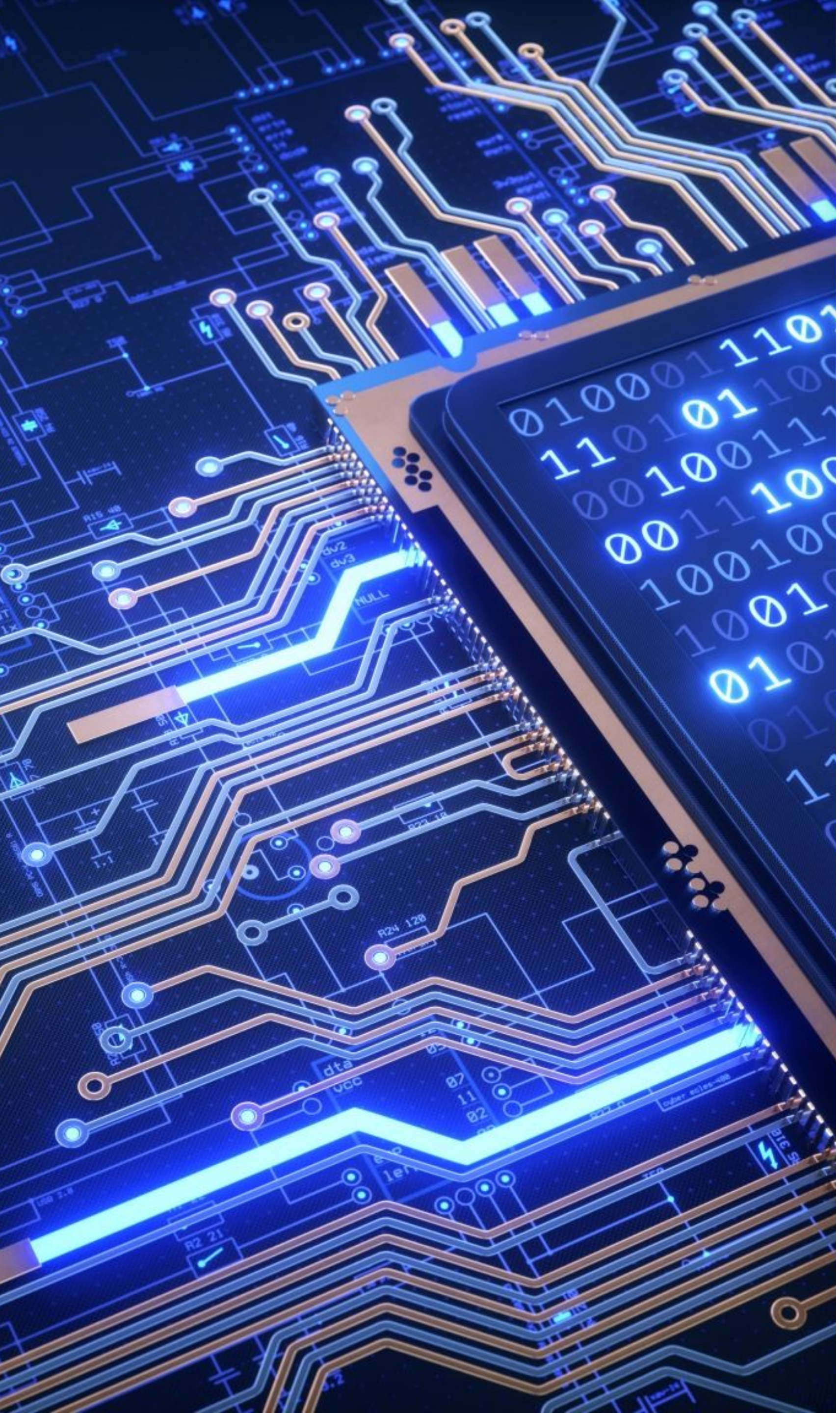
Lecture 1 – Introducing Embedded Systems and ARM Microcontroller

- By Dr. Min He

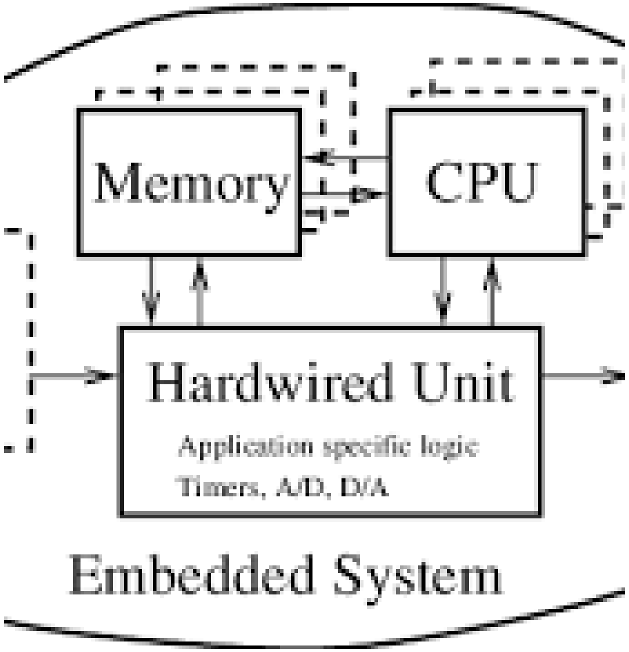


Outline

- ❑ What is an embedded system?
- ❑ Microcontroller vs. Microprocessor
- ❑ Embedded Systems’ Industry Roles
- ❑ Software Design: Desktop versus Embedded
- ❑ Introducing Microcontrollers
- ❑ ARM Cortex Microcontrollers
  - ➢ Inside the processor
  - ➢ System buses
  - ➢ Peripherals
  - ➢ GPIO
  - ➢ Interrupts
  - ➢ Operating Modes
  - ➢ Memory-mapped I/O Processor Architecture
- ❑ Lab Preparation

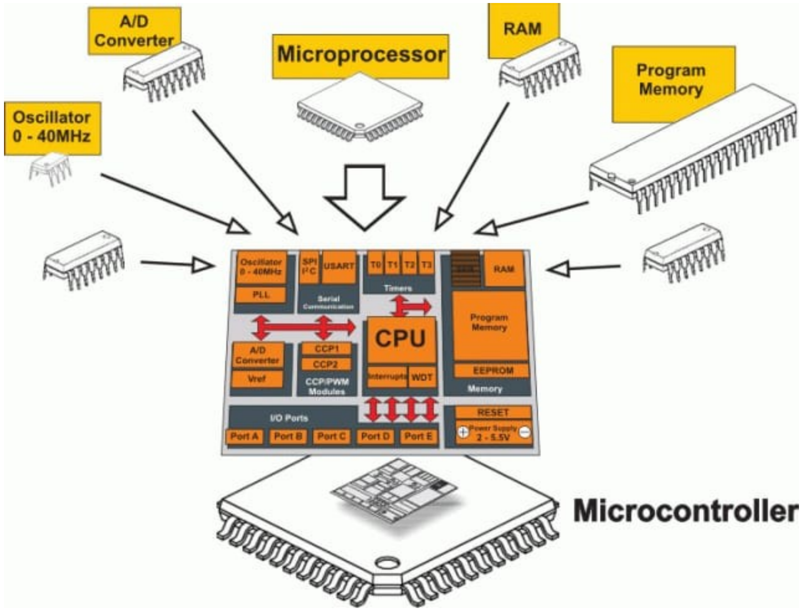
Embedded Systems -- Shape the World

- An **embedded system** is a computer system—a combination of a computer processor, computer memory, and input/output peripheral devices—that has a dedicated function within a larger mechanical or electronic system. – Wikipedia.
- Simply put: An embedded system is essentially a computer that is designed to control or perform certain functions either by itself or within a larger system.
- At its core, a processor is used to carry out computations and perform real-time operations.



Microprocessor vs. Microcontroller

- **Microprocessor**: a central processing unit that executes and manages the logical instructions.
- **Microcontroller**: includes one or more CPUs along with memory and programmable input/output peripherals.



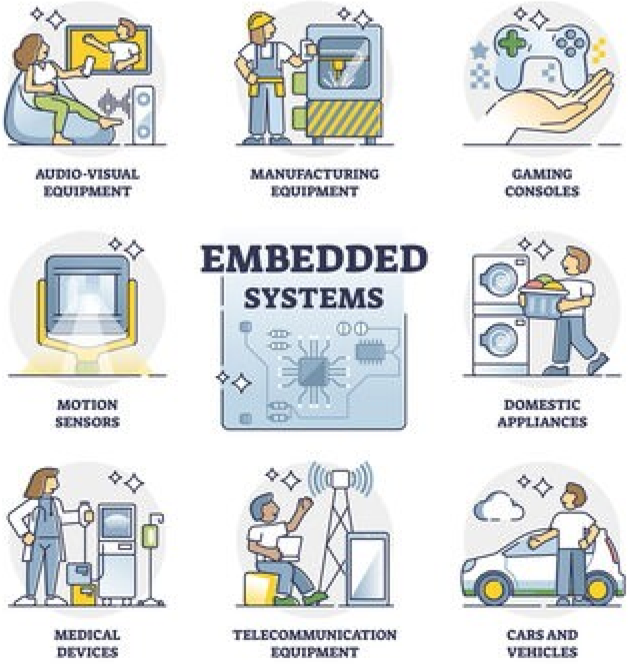
Industry Roles

Embedded systems play a crucial role in various industries, including automotive, healthcare, consumer electronics, industrial automation, and IoT (Internet of Things). Here are some examples:

- ✓ IoT Industry:
- ✓ Automotive Industry:
- ✓ Consumer Electronics:
- ✓ Medical Devices:
- ✓ Industrial Automation:
- ✓ Robotics:

•Current career opportunities:

- https://www.glassdoor.com/Salaries/embedded-systems-engineer-salary-SRCH_KO0,25.htm

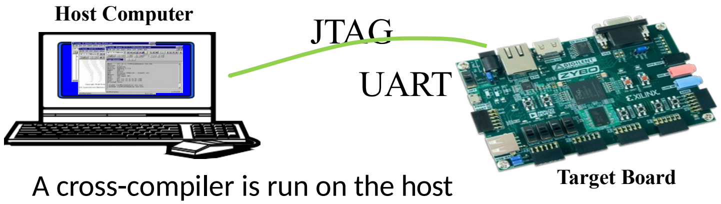


Software Design: Desktop versus Embedded

- Desktop development: written, debugged, and run on the same machine.

- OS loads the program into the memory when the program has been requested to run
- Address resolution takes place at the time of loading by a program called the loader
    - The loader is included in the OS

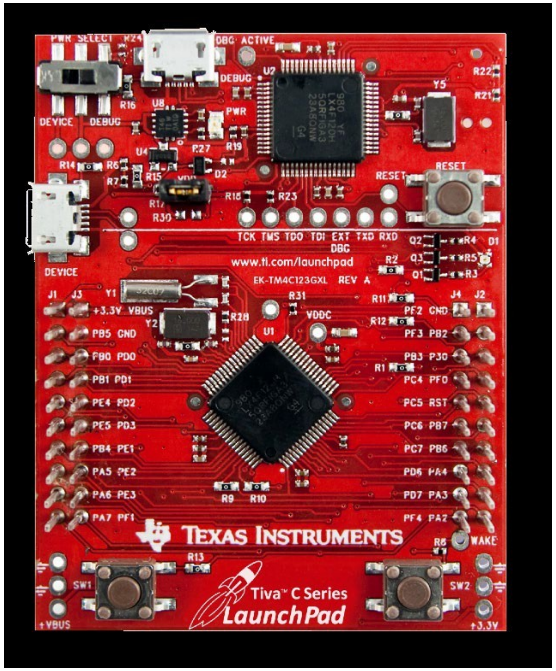
- Development takes place on one machine (host) and is downloaded to the embedded system (target).



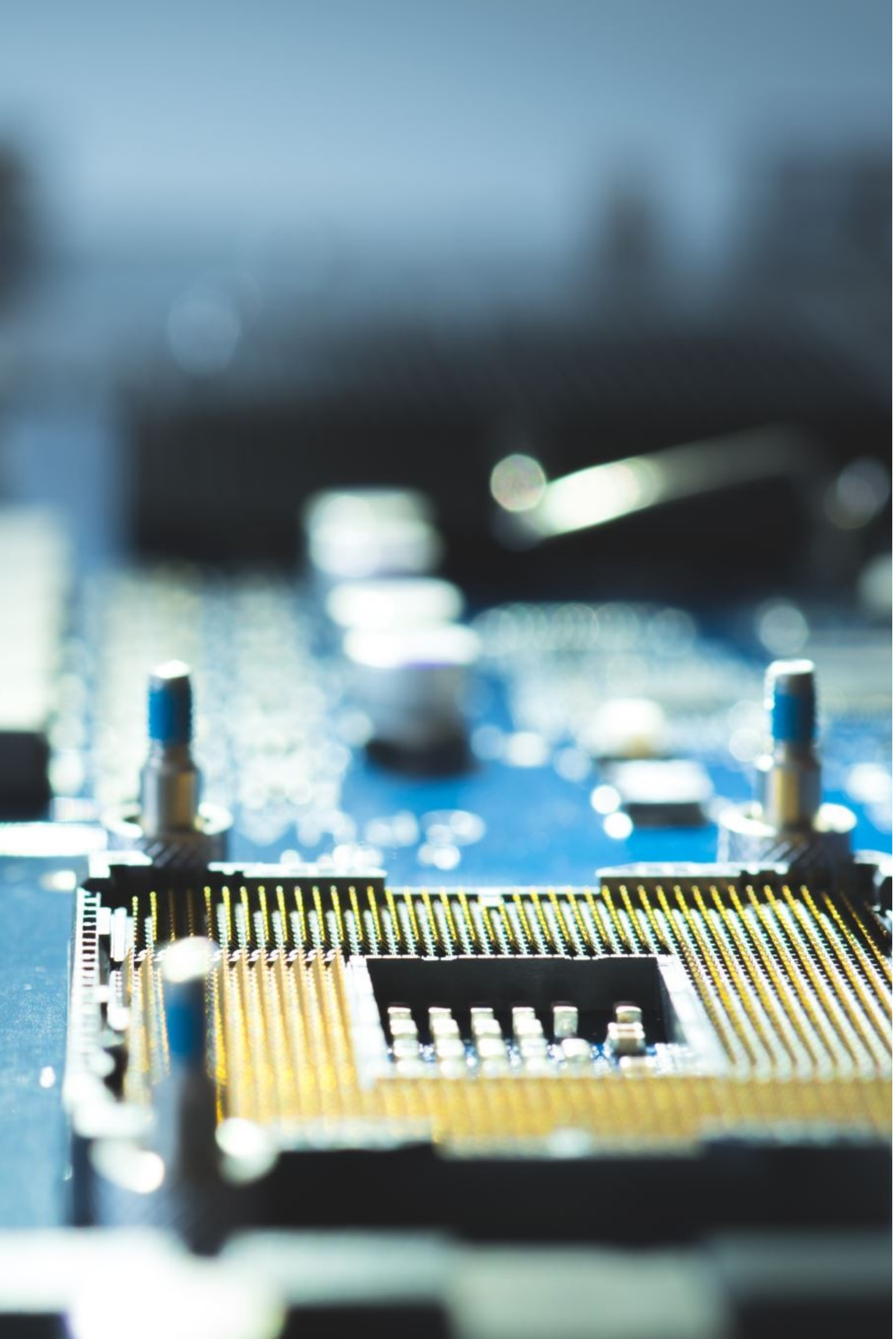
A cross-compiler is run on the host

- The programmer glues into one executable file called ELF
    - Boot code, application code, RTOS, and ISRs
    - Address resolution takes place during the *gluing* stage
- The executable file is downloaded into the target system through different methods
    - Ethernet, serial, JTAG, BDM, ROM programmer

Microcontroller



- A microcontroller is a small computer on an integrated circuit, or chip, that includes a CPU, memory, and peripheral input/output devices.
- Microcontrollers are programmed to build embedded systems that perform a specific task and manage other components within the system, including memory, such as RAM or ROM, and input/output devices that can include LED displays, switches and various types of sensors.
- In order for the hardware components to function, embedded software is used to provide instructions for the system.
- Microcontrollers are often programmed using higher-level languages such as C/C++ & Python and low-level language such as assembly.



Microcontroller Put it simple

I want to do “stuff”. I need…

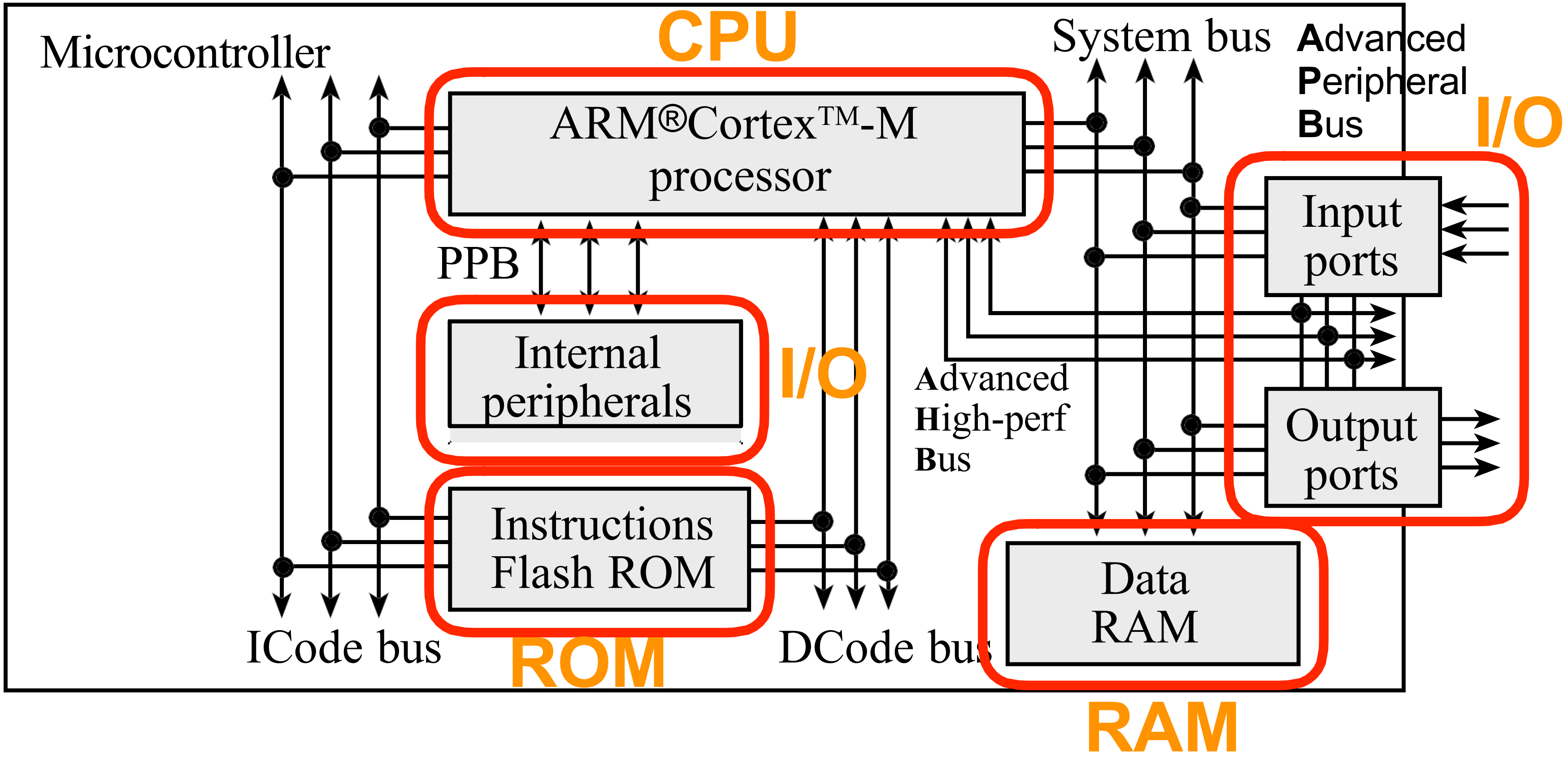
- **CPU**: Thing that performs calculations.
- **ROM:** Place to store a program.
- **RAM:** Place to store data.
- **I/O:** Ability to talk to other things.

Introducing TI TM4C123

ARM Cortex-M4 processor

- *Harvard* architecture:
  - Separate buses for instructions and data
  - vs Von Neumann architecture - shared bus
- 32-bit RISC machine:
  - Simpler (easier to implement / faster) instructions
  - Program requires more instructions (more memory)
  - vs CISC: uses less memory
- *Pipelining* effectively provides single cycle operation for many instructions
  - 3 stage pipeline: Fetch, Decode, Execute
- Thumb-2 configuration employs both 16 and 32 bit instructions
  - Increases code density

ARM Cortex-M4 Microcontroller



TM4C123 GPIO are on both buses

System Buses

Data are exchanged with memory and I/O via the system bus interface.

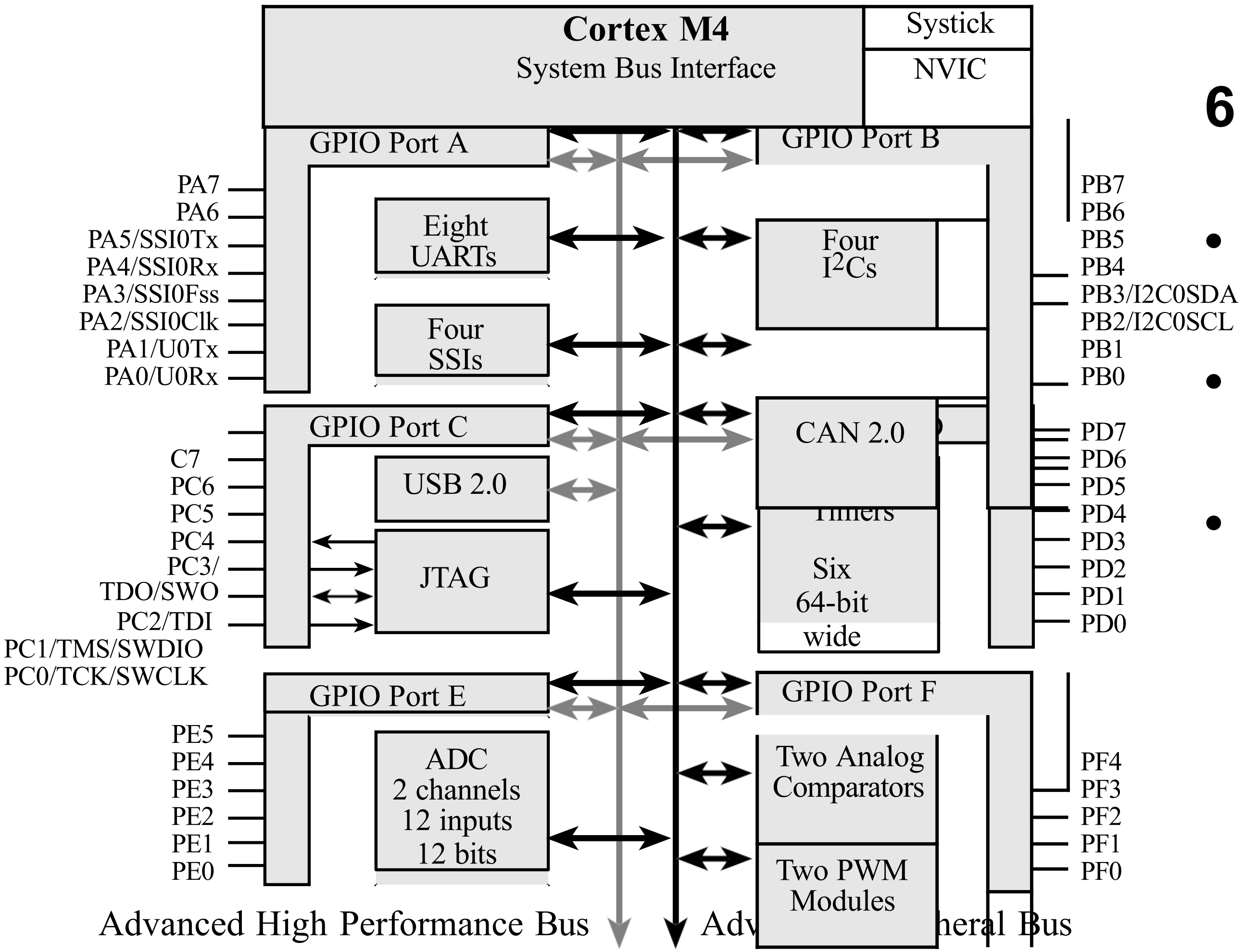
- PPB: Private Peripheral Bus. Used for communication between processor and internal peripherals, such as NVIC(nested vectored interrupt controller).
- APB: Advanced Peripheral Bus. Used for communication between processor and external peripherals
- AHB: Advanced High–Performance Bus. Used for high speed external devices like USB

Peripherals

Peripherals on a microcontroller are additional hardware components integrated into the microcontroller chip to provide extra functionality beyond the core processing capabilities

- General-Purpose Input/Output(GPIO)
- Analog-to-Digital Converter (ADC)/Digital-to-Analog Converter (DAC)
- Timers and Counters: Systick timer, general purpose timers
- Serial Communication Interfaces: SSI, UART, I2C
- PWM (Pulse Width Modulation)
- Interrupt Controller: NVIC(Nested Vectored Interrupt Controller)
- Watchdog Timer
- Communication Peripherals: CAN
- Memory Controllers
- USB (Universal Serial Bus) Controllers
- Clock and Oscillator Circuits: PLL

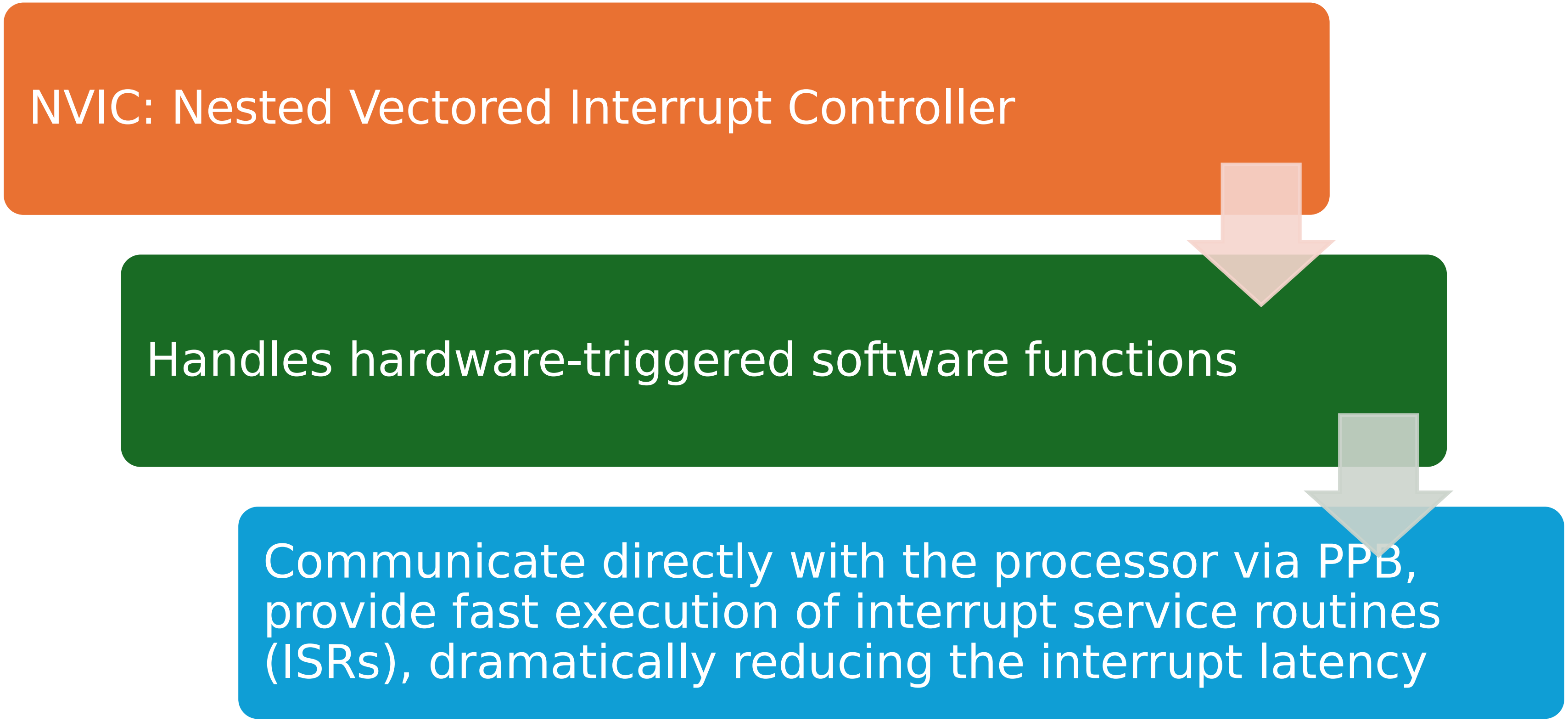
GPIO



6 General-Purpose I/O (GPIO) ports

- Four 8-bit ports (A, B, C, D)
- One 6-bit port (E)
- One 5-bit port (F)

Interrupts



NVIC: Nested Vectored Interrupt Controller

Handles hardware-triggered software functions

Communicate directly with the processor via PPB, provide fast execution of interrupt service routines (ISRs), dramatically reducing the interrupt latency

Two Operating Modes

- *Thread mode* is the foreground operating context
  - This is the context in which your main program and associated subroutines operate
- *Handler mode* is the background operating context
  - This is the context in which the routines that service interrupt requests operate

Memory Mapped I/O Processor Architecture

In a system with *memory mapped I/O*

- I/O devices are connected like memory
- I/O devices are assigned addresses
- Software accesses I/O using these addresses
- Software inputs from an input device
  - same instructions as a memory read
- Software outputs from an output device
  - same instructions as a memory write

Lab Preparations

Follow the instructions in "Getting Started with Embedded Systems" to setup your computer for labs and projects:

- Install Keil 5.39 (MDK539)
- Install Stellaris ICDI Drivers
- Install DLLs: extra support for Keil simulation

Purchase components:

- Components are listed in the document: "CECS346 Components List.docx"