SIGPwny

Embedded FA2025 • 2025-09-15

Embedded 101: Fundamentals

Adarsh and Minh

Adarsh Krishnan

- SIGPwny Helper + Embedded Lead
- CS + Philosophy, Info/Math minors
- Involved with Embedded since eCTF 2024
- Fact: I help lead the Creative Writing Club at UIUC

Minh Duong

- Previous SIGPwny President
- BS-MCS program (last semester)
- Involved in SIGPwny's eCTF team since 2023
- Fun fact: I just paid $50 to cheese a CSAW challenge yesterday

Announcements

- Fall CTF 2025 is in less than a week!
    - Register now to get a free electronic badge! https://sigpwny.com/fallctf
    - Sunday September 21st, 12pm
    - CIF 3039

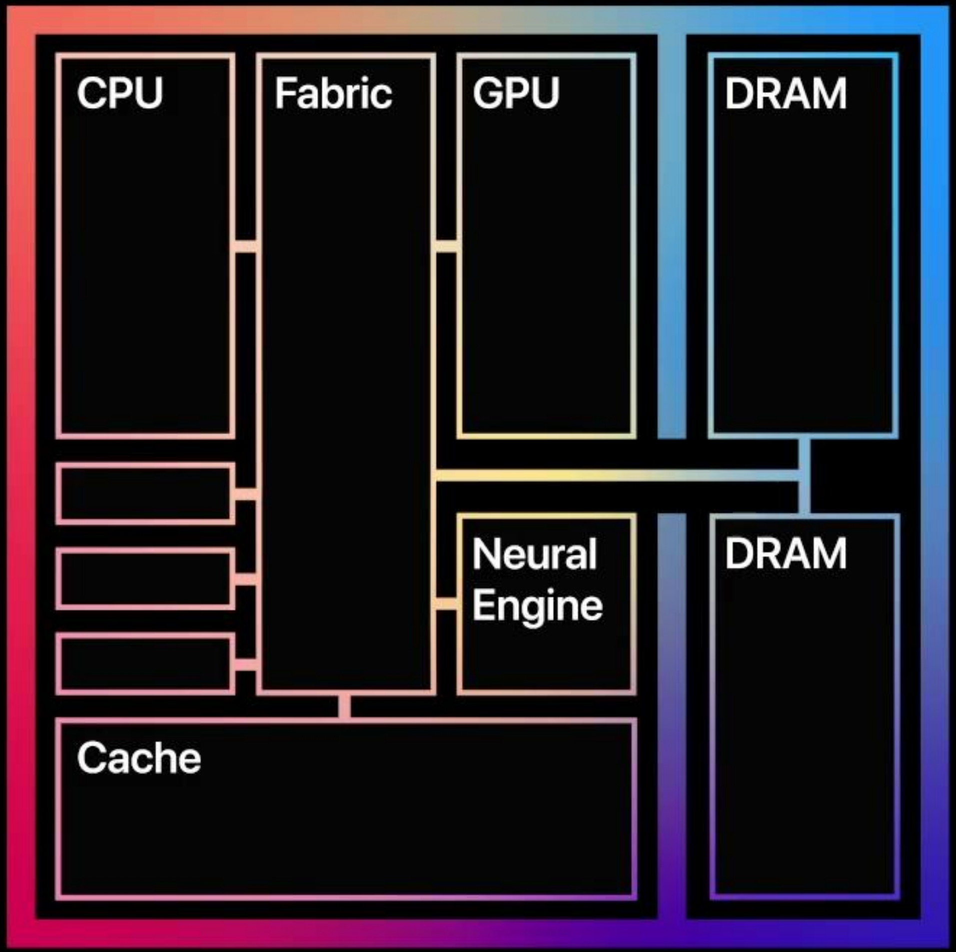
- We are using an ESP32 chip, which has Wi-Fi capabilities built-in!
    - Full color 160x120 display!!
    - Piezo buzzer!!!
    - Joystick!!!!
    - Buttons!!!!!

Microcontrollers

Microcontrollers (MCU)

- MCUs glues everything together
  - Peripherals are useless without some control logic to transform inputs into outputs
- MCUs are fully packaged (System-on-a-Chip)
  - Includes CPU core(s), RAM (memory), flash storage
- Typically lightweight and power-efficient for most embedded systems
  - Megabytes of RAM as opposed to gigabytes on your laptop
  - MUCH slower than your laptop (instructions executed at a much slower rate)
- Specific-purpose instead of general purpose (it's cheaper!)

Parts of a CPU (e.g. M1)

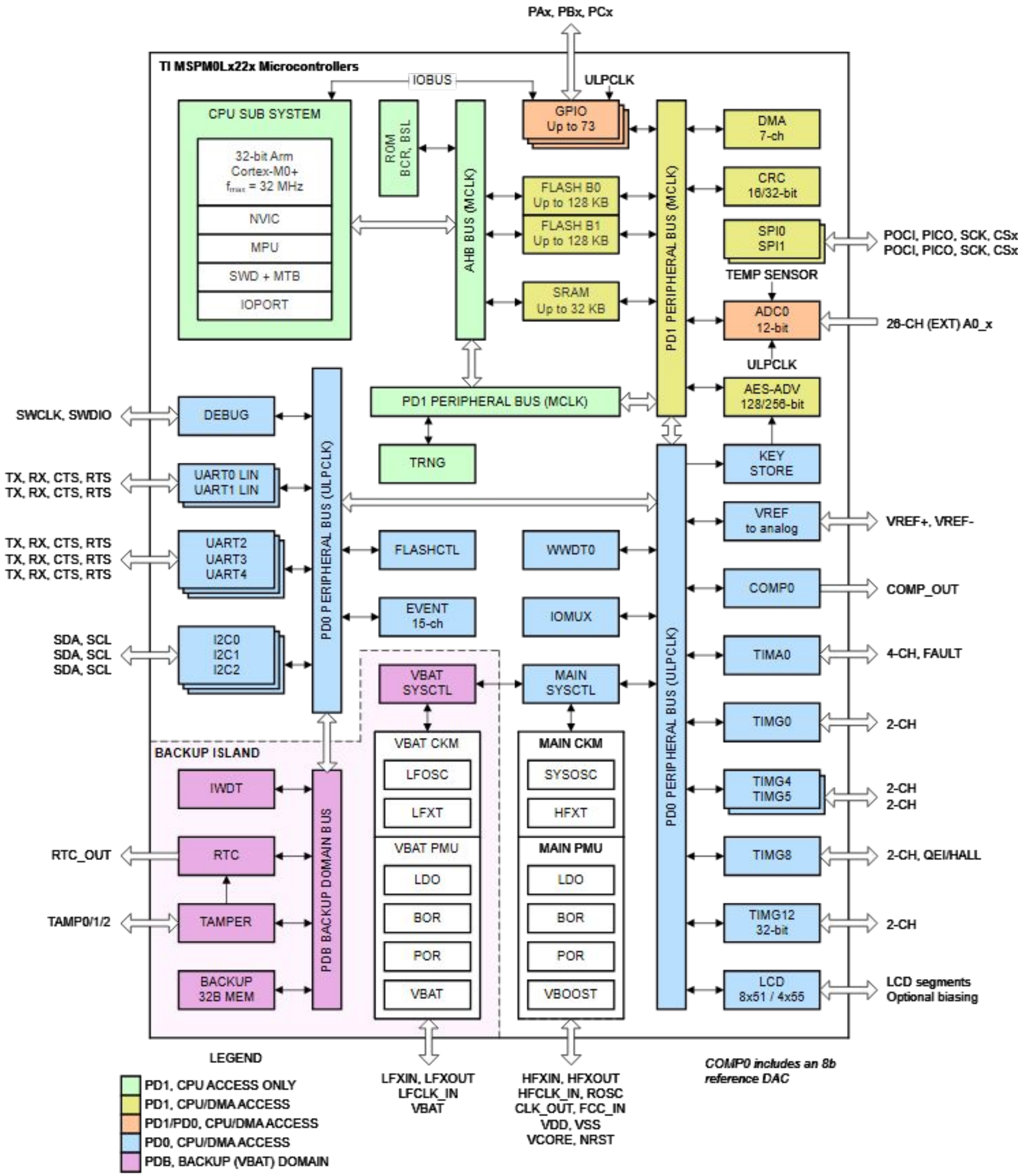


Unified memory architecture

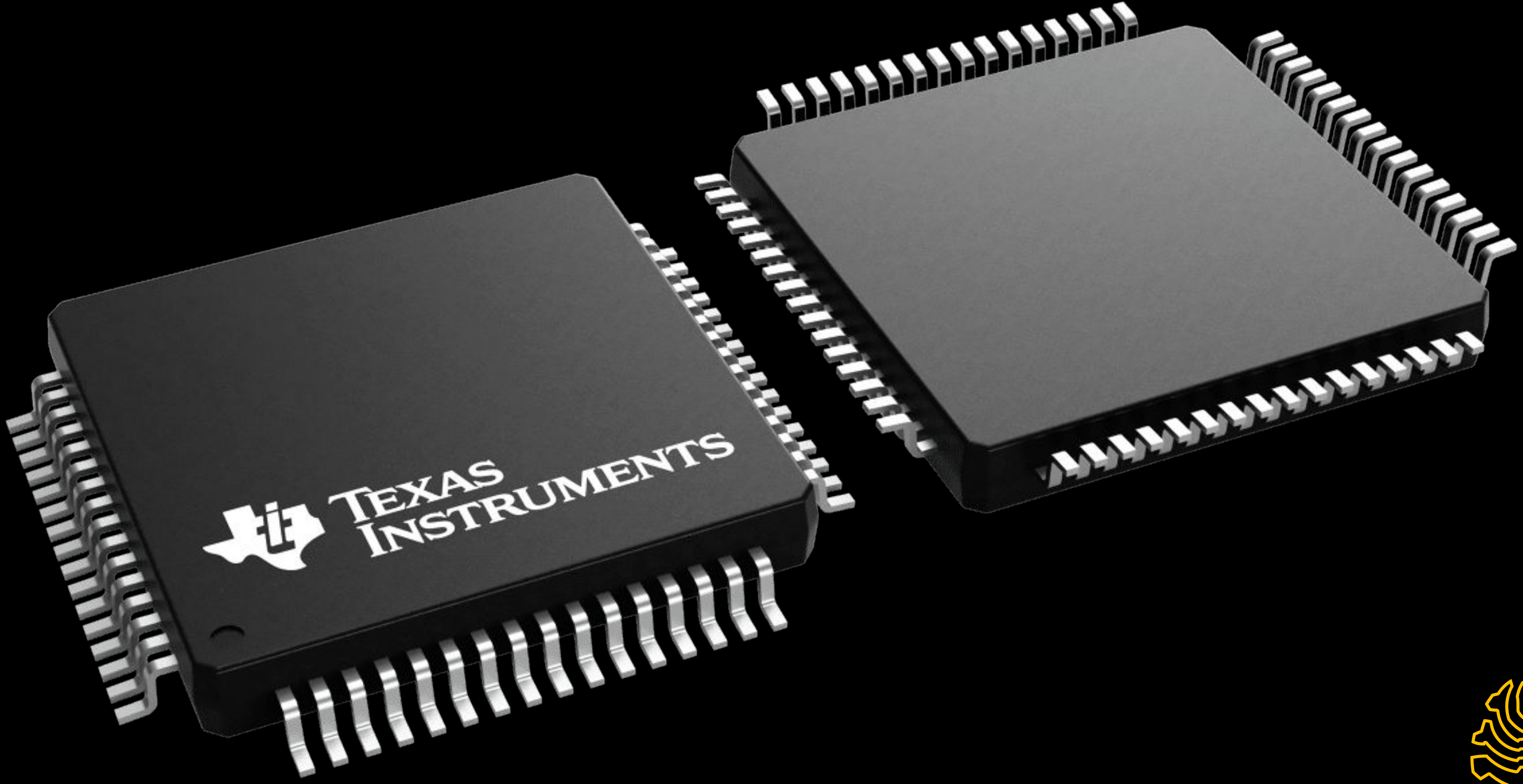
High bandwidth, low latency
Apple-designed package
Accessible to entire SoC

Parts of an MCU

- CPU cores
- RAM
- Peripherals
  - UART
  - Timers
  - TRNG
  - I²C
  - etc.



Pins



Memory (RAM vs Flash)

- Memory is represented as bytes - each byte has a memory address!
- A memory address looks like 0xFFFFFFFF
- Random Access Memory (RAM)
  - Volatile memory CPU has access to
- Flash Memory
  - Non-volatile memory that persists without power supply

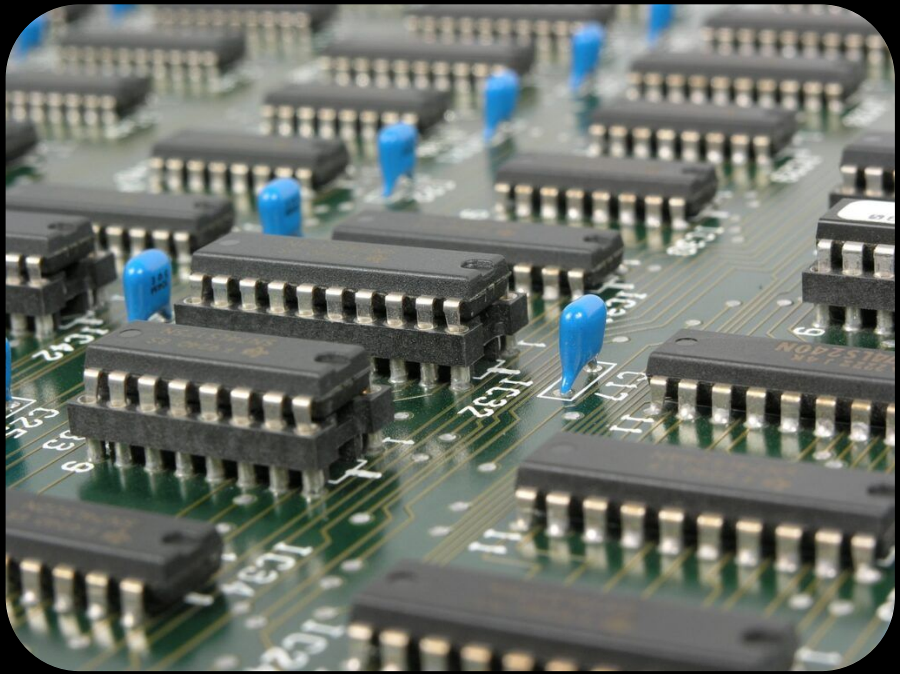
Peripherals

Sensors and Actuators

- Sensor converts a physical phenomenon into an electrical signal
  - Thermometers
  - Cameras
  - Infrared Receiver
- Actuator converts an electrical signal into a physical phenomenon
  - Motors
  - Solenoids
  - Infrared Transmitters
- Transducers can be either a sensor and/or actuator
  - Infrared transceivers can both transmit and receive

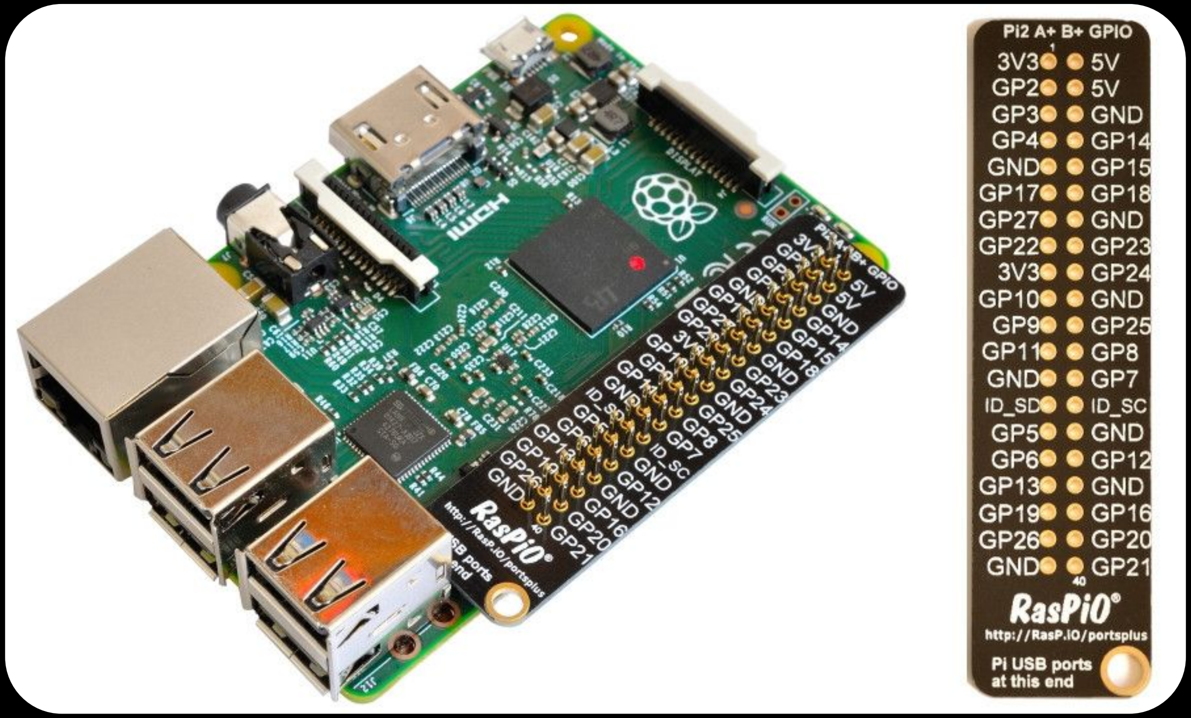
Integrated Circuits (ICs)

- Semiconductor that integrates collections of electronic circuits
  - Transistors, resistors, capacitors, diodes
- Control CPU functions of embedded systems
  - Retrieve and decode instructions from memory
  - Use instructions to perform computations for memory and I/O devices



General Purpose Input/Output (GPIO)

- Most peripherals have reserved pins
- If you need to control pins directly, GPIO allows you to control the state of pins (e.g. on or off)
- Unused by default, purpose defined and implemented by developer



Embedded Communication

How do Peripherals Communicate?

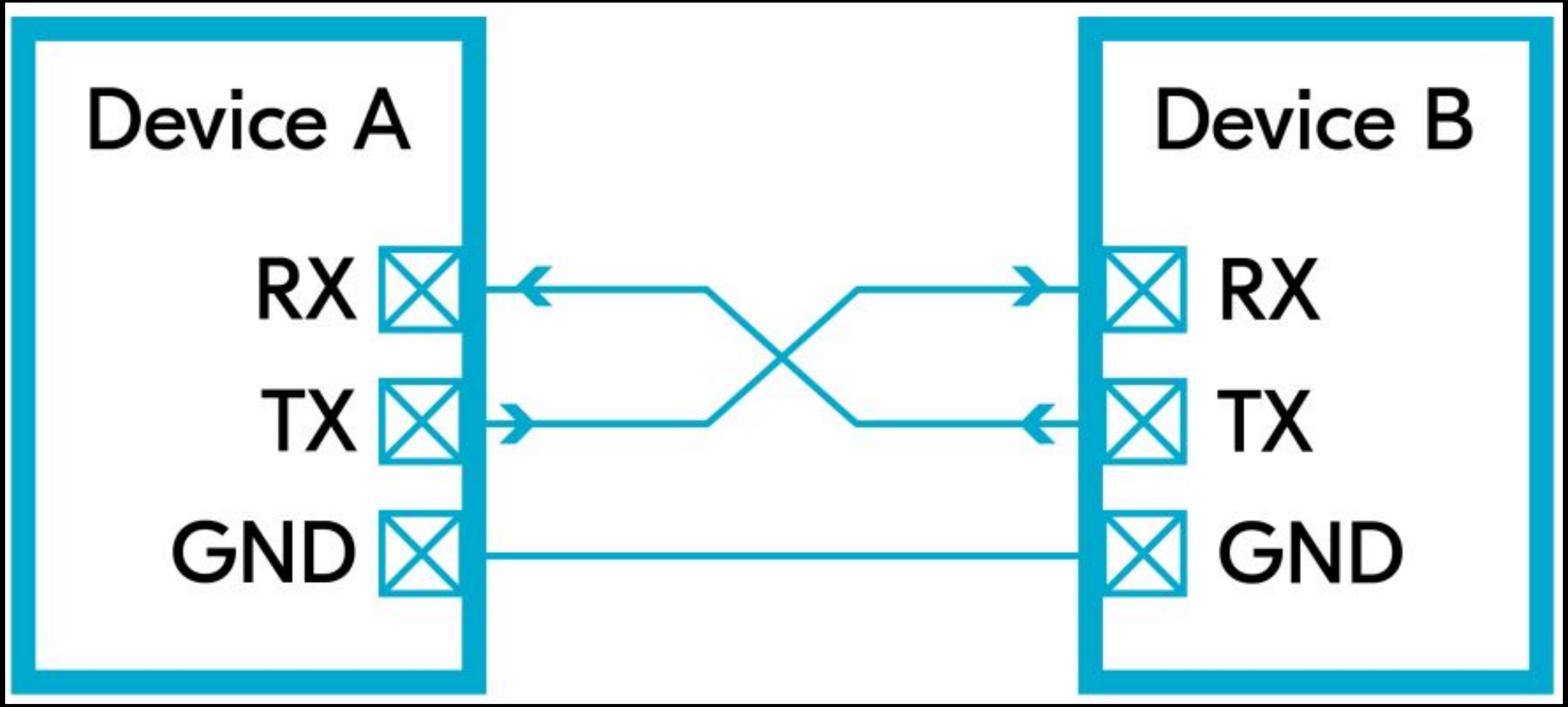
- Each part of an embedded device needs to communicate with each other
  - MCU to peripherals
  - MCU to another MCU
  - MCU to the world

- We use wire **protocols** to standardize these communications

Communication with Protocols

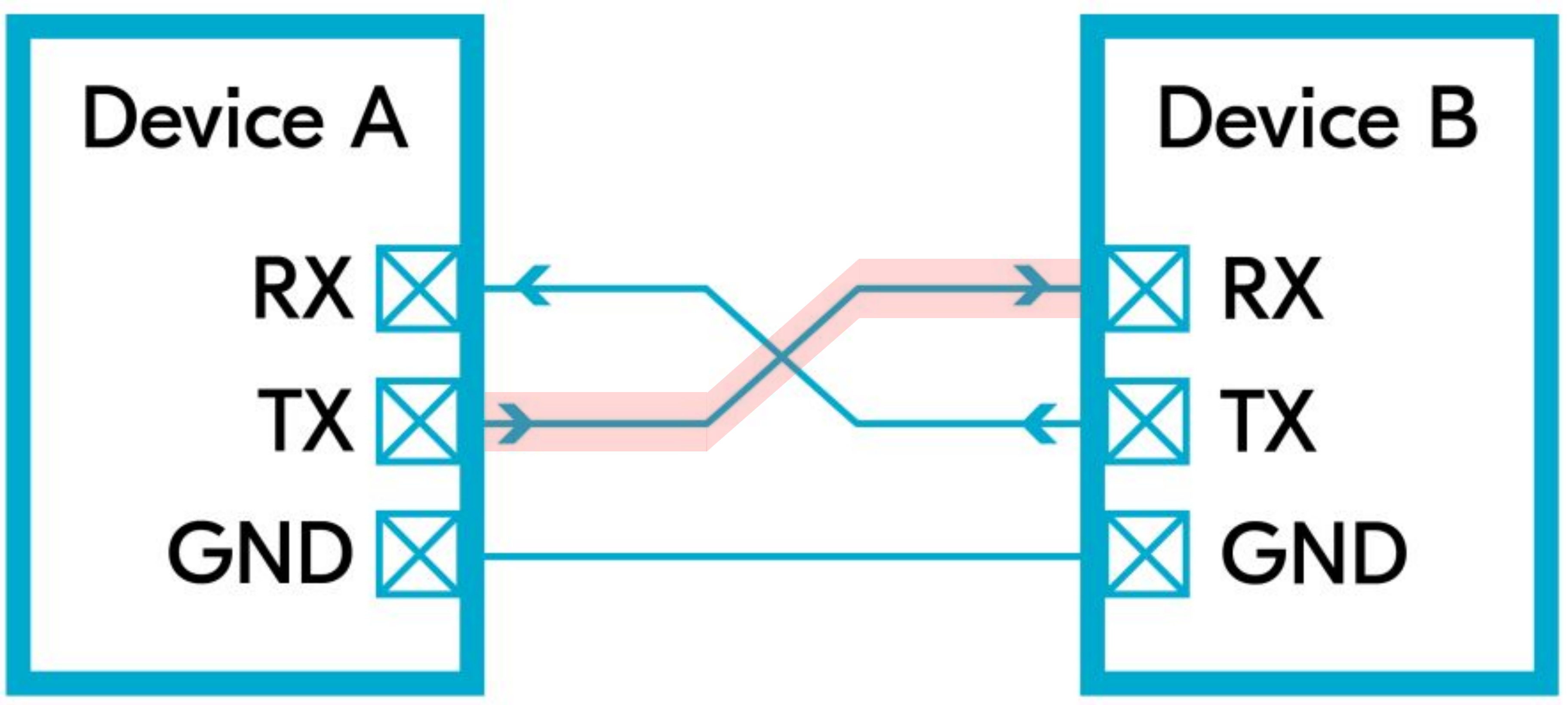
- There are many different protocols, each with their own advantages/disadvantages
  - I^2C allows a host device to communicate to MANY guest devices using only two wires
  - UART allows one-to-one communication using two wires, but is much easier to implement
  - SPI uses four wires but offers much faster data rates, which is good for flash chips where streams of data needs to be read or written

UART Step-by-step



TX - Transmitter
RX - Receiver
GND - Ground

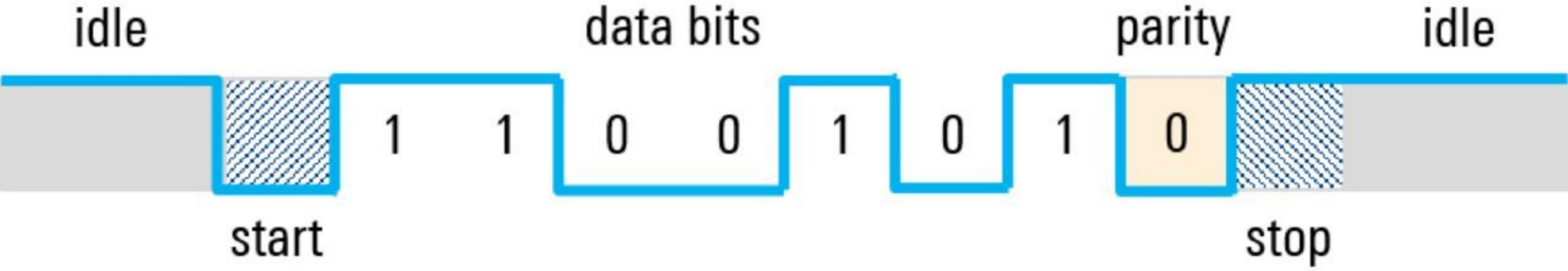
UART Step-by-step



Baud Rate - Number of bits per second
that can be transmitted by channel

Most common UART configuration:

- 8 data bits
- No parity
- 1 stop bit
- Baud rate of 115200 or 9600



https://www.rohde-schwarz.com/us/products/test-and-measurement/essentials-test-equipment/digital-oscilloscopes/understanding-uart_254524.html

UART Demo

Software and Firmware

Software vs Firmware

- Software - programs run by computer
- Typically managed by operating systems
    - Lots of overhead
    - Provides system security

- Firmware = software for hardware
- Firmware can be run with full "bare-metal" access
    - No OS managing code
- More efficient for embedded but potentially less safe

How does Firmware Control Peripherals?

- Microcontrollers usually come with built-in peripherals
  - UART peripheral
  - I^2C peripheral
  - TRNG peripheral
- Peripherals are mapped to reserved memory addresses
- Software can control peripherals by reading/writing values to these memory addresses

Datasheets

Texas Instruments
www.ti.com

MSPM0L2228, MSPM0L2227
MSPM0L1228, MSPM0L1227
SLASF94A – MAY 2024 – REVISED OCTOBER 2024

Table of Contents

Datasheets

Table 8-4. Memory Organization

MEMORY REGION	SUBREGION	MSP0L1227, MSPM0L2227	MSPM0L1228, MSPM0L2228
Code (Flash Bank 0)	MAIN ECC Corrected	64KB[1] 0x0000.0000 to 0x0000.FFFF	128KB[1] 0x0000.0000 to 0x0001.FFFF
	MAIN ECC Uncorrected	0x0040.0000 to 0x0040.FFFF	0x0040.0000 to 0x0041.FFFF
	Flash ECC code	0x0080.0000 to 0x0080.FFFF	0x0080.0000 to 0x0081.FFFF
Code (Flash Bank 1)	MAIN ECC Corrected	64KB[1] 0x0001.0000 to 0x0001.FFFF	128KB[1] 0x0002.0000 to 0x0003.FFFF
	MAIN ECC Uncorrected	0x0041.0000 to 0x0041.FFFF	0x0042.0000 to 0x0043.FFFF
	Flash ECC code	0x0081.0000 to 0x0081.FFFF	0x0082.0000 to 0x0083.FFFF
SRAM (SRAM)	SRAM "ECC Checked"	32KB 0x2000.0000 to 0x2000.7FFF	32KB 0x2000.0000 to 0x2000.7FFF
	Parity checked	0x2010.0000 to 0x2010.7FFF	0x2010.0000 to 0x2010.7FFF
	Un-checked	0x2020.0000 to 0x2020.7FFF	0x2020.0000 to 0x2020.7FFF
	ECC/parity code	0x2030.0000 to 0x2030.7FFF	0x2030.0000 to 0x2030.7FFF

Datasheets

Table 8-5. Peripherals Summary

PERIPHERAL NAME	BASE ADDRESS
ADC0	0x40004000
COMP0	0x40008000
VREF	0x40030000
LCD	0x40070000
WWDT0	0x40080000
TIMG0	0x40084000
TIMG4	0x4008C000
TIMG5	0x4008E000
TIMG8	0x40090000
LFSS (SPM, TIO)	0x40094000
RTC_A	0x40095100
IWDT	0x40095300
GPIOA	0x400A0000
GPIOB	0x400A2000
GPIOC	0x400A4000
KEYSTORE	0x400AC000
SYSCTL	0x400AF000
DEBUGSS	0x400C7000
EVENT	0x400C9000
NVM	0x400CD000
I2C0	0x400F0000
I2C1	0x400F2000
I2C2	0x400F4000
UART2	0x40100000
UART3	0x40102000
UART4	0x40104000
UART0	0x40108000
UART1	0x4010A000

21.3 UART Registers

Table 21-12 lists the memory-mapped registers for the UART registers. All register offset addresses not listed in Table 21-12 should be considered as reserved locations and the register contents should not be modified.

Table 21-12. UART Registers

Offset	Acronym	Register Name	Group	Section
800h	PWREN	Power enable		Go
804h	RSTCTL	Reset Control		Go
808h	CLKCFG	Peripheral Clock Configuration Register		Go
814h	STAT	Status Register		Go
1000h	CLKDIV	Clock Divider		Go
1008h	CLKSEL	Clock Select for Ultra Low Power		Go

21.3.35 TXDATA (Offset = 1120h) [Reset = 00000000h]

TXDATA is shown in Figure 21-50 and described in Table 21-48.

Return to the Summary Table.

UART Transmit Data Register. This register is the transmit data register (the interface to the FIFOs). For transmitted data, if the FIFO is enabled, data written to this location is pushed onto the transmit FIFO. If the is disabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). A write this register initiates a transmission from the UART.

Figure 21-50. TXDATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
RESERVED																								DATA					
R/W-0h																								R/W-0h					

Table 21-48. TXDATA Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R/W	0h	
7-0	DATA	R/W	0h	Data Transmitted or Received Data that is to be transmitted via UART is written to this field. When read, this field contains the that was received by the UART.

Datasheets in Action!

```
char my_string[] = "Hello, world!";
for (unsigned int i = 0; i < 13; i++) {
    // checks if UART is ready to send
    if (*0x40001030 & (1 << 5)) {
        return E_OVERFLOW;
    }
    *0x40001034 = my_string[i];
}
```

Manipulating peripherals is just reading and writing fields/values to some defined memory addresses!

HALs and Embedded SDKs

- “Hardware Abstraction Layers” (HALs) are libraries which make interfacing with the hardware easier
  - Abstracts technical details away to make software development better

- Manufacturers will create “Software Development Kits” (SDKs) to assist developers with writing firmware
  - SDKs are more to be used by software and application code
  - Will often include HALs for specific device and additional code, such as code to control external peripherals, such as displays or LEDs

Would You Rather?

```
char my_string[] = "Hello, world!";
for (unsigned int i = 0; i < 13;
i++) {
    if (*0x40001030 & (1 << 5)) {
        return E_OVERFLOW;
    }
    *0x40001034 = my_string[i];
}
```

```
char my_string[] = "Hello, world!";
uart0_write(my_string);
```

Next Meetings

`2025-09-22` • **Next Monday**

- Embedded 102: Microcontroller Programming
- Learn how to program a microcontroller!
- We will have microcontrollers available for you to program:
    - UART peripheral to print “Hello, world!” to a console
    - GPIO peripheral to trigger an LED

Meeting content can be found at sigpwny.com/meetings.

SIGPwny