, no lights. When a bug occurs, it typically stops working…
- Reliability – telegraph is not allowed to crash, which is typical of embedded systems. The sw must function without human intervention.
- Memory Space – telegraph has limited memory, typically 32 kbytes of memory for its program and another 32 kbytes for its data. (kinda dated)
- Program Installation – we need special tools to install sw on these kinds of systems.

Cordless Bar Code Scanner

- No problem with throughput…there just isn't that much data
- Power consumption – since the scanner is cordless, its battery is its only source of power. It is intended to be handheld, which limits its weight. How long must the battery last? An eight hour shift is probably the least we can tolerate.
- What are some ways that the battery life could be mitigated/extended?

Laser Printer

- Most have fairly substantial microprocessors embedded in them.
- The microprocessor is responsible for getting data from any of various communication ports, sensing when users press front panel buttons, presenting messages to the user on the front panel, sensing paper jams, noticing out of paper, etc.
- It also must deal with the laser engine, which is that part of the printer responsible for putting marks on the paper.

Laser Printer (continued)

- Users expect a nearly instantaneous response when pressing a front panel button, no matter what the microprocessor is doing.

Underground Tank Monitor

- This system watches the levels of gasoline in underground tanks at a gas station.
- Its principal purpose is to detect leaks before the gas station turns into a toxic waste dump by mistake and to set off a loud alarm if it discovers a leak.
- The system has a panel of 16 buttons, a 20 character LCD as a display, and a thermal printer.
- The user interacts with various buttons to tell the system to display or print various information such as the gasoline levels or time of day or system status.

Underground Tank Monitor

- The system has two float levels in each tank, namely one that indicates the level of gasoline and the other that indicates the level of water in the tank. (Water always accumulates in such tanks).
- It also reads the temperature at various levels in the tank because gasoline expands and contracts considerable with variations in temperature.
- Cost – A gas station owner buys one of the systems because some government agency tells him that he has to. Therefore, we might use a cheap 8 bit microprocessor that can barely add two numbers let alone deal with the expansion coefficient of gasoline in a reasonable way.

Nuclear Reactor Monitor

- A very simple example in which we can learn from a system controlling a hypothetical nuclear reactor.
- It monitors two temperatures, which are always supposed to be equal. It they ever differ, it is supposed to set off the alarm.

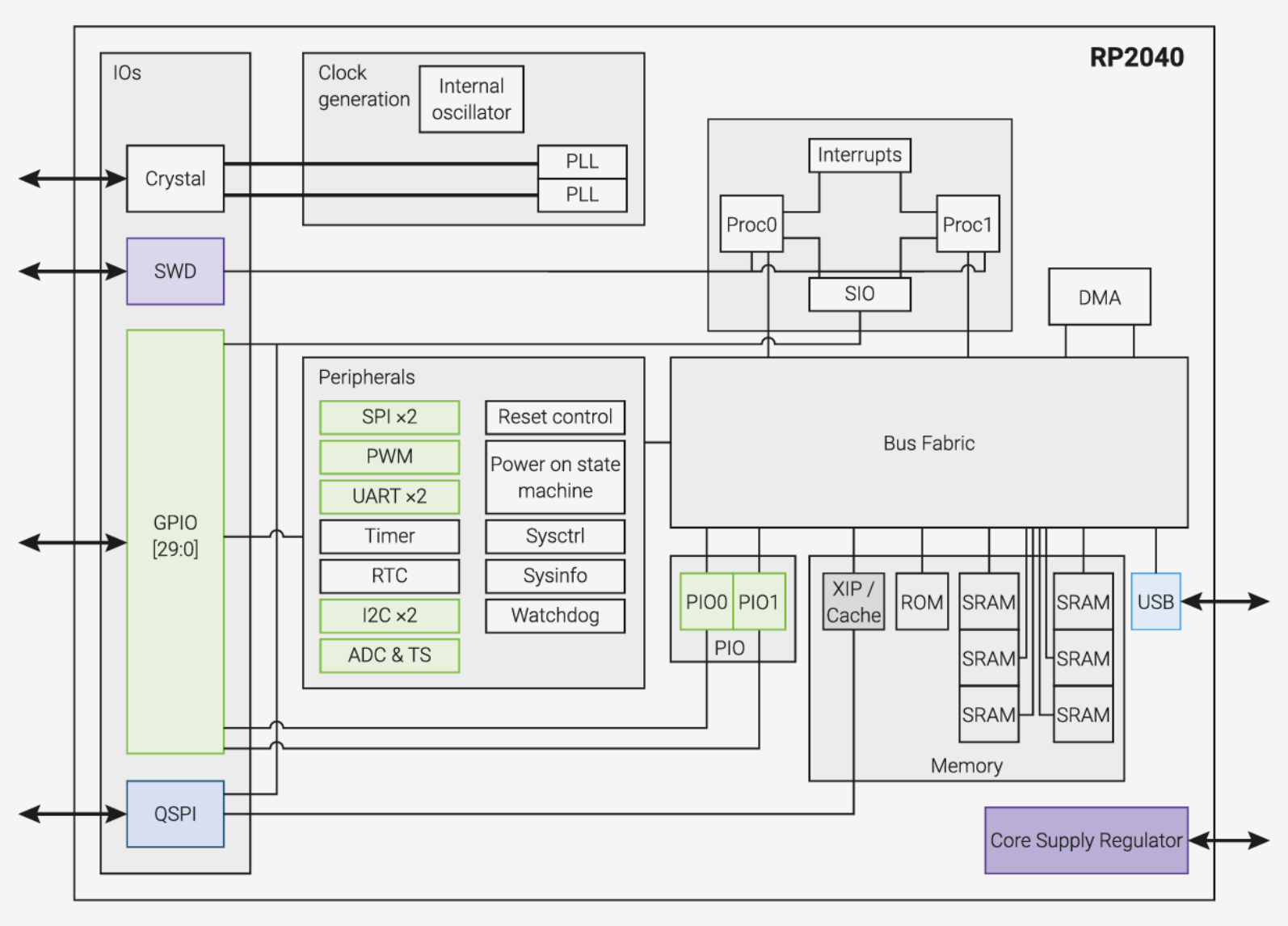
Death Ray

- Requirements:
  - Operation:
  - Physical:
  - Power:
  - Safety:

Typical Hardware

Processor	Bus Width	Ext Memory	Int Memory	Periphials	Speed(MIPS)
Zilog Z8 fam	8	64K	0	2 Timers	1
Intel 8051 fam	8	64K	64K	3 Timers	1
Zilog Z80 fam	8	64K	1MB	Various	2
Intel 80188	8	1MB	0	3 Timers 2 DMA	2
Motorola 68000	32	4GB	0	Varies	10
Motorola PowerPC	32	64MB	0	Many	75
ARM M4F TM4C123G	32	4GB	256K/32K F/R	Lots	
Dual ARM Cortex-M0+	32	4GB	0 KB / 32K FLASH/RAM	Many	

RP2040 Block Diagram



RP 2040

- Flash Memory..
- Cost..
- First Look at the RP2040 – Raspberry Pi Microcontroller
- Getting Started with Rust on a Raspberry Pi Pico (Part 1)
- Gobs of articles and tutorials for the Pi Pico

Embedded Systems

- Embedded systems are known for what they do NOT have – note the following
  - Keyboard
  - Screen
  - Disk drive
  - Compact discs, speakers, microphones,diskettes, modems, etc.

Chapter 1 summary

- An embedded system is any computer system hidden inside a product other than a computer.
  - Throughput
  - Response time
  - Testability
  - Debugability
  - Reliability
  - Memory space
  - Program installation
  - Power consumption
  - Processor hogs
  - Cost