# 24EC504 – Unit I – Part 1: Embedded System Basics and ARM Cortex-M3

## 1. Embedded System Basics

### 1.1 Definition

An embedded system is a special-purpose combination of computing hardware and software designed to perform one dedicated function within a larger mechanical or electrical system, rather than serving as a general-purpose computer. It is built around a microprocessor or microcontroller, works under real-time constraints, and is optimized for cost, power consumption and size. Examples: the ECU in a car, a digital camera's controller, a pacemaker.

### 1.2 Essential Components

- **Microprocessor or microcontroller:** the "brain" that executes instructions.
- **Memory:** ROM/Flash stores the program permanently; RAM holds temporary runtime data.
- **Input/output interfaces:** let the system sense and act on the environment (sensors, actuators, communication ports).
- **Embedded software (firmware):** programs the specific behaviour of the device.
- **Supporting blocks:** power-management circuitry and, where needed, a real-time clock or timer modules.

### 1.3 Microprocessor vs Microcontroller

| Aspect | Microprocessor | Microcontroller |
|---|---|---|
| Contents | CPU only (ALU, control unit, registers) | CPU + memory + I/O peripherals on one chip |
| External chips | Needs external RAM/ROM and I/O | Largely self-contained ("computer on a chip") |
| Typical use | General-purpose computing (PCs) | Dedicated embedded applications |
| Cost, size, power | Higher | Smaller, cheaper, more power-efficient |

### 1.4 Harvard Architecture

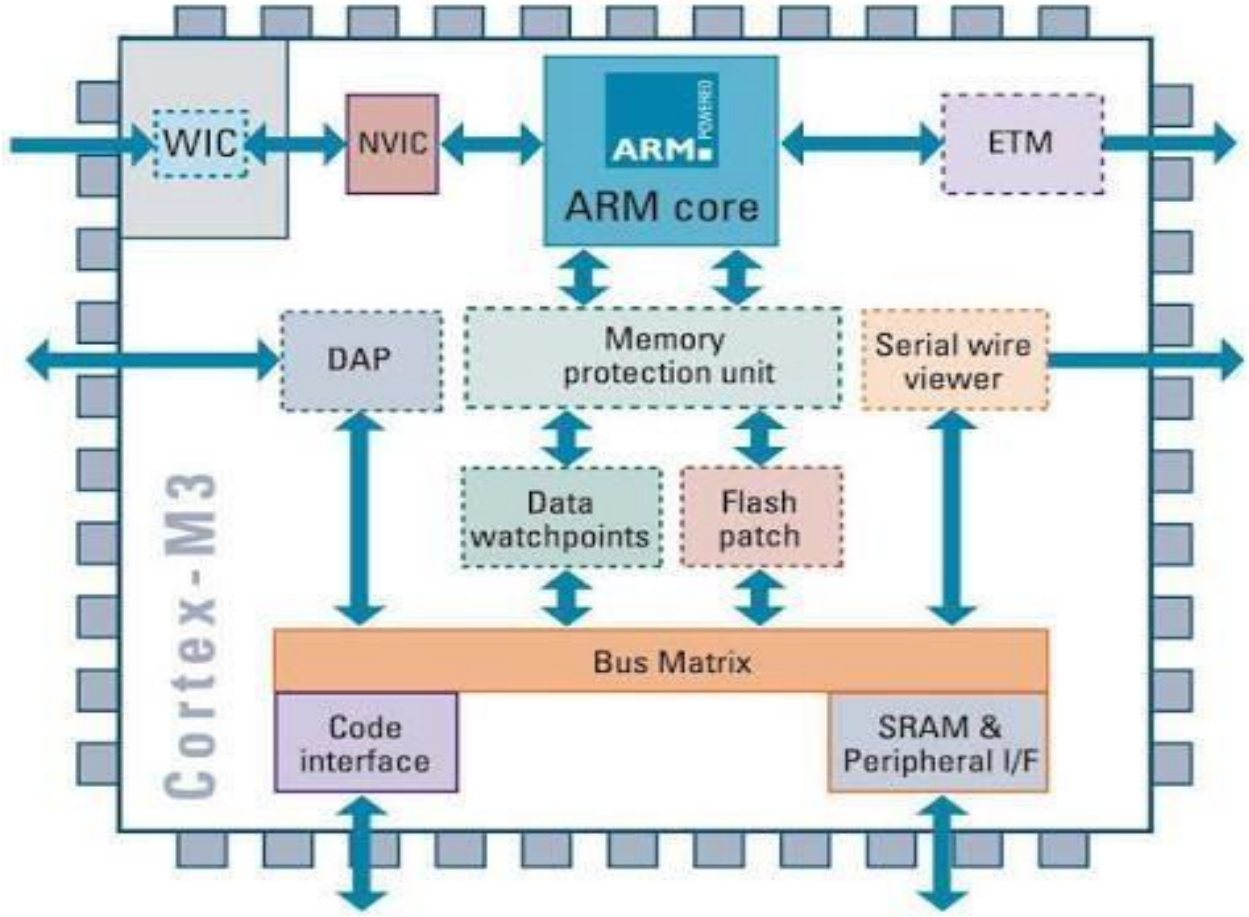
Harvard architecture uses physically separate memory and separate data/address buses for instructions and data. Von Neumann architecture shares one memory and bus for both. With Harvard, the CPU can fetch an instruction and access data in the same clock cycle, which improves throughput. It is widely used in microcontrollers, including the Cortex-M3 and ESP32.

### 1.5 Deterministic Behaviour

Deterministic means the response time to an event, such as an interrupt, is fixed and predictable: the same number of clock cycles every time, whatever else the system is doing. In the Cortex-M3 this comes from hardware such as the NVIC, which enters interrupts with fixed latency instead of relying on software polling. It is essential for motor control and medical devices, where a late or inconsistent response could cause failure.

## 2. ARM Cortex-M3 Overview

The Cortex-M3 is the processor core ARM designed specifically for microcontrollers. It was released in 2006 as the first member of the Cortex family, with the goal of doing more useful work per clock cycle, per rupee and per milliwatt. It packs high-end features (built-in interrupt controller, memory protection, debug hardware) into a small, efficient core.




**Why it matters**

- **Low power consumption:** battery products such as wireless sensors and medical wearables last longer.

- **Enhanced determinism:** the interrupt response takes a fixed, known number of cycles.
- **Improved code density:** the instruction set lets the compiler produce smaller code, so more fits in limited flash.
- **Ease of use:** engineers from 8-bit or 16-bit chips can move over easily, programming entirely in C.
- **Lower cost:** fewer logic gates than older 32-bit designs, so Cortex-M3 microcontrollers can sell for under a dollar.
- **Wide tool support:** many companies provide compilers and development boards.

### 3. Cortex-M3 Architecture, Block by Block

| Block | What it does |
|---|---|
| ARM core | The processing engine: fetches, decodes and executes instructions. |
| NVIC (Nested Vectored Interrupt Controller) | Manages interrupts. "Nested": a higher-priority interrupt can pause a lower-priority handler. "Vectored": jumps straight to the correct handler through the vector table, with no polling. |
| WIC (Wake-up Interrupt Controller) | Watches for interrupts in deep sleep so the chip can wake instantly without staying fully awake. |
| ETM (Embedded Trace Macrocell) | Optional block that records which instructions the core executed, to find hard-to-reproduce bugs. |
| DAP (Debug Access Port) | Lets an external debugger (JTAG or 2-wire Serial Wire Debug) read and write memory and registers. |
| MPU (Memory Protection Unit) | Optional hardware that defines which memory a program may access and how. |
| Data watchpoints | Alert the debugger the moment a watched address or register changes. |
| Flash patch | Lets a debugger redirect execution (breakpoint or patch) without reprogramming flash. |
| Serial wire viewer (SWV) | Single-pin output streaming debug and profiling data without stopping the processor. |
| Bus matrix | Internal "traffic junction" connecting core, memories and peripherals so several transfers happen at once. |

| Block | What it does |
|---|---|
| Code interface | Dedicated bus for fetching instructions from flash/code memory. |
| SRAM and peripheral interface | Bus path for working data and on-chip peripherals such as timers and communication ports. |

### 4. Instruction Sets: ARM, Thumb and Thumb-2

Older ARM processors ran two instruction sets and had to switch between them:

- **ARM instructions:** 32 bits wide; powerful but use more memory.
- **Thumb instructions:** 16 bits wide; a smaller subset that saves flash but with reduced capability.

Switching states cost time and added complexity. **Thumb-2** solves this: a single instruction set that freely mixes 16-bit and 32-bit instructions in the same program. The compiler uses compact 16-bit instructions for simple, frequent operations and 32-bit instructions where more power is needed. The primary advantage is near-ARM performance with code density close to pure Thumb.

The Cortex-M3 uses only Thumb-2 (plus classic Thumb) and never enters ARM state, so there is no state-switching overhead. It therefore cannot run old pure-ARM-state code.

### 5. Memory Map

The Cortex-M3 has a fixed 4 GB address space with a predictable layout, so every Cortex-M3 chip from any manufacturer places code, RAM and peripherals in the same regions. This makes software more portable.

| Region | Address range | Use |
|---|---|---|
| CODE | 0x00000000 – 0x1FFFFFFF | Program code (usually flash); also the exception vector table after power-up |
| SRAM | 0x20000000 – 0x3FFFFFFF | Main working memory for variables and data |
| Peripherals | 0x40000000 – 0x5FFFFFFF | On-chip devices (timers, UARTs, ADCs) accessed with ordinary read/write instructions |
| External RAM | 0x60000000 – 0x9FFFFFFF | Memory connected outside the chip |
| External device | 0xA0000000 – 0xDFFFFFFF | Peripherals connected outside the chip |

| Region | Address range | Use |
|---|---|---|
| System level | 0xE0000000 – 0xFFFFFFFF | Private components: NVIC, MPU control registers, debug hardware |

**6. Bus Interfaces**

A bus is an internal highway carrying data between the core and memory or peripherals. Having several buses lets the Cortex-M3 fetch an instruction and move data at the same moment.

- **Code memory buses (I-Code and D-Code):** two dedicated paths optimized for fast instruction fetch from code memory.
- **System bus:** general-purpose path to SRAM, on-chip peripherals, external RAM and external devices.
- **Private peripheral bus:** a separate path for the system's own built-in components, such as debug hardware.

**7. Memory Protection Unit (MPU)**

The MPU is an optional hardware gatekeeper for memory. Software defines rules about which regions a program may touch and how.

- If a program breaks a rule (for example, a user task writes into OS memory), the MPU raises a fault exception and a handler decides the response.
- An OS can use the MPU to shield kernel data from untrusted application code.
- Regions can be marked read-only to prevent accidental overwrites, or walled off between tasks.

The result is a more robust system: one buggy piece of code is far less likely to silently corrupt the rest.

**8. Registers**

Registers are the processor's tiny, ultra-fast storage locations, much faster than RAM. The Cortex-M3 has R0–R15 plus special registers.

| Register | Role |
|---|---|
| R0–R7 (low registers) | 32-bit general purpose; accessible to both 16-bit Thumb and 32-bit Thumb-2 instructions |
| R8–R12 (high registers) | 32-bit general purpose; only Thumb-2 (32-bit) instructions can reach all of them |
| R13 – Stack Pointer (SP) | Top of the stack (last-in-first-out). Two versions: Main Stack Pointer (MSP) for the OS kernel and exception handlers, Process Stack Pointer (PSP) for application code |

| Register | Role |
|---|---|
| R14 – Link Register (LR) | Stores the return address when a function is called |
| R15 – Program Counter (PC) | Address of the instruction being executed. Because of pipelining, reading PC returns a value 4 bytes ahead of the running instruction |
| Program Status Registers (PSRs) | Arithmetic flags and execution mode |
| PRIMASK, FAULTMASK, BASEPRI | Interrupt mask registers: selectively block interrupts or exceptions |
| CONTROL register | Selects the active stack pointer (MSP or PSP) and privileged or unprivileged mode |

**9. Where the Cortex-M3 Is Used, and Why**

- **Low-cost microcontrollers (toys, appliances):** low gate count keeps cost down; low power and C programming ease the move from 8-bit designs.
- **Automotive electronics:** needs guaranteed fast responses; low interrupt latency, up to 240 external interrupts and an optional MPU make it dependable.
- **Data communication devices (Bluetooth, ZigBee):** Thumb-2 is efficient at bit-level manipulation for packing and unpacking protocol data, while staying power-efficient.
- **Industrial control:** quick interrupt handling and strong fault-detection hardware stop small errors becoming failures.
- **Consumer products needing more computing power:** the MPU lets complex software (menus, connectivity, sensors) run safely by isolating faults.