CS466 Chapter 2

Hardware Fundamentals

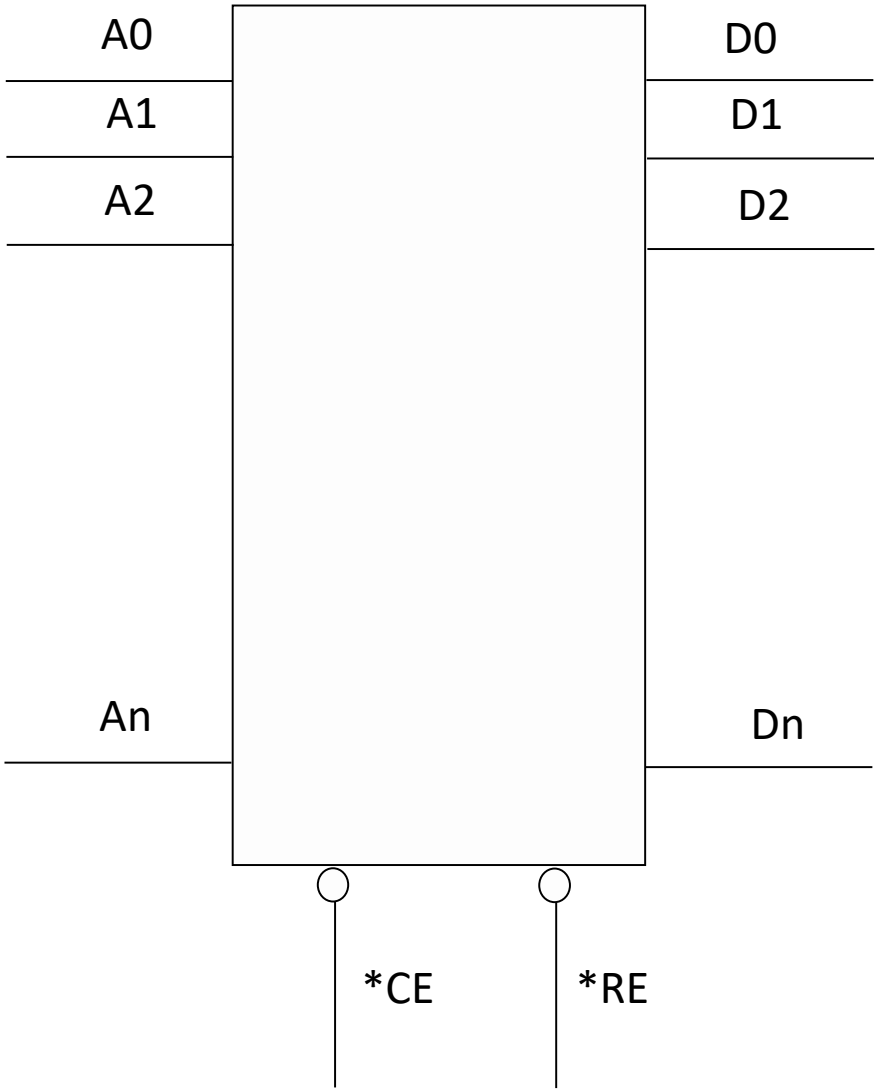
2. Hardware Fundamentals

- Terminology
- Gates – And,Or,Nand,Nor,XOR,
- Bubbles
- Negated input gates
- Power and ground and decoupling
- Open collector and tri-state outputs
- Signal loading
- Timing Diagrams
- D Flip Flops
- Hold Time and Setup Time
- Clocks

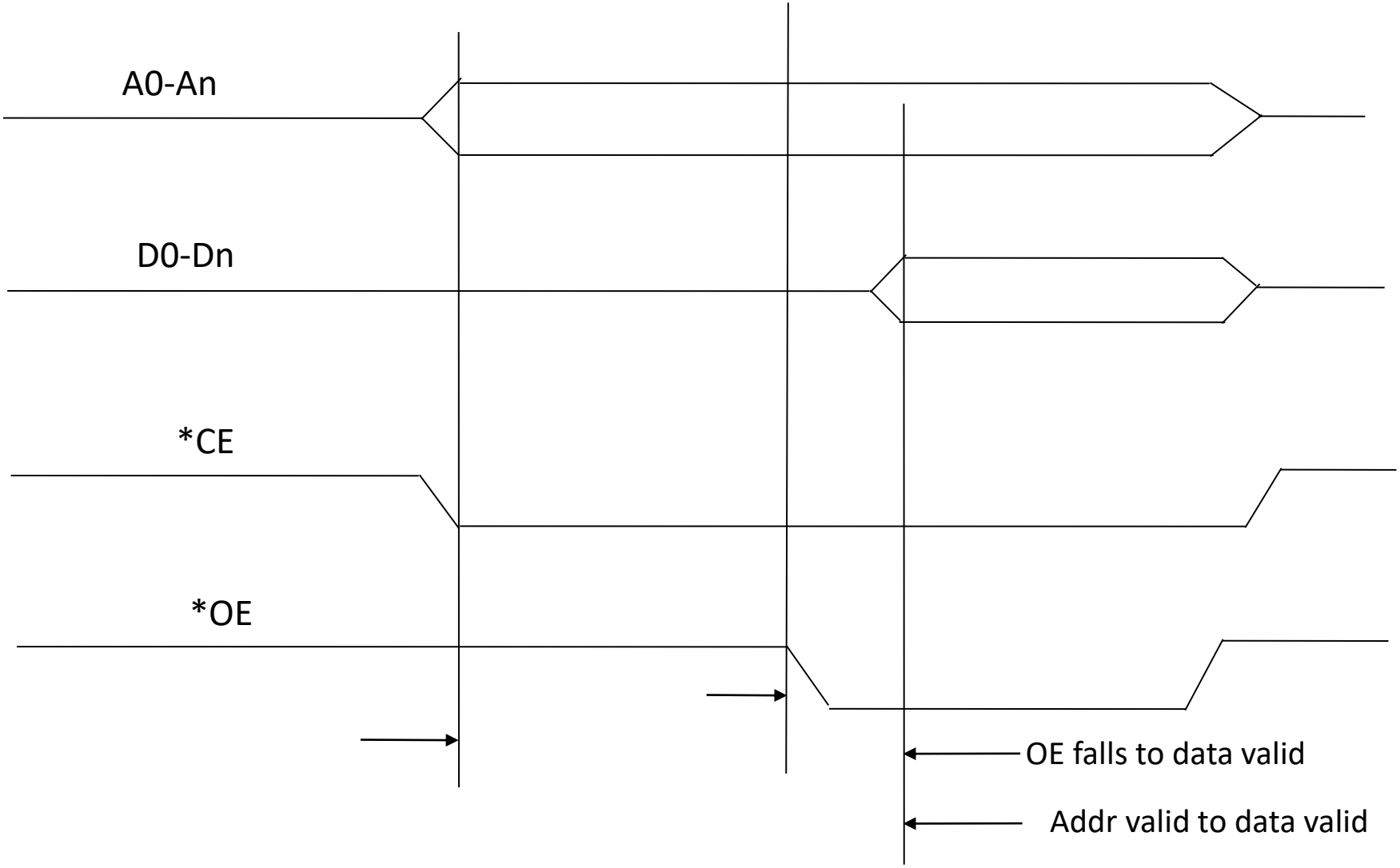
Hardware Fundamentals (cont)

- Read-Only Memory
  - microprocessor can read instructions from ROM quickly, but not as quick as RAM
  - Cannot write new data to the ROM
  - ROM remembers the data, even after power cycled
  - Typically, when the power is turned on, the microprocessor will start fetching instructions from the still-remembered program in ROM

ROM



Timing Diagram for a Typical ROM



Available ROMs

- Masked ROM or just ROM
- PROM or programmable ROM(once only)
- EPROM (erasable via ultraviolet light)
- Flash (can be erased and re-written about 10000 times, usually must write a whole block not just 1 byte or 2 bytes, slow writing, fast reading) This is the flash memory that we have in our projects
- EEROM (electrically erasable read-only memory, also known as EEPROM—both reading and writing are very slow but can program millions of times…useless for storing a program but good for say configuration information. (does get old)

RAM (random access memory)

- The microprocessor can read the data from RAM quickly, usually even faster than from ROM
- The microprocessor can write new data quickly to RAM
- RAM data is lost if power is turned off
- Not a good place for bootstrap programs!
- Static RAM remembers its data with no external assistance
- Dynamic RAM (DRAM) requires a refresh cycle from circuitry and is beyond the scope of text.

Chapter 2 Summary

- Most semiconductor parts, **chips**, are sold in plastic or ceramic packages and are connected to one another by being soldered to a printed circuit board.
- Electrical Engineers draw **schematic diagrams** to indicate what components are needed in each circuit and how they are to be connected.
- Digital signals are always in one of two states: **high** and **low**. A signal is said to be **asserted** when the condition that it signals is true. Some signals are asserted when they are high, others when they are low.
- Each chip has a collection of pins that are inputs, and a collection that are outputs.
- The standard semiconductor **gates** perform Boolean NOT, AND, OR, and XOR functions on their inputs.

Chapter 2 Summary(cont)

- Most chips have a pin to be connected to VCC and a pin to be connected to ground. These provide power to the chip.
- **Decoupling capacitors** prevent local brownouts in a circuit.
- A signal that no output is driving is a **floating** signal.
- **Open collector** devices can drive their outputs low or let them float high but they cannot drive them high. The may be connected together and the signal will be low if any of the outputs is pulling it low.
- **Tri-state** devices can drive their outputs both high and low or let them float. The designer must ensure that only one of these devices is driving the signal at the same time if multiple outputs are connected together to avoid “bus fights”.
- A dot on a schematic represents lines crossing one another on the schematic which are electrically connected.

Chapter 2 summary (cont)

- A single output can only drive a certain number of inputs. Too many inputs leads to an overloaded signal.
- **Timing diagrams** show the relationship among events in a circuit.
- Various important time specifications for circuits include the **setup time**, the **hold time**, and the **clock to Q time**.
- **D flip-flops** are 1 bit memory devices.
- The most common types of memory are **RAM**, **ROM**, **PROM**, **EPROM**, **EEROM**, and **flash**.

Lecture #2 Notes:

- Class ends at 7:40, If I talk past, let me know…
- I have several Linux laptops that are available for loan. I will want them back. For right now they are all at my house if you want to borrow one it takes a drive..
- To stay on track it is important that you complete lab #1 this week. The writeup is due a week from Thursday but this will be the last two-week lab for a while.
- I will be handing out lab #2 soon.

Lab 1

- I'll run a short demo of what lab work should look like.
- Questions?

Typical File Types

- Independent
  - .c -- Program Source
  - .h -- Module interface, hardware interface. Others
  - .ld -- Linker directive
  - Makefile
- Dependent
  - .o -- Object file (:= compiled translation unit)
  - .d -- Debug information
  - .bin -- Binary image
  - .elf or .axf (https://en.wikipedia.org/wiki/Executable_and_Linkable_Format)
  - .map -- List of symbols
- Other

Linker File (this is a really simple linker file)

- MEMORY

```
MEMORY
{
    FLASH (rx) : ORIGIN = 0x00000000, LENGTH = 0x00040000
    SRAM (rwx) : ORIGIN = 0x20000000, LENGTH = 0x00008000
}
```

- SECTIONS

```
SECTIONS
{
    .text :
    {
        _text = .;
        KEEP(*(.isr_vector))
        *(.text*)
        *(.rodata*)
        _etext = .;
    } > FLASH
```

Linker File (this is a really simple linker file)

- SECTIONS (Continued)

```
    .data : AT(ADDR(.text) + SIZEOF(.text))
    {
        _data = .;
        *(vtable)
        *(.data*)
        _edata = .;
    } > SRAM

    .bss :
    {
        _bss = .;
        *(.bss*)
        *(COMMON)
        _ebss = .;
    } > SRAM
}
```

What Happens Before main()?

1. Power Applied

   "Look at startup_gcc.c"
2. Processor Stack Register Set (ARM Thing)
3. PC Loaded
4. Processor Set Loose
5. Copy .data (initialized data) segment to RAM
6. Zero .bss (uninitialized data) segment in RAM
7. If C++, there are more steps, Static Initializers, ect
8. Initialize Interrupt Processor
9. Call main()

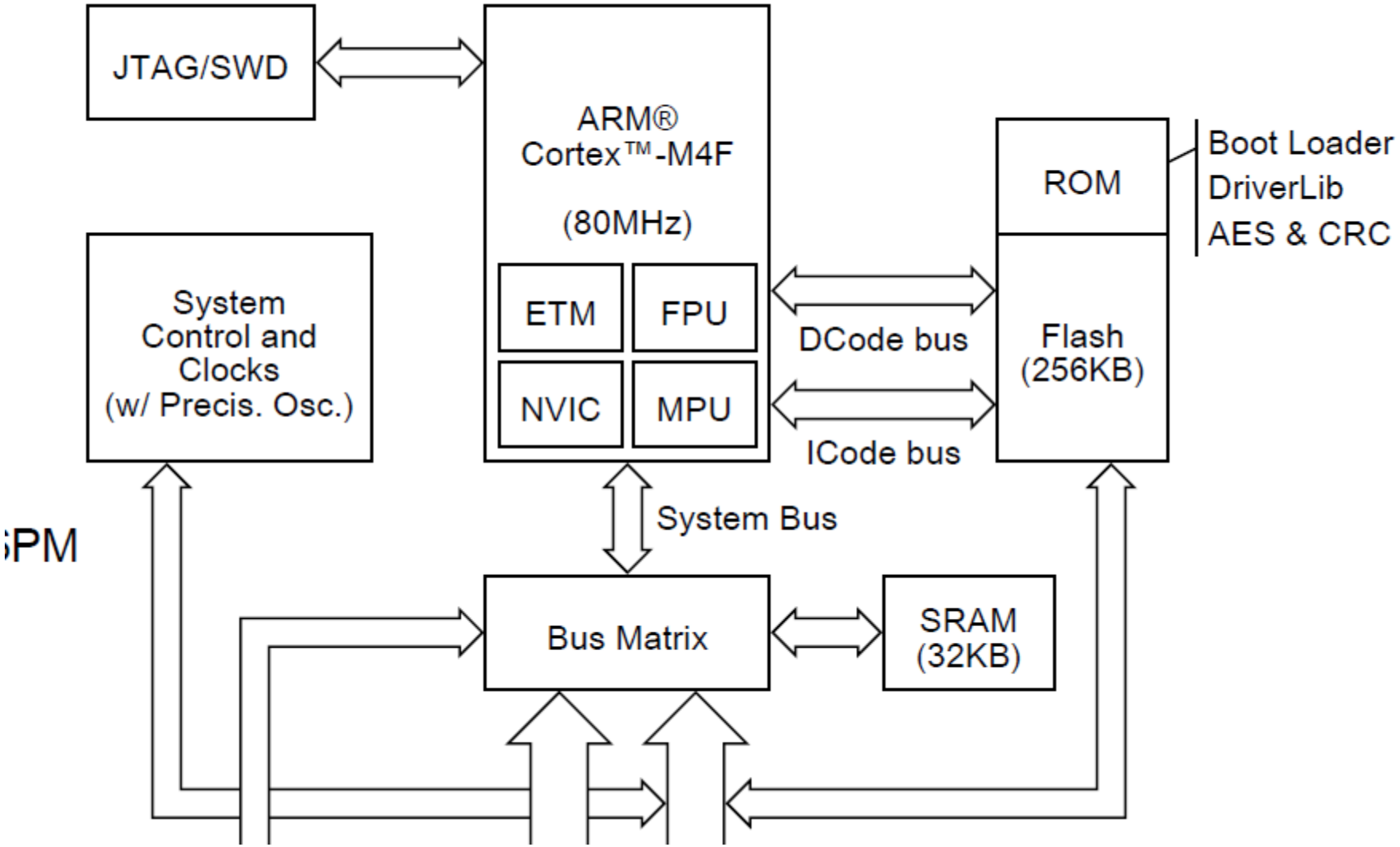
What Happens if Main Returns?

1. Main should never return.

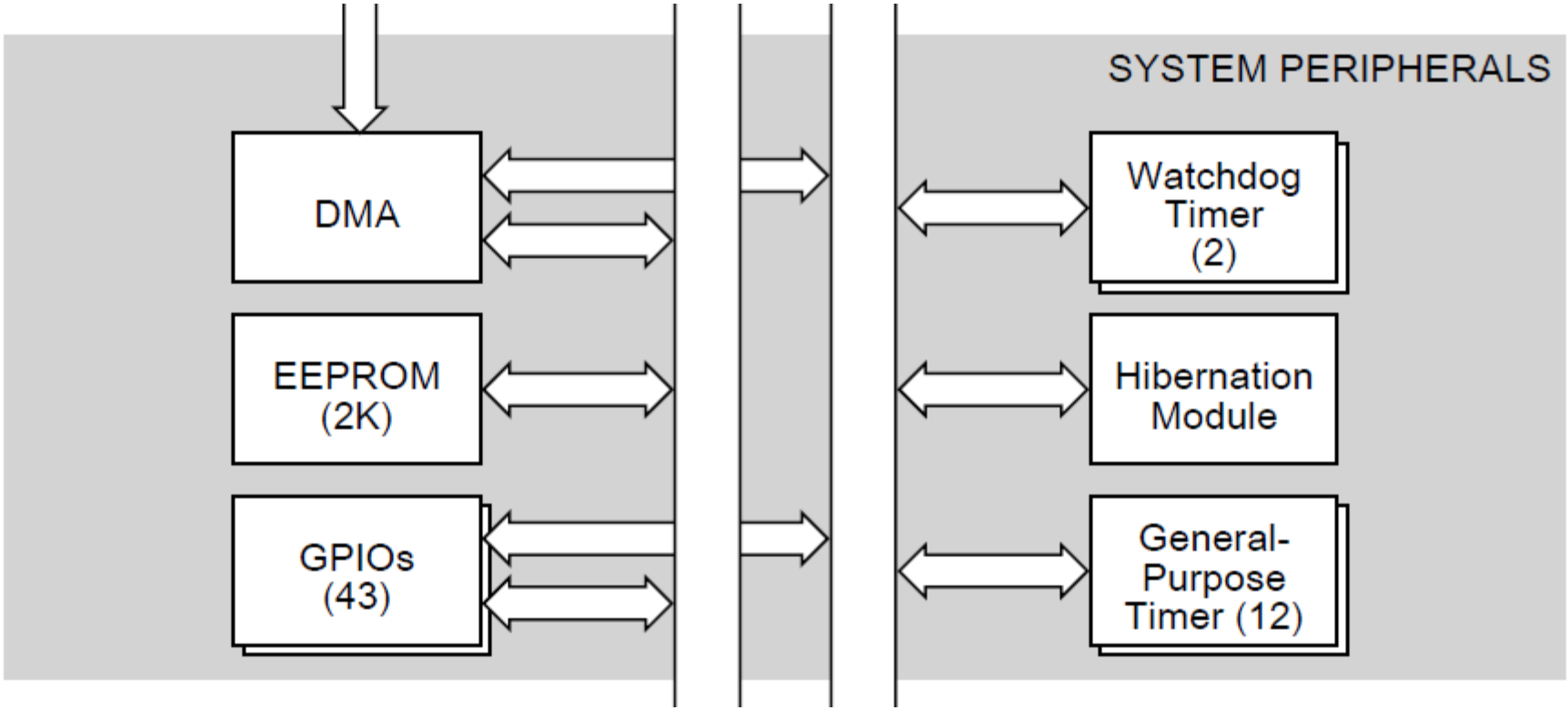
```
    //
    SystemCoreClock = 80000000;  // Required for FreeRTOS.

    SysCtlClockSet( SYSCTL_SYSDIV_2_5 |
                    SYSCTL_USE_PLL |
                    SYSCTL_XTAL_16MHZ |
                    SYSCTL_OSC_MAIN);


    //
    // Call the application's entry point.
    //
    main();
}
```

2. What Does a call to exit() do?

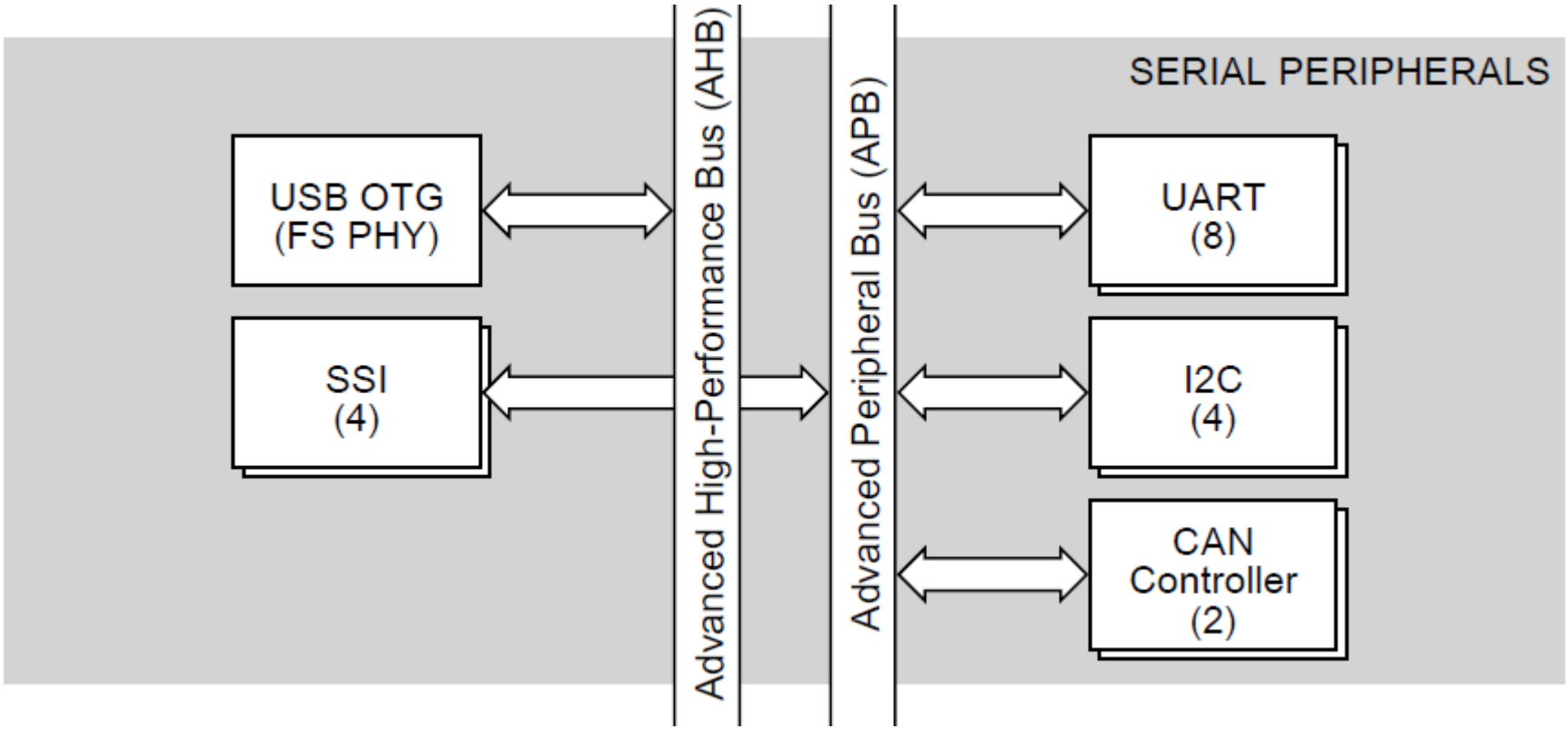
TM4C123GH66PM High-Level Block Diagram



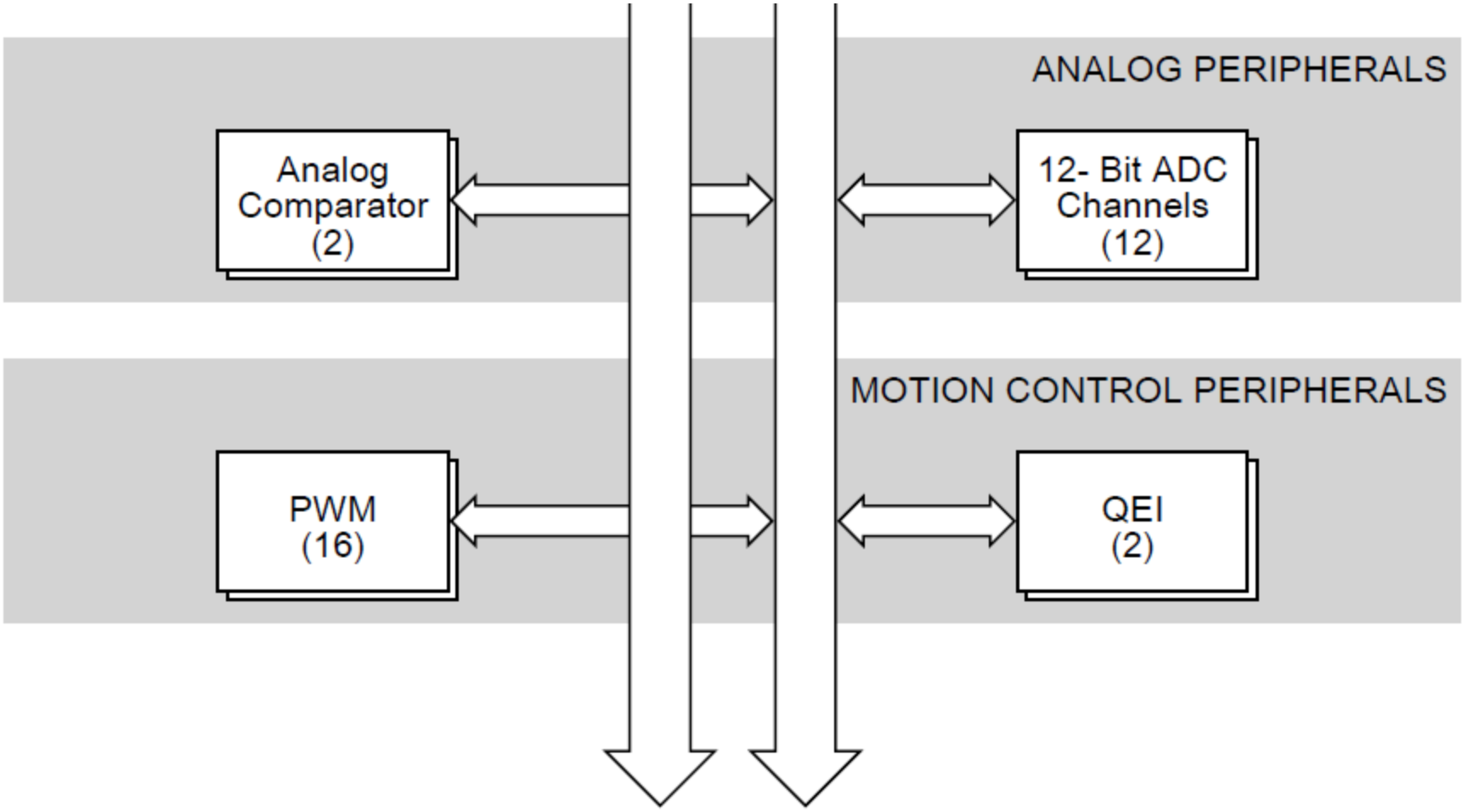
TM4C123GH66PM High-Level Block Diagram



TM4C123GH66PM High-Level Block Diagram



TM4C123GH66PM High-Level Block Diagram



Lab 1: Contrived Issues

1. What's with 1Hz, 15Hz and 13 HZ.
2. What's Up with SW2
3. Why are there lock registers at all
4. How do I write a Timing Test Function?
5. "You will need to figure how to feed the function and how to calibrate it. I want you to do it empirically for now but we'll use some high resolution timers in the future"

Lab 2:

1. Start your FreeRTOS download Thursday in a window as it will take a long time.
2. Is the same requirements as Lab 1
3. Adds an interrupt Routine
4. What is a semaphore?
5. Generic RTOS OS task idiom.