2 unit iot - iot

Embedded Systems (Jawaharlal Nehru Technological University, Kakinada)

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY(A)
(An AUTONOMOUS Institution)
Approved by AICTE, New Delhi * Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA* Accredited by NAAC A+ Grade with CGPA of 3.40
Recognized by UGC Under Sections 2(f) and 12(B) of the UGC Act, 1956
Aditya Nagar, ADB Road, Surampalem, Gandepalli Mandal, Kakinada District - 533437, A.P
Ph. 99591 76665. Email: office@acet.ac.in. www.acet.ac.in

IOT and Applications

Dr.R.V.S.Lalitha, Professor, Department of CSE, 8008379819

UNIT II:

Elements of IoT: Hardware Components- Computing- Arduino, Raspberry Pi, ARM Cortex-A class processor, Embedded Devices – ARM Cortex-M class processor, Arm Cortex-M0 Processor Architecture, Block Diagram, Cortex-M0 Processor Instruction Set, ARM and Thumb Instruction Set.

Hardware components

Embedded system denotes a system that embeds software into a computing platform. The system is dedicated for either an application(s), specific part of an application, product or a component of a large system.

Embedded device refers to a device, which embeds software into a computing platform and performs the computations and communication for specific systems.

Microcontroller unit (MCU) means a single-chip VLSI unit (also called microcomputer), which may be having limited computational capabilities. The MCU possesses memory, flash, enhanced input-output capabilities and a number of on-chip functional units.

Flash memory is an electronic non-volatile computer memory storage medium that can be electrically erased and reprogrammed. The two main types of flash memory, NOR flash and NAND flash, are named for the NOR and NAND logic gates. Both use the same cell design, consisting of floating gate MOSFETs.

Timer refers to a device which enables initiating new action(s) on timer start, on the clock inputs, time outs or when the number of clock inputs equal to a preset value.

Port refers to a device that enables input output (IO) communication between the MCU and another device such as a sensor or actuator or keypad or with an external computing device.

USB port connects the device hardware to a computer, downloads the developed codes into the device from the computer, or sends the codes from the device to the computer. USB port can also provide the power for charging the battery of the connected platform, thus an external charger is not needed.

GPIO pins refer to General Purpose Input-Output pins. A pin that can be used in addition to digital input and output for other purposes, such as Rx and Tx or SDA and SCK, PWMs, analog inputs, outputs or timer outputs. The Rx and Tx pins are used during UART protocol-based reception and transmission, SDA and SCK are used during use of I2C protocol-based serial data and clock communication.

Rx Receiver

Tx Transmitter

SDA Serial Data

SCK Serial Clock

I2C Inter-Integrated Circuit

UART Universal Asynchronous Receiver and Transmitter

MCU Microcontroller Unit

SoC System on a chip

Board is an electronic hardware—an electronic circuit board with MCU or SoC, circuits and connectors, which provide the connections to other ICs and circuit components. The ICs and circuit components can also be inserted or joined or put in place onto the board, by surface mount technology. The board may also have battery, power supply, voltage regulator or connections for the power.

Platform denotes a set consisting of computing and communication hardware, software and operating system (OS). A platform enables working with different software, APIs, IDE and middleware. A platform may enable the development of codes at the development stage. It may also enable prototype development for an application(s) or specific parts of an application.

Module (hardware) is smaller form-factor hardware which can be placed onto a board. The module may embed the software. It may enable use of the board circuit with shorter form factor. An example is RF module placed onto an electronic board.

Shield means a supporting circuit with connection pins, socket(s) and supporting software.

The supporting circuit enables the connectivity of a board or computing platform to external circuits. The circuit connects the elements that can be plugged onto a board or platform. Usage of the supporting circuit provides extra features, such as connectivity to wireless devices, such as ZigBee, ZigBee IP, and Bluetooth LE, Wi-Fi or GSM or RF module or to a wired device, such as Ethernet shield. The Ethernet shield enables wired connection of a platform to the Ethernet controller and through an external standard LAN socket enables connectivity to a wired or Wi-Fi modem for the Internet. Shield is the term used in Arduino hardware for the supporting circuits.

Header means plastic-coated strip or plastic-capped plug-in which is placed on top of the pin holes when making connection of the wires without electronic soldering. A header also provides jumpers. Six-pin header means plastic 6-pin plug-in that connects the 6 pin holes. This header is a component, distinct from enveloping words in a data stack for communication at a layer according to a protocol.

Jumper denotes a wire with a solid tip at each end which is normally used to interconnect the components on an electronic-circuit breadboard. Jumpers are used for transferring IOs or signals to or from the pins.

Interrupt means an action in which a running program interrupts an hardware signal, such as timer timeout or on execution of a software instruction for interrupt. For example, a program interrupts for sending the data or for accepting a newly added device in the system or on execution of the INT instruction.

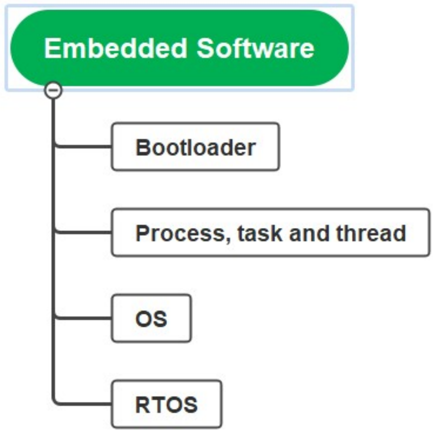
Integrated Development Environment (IDE) means a set of software components and modules which provide the software environment for developing and prototyping.

Operating system (OS) is a system software which facilitates the running of processes, allocation of memory, system calls to the IOs, facilitates the use of network subsystems, and which does devices management, priority allocations of processes and threads, and Prototyping the Embedded Devices for IoT and M2M 295 enables multitasking and running of number of threads. The OS enables many system functions using the given computing device hardware.

8.2 EMBEDDED COMPUTING BASICS

Embedding means embedding function software into a computing hardware to enable a system function for the specific dedicated applications. A device embeds software into the computing and communication hardware, and the device functions for the applications.

8.2.1 Embedded Software



Bootloader

Bootloader is a program which runs at the start of a computing device, such as an MCU. A bootloader initiates loading of system software (OS) when the system power is switched on, and power-on-self test completes.

Operating System

An operating system (OS) facilitates the use of system hardware and networking capabilities. When a load of the OS into RAM completes then the MCU starts the normal operational runtime environment. When the device is executing multiple tasks or threads, then also an OS is required. The OS controls the multiple processes and device functions.

Process, task and thread are the set of instructions which run under the control of the OS.

The OS enables memory allocation to different processes, and prioritising of the processes enables the use of network hardware and device hardware functions and execution of software components and processes.

The OS is at flash memory of the device. It may be required to load functions at the device RAM.

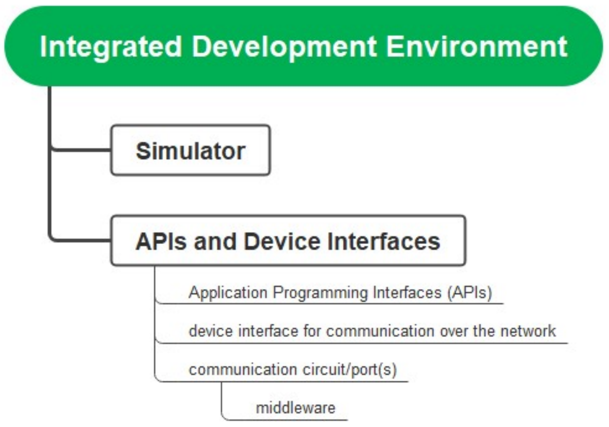
An OS may be open source, such as Linux or its distribution. Linux distribution means a package or set of software components and modules bundled together for specific functions or for specific hardware, and distributed for wider usages and applications.

For example, Arduino Linux distribution runs in Arduino circuit boards and the Linux functions enable developing the application programs for using the Arduino.

Real-Time Operating System

Real-Time Operating System (RTOS) is an OS that enables real-time execution of processes on computing and communication hardware. RTOS uses prioritisation and priority allocation concept to enable the execution of processes in real-time.

8.2.2 Integrated Development Environment


Integrated Development Environment
Simulator
APIs and Device Interfaces
Application Programming Interfaces (APIs)
device interface for communication over the network
communication circuit/port(s)
middleware

Integrated Development Environment (IDE) is a set of software components and modules which provide the software and hardware environment for developing and prototyping. An IDE enables the codes development on a computer, and later on enables download of the codes on the hardware platform. IDE enables software that communicates with the Internet web server or cloud service.

IDE consists of the device APIs, libraries, compilers, RTOS, simulator, editor, assembler, debugger, emulators, logic analyser, and application codes' burner for flash, EPROM and EEPROM and other software components for integrated development of a system. IDE may be open source. For example, Arduino has open source IDE from the Arduino website. The IDE or prototype tool enables a prototype design. IDE is used for developing embedded hardware and software platforms, simulating, and debugging. IDE is a tool for software development of embedded devices, which makes application development controller system. The library consists of a number of programs. The library has programs for each serial-interface protocol, which can be used in the device. The program enables using the protocol-specific programs directly, such as using a program for reading an RFID tag or using a program for sending data to the USB port for onward transmission on the Internet.

Simulator

Simulator is software that enables development on the computer without any hardware, and then prototyping hardware can be connected for embedding the software and further tests.

APIs and Device Interfaces

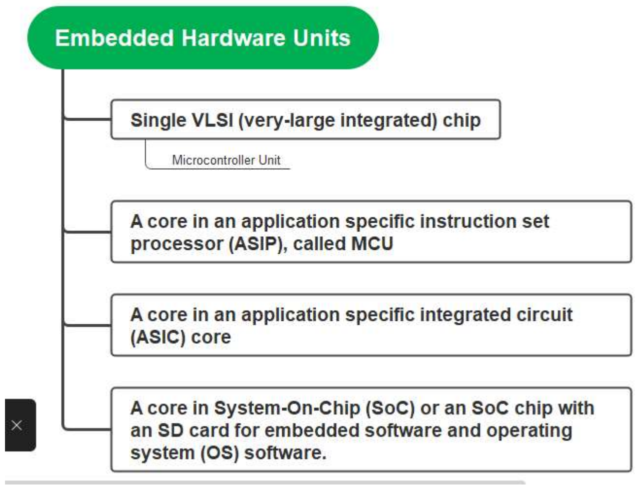
Software consists of device Application Programming Interfaces (APIs) and device interface for communication over the network and communication circuit/port(s) which also includes a middleware.

The middleware creates IPv4, IPv6, 6LowPAN, MQTT, COAP, LWM2M, REST and other communication protocol stacks.

Device Interfaces

A connectivity interface consists of communication APIs, device interfaces and processing units. Software commands the action on the message or information received followed by hardware port outputs for the actuators.

8.2.3 Embedded Hardware Units


Embedded Hardware Units
Single VLSI (very-large integrated) chip
Microcontroller Unit
A core in an application specific instruction set processor (ASIP), called MCU
A core in an application specific integrated circuit (ASIC) core
A core in System-On-Chip (SoC) or an SoC chip with an SD card for embedded software and operating system (OS) software.

The hardware includes the following:

● Single VLSI (very-large integrated) chip

● A core in an application specific instruction set processor (ASIP), called MCU

● A core in an application specific integrated circuit (ASIC) core

● A core in System-On-Chip (SoC) or an SoC chip with an SD card for embedded software and operating system (OS) software.

Following subsections describe MCU, SoC and selection of platform for prototyping.

Microcontroller Unit

An MCU is a single-chip VLSI unit (also called microcomputer) which though having limited computational capabilities, possesses enhanced input-output capability and has a number of on-chip functional units, such as Internal RAM, flash, IO ports, GPIOs, serial interfaces, timers, serial ports and timers.

Application-specific MCUs have additional on-chip functional units, such as PWM circuits (1, 2 or 3), ADC (1, 2, 4 or higher) and other functional units. Figure 8.1 shows the on-chip functional units in a microcontroller.

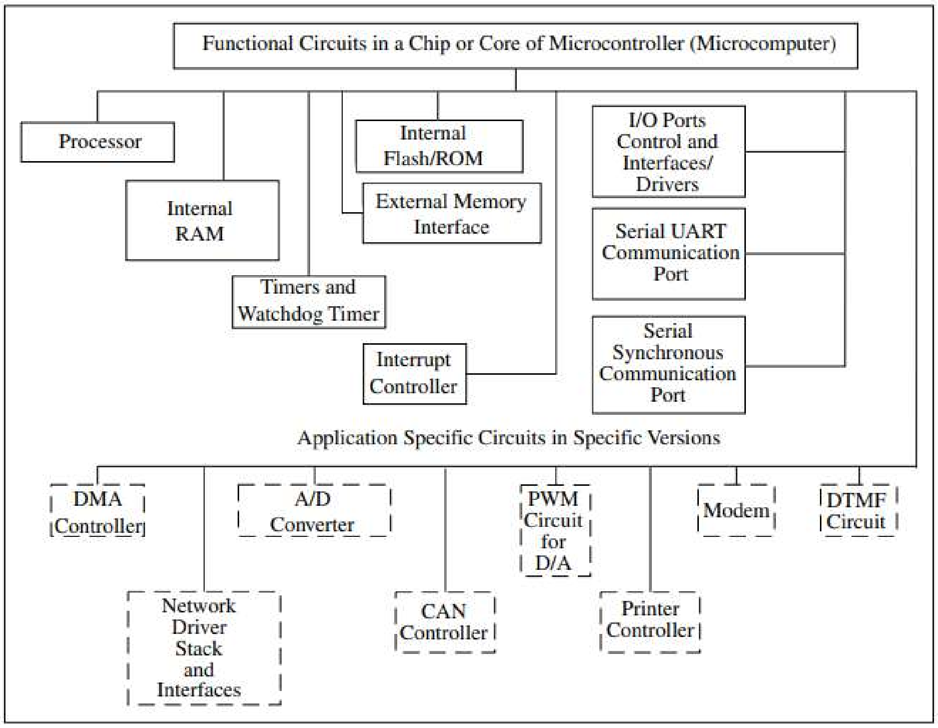

Functional Circuits in a Chip or Core of Microcontroller (Microcomputer)
Processor
Internal RAM
Timers and Watchdog Timer
Internal Flash/ROM
External Memory Interface
Interrupt Controller
I/O Ports Control and Interfaces/ Drivers
Serial UART Communication Port
Serial Synchronous Communication Port
Application Specific Circuits in Specific Versions
DMA Controller
A/D Converter
PWM Circuit for D/A
Modem
DTMF Circuit
Network Driver Stack and Interfaces
CAN Controller
Printer Controller

Figure 8.1 Microcontroller, on-chip functional units and application specific units

An MCU is an IC chip, available from a number of sources, such as ATMEL, Nexperia, Microchip, Texas Instruments or Intel. A source may manufacture different types, families and groups of MCUs, like ATMEL manufactures AVR®8 and AVR®32 families of MCUs. A family of MCUs can have different versions.

MCU can be of 8-bit, 16-bit or 32-bit family.

● MCU clock frequency can be 8 MHz, 16 MHz, 100 MHz, 200 MHz or higher. The clock frequency depends on the version and family. Performance defines number of instructions executed per sec that primarily depends on the clock frequency also. A metric for performance is Million Instructions Per Second (MIPS). Another metric for performance is Million Floating Point Operations Per Second (MFLOPS).

● MCU includes RAM which can be 4 kB, 16 kB, 32 B or higher. RAM read and write of byte takes an instruction cycle each. RAM is used for temporary variables, stacks and MCU includes EEPROM and flash memory, which may be 512 B, 1 kB, 2 kB, 4 kB, 16 kB, 64 kB, 128 kB, 512 kB or higher. Flash stores the programs, data, tables and required information during building and testing stages, and then stores a final version of the application program in the embedded device.

● MCU includes timers, I/O ports, GPIO pins, serial synchronous and asynchronous ports and interrupt controllers.

● MCU includes several functional units in specific version, such as ADC, multi-channel ADC or ADC with programmable positive and negative reference voltage pins, PWMs, RTC, I2C, CAN and USB ports, LCD interface, ZigBee interface, Ethernet, modem or other functional units, depending on a specific source, family, group and version.

System-on-Chip

Complex embedded devices, such as mobile phones, consist of a circuit which is designed on a single silicon chip. The circuit consists of multiple processors, hardware units and software. An SoC is a system on a VLSI chip that has multiple processors, software and all the needed digital as well as analog circuits' on chip. An SoC embeds processing circuit with memory and is specific to dedicated application(s) and may have the analog circuits.

SoC may associate an external SD card in a mobile phone. The card stores the external programs and operating system and enables the use of the chip distinctly for distinct purposes. Secure Digital Association created the SDIO (Secure Data Input-Output) card.

The card consists of standard, mini, micro or nano form. It consists of flash memory and communication protocol. An SoC can be from different sources, for examples, Raspberry Pi and BeagleBone.

Selection of Embedded Platform

Selection among the number of different available platforms depends on a number of factors like price, open source availability, ease of application development and needed capabilities, performance required from IoT device and suitability for developing and using for prototyping and **designing.**

Hardware

The choice, beside the price, of embedded hardware depends on the following:

● Processor speed required which depends upon the applications and services. For example, image and video processing need the high-speed processors

● RAM need which may be 4 kB or higher depending upon the OS and applications. For example, requirement is 256 kB for using the Linux distribution. New generation mobile phones have over one GB RAM.

● Connection needs to ZigBee, ZigBee IP, Bluetooth LE, Wi-Fi or Wired Ethernet for networking using a supporting circuit (shield)

● USB host

● Sensor, actuator and controllers interfacing circuits, such as ADC, UART, I2C, SPI, CAN single or multiples

● Power requirements, V- and V+, 0 V and 3.3 V or 0 V and 5 V or other.

Software Platforms and Components

Selection among a number of different available software depends on hardware platform, open source availability of software components, cost of availability or development of other required components for applications and services.

The choice of embedded software, beside the price, depends on the following: IDE with device APIs, libraries, OS or RTOS, emulator, simulator and other environment components, middleware with communication and Internet protocols, and Cloud and sensor-cloud platform for applications development, data storage and services.

Open Source Framework for IoT Implementation

Eclipse IoT (www.iot.eclipse.org) provides open source implementations of a number of standards including MQTT CoAP, LWM2M and services and frameworks that enable an Open Internet of Things software. Eclipse tool work with Lua. An IoT programming language is Lua. IoT can be programmed in any open source language like Python or Java or PHP also using the tools.

Arduino development tools provide a set of software that includes an IDE and the Arduino programming language for a hardware specification for interactive electronics that can sense and control more of the physical world.

Middleware

OpenIoT is an open source middleware. It enables communication with sensor clouds and enables cloudbased 'sensing as a service', IoTSyS is middleware. It enables provisioning of communication stack for smart devices using IPv6 and a number of other standard protocols.

Web Services

Web server or cloud server or clients use SOAP, REST, JSON, HTTP, HTTPS, web sockets, web APIs, URI. These provide the building blocks (software components) for writing the codes for Web Services.

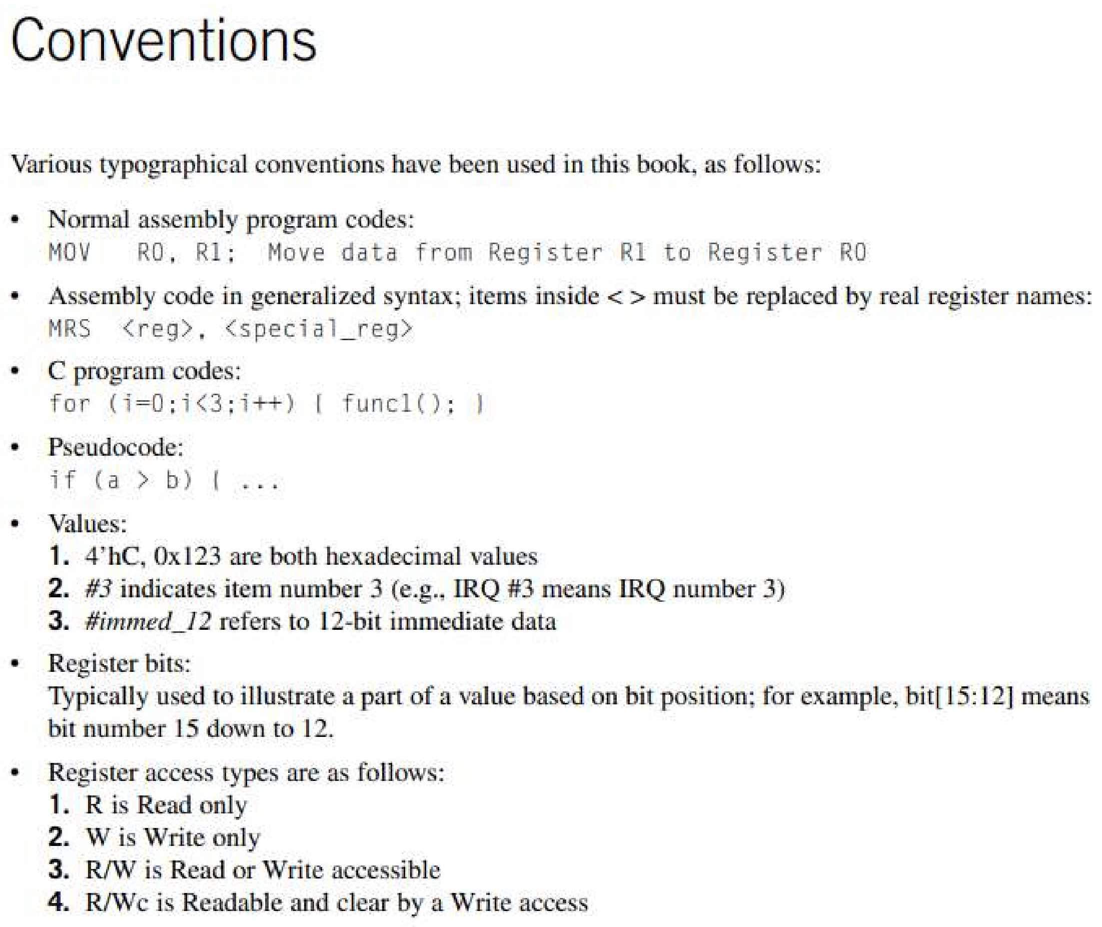
Cloud Applications Development Platforms

The cloud-based development platforms are also widely used for IoT because of world wide availability, storage and platform specific features.

Xively (Pachube/Cosm) is another cloud application development platform for sensor data. Xively is open source for basic services. It is software and server platform for data capture, data visualization real-time and other features over the Internet. It is an open source platform for Arduino open source electronics prototyping platform connectivity with web.

Nimbits enables IoT on an open source distributed cloud. PaaS at the Nimbits cloud deploys an instance of Nimbits server at the device nodes.

8.3 ARM processor



Conventions

Various typographical conventions have been used in this book, as follows:

- Normal assembly program codes:
```
MOV   R0, R1;  Move data from Register R1 to Register R0
```
- Assembly code in generalized syntax; items inside < > must be replaced by real register names:
```
MRS  <reg>, <special_reg>
```
- C program codes:
```
for (i=0;i<3;i++) { func1(); }
```
- Pseudocode:
```
if (a > b) { ...
```
- Values:
  1. 4'hC, 0x123 are both hexadecimal values
  2. *#3* indicates item number 3 (e.g., IRQ #3 means IRQ number 3)
  3. *#immed_12* refers to 12-bit immediate data
- Register bits:
  Typically used to illustrate a part of a value based on bit position; for example, bit[15:12] means bit number 15 down to 12.
- Register access types are as follows:
  1. R is Read only
  2. W is Write only
  3. R/W is Read or Write accessible
  4. R/Wc is Readable and clear by a Write access

The ARM Cortex™-M3 processor, the first of the Cortex generation of processors released by ARM in 2006, was primarily designed to target the 32-bit microcontroller market. The Cortex-M3 processor provides excellent performance at low gate count and comes with many new features previously available only in high-end processors. The Cortex-M3 addresses the requirements for the 32-bit embedded processor market in the following ways:

• Greater performance efficiency: allowing more work to be done without increasing the frequency or power requirements

• Low power consumption: enabling longer battery life, especially critical in portable products including wireless networking application

- Enhanced determinism: guaranteeing that critical tasks and interrupts are serviced as quickly as possible and in a known number of cycles
- Improved code density: ensuring that code fits in even the smallest memory footprints • Ease of use: providing easier programmability and debugging for the growing number of 8-bit and 16-bit users migrating to 32 bits
- Lower cost solutions: reducing 32-bit-based system costs close to those of legacy 8-bit and 16-bit devices and enabling low-end, 32-bit microcontrollers to be priced at less than US$1 for the first time

• Wide choice of development tools: from low-cost or free compilers to full-featured development suites from many development tool vendors

History

ARM was formed in 1990 as Advanced RISC Machines Ltd., a joint venture of Apple Computer, Acorn Computer Group, and VLSI Technology. In 1991, ARM introduced the ARM6 processor family, and VLSI became the initial licensee. Subsequently, additional companies, including Texas Instruments, NEC, Sharp, and ST Microelectronics, licensed the ARM processor designs, extending the applications of ARM processors into mobile phones, computer hard disks, personal digital assistants (PDAs), home entertainment systems, and many other consumer products.

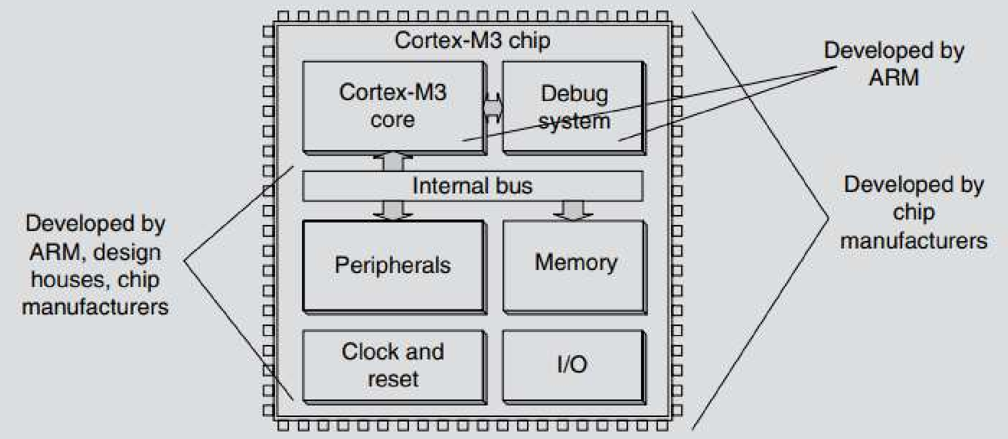


FIGURE 1.1

The Cortex-M3 Processor versus the Cortex-M3-Based MCU.

Over the past several years, ARM extended its product portfolio by diversifying its CPU development, which resulted in the architecture version 7 or v7. In this version, the architecture design is divided into three profiles:

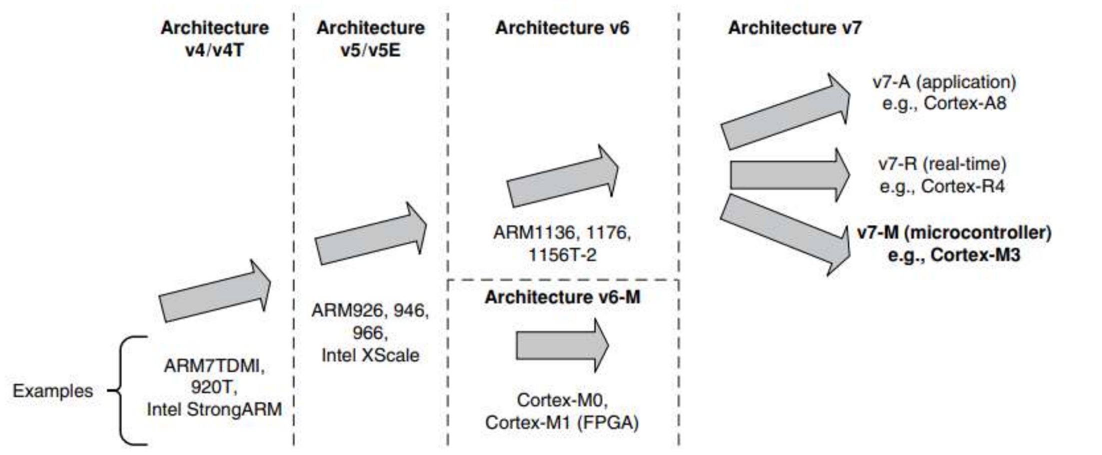
• The A profile is designed for high-performance open application platforms.

• The R profile is designed for high-end embedded systems in which real-time performance is needed. • The M profile is designed for deeply embedded microcontroller-type systems. Let's look at these profiles in a bit more detail:

• A Profile (ARMv7-A): Application processors which are designed to handle complex applications such as high-end embedded operating systems (OSs) (e.g., Symbian, Linux, and Windows Embedded). These processors requiring the highest processing power, virtual memory system support with memory management units (MMUs), and, optionally, enhanced Java support and a secure program execution environment. Example products include high-end mobile phones and electronic wallets for financial transactions.

• R Profile (ARMv7-R): Real-time, high-performance processors targeted primarily at the higher end of the real-time1 market—those applications, such as high-end breaking systems and hard drive controllers, in which high processing power and high reliability are essential and for which low latency is important.

• M Profile (ARMv7-M): Processors targeting low-cost applications in which processing efficiency is important and cost, power consumption, low interrupt latency, and ease of use are critical, as well as industrial control applications, including real-time control systems.



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
ARM926, 946, 966, Intel XScale
Examples
ARM7TDMI, 920T, Intel StrongARM
Cortex-M0, Cortex-M1 (FPGA)

FIGURE 1.2

The Evolution of ARM Processor Architecture.

The ARMv7-M architecture contains the following key areas:

• Programmer’s model

• Instruction set

• Memory model

• Debug architecture

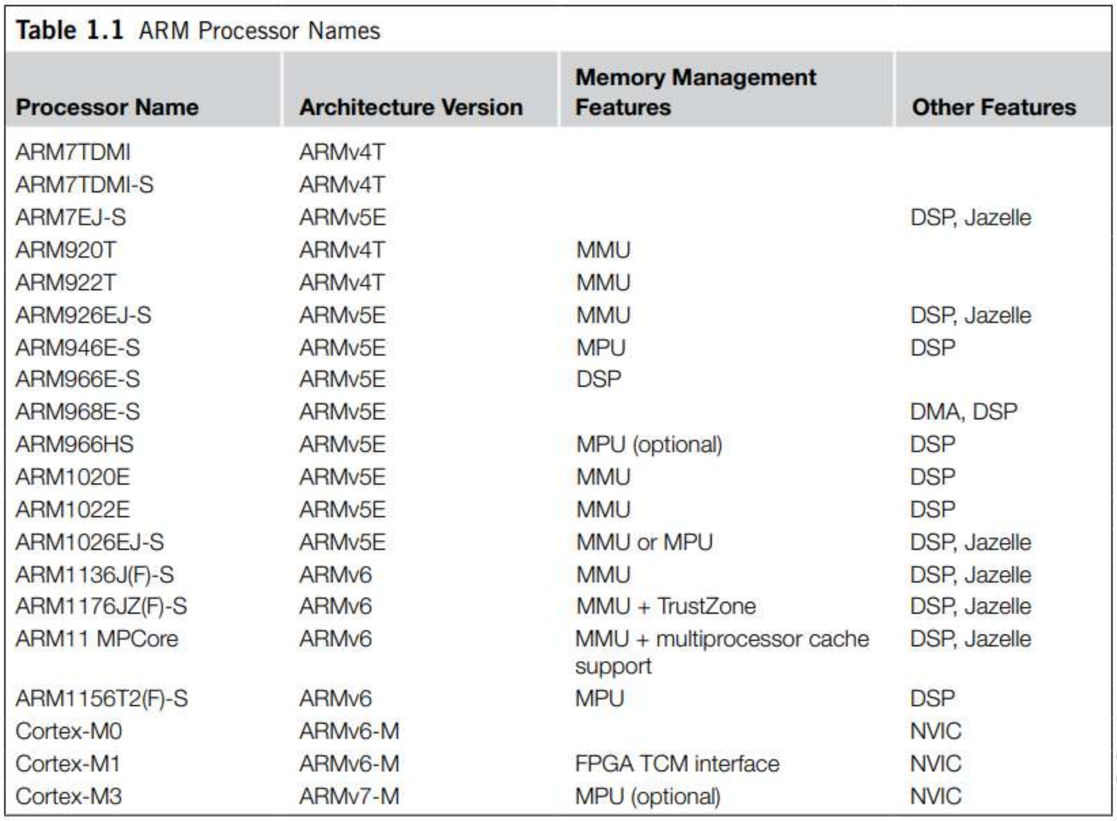


Table 1.1 ARM Processor Names

Processor Name	Architecture Version	Memory Management Features	Other Features
ARM7TDMI	ARMv4T		
ARM7TDMI-S	ARMv4T		
ARM7EJ-S	ARMv5E		DSP, Jazelle
ARM920T	ARMv4T	MMU	
ARM922T	ARMv4T	MMU	
ARM926EJ-S	ARMv5E	MMU	DSP, Jazelle
ARM946E-S	ARMv5E	MPU	DSP
ARM966E-S	ARMv5E	DSP	
ARM968E-S	ARMv5E		DMA, DSP
ARM966HS	ARMv5E	MPU (optional)	DSP
ARM1020E	ARMv5E	MMU	DSP
ARM1022E	ARMv5E	MMU	DSP
ARM1026EJ-S	ARMv5E	MMU or MPU	DSP, Jazelle
ARM1136J(F)-S	ARMv6	MMU	DSP, Jazelle
ARM1176JZ(F)-S	ARMv6	MMU + TrustZone	DSP, Jazelle
ARM11 MPCore	ARMv6	MMU + multiprocessor cache support	DSP, Jazelle
ARM1156T2(F)-S	ARMv6	MPU	DSP
Cortex-M0	ARMv6-M		NVIC
Cortex-M1	ARMv6-M	FPGA TCM interface	NVIC
Cortex-M3	ARMv7-M	MPU (optional)	NVIC

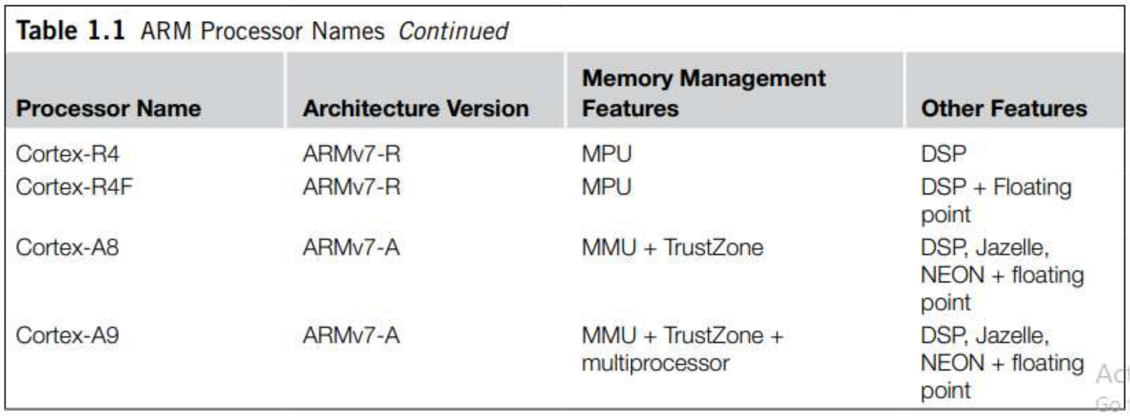


Table 1.1 ARM Processor Names *Continued*

Processor Name	Architecture Version	Memory Management Features	Other Features
Cortex-R4	ARMv7-R	MPU	DSP
Cortex-R4F	ARMv7-R	MPU	DSP + Floating point
Cortex-A8	ARMv7-A	MMU + TrustZone	DSP, Jazelle, NEON + floating point
Cortex-A9	ARMv7-A	MMU + TrustZone + multiprocessor	DSP, Jazelle, NEON + floating point

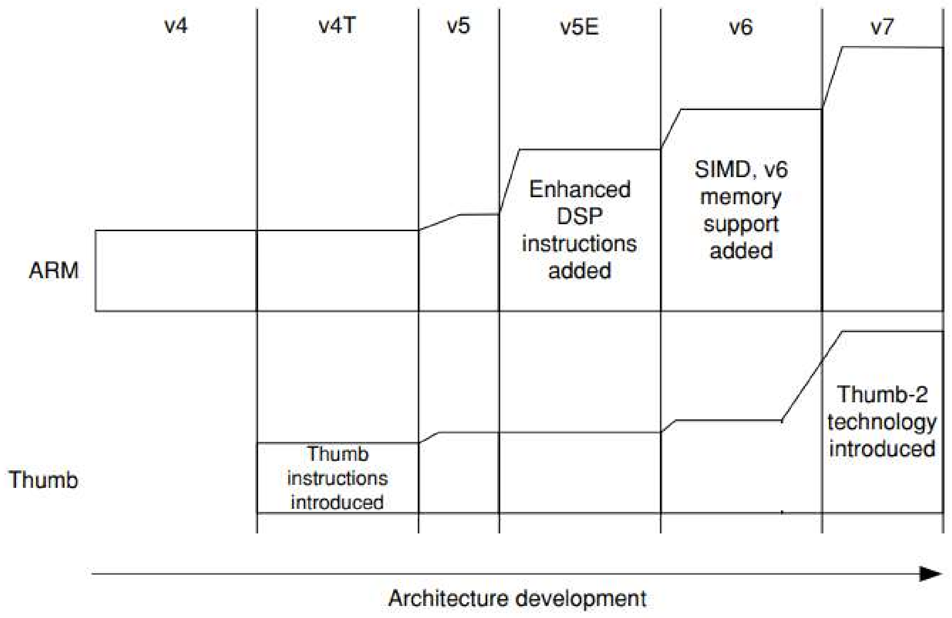

v4
v4T
v5
v5E
v6
v7
ARM
Enhanced DSP instructions added
SIMD, v6 memory support added
Thumb
Thumb instructions introduced
Thumb-2 technology introduced
Architecture development

FIGURE 1.3

Instruction Set Enhancement.

The Thumb-2 Technology and Instruction Set Architecture

The Thumb-23 technology extended the Thumb Instruction Set Architecture (ISA) into a highly efficient and powerful instruction set that delivers significant benefits in terms of ease of use, code size, and performance.

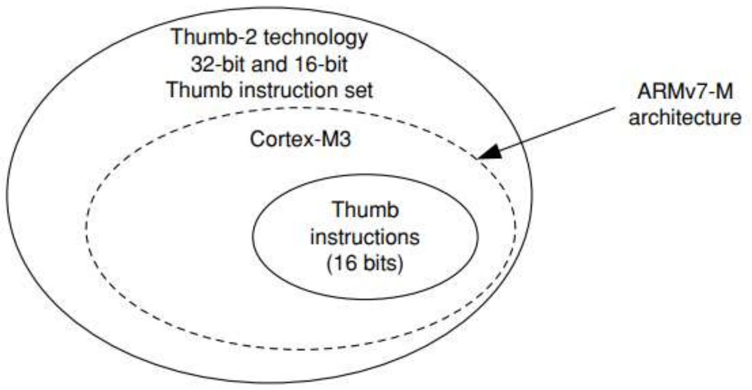

Thumb-2 technology 32-bit and 16-bit Thumb instruction set
Cortex-M3
Thumb instructions (16 bits)
ARMv7-M architecture

FIGURE 1.4

The Relationship between the Thumb Instruction Set in Thumb-2 Technology and the Traditional Thumb.

1.5 Cortex-M3 Processor Applications With its high performance and high code density and small silicon footprint, the Cortex-M3 processor is ideal for a wide variety of applications:

• Low-cost microcontrollers: The Cortex-M3 processor is ideally suited for low-cost microcontrollers, which are 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: Another ideal application for the Cortex-M3 processor is in the automotive industry. 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.

Cortex M3

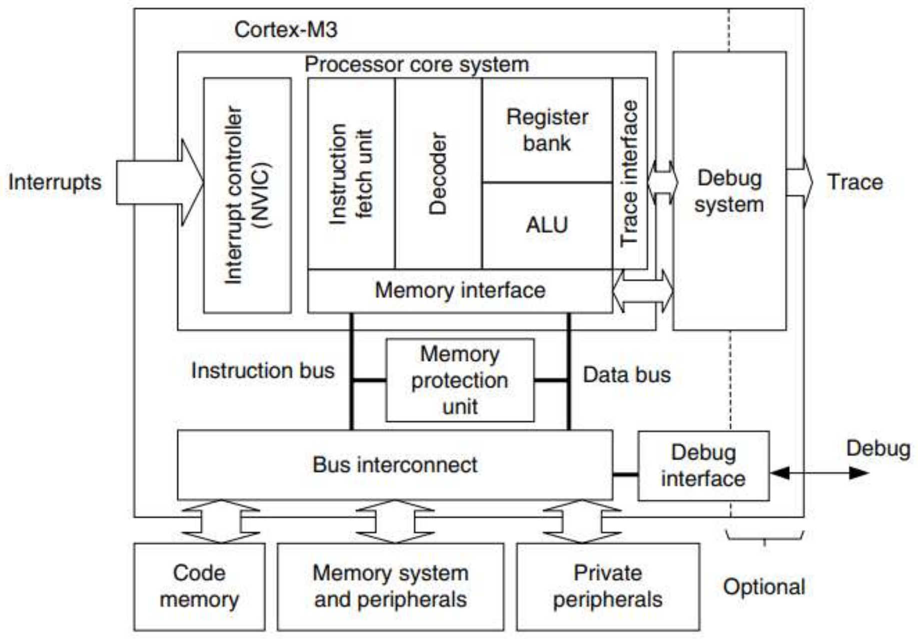


FIGURE 2.1

A Simplified View of the Cortex-M3.

Registers

1. R0–R12: General-Purpose Registers R0–R12 are 32-bit general-purpose registers for data operations. Some 16-bit Thumb® instructions can only access a subset of these registers (low registers, R0–R7).
2. R13: Stack Pointers The Cortex-M3 contains two stack pointers (R13). They are banked so that only one is visible at a time. The two stack pointers are as follows:
• Main Stack Pointer (MSP): The default stack pointer, used by the operating system (OS) kernel and exception handlers.
• Process Stack Pointer (PSP): Used by user application code. The lowest 2 bits of the stack pointers are always 0, which means they are always word aligned.

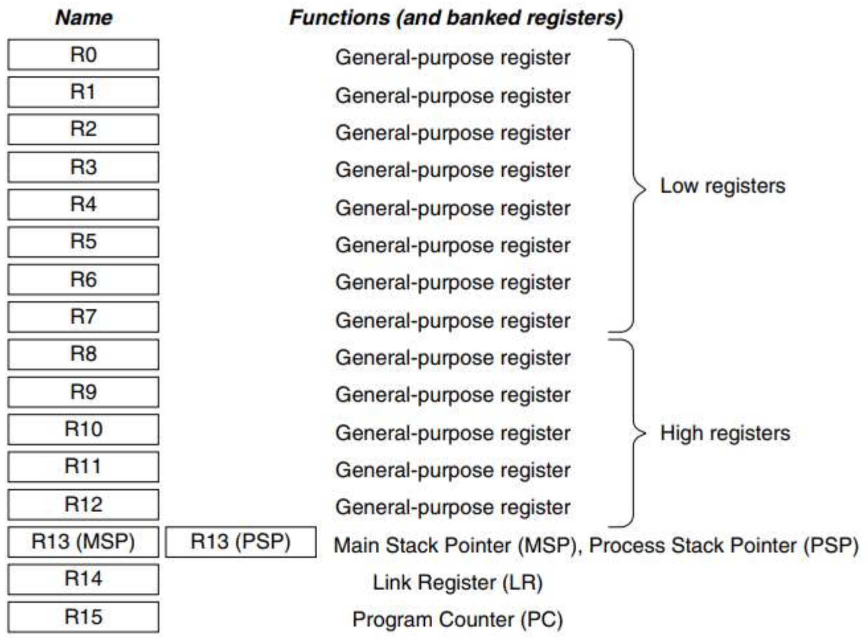


FIGURE 2.2
Registers in the Cortex-M3.

R14: The Link Register When a subroutine is called, the return address is stored in the link register.
R15: The Program Counter The program counter is the current program address. This register can be written to control the program flow.
Special Registers The Cortex-M3 processor also has a number of special registers. They are as follows:
• Program Status registers (PSRs)
• Interrupt Mask registers (PRIMASK, FAULTMASK, and BASEPRI)
• Control register (CONTROL) These registers have special functions and can be accessed only by special instructions. They cannot be used for normal data processing

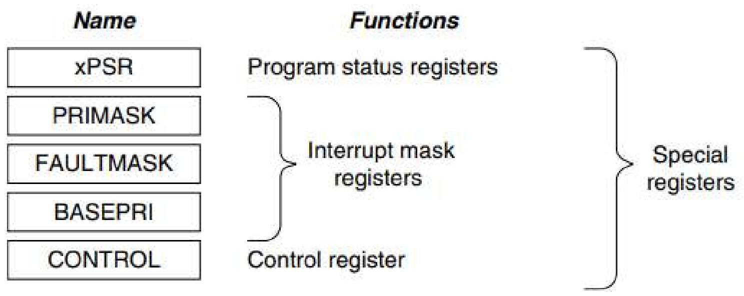


FIGURE 2.3

Special Registers in the Cortex-M3.

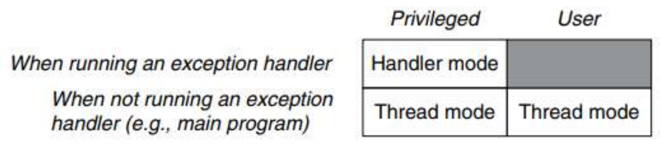
Table 2.1 Special Registers and Their Functions

Register	Function
xPSR	Provide arithmetic and logic processing flags (zero flag and carry flag execution status, and current executing interrupt number
PRIMASK	Disable all interrupts except the nonmaskable interrupt (NMI) and har
FAULTMASK	Disable all interrupts except the NMI
BASEPRI	Disable all interrupts of specific priority level or lower priority level
CONTROL	Define privileged status and stack pointer selection

For more information on these registers, see Chapter 3.

Operation Modes

The Cortex-M3 processor has two modes and two privilege levels. The operation modes (thread mode and handler mode) determine whether the processor is running a normal program or running an exception handler like an interrupt handler or system exception handler. The privilege levels (privileged level and user level) provide a mechanism for safeguarding memory accesses to critical regions as well as providing a basic security model. When the processor is running a main program (thread mode), it can be either in a privileged state or a user state, but exception handlers can only be in a privileged state. When the processor exits reset, it is in thread mode, with privileged access rights. In the privileged state, a program has access to all memory ranges (except when prohibited by MPU settings) and can use all supported instructions. Software in the privileged access level can switch the program into the user access level using the control register. When an exception takes place, the processor will always switch back to the privileged state and return to the previous state when exiting the exception handler. A user program cannot change back to the privileged state by writing to the control register. It has to go through an exception handler that programs the control register to switch the processor back into the privileged access level when returning to thread mode.



	Privileged	User
When running an exception handler	Handler mode	▨
When not running an exception handler (e.g., main program)	Thread mode	Thread mode

FIGURE 2.4

Operation Modes and Privilege Levels in Cortex-M3.

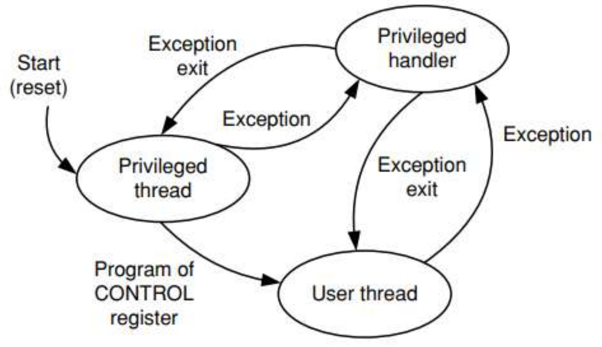


FIGURE 2.5

Allowed Operation Mode Transitions.

The Built-In Nested Vectored Interrupt Controller

The Cortex-M3 processor includes an interrupt controller called the Nested Vectored Interrupt Controller (NVIC). It is closely coupled to the processor core and provides a number of features as follows:

• Nested interrupt support

Nested Interrupt Support The NVIC provides nested interrupt support. All the external interrupts and most of the system exceptions can be programmed to different priority levels

• Vectored interrupt support

The Cortex-M3 processor has vectored interrupt support. When an interrupt is accepted, the starting address of the interrupt service routine (ISR) is located from a vector table in memory. There is no need to use software to determine and branch to the starting address of the ISR. Thus, it takes less time to process the interrupt request.

• Dynamic priority changes support

Priority levels of interrupts can be changed by software during run time. Interrupts that are being serviced are blocked from further activation until the ISR is completed, so their priority can be changed without risk of accidental re-entry.

• Reduction of interrupt latency

The Cortex-M3 processor also includes a number of advanced features to lower the interrupt latency. These include automatic saving and restoring some register contents, reducing delay in switching from one ISR to another, and handling of late arrival interrupts.

• Interrupt masking

Interrupts and system exceptions can be masked based on their priority level or masked completely using the interrupt masking registers BASEPRI, PRIMASK, and FAULTMASK. They can be used to ensure that time-critical tasks can be finished on time without being interrupted.

The Cortex-M3 has a predefined memory map. This allows the built-in peripherals, such as the interrupt controller and the debug components, to be accessed by simple memory access instructions. Thus, most system features are accessible in C program code. The predefined memory map also allows the Cortex-M3 processor to be highly optimized for speed and ease of integration in system-on-a-chip (SoC) designs. Overall, the 4 GB memory space can be divided into ranges as shown in Figure 2.6.

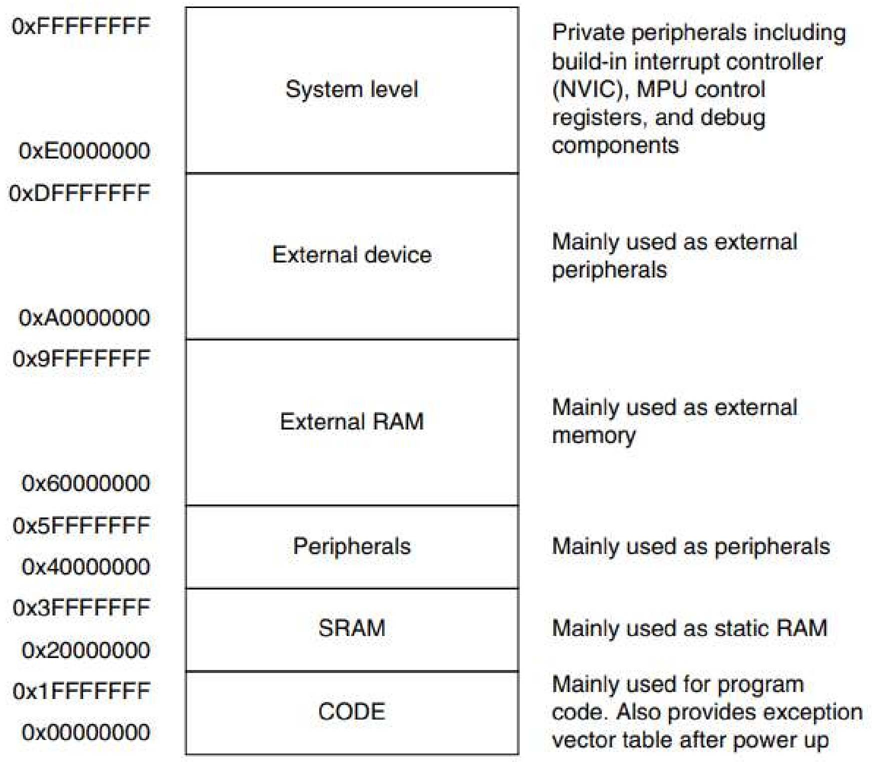

0xFFFFFFFF
0xE0000000
System level
Private peripherals including build-in interrupt controller (NVIC), MPU control registers, and debug components
0xDFFFFFFF
0xA0000000
External device
Mainly used as external peripherals
0x9FFFFFFF
0x60000000
External RAM
Mainly used as external memory
0x5FFFFFFF
0x40000000
Peripherals
Mainly used as peripherals
0x3FFFFFFF
0x20000000
SRAM
Mainly used as static RAM
0x1FFFFFFF
0x00000000
CODE
Mainly used for program code. Also provides exception vector table after power up

FIGURE 2.6

The Cortex-M3 Memory Map.

The Bus Interface There are several bus interfaces on the Cortex-M3 processor. They allow the Cortex-M3 to carry instruction fetches and data accesses at the same time. The main bus interfaces are as follows:

• Code memory buses

The code memory region access is carried out on the code memory buses, which physically consist of two buses, one called I-Code and other called D-Code. These are optimized for instruction fetches for best instruction execution speed.

• System bus

The system bus is used to access memory and peripherals. This provides access to the Static Random Access Memory (SRAM), peripherals, external RAM, external devices, and part of the systemlevel memory regions.

• Private peripheral bus

The private peripheral bus provides access to a part of the system-level memory dedicated to private peripherals, such as debugging components.

MPU

The Cortex-M3 has an optional MPU. This unit allows access rules to be set up for privileged access and user program access. When an access rule is violated, a fault exception is generated, and the fault exception handler will be able to analyze the problem and correct it, if possible. The MPU can be used in various ways. In common scenarios, the OS can set up the MPU to protect data use by the OS kernel and other privileged processes to be protected from untrusted user programs. The MPU can also be used to make memory regions read-only, to prevent accidental erasing of data or to isolate memory regions between different tasks in a multitasking system. Overall, it can help make embedded systems more robust and reliable. The MPU feature is optional and is determined during the implementation stage of the microcontroller or SoC design.

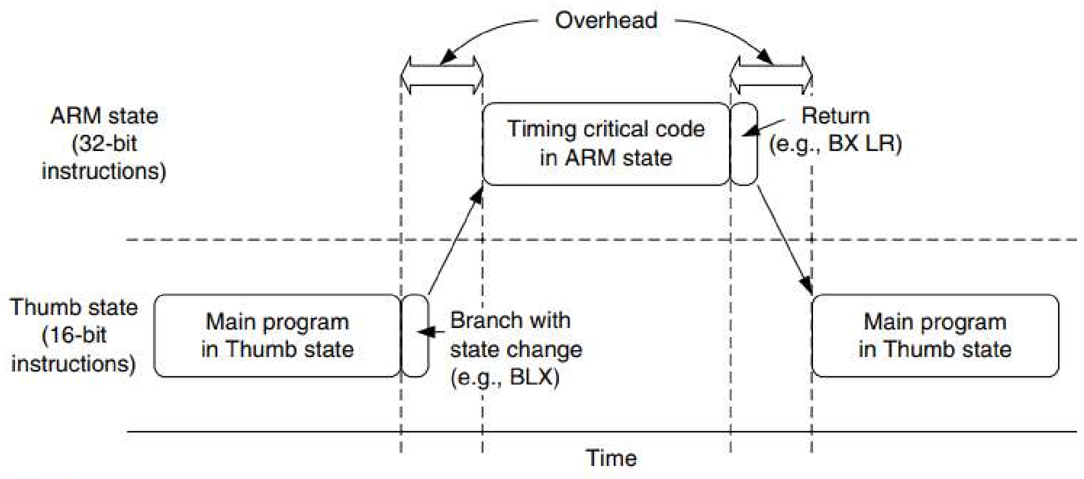

Overhead
ARM state (32-bit instructions)
Timing critical code in ARM state
Return (e.g., BX LR)
Thumb state (16-bit instructions)
Main program in Thumb state
Branch with state change (e.g., BLX)
Main program in Thumb state
Time

FIGURE 2.7

Switching between ARM Code and Thumb Code in Traditional ARM Processors Such as the ARM7.

The Cortex-M3 processor has a number of interesting and powerful instructions. Here are a few examples:

• UFBX, BFI, and BFC: Bit field extract, insert, and clear instructions • UDIV and SDIV: Unsigned and signed divide instructions

• WFE, WFI, and SEV: Wait-For-Event, Wait-For-Interrupts, and Send-Event; these allow the processor to enter sleep mode and to handle task synchronization on multiprocessor systems

• MSR and MRS: Move to special register from general-purpose register and move special register to general-purpose register; for access to the special registers

Interrupts and Exceptions

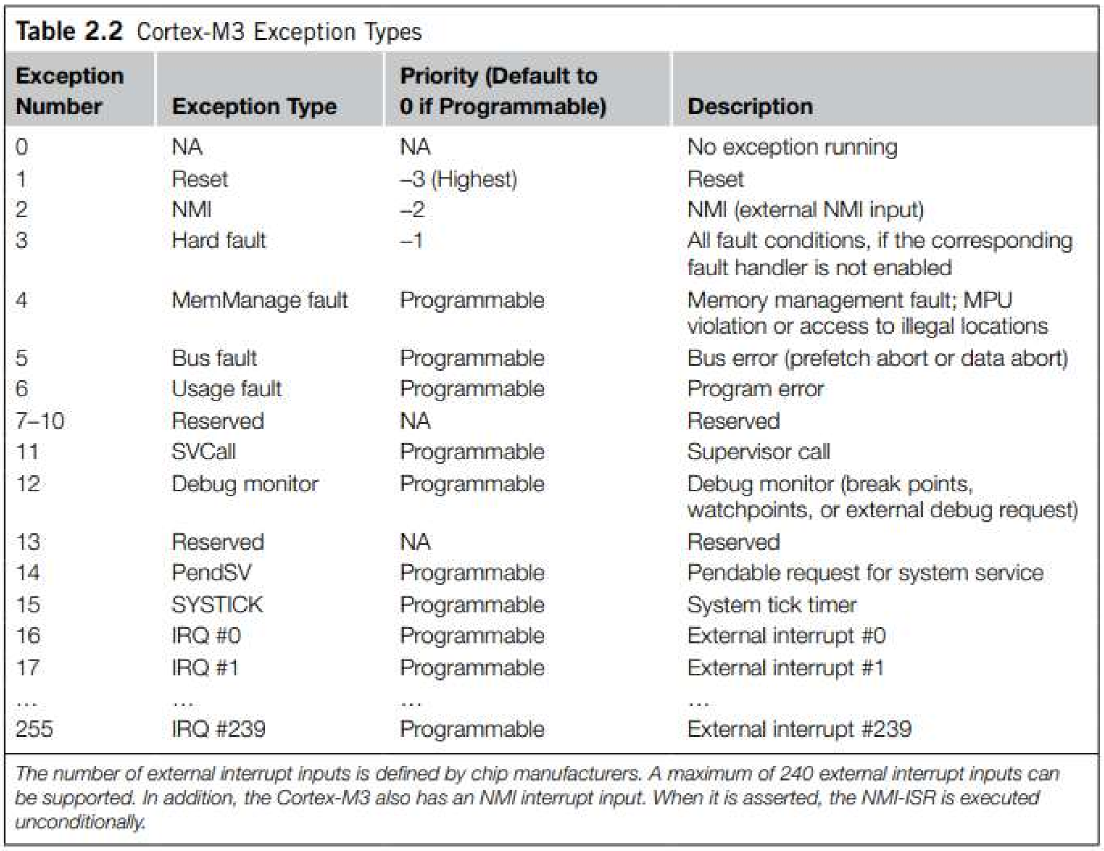


Table 2.2 Cortex-M3 Exception Types

Exception Number	Exception Type	Priority (Default to 0 if Programmable)	Description
0	NA	NA	No exception running
1	Reset	−3 (Highest)	Reset
2	NMI	−2	NMI (external NMI input)
3	Hard fault	−1	All fault conditions, if the corresponding fault handler is not enabled
4	MemManage fault	Programmable	Memory management fault; MPU violation or access to illegal locations
5	Bus fault	Programmable	Bus error (prefetch abort or data abort)
6	Usage fault	Programmable	Program error
7–10	Reserved	NA	Reserved
11	SVCall	Programmable	Supervisor call
12	Debug monitor	Programmable	Debug monitor (break points, watchpoints, or external debug request)
13	Reserved	NA	Reserved
14	PendSV	Programmable	Pendable request for system service
15	SYSTICK	Programmable	System tick timer
16	IRQ #0	Programmable	External interrupt #0
17	IRQ #1	Programmable	External interrupt #1
...	...	...	...
255	IRQ #239	Programmable	External interrupt #239

The number of external interrupt inputs is defined by chip manufacturers. A maximum of 240 external interrupt inputs can be supported. In addition, the Cortex-M3 also has an NMI interrupt input. When it is asserted, the NMI-ISR is executed unconditionally.

Debugging Support

The Cortex-M3 processor includes a number of debugging features, such as program execution controls, including halting and stepping, instruction breakpoints, data watchpoints, registers and memory accesses, profiling, and traces. The debugging hardware of the Cortex-M3 processor is based on the CoreSight™ architecture. Unlike traditional ARM processors, the CPU core itself does not have a Joint Test Action Group (JTAG) interface. Instead, a debug interface module is decoupled from the core, and a bus interface called the Debug Access Port (DAP) is provided at the core level. Through this bus interface, external debuggers can access control registers to debug hardware as well as system memory, even when the processor is running. The control of this bus interface is carried out by a Debug Port (DP) device. The DPs currently available are the Serial-Wire JTAG Debug Port (SWJ-DP) (supports the traditional JTAG protocol as well as the Serial-Wire protocol) or the SW-DP (supports the Serial-Wire protocol only). A JTAG-DP module from the ARM CoreSight product family can also be used. Chip manufacturers can choose to attach one of these DP modules to provide the debug interface.

Characteristics Summary

The benefits and advantages are summarized in this section.

2.11.1 High Performance

The Cortex-M3 processor delivers high performance in microcontroller products:

• Many instructions, including multiply, are single cycle. Therefore, the Cortex-M3 processor outperforms most microcontroller products.

• Separate data and instruction buses allow simultaneous data and instruction accesses to be

performed.

• The Thumb-2 instruction set makes state switching overhead history. There's no need to spend time switching between the ARM state (32 bits) and the Thumb state (16 bits), so instruction cycles and program size are reduced. This feature has also simplified software development, allowing faster time to market, and easier code maintenance.

• The Thumb-2 instruction set provides extra flexibility in programming. Many data operations can now be simplified using shorter code. This also means that the Cortex-M3 has higher code density and reduced memory requirements.

• Instruction fetches are 32 bits. Up to two instructions can be fetched in one cycle. As a result, there's more available bandwidth for data transfer.

• The Cortex-M3 design allows microcontroller products to operate at high clock frequency (over 100 MHz in modern semiconductor manufacturing processes). Even running at the same frequency as most other microcontroller products, the Cortex-M3 has a better clock per instruction (CPI) ratio. This allows more work per MHz or designs can run at lower clock frequency for lower power consumption.

2.11.2 Advanced Interrupt-Handling Features

The interrupt features on the Cortex-M3 processor are easy to use, very flexible, and provide high interrupt processing throughput:

• The built-in NVIC supports up to 240 external interrupt inputs. The vectored interrupt feature considerably reduces interrupt latency because there is no need to use software to determine which IRQ handler to serve. In addition, there is no need to have software code to set up nested interrupt support.

2.11 Characteristics Summary 23

• The Cortex-M3 processor automatically pushes registers R0–R3, R12, Link register (LR), PSR, and PC in the stack at interrupt entry and pops them back at interrupt exit. This reduces the IRQ handling latency and allows interrupt handlers to be normal C functions.

• Interrupt arrangement is extremely flexible because the NVIC has programmable interrupt priority control for each interrupt. A minimum of eight levels of priority are supported, and the priority can be changed dynamically.

• Interrupt latency is reduced by special optimization, including late arrival interrupt acceptance and tail-chain interrupt entry.

• Some of the multicycle operations, including Load-Multiple (LDM), Store-Multiple (STM), PUSH, and POP, are now interruptible.

- On receipt of an NMI request, immediate execution of the NMI handler is guaranteed unless the system is completely locked up. NMI is very important for many safety-critical applications.

2.11.3 Low Power Consumption

The Cortex-M3 processor is suitable for various low-power applications:

- The Cortex-M3 processor is suitable for low-power designs because of the low gate count.

- It has power-saving mode support (SLEEPING and SLEEPDEEP). The processor can enter sleep mode using WFI or WFE instructions. The design has separated clocks for essential blocks, so clocking circuits for most parts of the processor can be stopped during sleep.

- The fully static, synchronous, synthesizable design makes the processor easy to be manufactured using any low power or standard semiconductor process technology.

2.11.4 System Features

The Cortex-M3 processor provides various system features making it suitable for a large number of applications:

- The system provides bit-band operation, byte-invariant big endian mode, and unaligned data access support.

- Advanced fault-handling features include various exception types and fault status registers, making it easier to locate problems.

- With the shadowed stack pointer, stack memory of kernel and user processes can be isolated. With the optional MPU, the processor is more than sufficient to develop robust software and reliable products.

2.11.5 Debug Supports

The Cortex-M3 processor includes comprehensive debug features to help software developers design their products:

- Supports JTAG or Serial-Wire debug interfaces

- Based on the CoreSight debugging solution, processor status or memory contents can be accessed even when the core is running

- Built-in support for six breakpoints and four watchpoints

- New debugging features, including fault status registers, new fault exceptions, and Flash Patch operations, make debugging much easier.

CISC

CISC stands for Complex Instruction Set Computer. It comprises a complex instruction set. It incorporates a variable-length instruction format. Instructions that require register operands can take only two bytes.

The instructions that require two memory addresses can take five bytes to include the complete instruction code. Thus, CISC has a variable-length encoding of instructions and the execution of instructions may take a varying number of clock cycles. The CISC processor provides direct manipulation of operands that are in memory.

RISC

Many of today's most popular 32-bit microcontrollers use **RISC** technology. Unlike CISC processors, RISC engines generally execute each instruction in a single clock cycle, which typically results in faster execution than on a CISC processor with the same clock speed.

Thumb Instruction Set

The Thumb instruction set consists of 16-bit instructions that act as a compact shorthand for a subset of the 32-bit instructions of the standard ARM. Every Thumb instruction could instead be executed via the equivalent 32-bit ARM instruction.