香 港 中 文 大 學（深 圳）
The Chinese University of Hong Kong, Shenzhen

ECE3080
Microprocessors and Computer Systems

Introduction of Embedded System

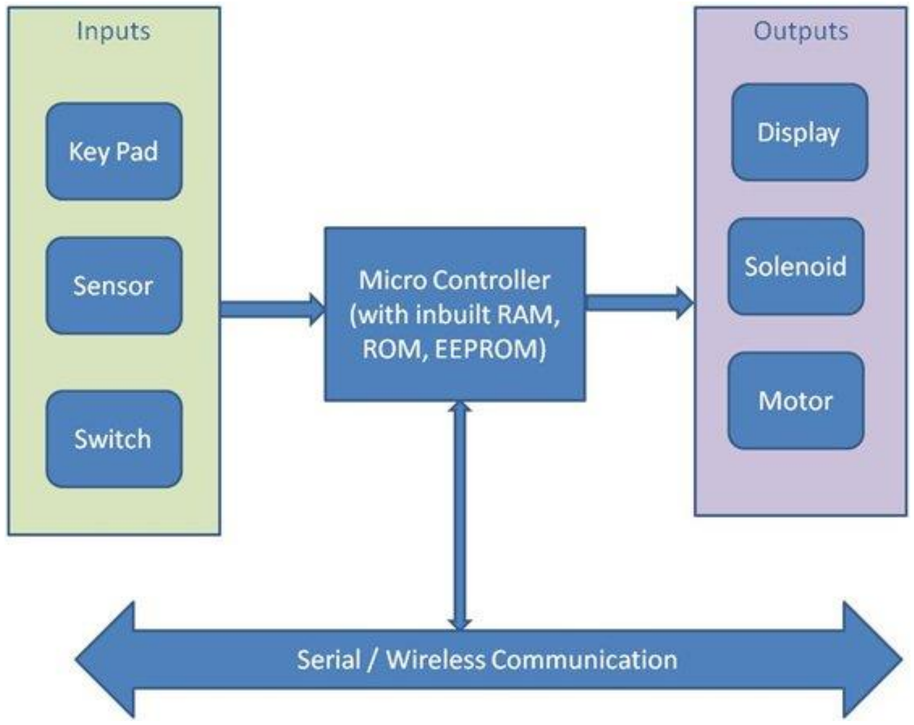
Instructor：Tin Lun LAM

E-mail: tllam@cuhk.edu.cn

URL: https://myweb.cuhk.edu.cn/tllam

What is Embedded System?

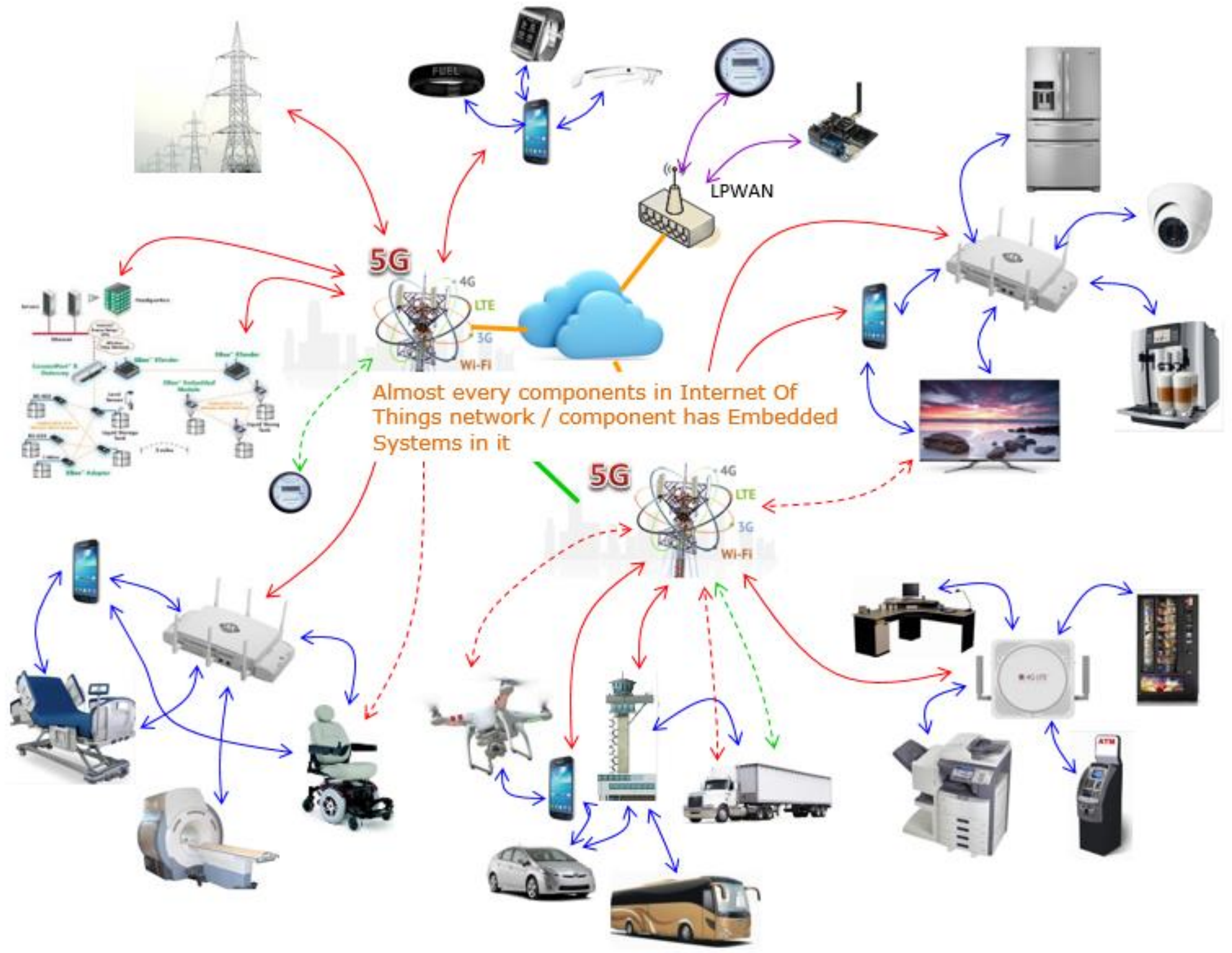
An embedded system is a specialized computer system—a combination of a computer processor, computer memory, and input/output peripheral devices—that has a dedicated function within a larger mechanical or electronic system. Wikipedia



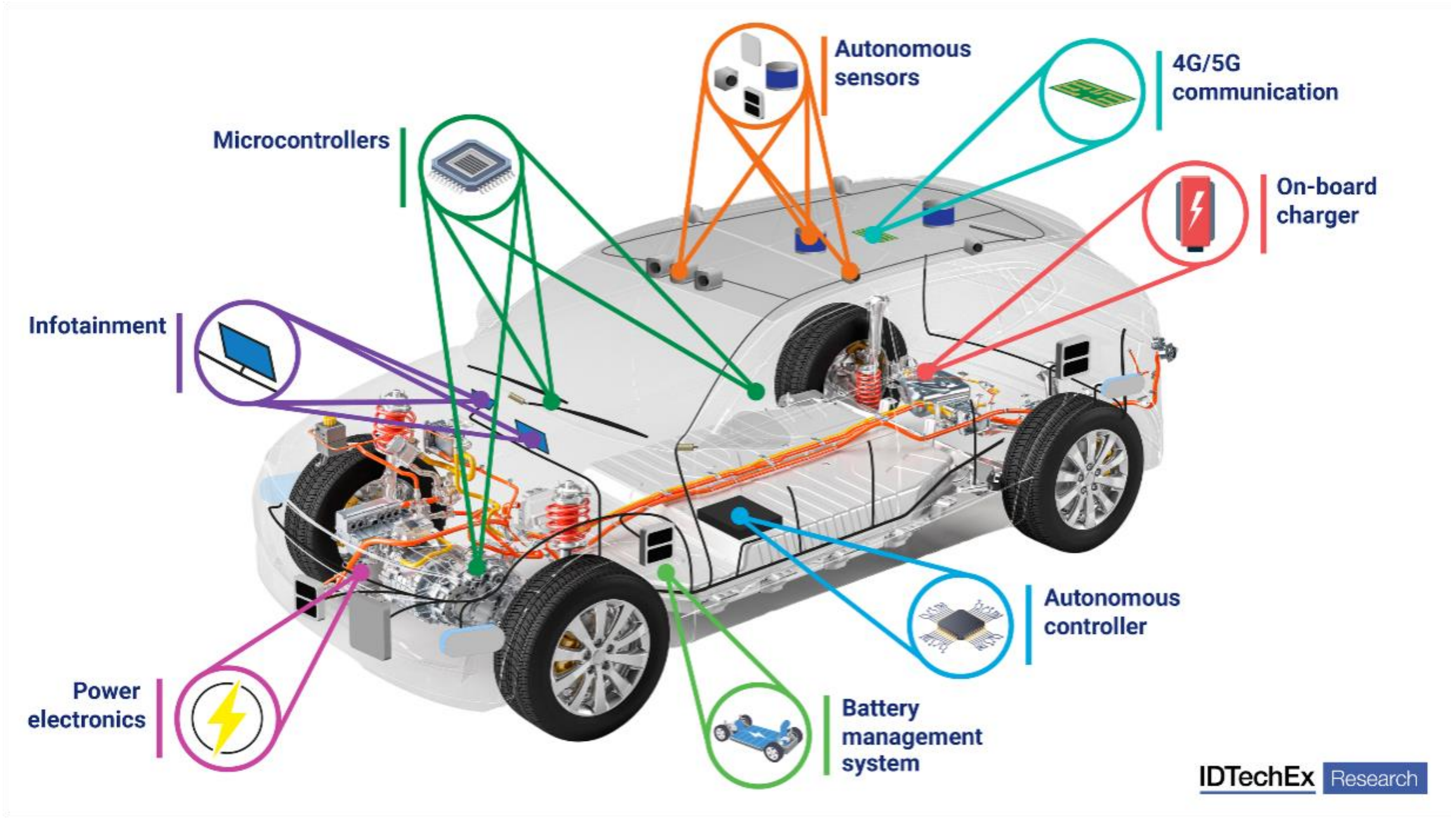
Inputs
Key Pad
Sensor
Switch
Micro Controller (with inbuilt RAM, ROM, EEPROM)
Outputs
Display
Solenoid
Motor
Serial / Wireless Communication

Examples of Embedded Systems

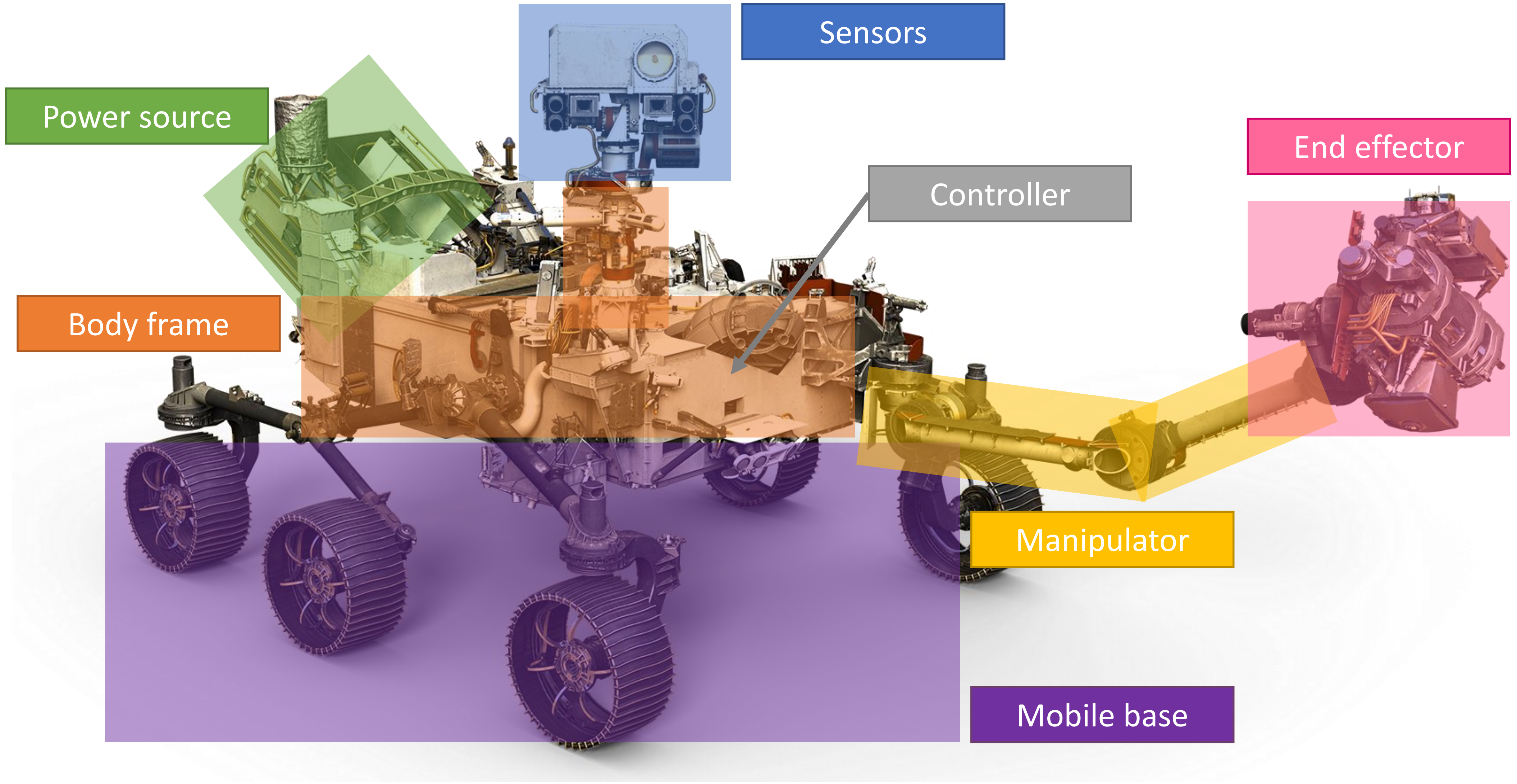
Examples of Embedded Systems


5G
4G
LTE
3G
Wi-Fi
LPWAN
Almost every components in Internet Of Things network / component has Embedded Systems in it
5G
4G
LTE
3G
Wi-Fi

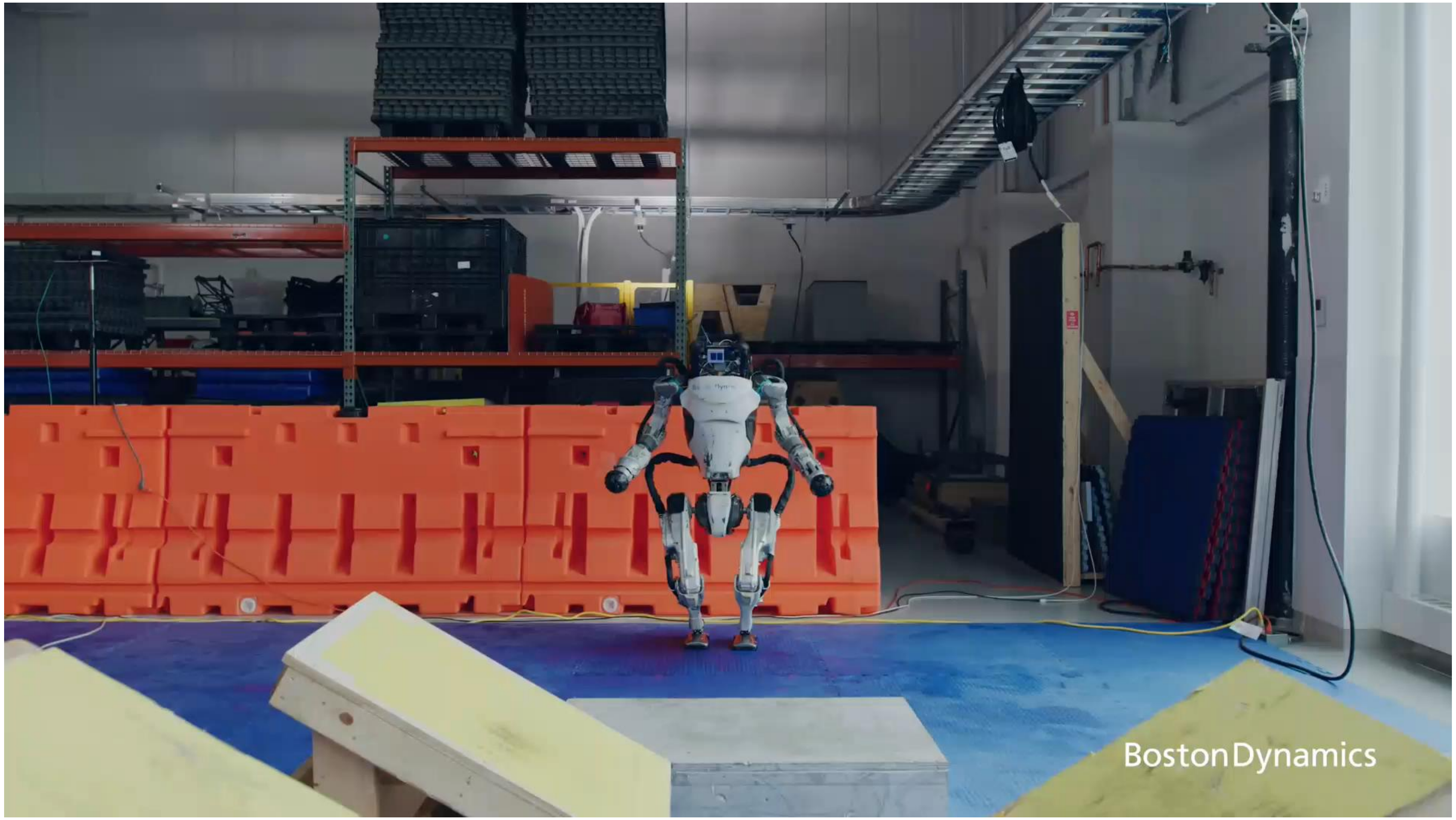
Examples of Embedded Systems



Examples of Embedded Systems



Examples of Embedded Systems - Robot



https://www.youtube.com/watch?v=tF4DML7FIWk

< Embedded System >

Tablets?
Smartphones?

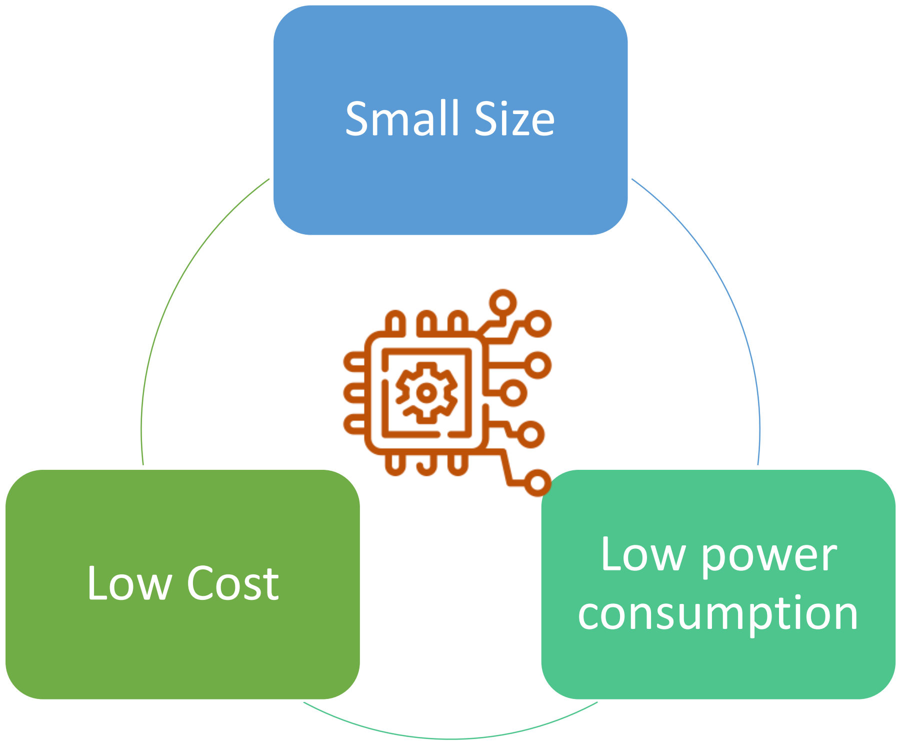
< General Purpose PC >

Personal computers, Servers, Super computers, Workstations

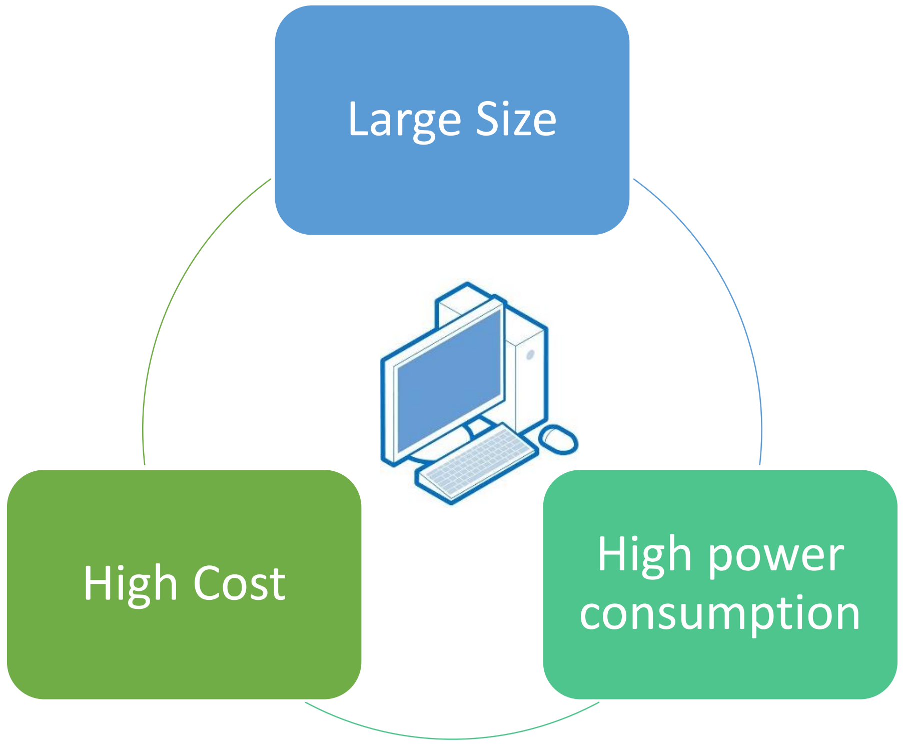
What is NOT an Embedded System?

Characteristics of Embedded Systems

Requirements of Embedded Systems

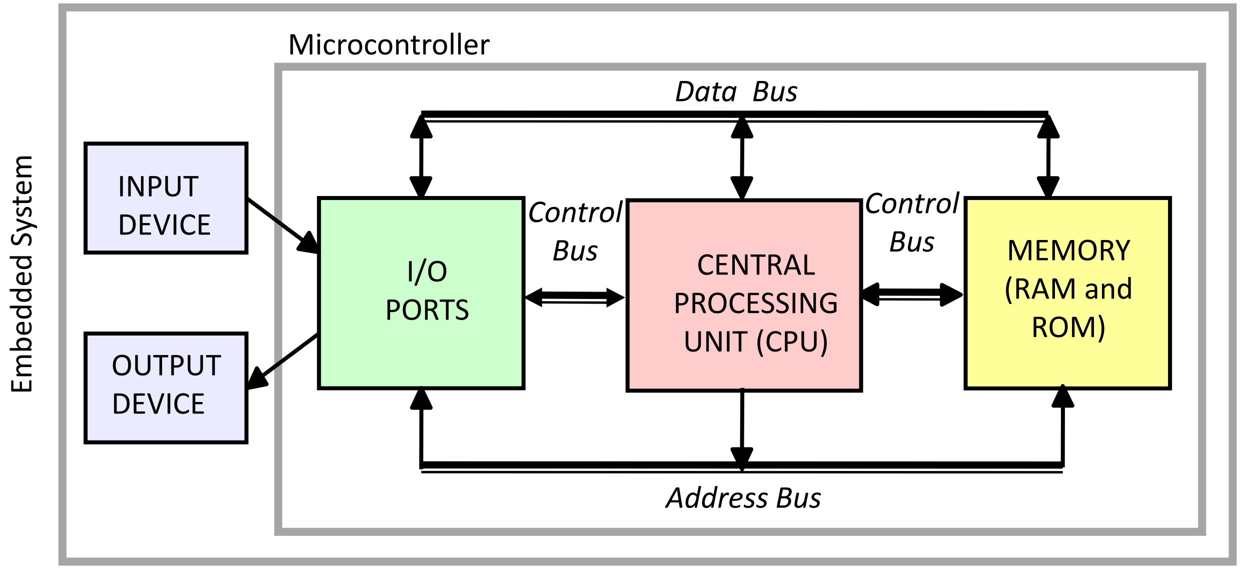


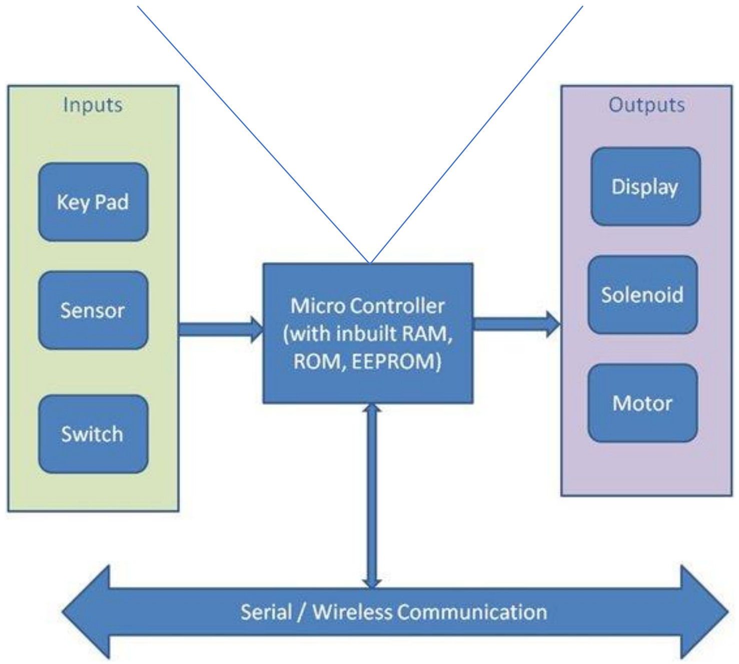
General-purpose Computer



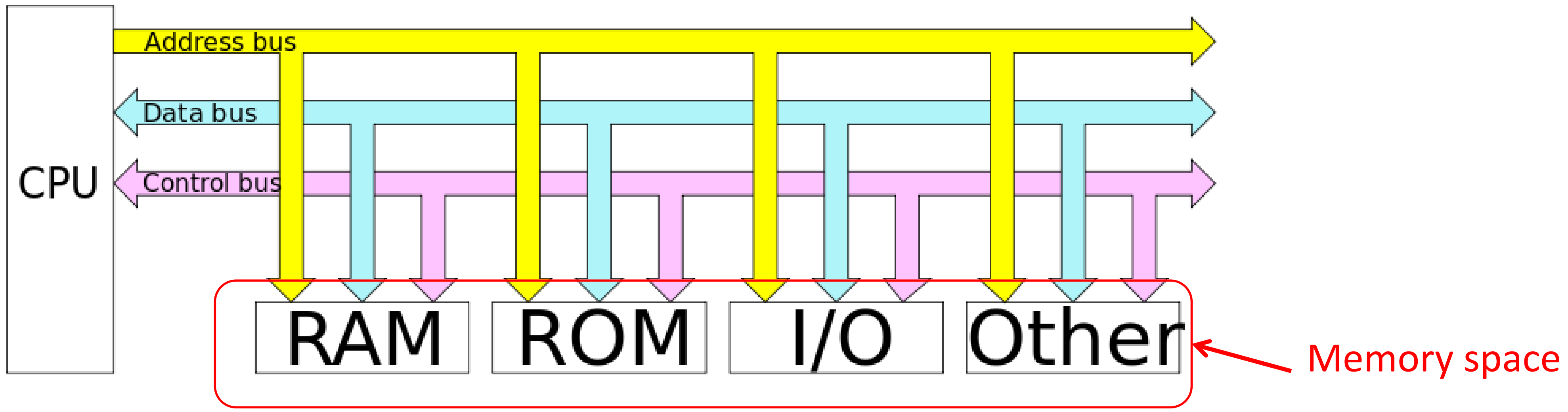
Components of Microcontrollers

- **Processor** - Makes decisions and executes instructions
- **ROM** - Read Only Memory (stores program)
- **RAM** - Random Access Memory (stores temporary data)
- **Buses** - Connections between Processor, ROM, RAM, and Peripherals.
- **Timer** - Let the system work in cycles and coordinate different tasks
- **Interrupt Controller** - Handle job requests from different parts
- **Peripherals** – General-purpose I/O, USART, ADC, …




Inputs
Key Pad
Sensor
Switch
Micro Controller (with inbuilt RAM, ROM, EEPROM)
Outputs
Display
Solenoid
Motor
Serial / Wireless Communication

Basic Computer Architecture

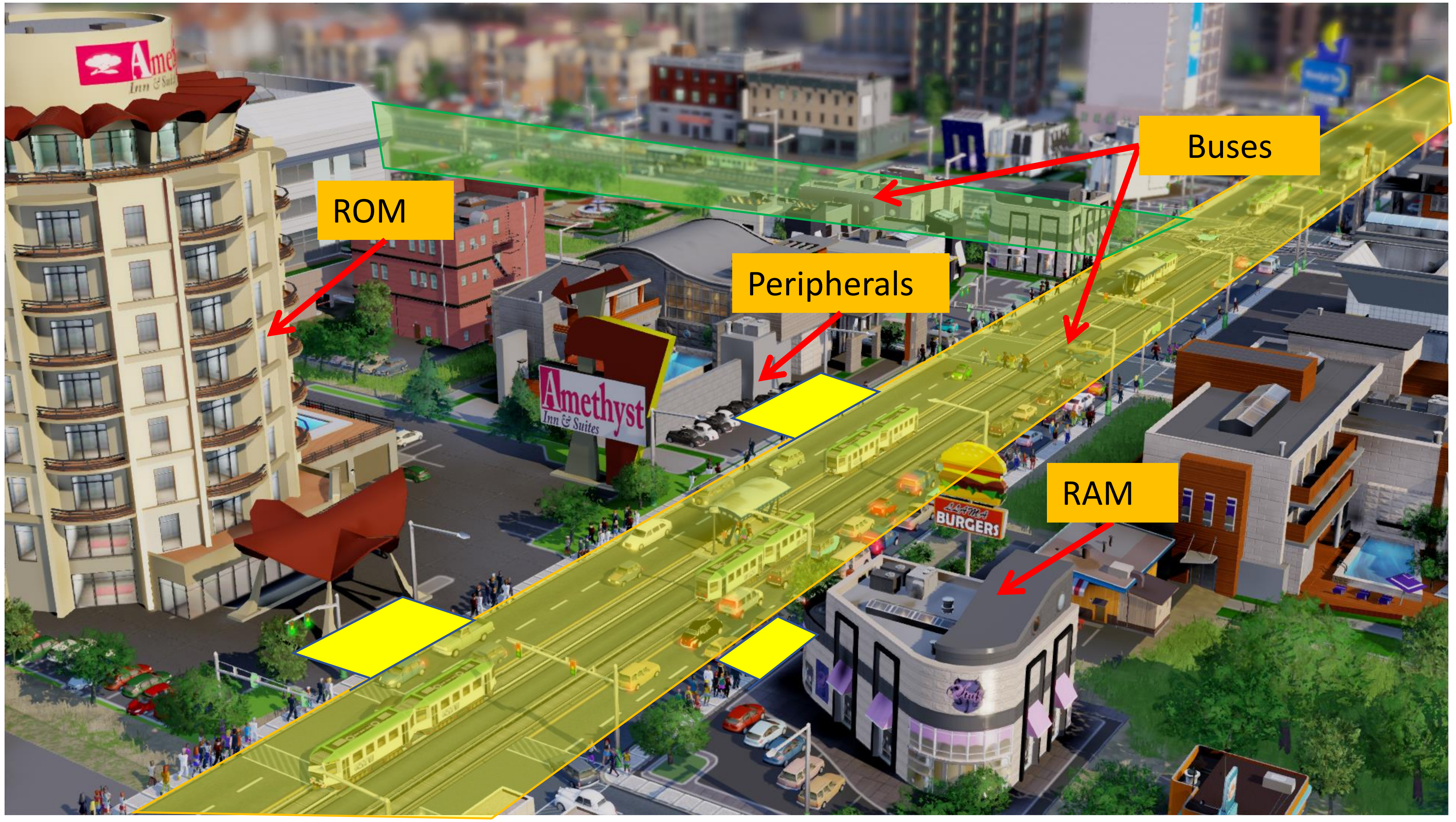


RAM, ROM, I/O and others are assigned to different memory space, and connected to the processor with Address bus, Data bus and Control bus.

- Address bus is unidirectional and address information is generated by the processor.
- Data bus is bi-directional for moving data.
- Control bus carries control signal (like write enable) is generate by the processor to control the data flow. Special signals Interrupts are generated from I/O and others. So Control bus is bi-directional.

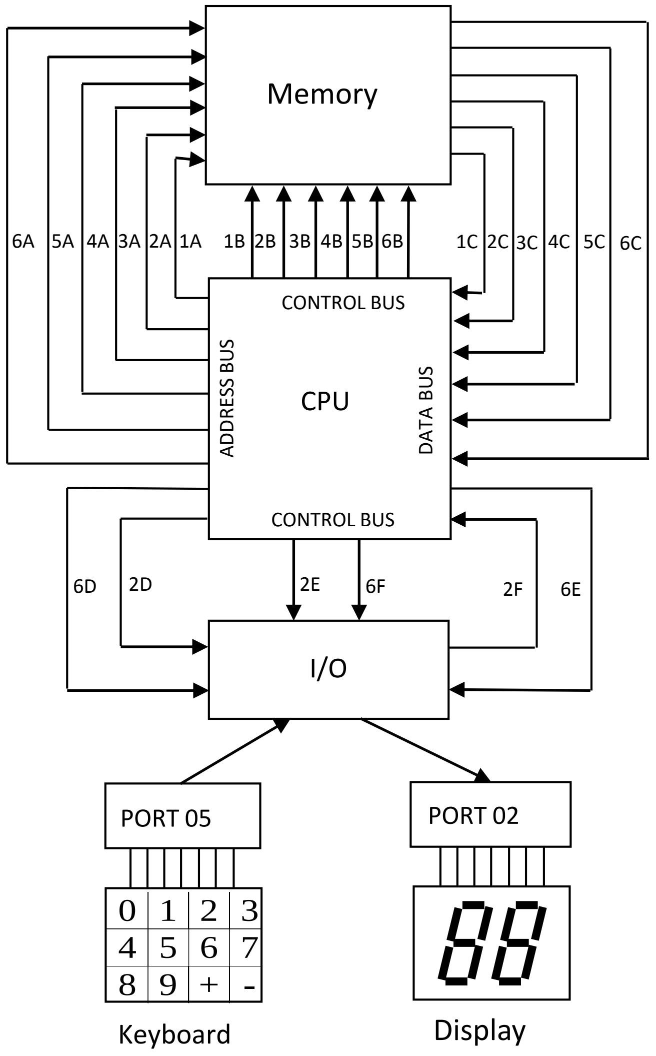
SimCity analogy of Memory Space

http://web-vassets.ea.com/Assets/Resources/Image/Screenshots/SimCity%20Hotels%20Block.png?cb=1412974396



Example of the Execution of a three-step Computer Program

香 港 中 文 大 學 (深 圳)
The Chinese University of Hong Kong, Shenzhen



(D. Hall - Figure 2-6)

PROGRAM

1. Input a value form port 05.
2. Add 7 to this value.
3. Output the result to port 02.

INPUT FROM PORT 05H
ADD 07H
OUTPUT TO PORT 02

SEQUENCE

1A *CPU* sends out *address of the first instruction* to *memory*.

1B CPU sends out *memory read* control signal to enable memory.

1C *Instruction byte* (E4h) sent from memory to CPU on data bus. INPUT FROM

2A Address next memory location to get the rest of instruction.

2B Send memory read control signal to enable memory.

2C Port address byte (05h) sent from memory to CPU on data bus. PORT 05H

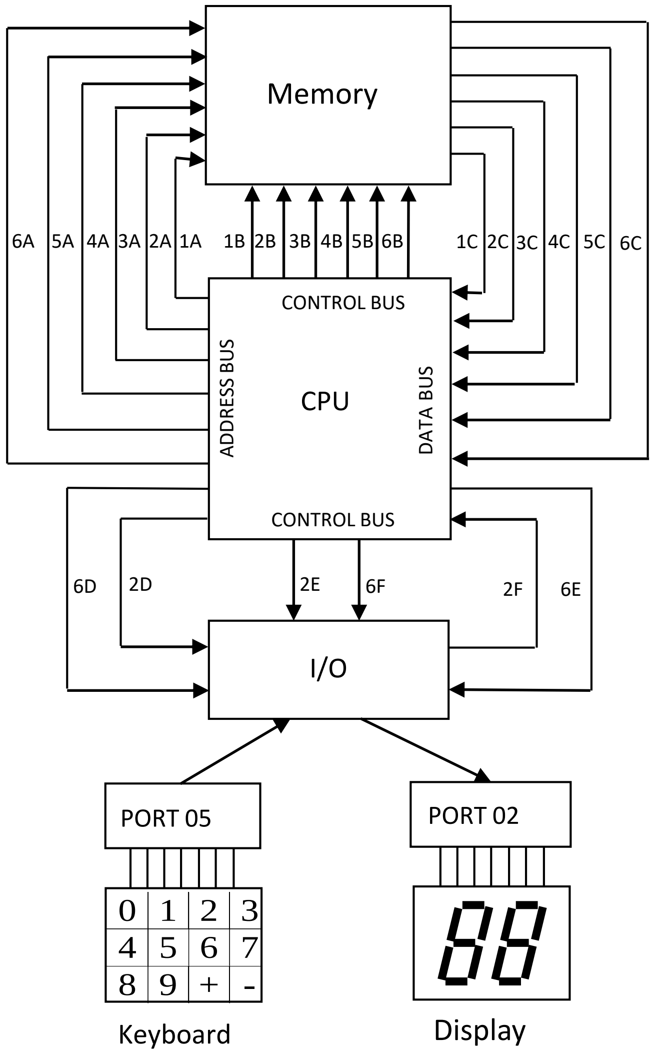
2D CPU sends out port address on address bus.

2E CPU sends out port read control signal to enable I/O port.

2F Data from port sent to CPU on data bus and store in an internal register.

(Complete the fetch and execution of 1st instruction)

Example of the Execution of a three-step Computer Program



PROGRAM

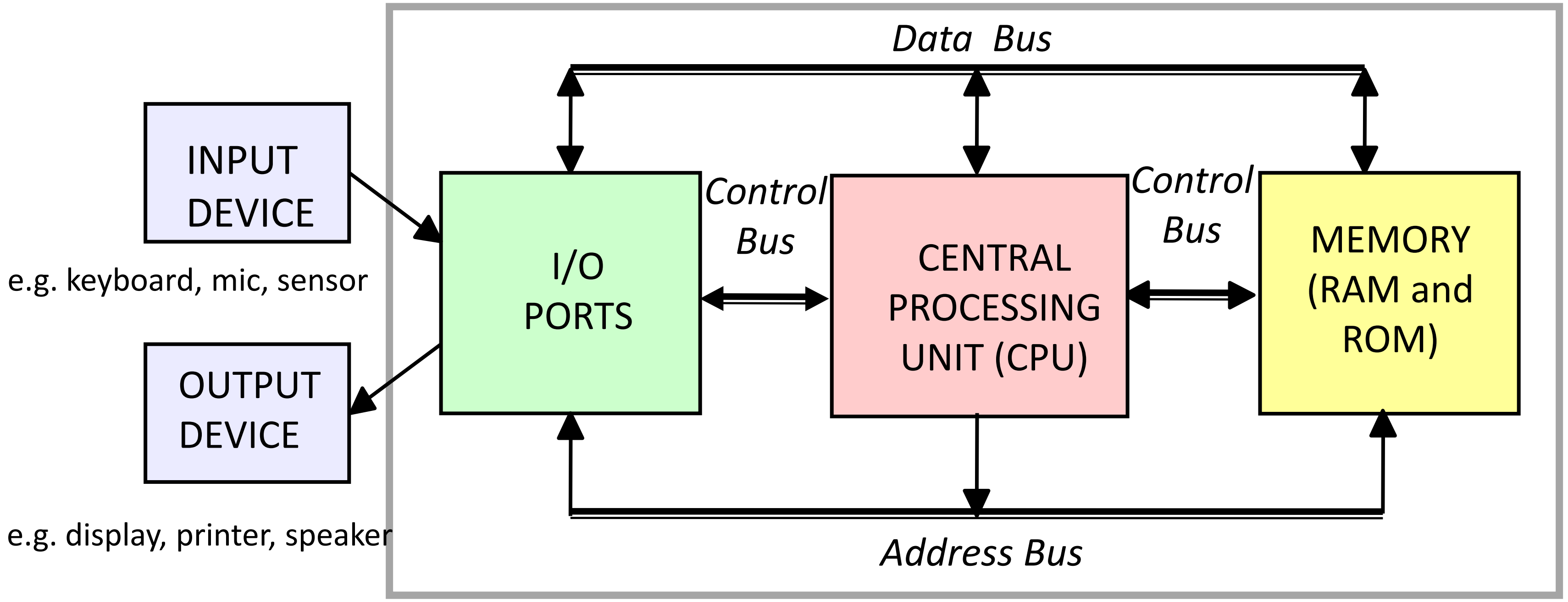
1. Input a value form port 05.
2. Add 7 to this value.
3. Output the result to port 02.

INPUT FROM PORT 05H
ADD 07H
OUTPUT TO PORT 02

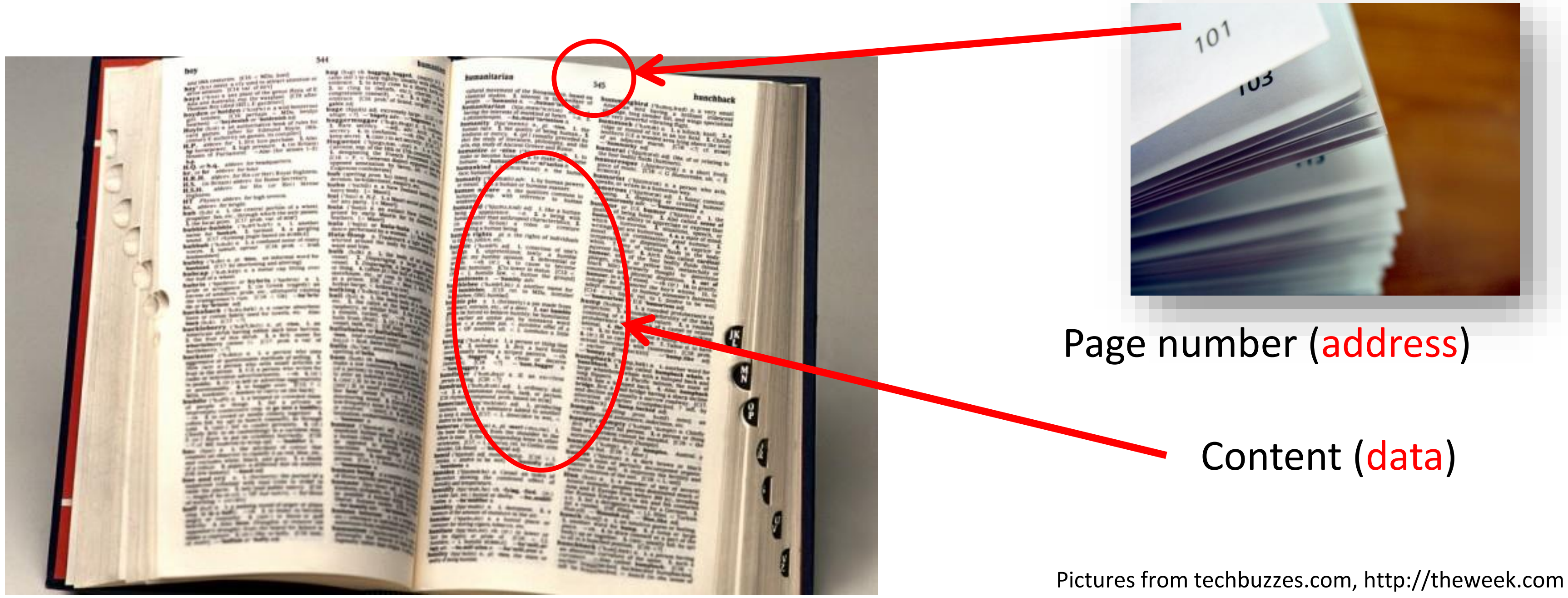
3A CPU sends address of next instruction to memory.	
3B CPU sends memory read control signal to enable memory.	
3C Instruction byte (04h) from memory sent to CPU on data bus.	ADD
4A CPU sends next address to memory to get rest of instruction.	
4B CPU sends memory read control signal to enable memory.	
4C Number 07h sent form memory to CPU on data bus.	07H
5A CPU sends address of next instruction to memory.	
5B CPU sends memory read control signal to enable memory.	
5C Instruction byte (E6h) from memory sent to CPU on data bus.	OUTPUT TO
6A CPU sends out next address to get rest of instruction.	
6B CPU sends out memory read control signal to enable memory.	
6C Port address byte (02h) sent from memory to CPU on data bus.	PORT 02
6D CPU sends out port address on address bus.	
6E CPU sends out data to port on data bus.	
6F CPU sends out output write signal to enable port.	

Total: 24 operations!

Memory


INPUT DEVICE
e.g. keyboard, mic, sensor
OUTPUT DEVICE
e.g. display, printer, speaker
Data Bus
I/O PORTS
Control Bus
CENTRAL PROCESSING UNIT (CPU)
Control Bus
MEMORY (RAM and ROM)
Address Bus

Address and Data of Memory

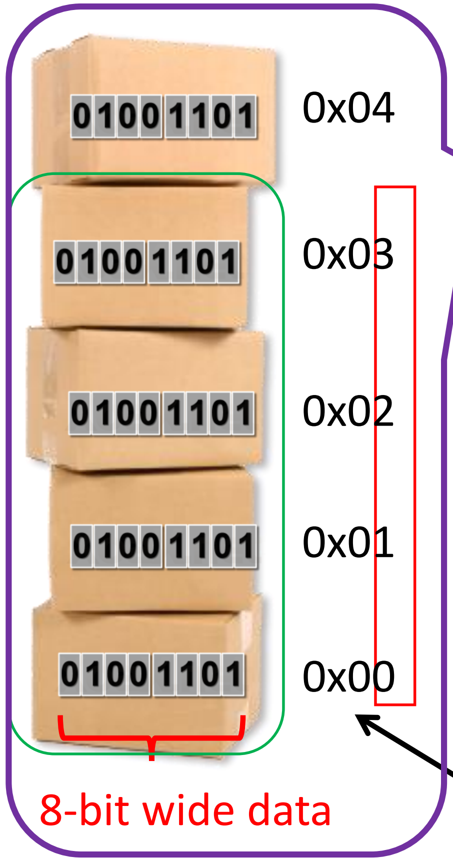


Pictures from techbuzzes.com, http://theweek.com

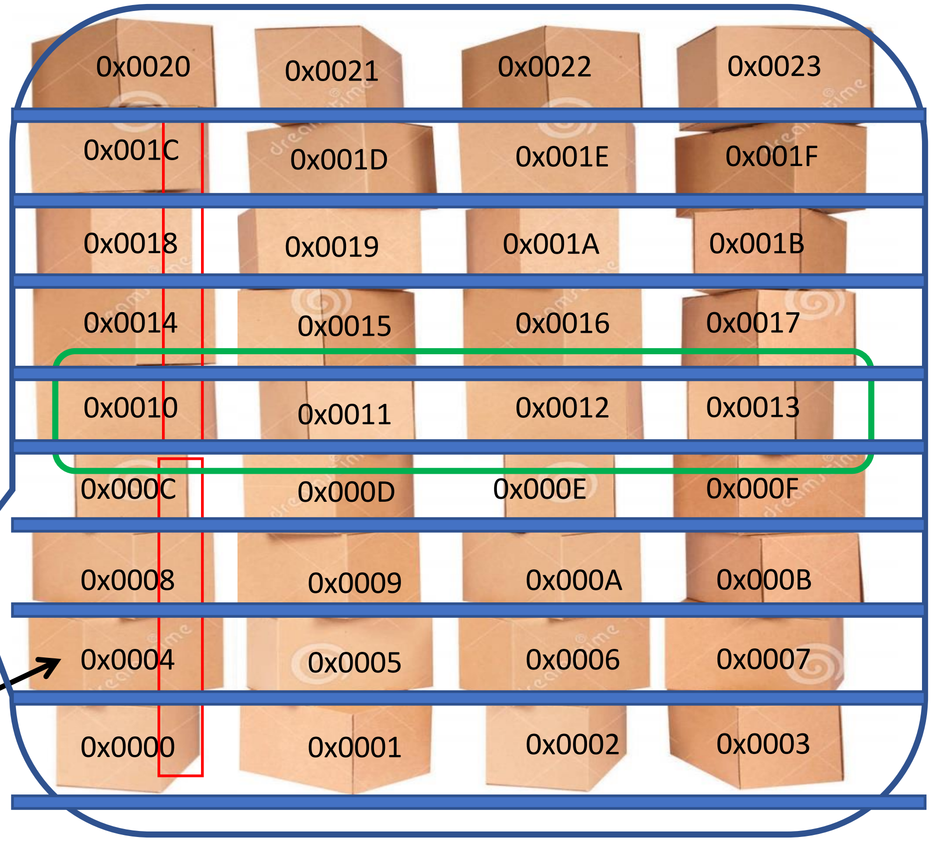
- Memory includes two parts: data and address.
- The computer memory space is like a book: *page number* is the address and the *content* on each page is the data. The computer stores fixed size data (e.g. 8-bit) in each memory unit.
- The role of address and data may not be that clear sometimes, e.g., when they are used in "pointer" or " indexing".

8-bit /16-bit and 32-bit system

- A memory unit stores one byte. It is not a problem for an 8-bit system. In 32-bit system, we always access 32 bits (4 bytes) at once.



- 8-bit systems: The address increment is +1 each.
- The last digit of the starting address of each 32-bit block are 0, 4, 8 and C.



address

Pictures from starrplanet.com, pixgood.com

For 32-bit systems, we put 4 boxes (4 units) in a row. In program, one address is referring to the 4 consecutive bytes.

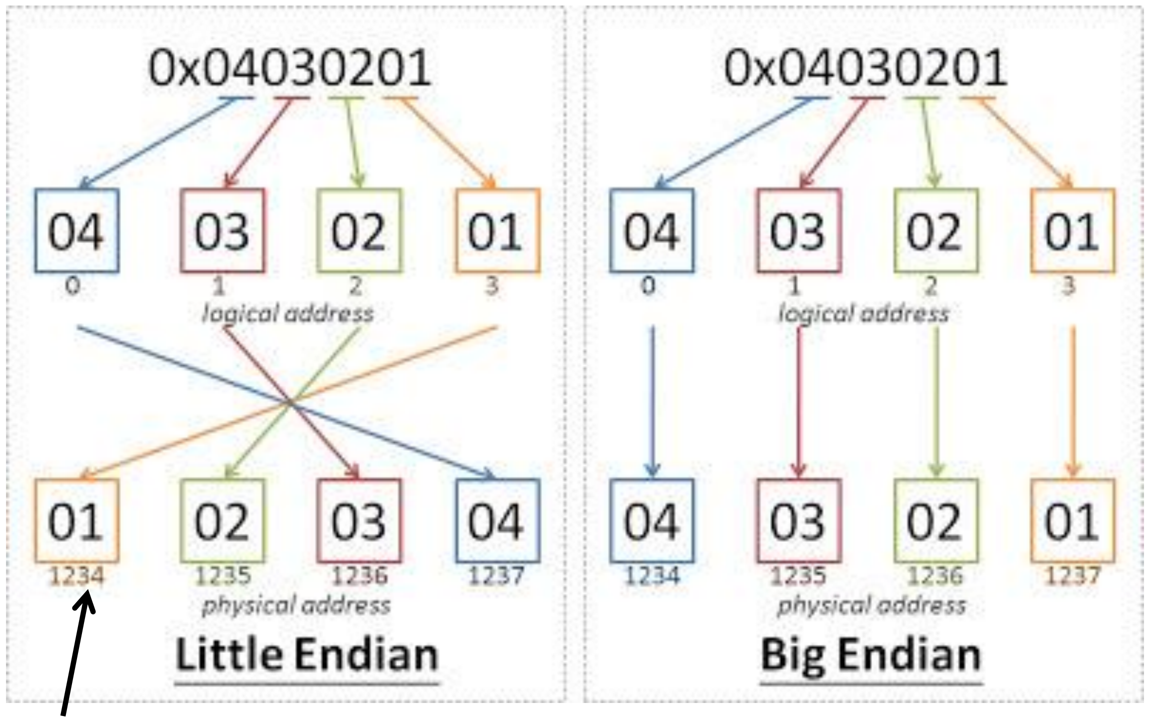
Big endian vs Little endian

- Big Endian: Most Significant Byte first. (MS byte at the lowest memory location)
- Little Endian: Least Significant Byte first. (LS byte at the lowest memory location)

Motorola 6800 & 68K, IBM Power, Atmel AVR32 architectures use big-endian.

Intel x86, x86-64, 6502, Z80, MCS-48, 8051, DEC Alpha, Atmel AVR, VAX architectures use little-endian.

Cortex-M3 supports both Big endian and little endian.



address

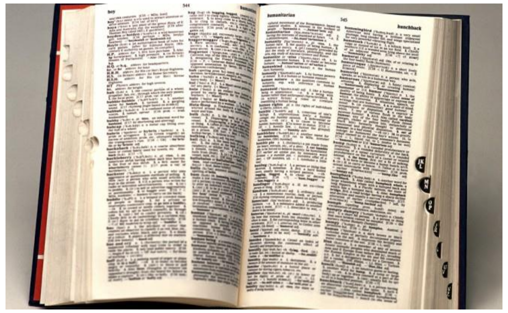
Pictures from starrplanet.com, http://albert-hu.blogspot.hk

Different kinds of memories

Computer memory stores data. There are different types of memories that have been designed for different purposes.

(1) Non-volatile memory (NVM). The data in the memory devices is kept even without power.

E.g. Mask Read-Only Memory (Mask ROM),
Programmable ROM (PROM),
Erasable PROM (EPROM),
Electrically Erasable PROM (EEPROM),
Flash memory,
Ferroelectric RAM (F-RAM),
Magnetic computer storage devices (e.g. hard disks, floppy disks, and magnetic tape), optical discs, and early methods for computer storage such as paper tape and punched cards.


Pictures from techbuzzes.com

- Q: How to erase/revise the memory content?

  Most of them can be erased/revised with some special methods. Some of them (Mask ROM, punched cards, paper tape) cannot be erased, like the printed book that its content cannot be erased.

Different kinds of memories

(2) Volatile memory (VM): The data in this memory devices can only be kept during power on. When the power is interrupted the stored data is lost immediately .

E.g. Dynamic RAM (DRAM),
Static Random-Access Memory (SRAM),
Pseudo SRAM (PSRAM),

- **The advantages of VM: The design is simple, and the cost is low.**
- Erasing/revising the contents of specify location is very easy and simple. So we call them Random Access Memory (RAM). They are like a note book; you can erase/rewrite the contents with an eraser and a pencil easily.

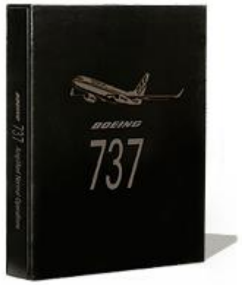
Pictures from dreamstime.com

Type of memory

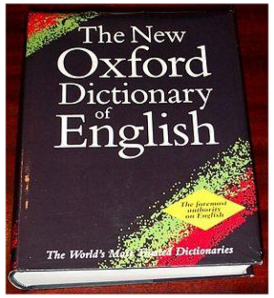
Base on the needs of the memory, there are a few types of memories that can be used for different purposes.

Volatile memory: store temporary data
Storage: DRAM, SRAM, PSRAM.

Non-volatile memory:
(1) store a series of instructions (program)



(2) store a group of predefined data (constant data)
e.g. character sets, tables for looking up, pictures, etc.

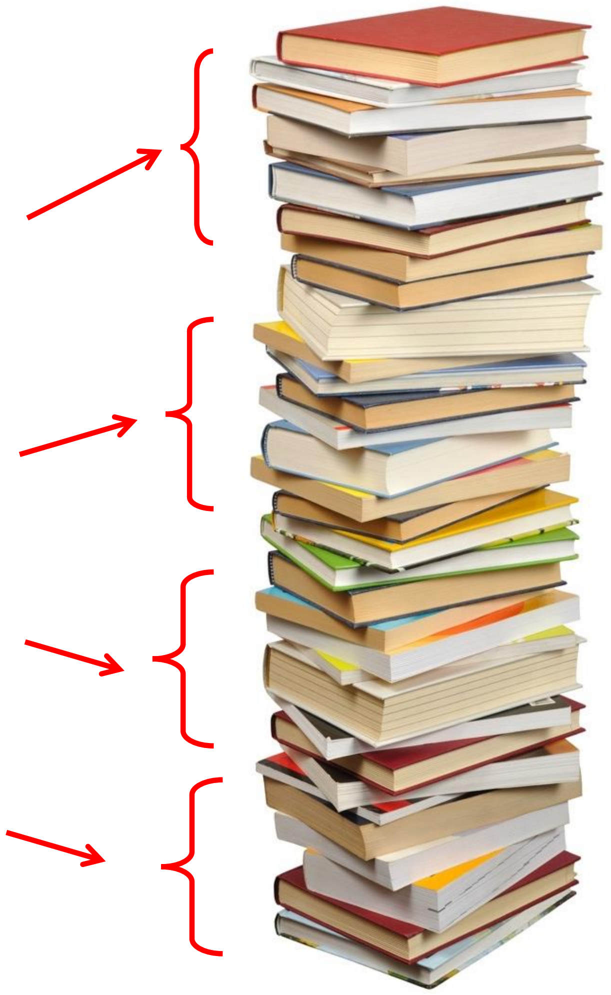

The New Oxford Dictionary of English
The World's Most Trusted Dictionaries

Storage: ROM, PROM

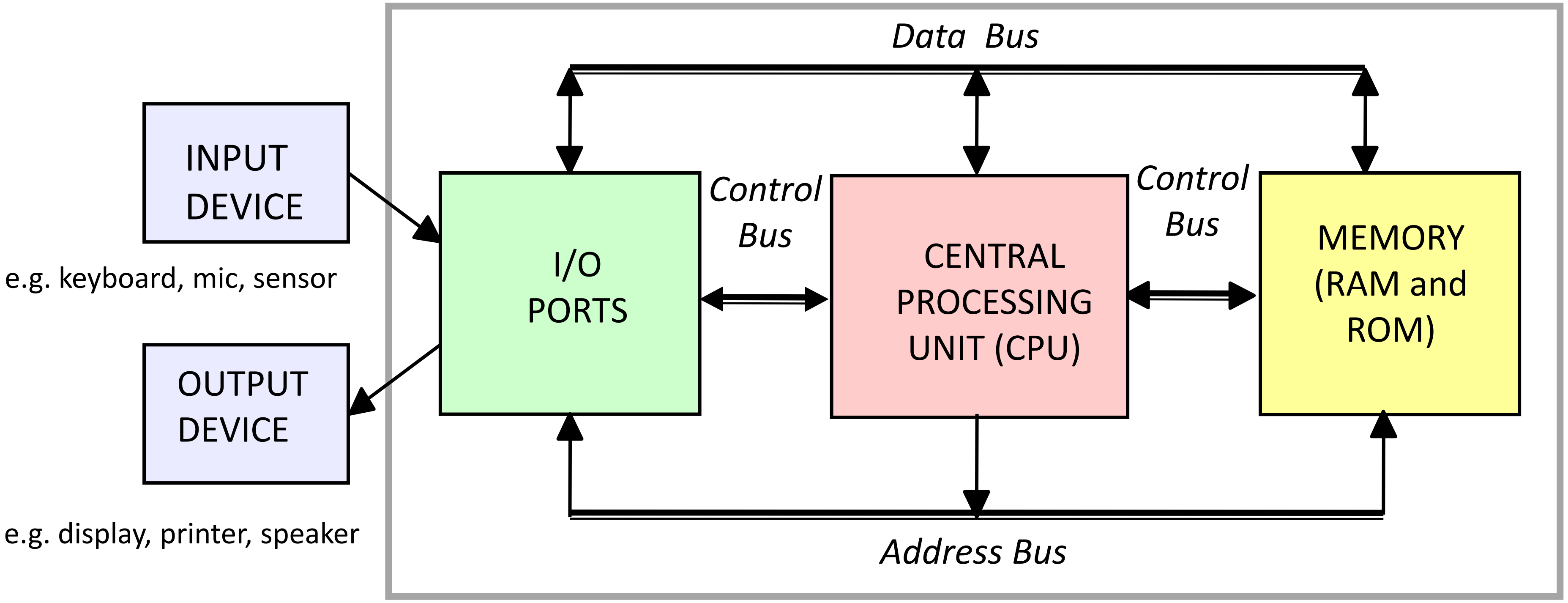
Pictures from http://www.dreamstime.com, http://www.arianedesign.com, http://en.wikipedia.org

Memory (address) space

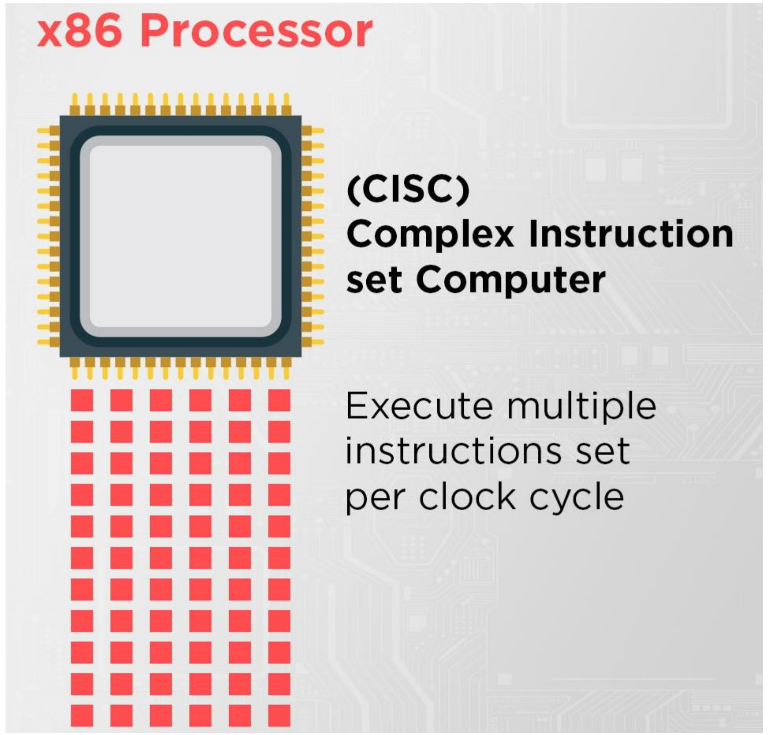
- The memory space (the address) of the embedded system is like a stack of books with different purposes.
- Some books may be like notebooks for storing temporary data.
- Some books may be like operation manuals, which contain instructions, programs.
- And some books may be like dictionaries, which contain some constant data.
- Some memory space can also be allocated to outputs/inputs (e.g. display board, motor controlling, sensors) and their control signals. peripherals
- These memories will occupy some memory address space.

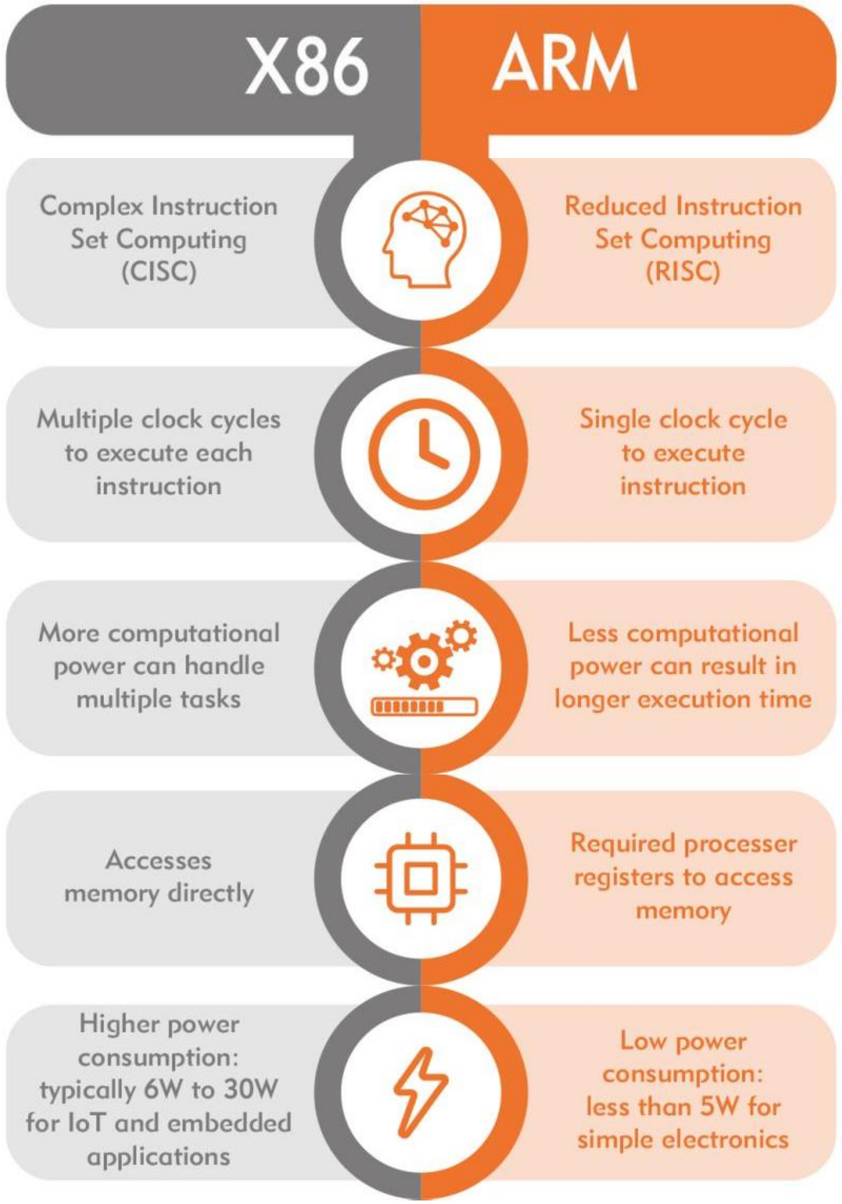

Pictures from http://coachingforleaders.com

CPU


INPUT DEVICE
e.g. keyboard, mic, sensor
OUTPUT DEVICE
e.g. display, printer, speaker
Data Bus
I/O PORTS
Control Bus
CENTRAL PROCESSING UNIT (CPU)
Control Bus
MEMORY (RAM and ROM)
Address Bus

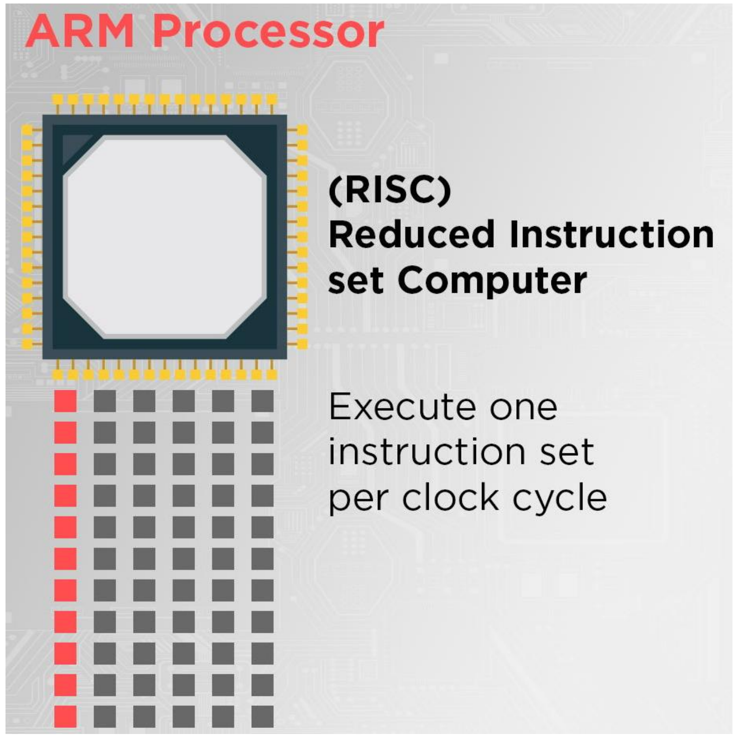
CPU - x86 vs ARM Processors


x86 Processor
(CISC)
Complex Instruction set Computer
Execute multiple instructions set per clock cycle

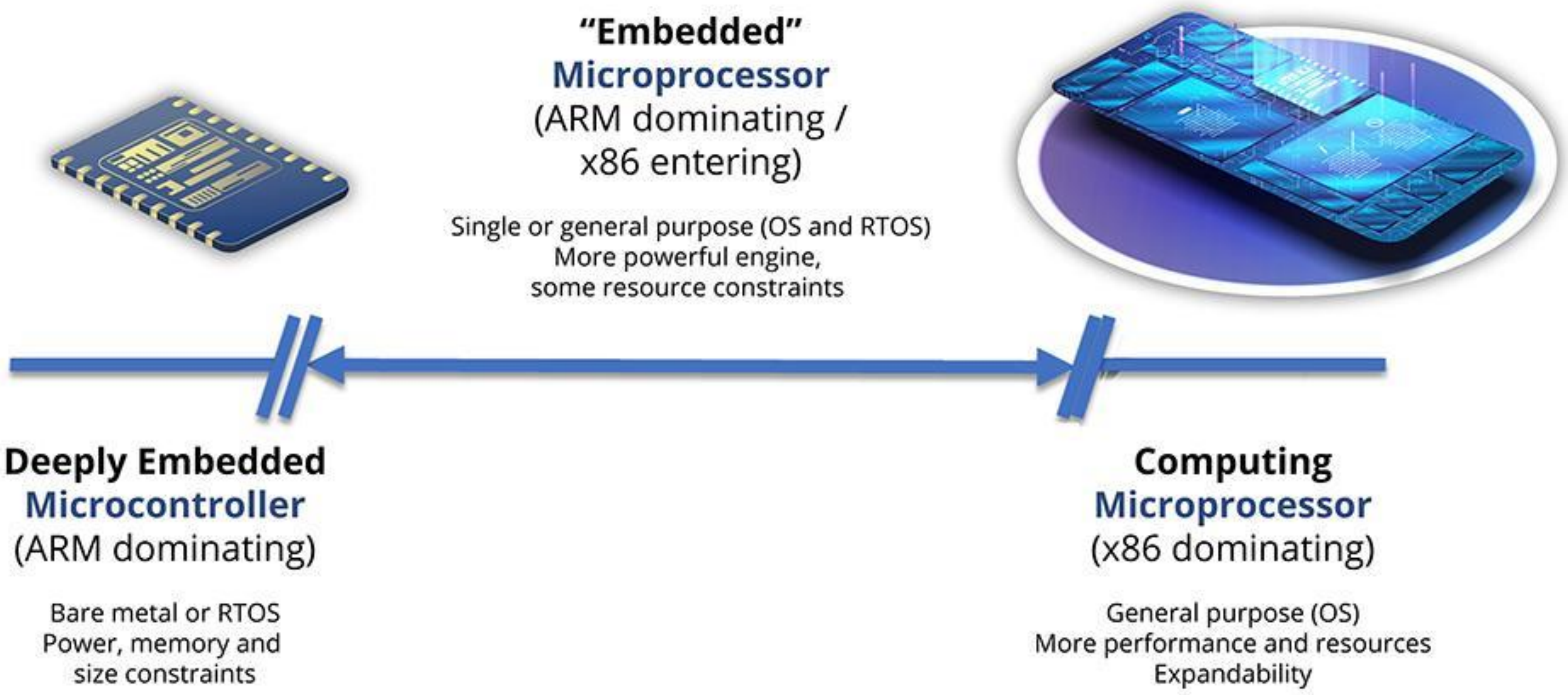


X86	ARM
Complex Instruction Set Computing (CISC)	Reduced Instruction Set Computing (RISC)
Multiple clock cycles to execute each instruction	Single clock cycle to execute instruction
More computational power can handle multiple tasks	Less computational power can result in longer execution time
Accesses memory directly	Required processer registers to access memory
Higher power consumption: typically 6W to 30W for IoT and embedded applications	Low power consumption: less than 5W for simple electronics

Low Cost, Small Size,
Low power consumption


ARM Processor
(RISC)
Reduced Instruction set Computer
Execute one instruction set per clock cycle

CPU - x86 vs ARM Processors



"Embedded"
Microprocessor
(ARM dominating /
x86 entering)

Single or general purpose (OS and RTOS)
More powerful engine,
some resource constraints

Deeply Embedded
Microcontroller
(ARM dominating)

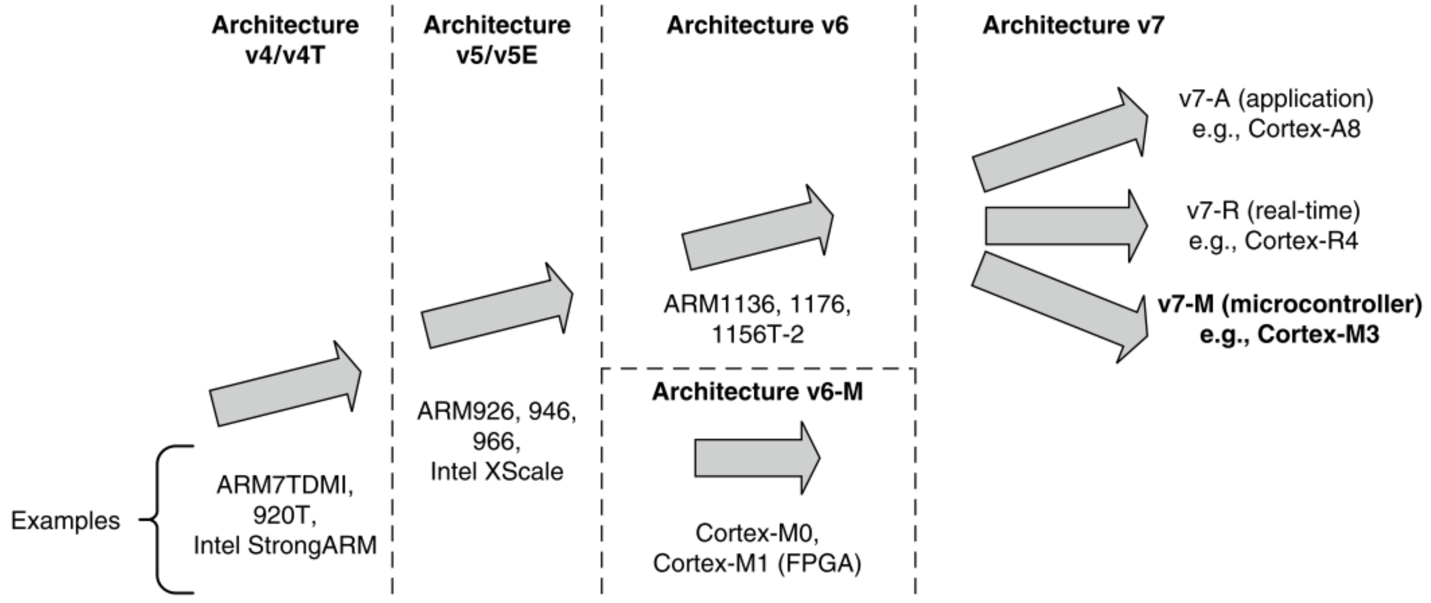
Bare metal or RTOS
Power, memory and
size constraints

Computing
Microprocessor
(x86 dominating)

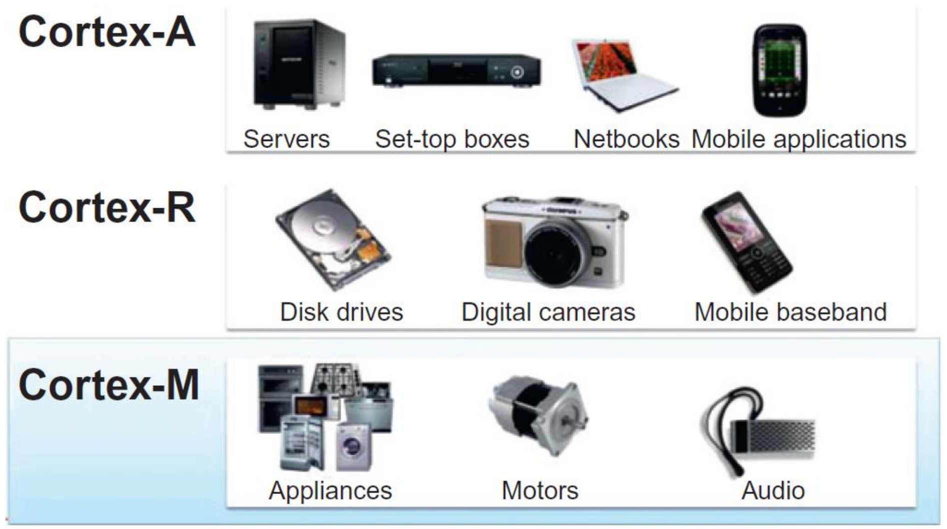
General purpose (OS)
More performance and resources
Expandability

History of ARM

- ARM was formed in 1990 as Advanced RISC Machines Ltd., a joint venture of Apple Computer, Acorn Computer Group, and VLSI Technology.
- ARM does not manufacture processors or sell the chips directly. Instead, ARM licenses the processor designs to business partners, such as VLSI, TI, NEC, Sharp, and ST Microelectronics.


Architecture v4/v4T
Architecture v5/v5E
Architecture v6
Architecture v7
v7-A (application) e.g., Cortex-A8
v7-R (real-time) e.g., Cortex-R4
v7-M (microcontroller) e.g., Cortex-M3
ARM1136, 1176, 1156T-2
Architecture v6-M
Cortex-M0, Cortex-M1 (FPGA)
ARM926, 946, 966, Intel XScale
Examples
ARM7TDMI, 920T, Intel StrongARM

The Evolution of ARM Processor Architecture



Features of Cortex-M3

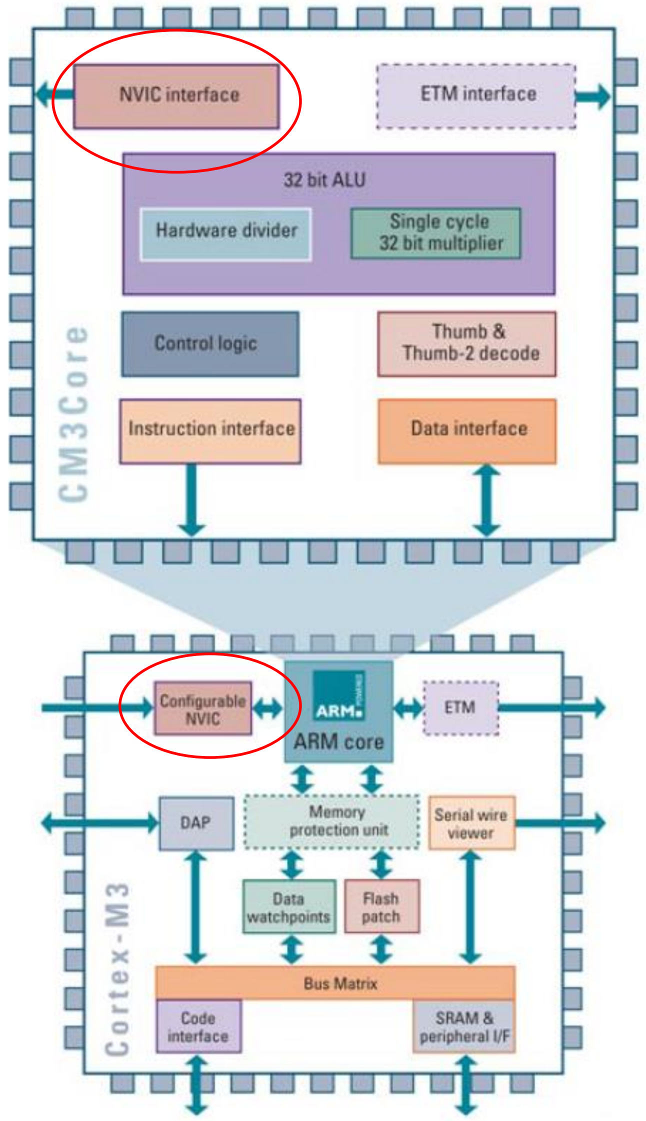
1. **Greater performance efficiency:** performing more tasks without increasing the clock frequency or power requirements.
2. **Low power consumption:** enabling longer battery life, especially critical in portable products. (point 1 & 2 make Cortex-M3 a good candidate for communications applications such as Bluetooth and ZigBee.)
3. **Enhanced determinism:** assuring critical tasks and interrupts are serviced quickly with low latency and in a known number of cycles (good for industrial control applications).
4. **Improved code density:** ensuring that codes use the smallest memory footprints.
5. **Ease of use:** providing easier programmability and debugging for 8-/16-bit users migrating to 32-bit systems.
6. **Lower cost solutions:** reducing 32-bit-based system costs close to those of legacy 8-/16-bit devices; less than US$1 for low-end, 32-bit microcontrollers.
7. **Wide choice of development tools:** from low-cost or free compilers to full-featured development suites from many development tool vendors.

Cortex-M3 Processor Applications

- ***Low-cost microcontrollers***: Commonly used in consumer products, from toys to electrical appliances. It is a highly competitive market due to the many well-known 8-bit and 16-bit microcontroller products on the market. Its lower power, high performance, and ease-of-use advantages enable embedded developers to migrate to 32-bit systems and develop products with the ARM architecture.
- ***Automotive***: The Cortex-M3 processor has very high-performance efficiency and low interrupt latency, allowing it to be used in real-time systems. The Cortex-M3 processor supports up to 240 external vectored interrupts, with a built-in interrupt controller with nested interrupt supports and an optional MPU, making it ideal for highly integrated and cost-sensitive automotive applications.
- ***Data communications***: The processor's low power and high efficiency, coupled with instructions in Thumb-2 for bit-field manipulation, make the Cortex-M3 ideal for many communications applications, such as Bluetooth and ZigBee.
- ***Industrial control***: In industrial control applications, simplicity, fast response, and reliability are key factors. Again, the Cortex-M3 processor's interrupt feature, low interrupt latency, and enhanced fault-handling features make it a strong candidate in this area.
- ***Consumer products***: In many consumer products, a high-performance microprocessor (or several of them) is used. The Cortex-M3 processor, being a small processor, is highly efficient and low in power and supports an MPU enabling complex software to execute while providing robust memory protection.

ARM Cortex-M3

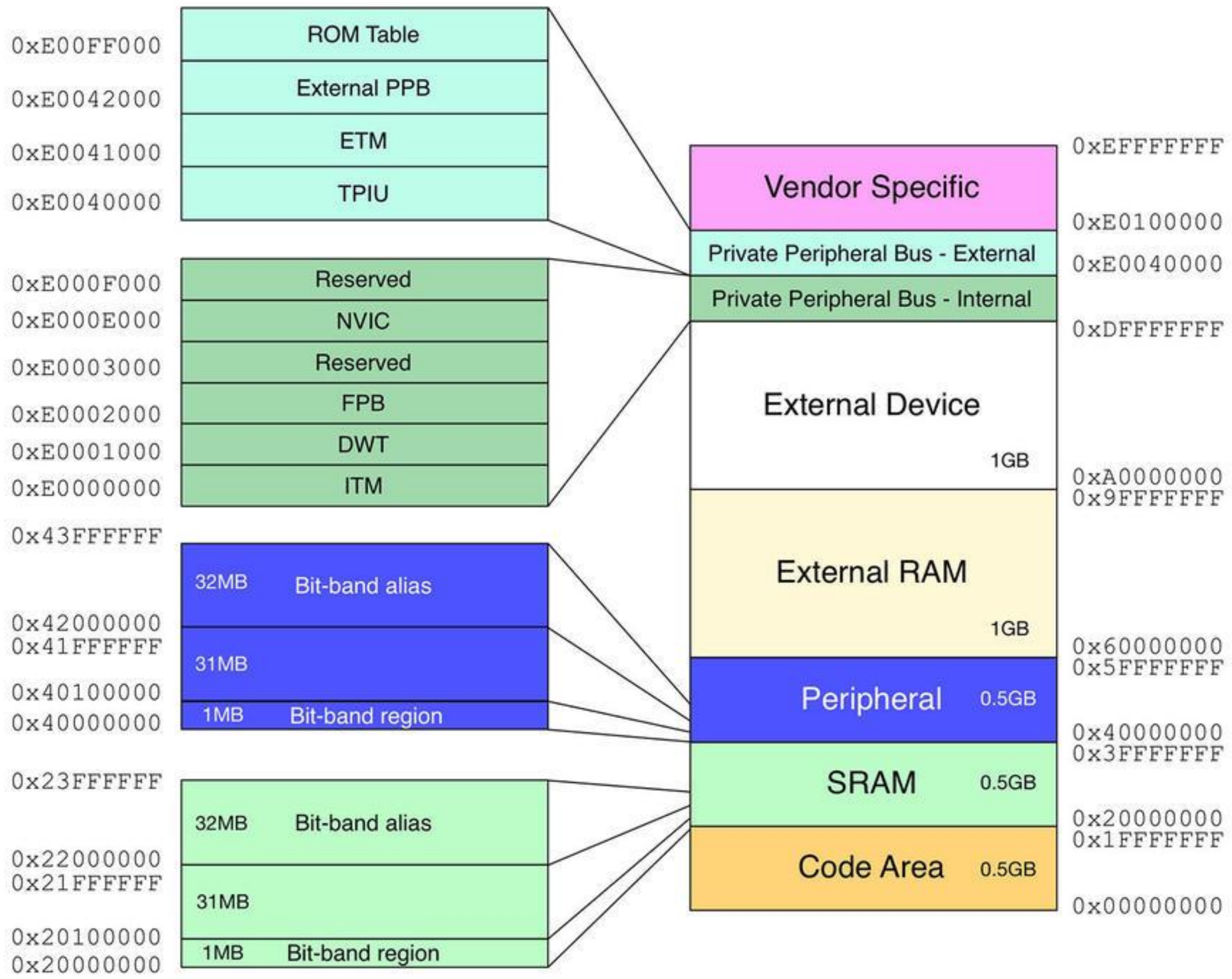
- The Cortex-M3 provides a standardized microcontroller core that goes beyond the CPU (32bit) to provide the entire heart of a microcontroller, including
  - Nested Vectored Interrupt Controller (NVIC)
  - System timer (SysTick)
  - Fixed memory map. *(Important features to be discussed later)*
  - Debug system



Ref: from Hitex

ARM Cortex-M3

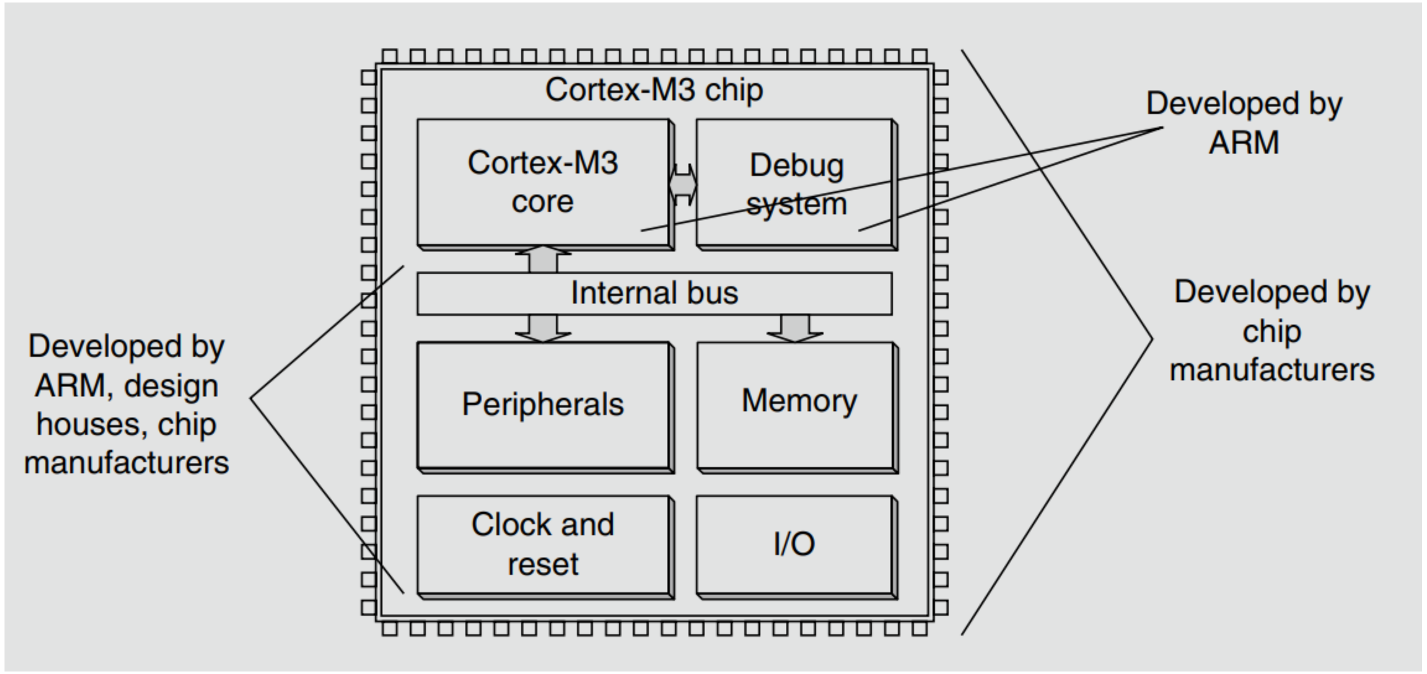
- Cortex-M3 has 4-Gbyte address space, which is split into well defined regions for Code, SRAM, peripherals and system peripherals.
- Cortex-M3 has a (modified) Harvard bus architecture -> it has multiple busses to allow performing operations in parallel (ARM7 uses Von Neumann architecture instead) .
- Unlike earlier ARM architectures, the Cortex family allows unaligned data accesses, providing most efficient use of the internal SRAM.
- The Cortex family also supports setting and clearing of bits within two 1-Mbyte regions of memory by a method called *bit banding* -> efficient access to peripheral registers and flags in memory without the need for a full Boolean processor to process one whole byte/word.



Fixed memory map for ARM Cortex-M cores

Cortex-M3-Based MCUs

- The Cortex-M3 processor is the central processing unit (CPU) of a microcontroller chip. In addition, a number of other components are required for the whole Cortex-M3 processor-based microcontroller. After chip manufacturers license the Cortex-M3 processor, they can put the Cortex-M3 processor in their silicon designs, adding memory, peripherals, input/output (I/O), and other features. Cortex-M3 processor-based chips from different manufacturers will have different memory sizes, types, peripherals, and features.



In the lab, we use the chip STM32 MCU provided by STMicroelectronics. (www.st.com/stm32)

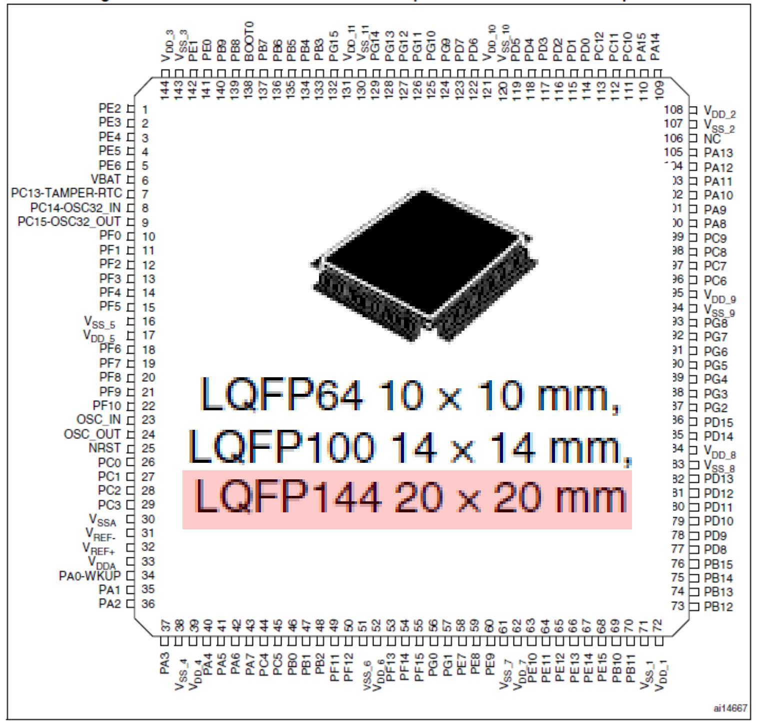
Ref: from Text book

Chip used in EIE3810 – STM32

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

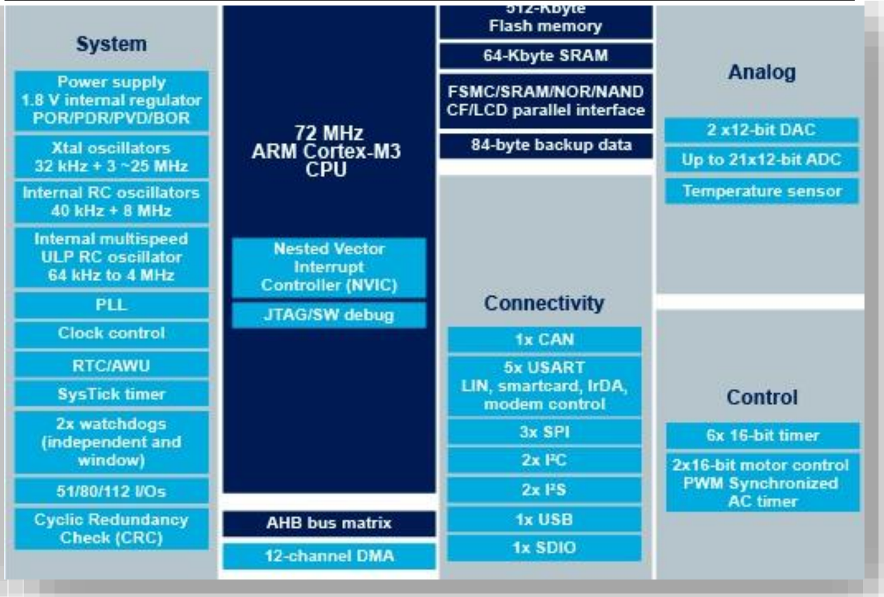
- The microcontroller that we use in our lab is STM32F103ZET6

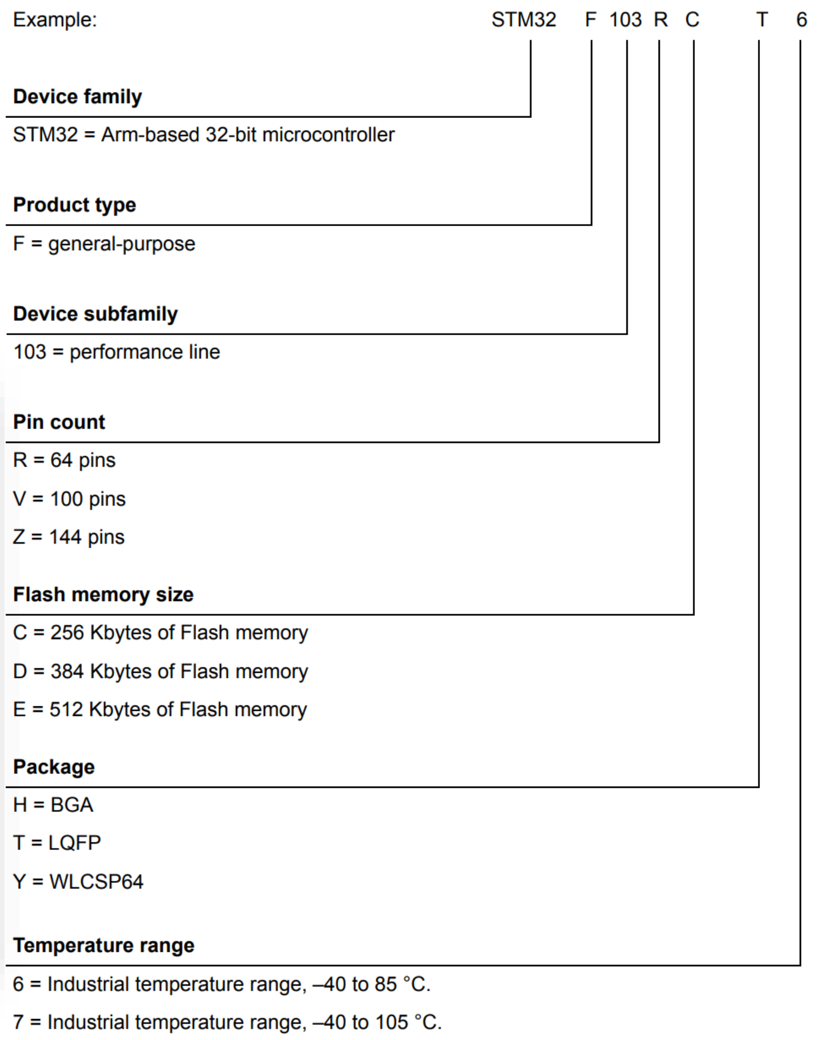
Figure 5.STM32F103xC and STM32F103xE performance line LQFP144 pinout



STM32F103xC, STM32F103xD, STM32F103xE

High-density performance line ARM®-based 32-bit MCU with 256 to 512KB Flash, USB, CAN, 11 timers, 3 ADCs, 13 communication interfaces





Example: STM32 F 103 R C T 6

Device family
STM32 = Arm-based 32-bit microcontroller

Product type
F = general-purpose

Device subfamily
103 = performance line

Pin count
R = 64 pins
V = 100 pins
Z = 144 pins

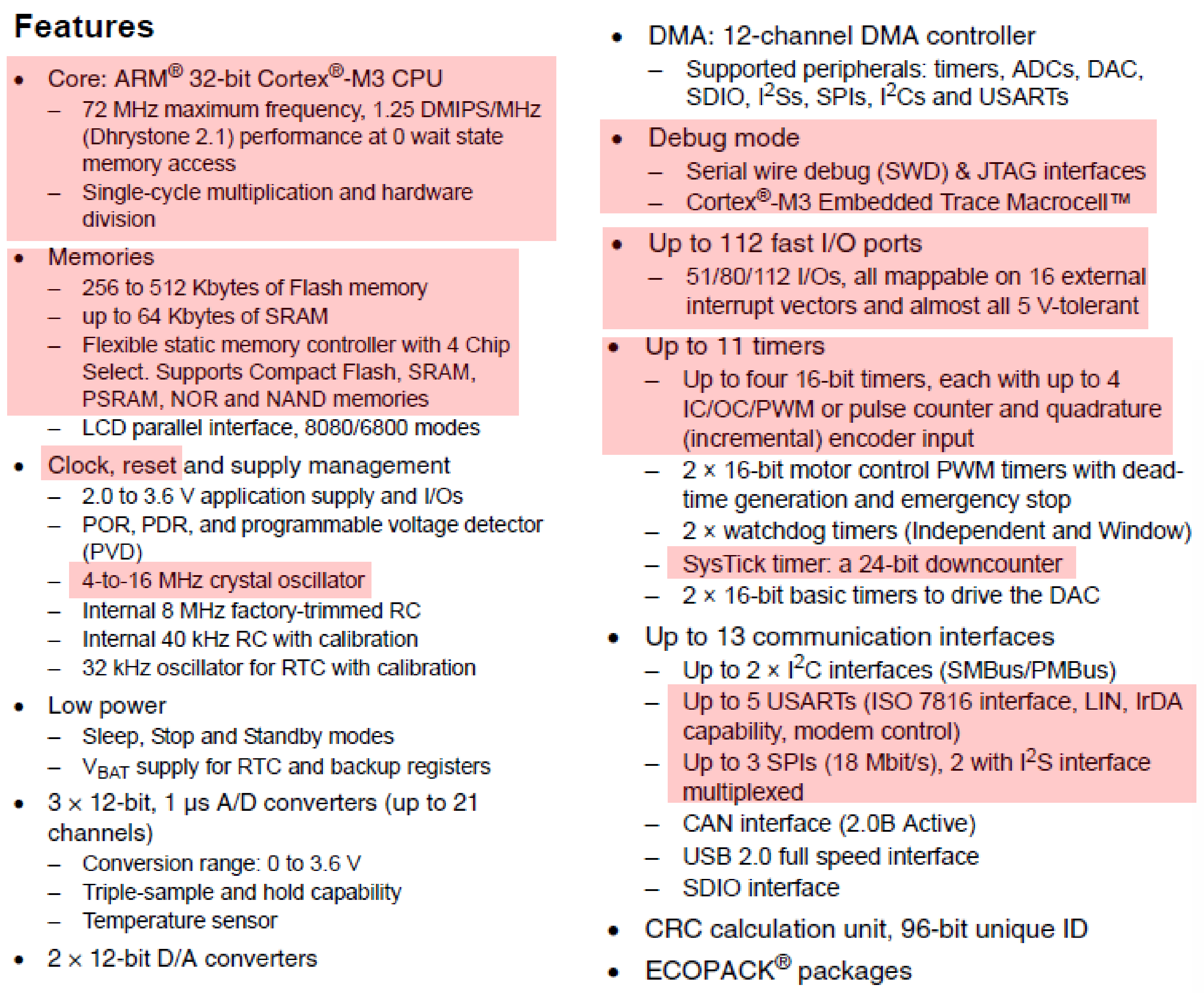
Flash memory size
C = 256 Kbytes of Flash memory
D = 384 Kbytes of Flash memory
E = 512 Kbytes of Flash memory

Package
H = BGA
T = LQFP
Y = WLCSP64

Temperature range
6 = Industrial temperature range, –40 to 85 °C.
7 = Industrial temperature range, –40 to 105 °C.

Pictures from www.st.com

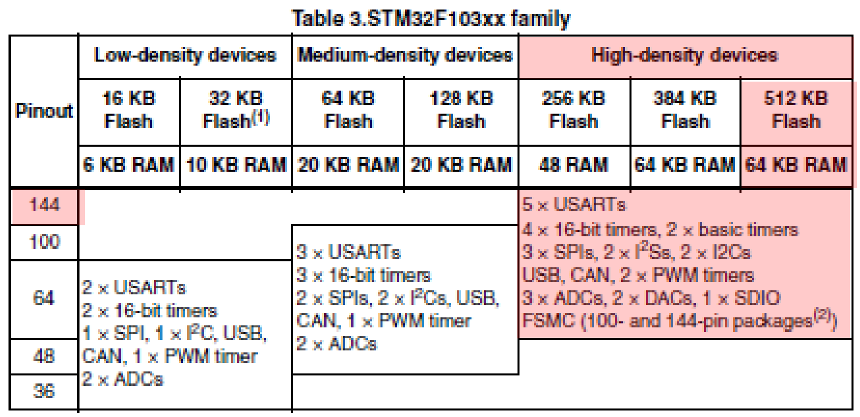
STM32 Basic Features



Features

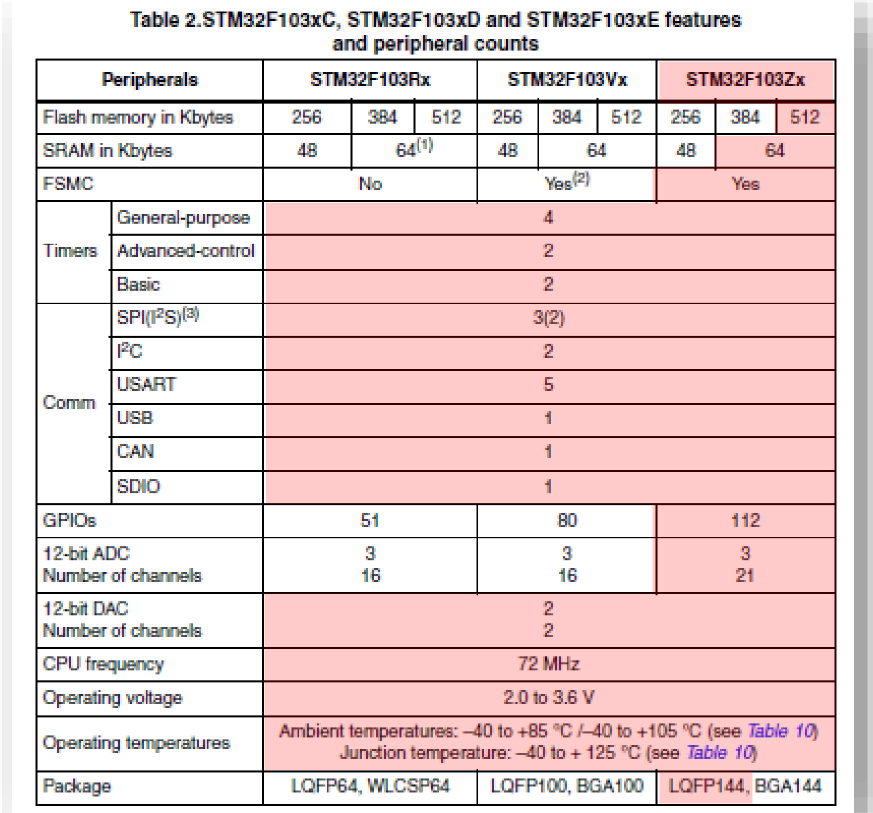
- Core: ARM® 32-bit Cortex®-M3 CPU
  - 72 MHz maximum frequency, 1.25 DMIPS/MHz (Dhrystone 2.1) performance at 0 wait state memory access
  - Single-cycle multiplication and hardware division
- Memories
  - 256 to 512 Kbytes of Flash memory
  - up to 64 Kbytes of SRAM
  - Flexible static memory controller with 4 Chip Select. Supports Compact Flash, SRAM, PSRAM, NOR and NAND memories
  - LCD parallel interface, 8080/6800 modes
- Clock, reset and supply management
  - 2.0 to 3.6 V application supply and I/Os
  - POR, PDR, and programmable voltage detector (PVD)
  - 4-to-16 MHz crystal oscillator
  - Internal 8 MHz factory-trimmed RC
  - Internal 40 kHz RC with calibration
  - 32 kHz oscillator for RTC with calibration
- Low power
  - Sleep, Stop and Standby modes
  - V_{BAT} supply for RTC and backup registers
- 3 × 12-bit, 1 µs A/D converters (up to 21 channels)
  - Conversion range: 0 to 3.6 V
  - Triple-sample and hold capability
  - Temperature sensor
- 2 × 12-bit D/A converters
- DMA: 12-channel DMA controller
  - Supported peripherals: timers, ADCs, DAC, SDIO, I^2Ss, SPIs, I^2Cs and USARTs
- Debug mode
  - Serial wire debug (SWD) & JTAG interfaces
  - Cortex®-M3 Embedded Trace Macrocell™
- Up to 112 fast I/O ports
  - 51/80/112 I/Os, all mappable on 16 external interrupt vectors and almost all 5 V-tolerant
- Up to 11 timers
  - Up to four 16-bit timers, each with up to 4 IC/OC/PWM or pulse counter and quadrature (incremental) encoder input
  - 2 × 16-bit motor control PWM timers with dead-time generation and emergency stop
  - 2 × watchdog timers (Independent and Window)
  - SysTick timer: a 24-bit downcounter
  - 2 × 16-bit basic timers to drive the DAC
- Up to 13 communication interfaces
  - Up to 2 × I^2C interfaces (SMBus/PMBus)
  - Up to 5 USARTs (ISO 7816 interface, LIN, IrDA capability, modem control)
  - Up to 3 SPIs (18 Mbit/s), 2 with I^2S interface multiplexed
  - CAN interface (2.0B Active)
  - USB 2.0 full speed interface
  - SDIO interface
- CRC calculation unit, 96-bit unique ID
- ECOPACK® packages

Table 3.STM32F103xx family



Pinout	Low-density devices		Medium-density devices		High-density devices		
	16 KB Flash	32 KB Flash[1]	64 KB Flash	128 KB Flash	256 KB Flash	384 KB Flash	512 KB Flash
	6 KB RAM	10 KB RAM	20 KB RAM	20 KB RAM	48 RAM	64 KB RAM	64 KB RAM
144					5 × USARTs 4 × 16-bit timers, 2 × basic timers 3 × SPIs, 2 × I^2Ss, 2 × I2Cs USB, CAN, 2 × PWM timers 3 × ADCs, 2 × DACs, 1 × SDIO FSMC (100- and 144-pin packages[2])		
100			3 × USARTs 3 × 16-bit timers 2 × SPIs, 2 × I^2Cs, USB, CAN, 1 × PWM timer 2 × ADCs				
64	2 × USARTs 2 × 16-bit timers 1 × SPI, 1 × I^2C, USB, CAN, 1 × PWM timer 2 × ADCs						
48							
36							

Table 2.STM32F103xC, STM32F103xD and STM32F103xE features and peripheral counts

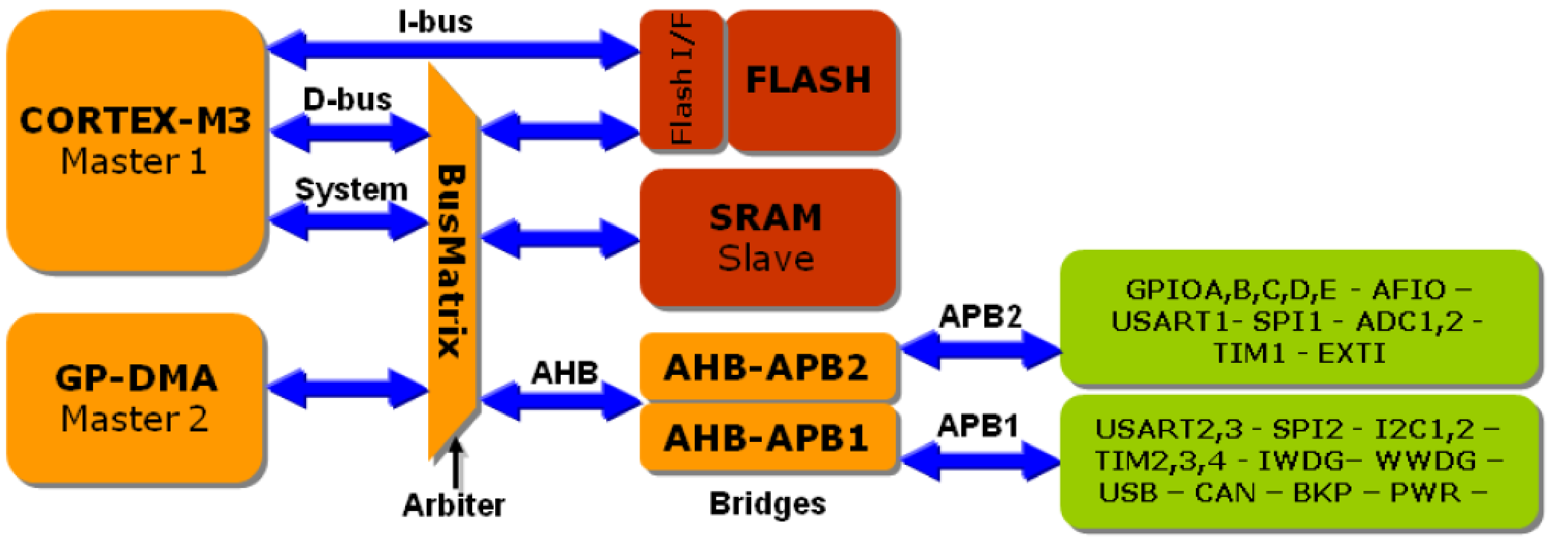


Peripherals		STM32F103Rx			STM32F103Vx			STM32F103Zx		
Flash memory in Kbytes		256	384	512	256	384	512	256	384	512
SRAM in Kbytes		48	64[1]		48	64		48	64	
FSMC		No			Yes[2]			Yes		
Timers	General-purpose	4								
	Advanced-control	2								
	Basic	2								
Comm	SPI(I^2S)[3]	3(2)								
	I^2C	2								
	USART	5								
	USB	1								
	CAN	1								
	SDIO	1								
GPIOs		51			80			112		
12-bit ADC Number of channels		3 16			3 16			3 21		
12-bit DAC Number of channels		2 2								
CPU frequency		72 MHz								
Operating voltage		2.0 to 3.6 V								
Operating temperatures		Ambient temperatures: –40 to +85 °C /–40 to +105 °C (see *Table 10*) Junction temperature: –40 to + 125 °C (see *Table 10*)								
Package		LQFP64, WLCSP64			LQFP100, BGA100			LQFP144, BGA144		

(Pictures from www.st.com)

STM32 System Architecture

- The STM32 has a 32-bit Cortex core that is connected to the FLASH memory with a dedicated Instruction bus (I-bus).
- The Cortex Data buses (D-bus) and System buses are connected to a matrix of ARM Advanced High-performance Bus (AHB).
- The internal SRAM is connected directly to the AHB bus matrix, same as the Direct Memory Access (DMA) unit.
- The Peripherals are connected to system by two ARM Advanced Peripheral Buses (APB). Each of the APB buses is bridged onto the AHB bus matrix.



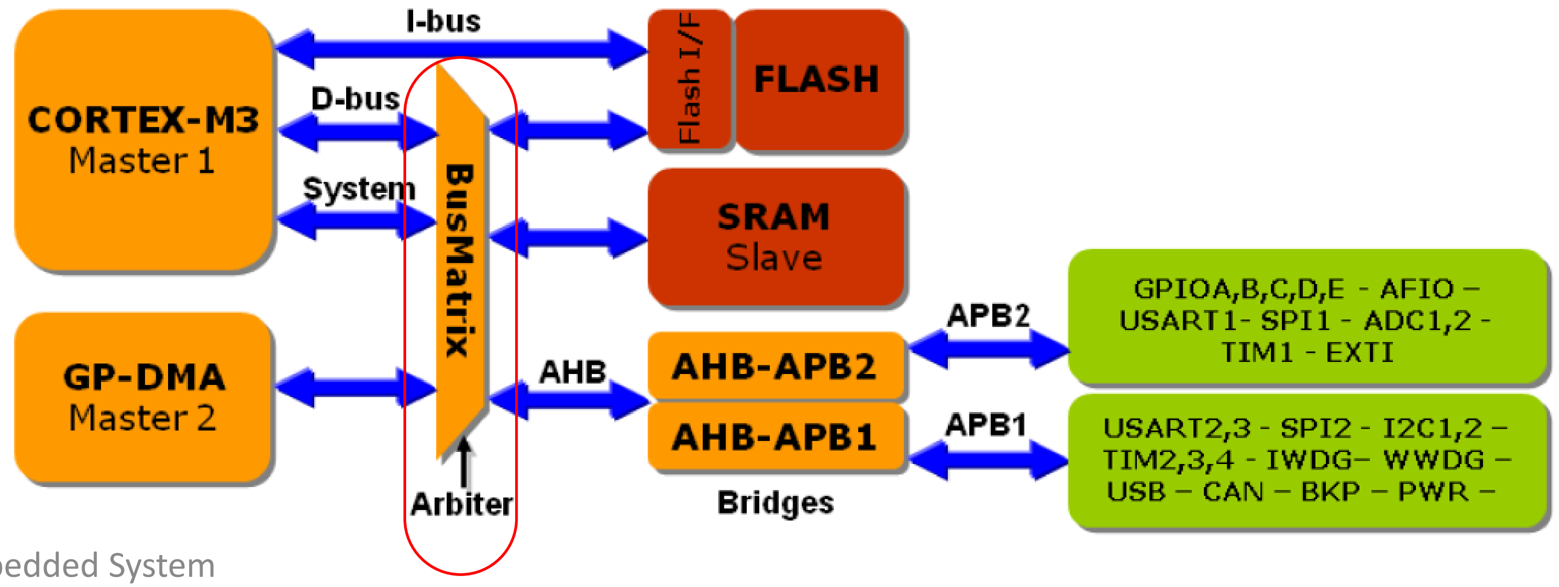
I-bus
CORTEX-M3 Master 1
D-bus
System
BusMatrix
Flash I/F
FLASH
SRAM Slave
GP-DMA Master 2
AHB
Arbiter
AHB-APB2
AHB-APB1
Bridges
APB2
APB1
GPIOA,B,C,D,E - AFIO – USART1 - SPI1 - ADC1,2 - TIM1 - EXTI
USART2,3 - SPI2 - I2C1,2 – TIM2,3,4 - IWDG – WWDG – USB – CAN – BKP – PWR –

Ref: from Hitex

STM32 System Architecture

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

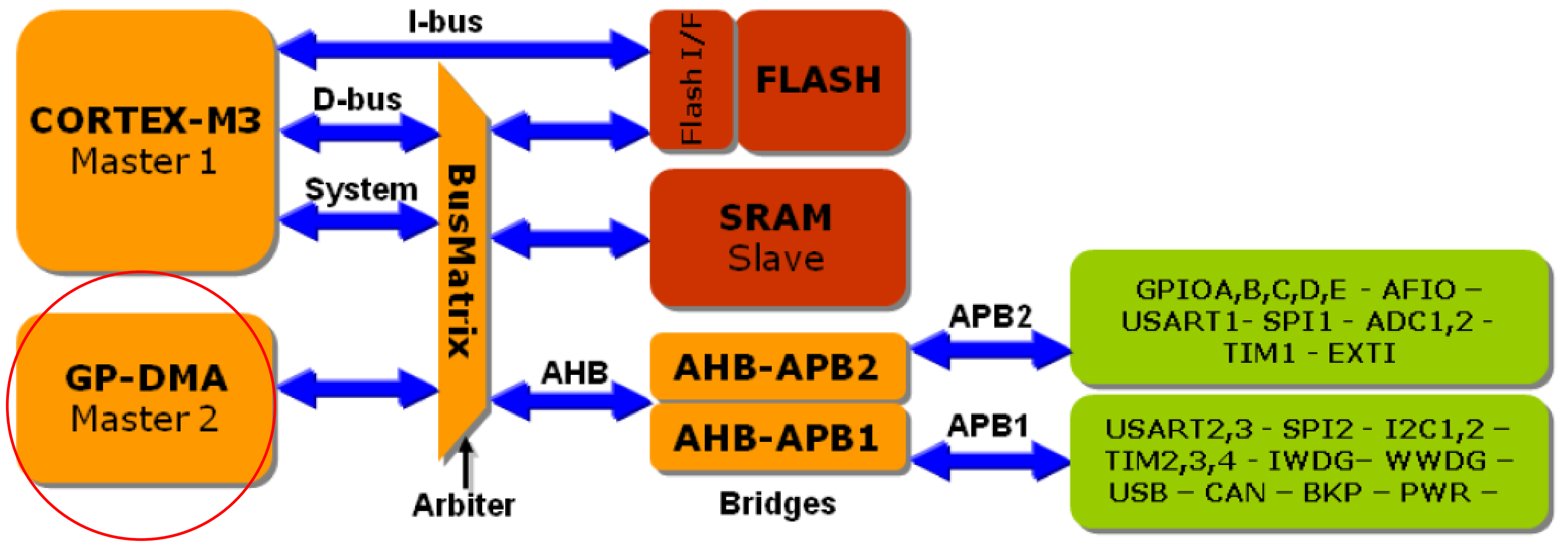
- The AHB bus matrix is clocked at the same speed as the Cortex core. However, the AHB busses have separate prescalers (a clock divider) and may be clocked at slower speeds to conserve power.
- APB2 can run at the full 72MHz, whereas APB1 is limited to 36MHz.
- Both the Cortex-M3 Core and the Direct Memory Access (DMA) unit can be bus masters (that initiates the transaction on the bus). Because of the inherent parallelism of the bus matrix, they will only arbitrate if they both are attempting to access the SRAM, APB1 or APB2 at the same time.
- The bus arbiter will guarantee 2/3 access time for the DMA and 1/3 for the Cortex CPU.



Ref: from Hitex

Direct Memory Access (DMA)

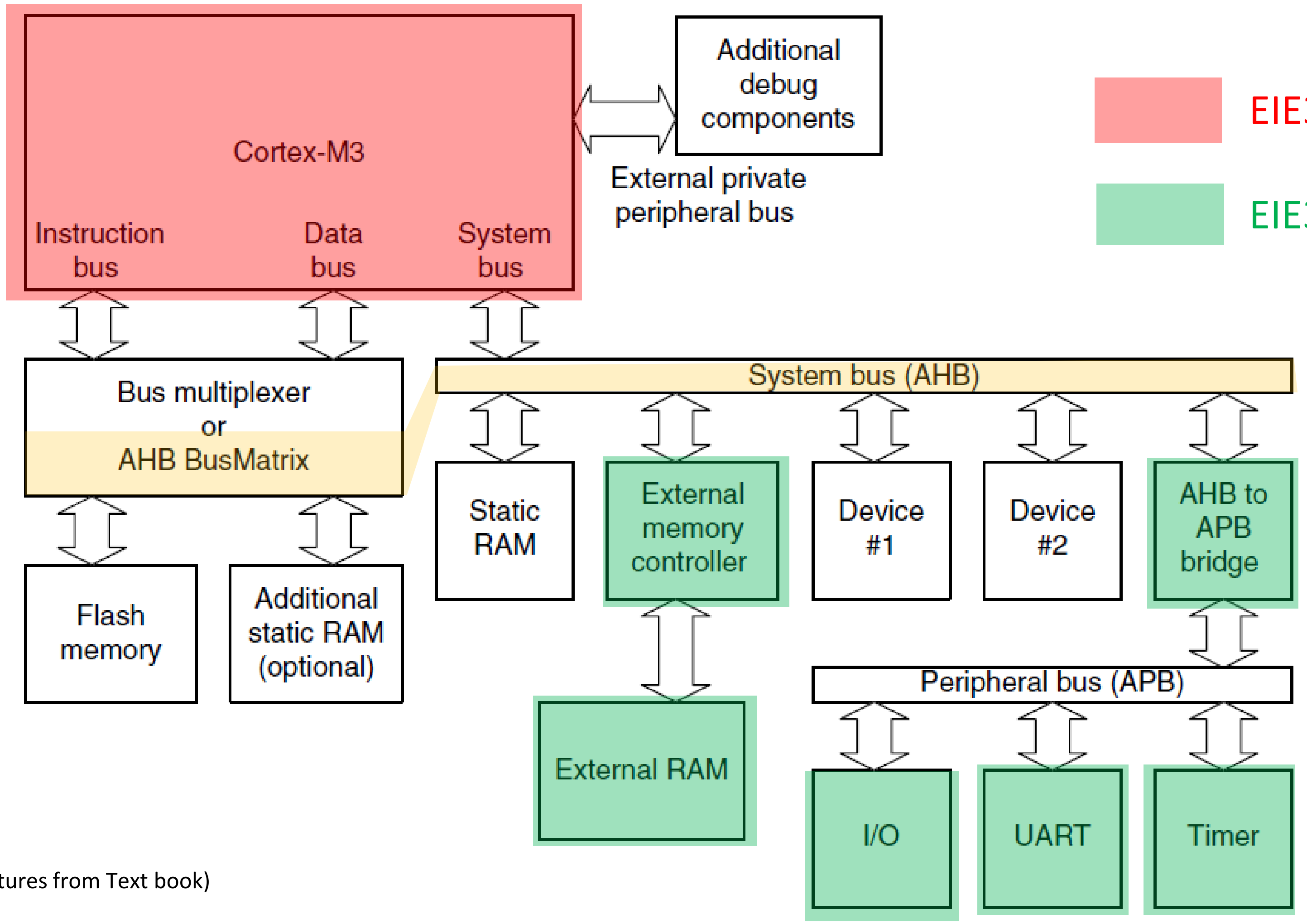
- DMA can perform autonomous data transfers from memory to memory, peripheral to memory, memory to peripheral and peripheral to peripheral without using CPU's resource.
- The memory to memory transfers will be performed as fast as the DMA channel can move the data.
- Cortex CPU can be used to move data between peripherals and the internal SRAM.
- DMA is optimized for the short but frequent data transfers that typically find in microcontroller applications.



CORTEX-M3
Master 1
GP-DMA
Master 2
I-bus
D-bus
System
BusMatrix
Arbiter
AHB
Flash I/F
FLASH
SRAM
Slave
AHB-APB2
AHB-APB1
Bridges
APB2
APB1
GPIOA,B,C,D,E - AFIO – USART1 - SPI1 - ADC1,2 - TIM1 - EXTI
USART2,3 - SPI2 - I2C1,2 – TIM2,3,4 - IWDG– WWDG – USB – CAN – BKP – PWR –

Ref: from Hitex

Study focus



🟥 EIE3080

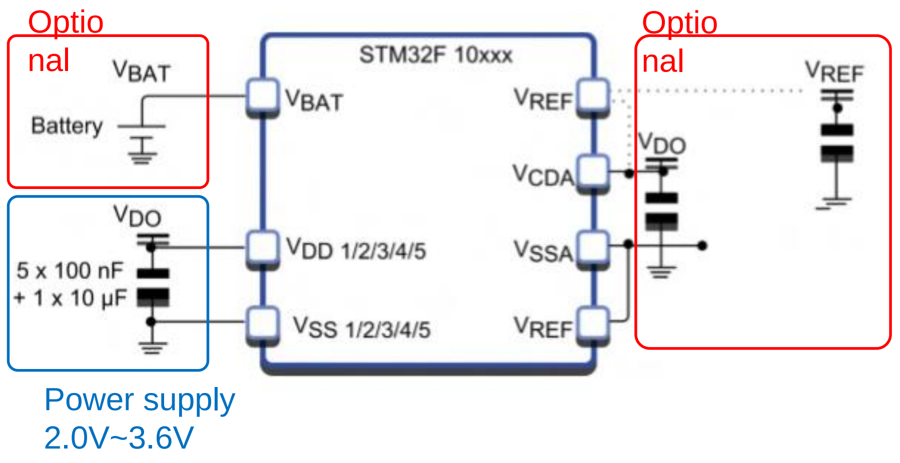
🟩 EIE3810

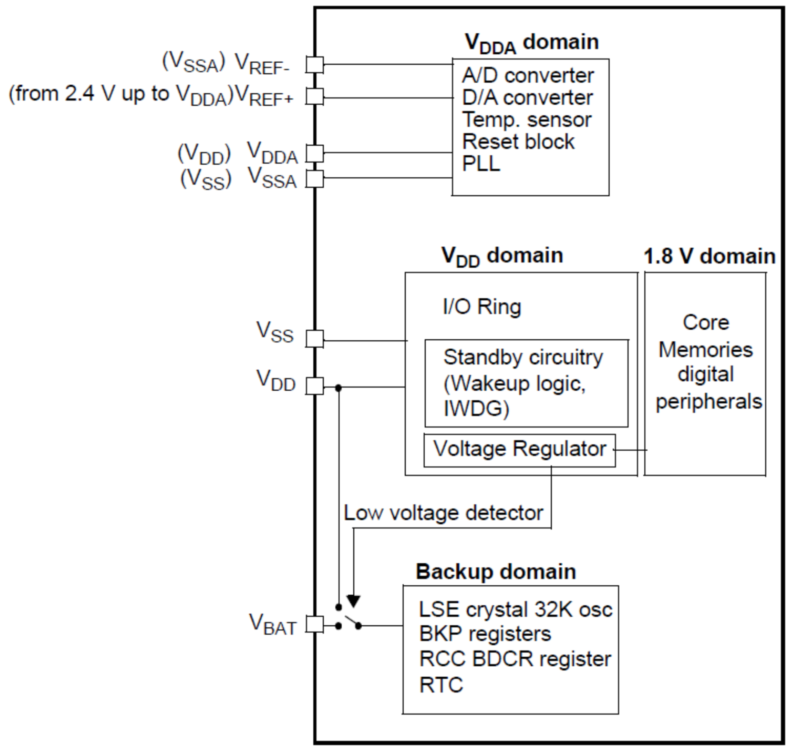
(Pictures from Text book)

Hardware Preparation of STM32 System

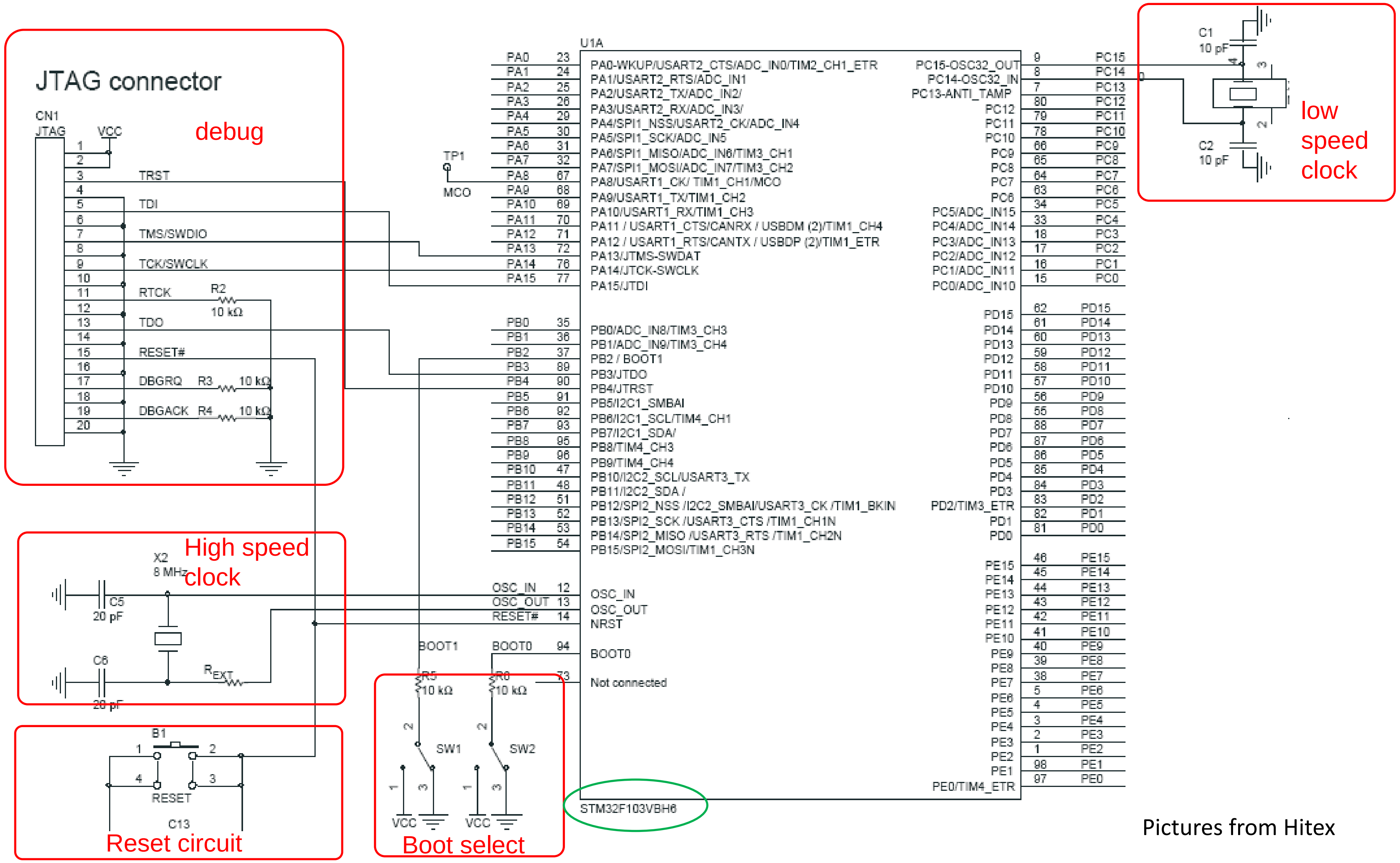
Minimal STM32 Hardware Settings

- It simply works with a single power supply in the range 2.0V to 3.6V.
- STM32 contains its own internal RC oscillators and reset circuit.
- An internal regulator is used to generate a 1.8V supply for the Cortex core.
- The real-time clock (RTC) and backup registers can be powered from the VBAT voltage when the main VDD supply is powered off.





Basic External Circuit

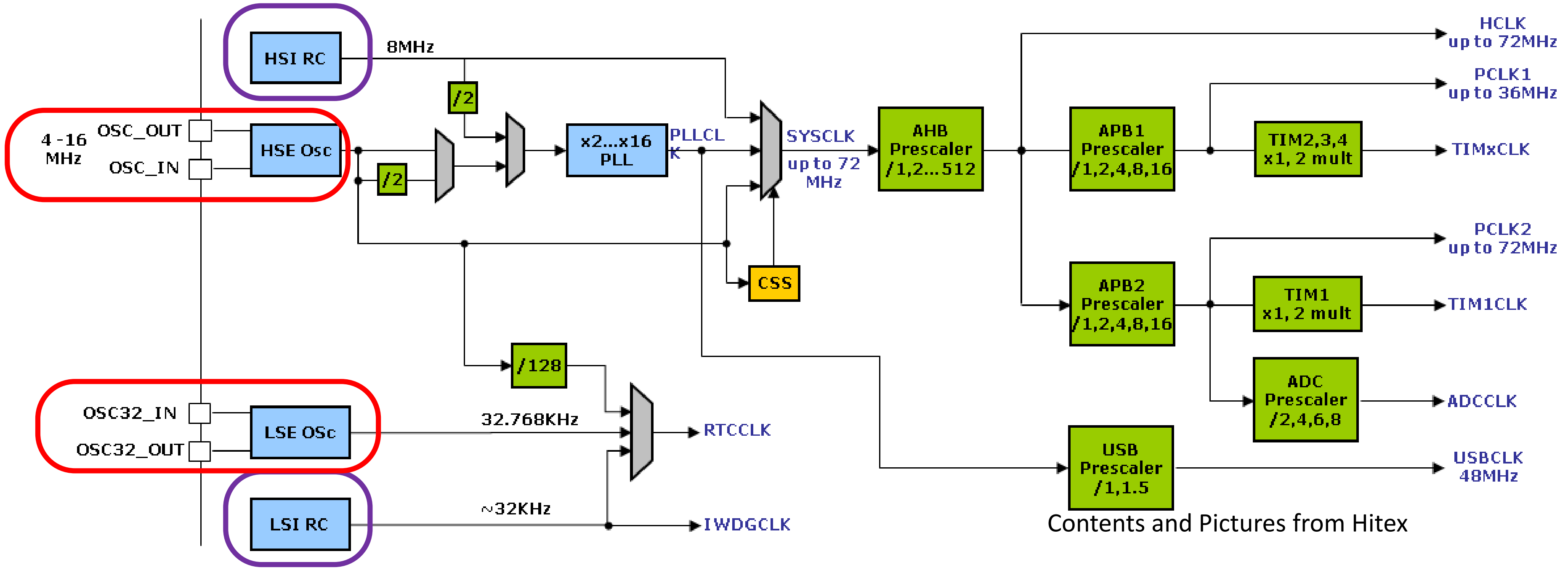


Pictures from Hitex

Internal Oscillators

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

- Microprocessor system is a huge-size state machine. It needs clock signals for synchronous operation.
- The STM32 has internal RC oscillators (the "HSI RC" and "LSI RC" below) that can provide a clock to the internal PLL. This will allow the microcontroller to run at its maximum 72-MHz clock frequency.
- BUT, the internal oscillators are not as accurate or stable as an external crystal; consequently for most designs you will need at least one external clock source.



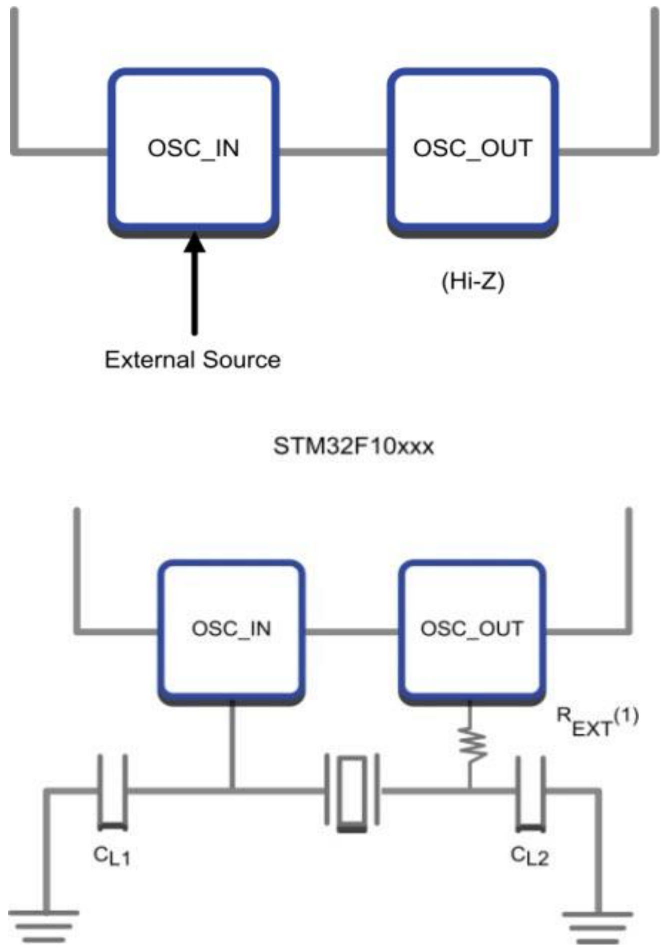
Contents and Pictures from Hitex

External Oscillators : HSE Osc

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

High Speed External Oscillator (HSE Osc)

- HSE is the main external clock source used to drive the Cortex processor and the STM32 peripheral clocks.
- It can be (i) a crystal/ceramic resonator or (ii) a user provided clock source.
- For user clock, it can be a square, sine or triangular waveform, but must have a duty cycle of 50% and a maximum frequency of 25 MHz.
- If an external crystal/ceramic resonator is used, it must be in the range 4 MHz – 16 MHz.
- Since the internal PLL multiplies the HSE Osc frequency up by integer values, <u>the external clock should be a factor of 72 MHz</u> so that you can easily derive the full (max.) operating frequency.



Contents and Pictures from Hitex

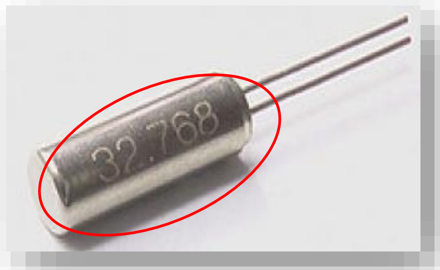
External Oscillators: LSE Osc

Low Speed External Oscillator (LSE Osc)

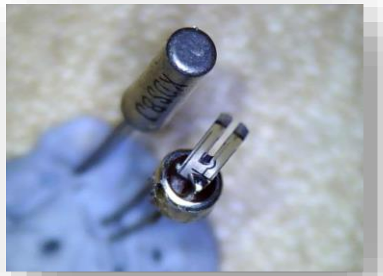
- This is the second clock source used for the real-time clock and the windowed watchdog.
- Like the HSE Osc, the LSE Osc can be an external crystal or a user-provided clock, which can again have a square, sine or triangular waveform, as long as the duty cycle is 50%.
- In both cases the LSE Osc should have a frequency of 32.768 kHz, to provide an accurate frequency for the real-time clock.

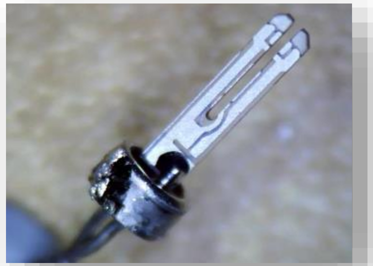
Why 32768?

- Note that the internal low speed oscillator can be used to supply the real-time clock, but it is not highly accurate.



http://en.wikipedia.org/wiki/Real-time_clock





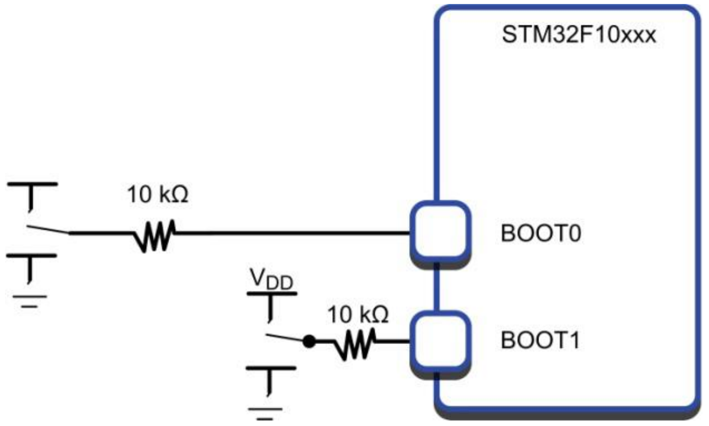
Pictures from http://www.mgsuperlabs.co.in

Boot Pins and Field Programming

- By changing the boot mode the microcontroller will alias different areas of the memory map to the bottom of memory. This allows the execution of code from (1) user FLASH, (2) internal SRAM or (3) system memory. These modes are selected by two external boot pins BOOT0 and BOOT1.
- If **system memory** is selected, the STM32 will start to execute a factory- programmed bootloader, which allows the user flash memory to be reprogrammed.
- USART1 is the default serial interface used to download code from a PC, so you will need to add an RS232 driver chip. (EIE3810 uses this method)

Table 1. Boot pin configuration

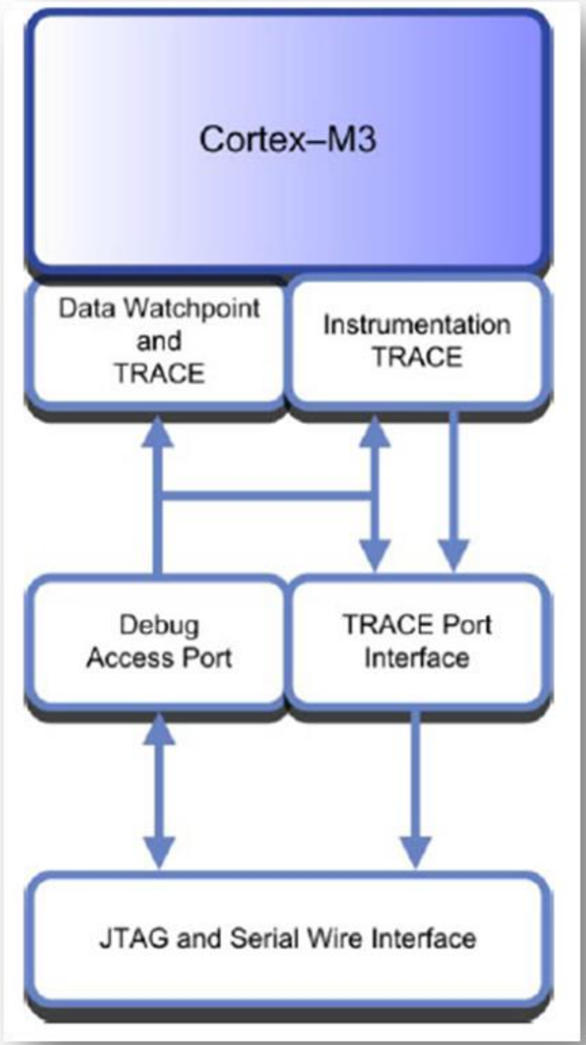
Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
X	0	User Flash memory	User Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

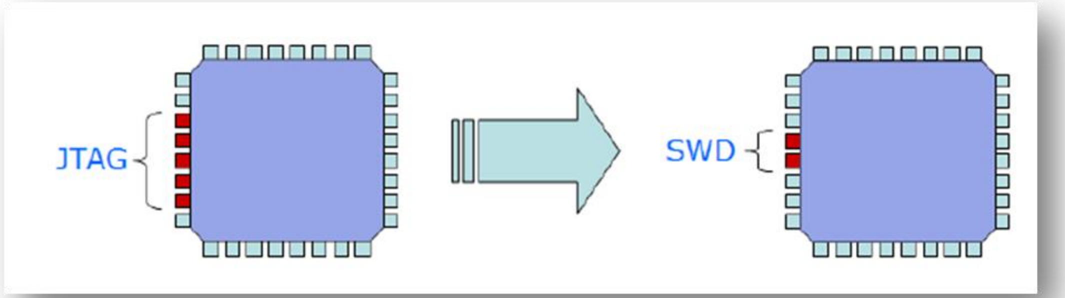


Contents and pictures from Hitex

Debug Port

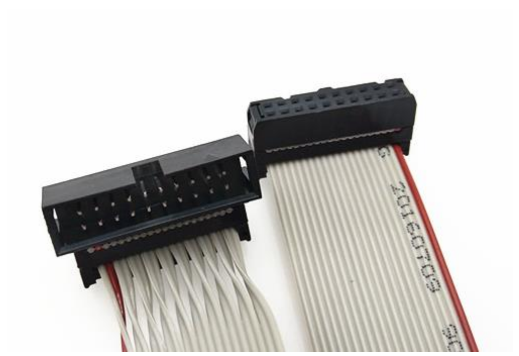
- How to debug your program if it is not working as designed?
- Hardware debug port allows a debugger to connect to the STM32.
- The Cortex CoreSight debug system supports two connection standards: the five-pin JTAG port and the 2-pin Cortex serial wire port. Both configurations occupy some GPIO (general-purpose input/output).
- After reset, the Cortex CPU places these pins in their alternate function setting so that the debug port is available.
- If you wish to use them for I/O, you must program the alternate function registers to convert them back to GPIO pins.

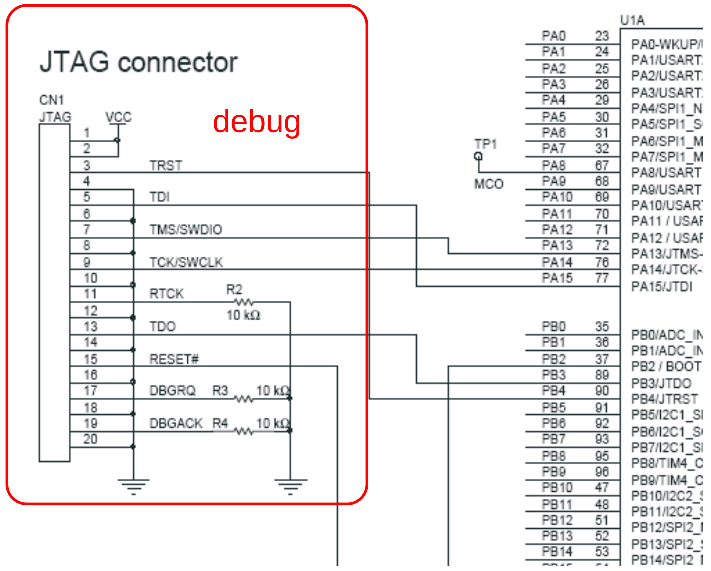


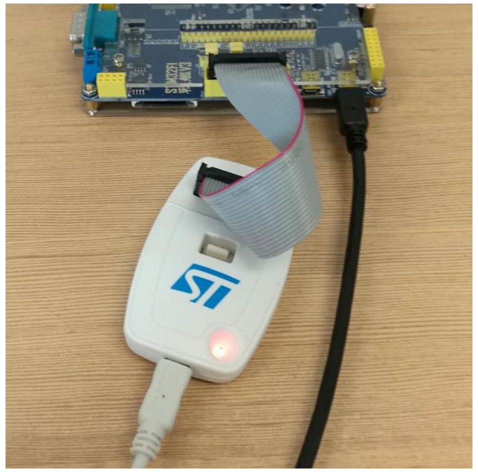


Cortex debug system

- The five-pin JTAG interface is brought out to a 20-pin IDC type connector that has a standard pinning for all JTAG tools.
- The serial wire interface uses Port A 13 for the serial data and Port A 14 for the serial clock.
- (*CoreSight* debug system will not be practiced in EIE3810 due to limited time available.)







STMicroelectronics ST-LINK

End