1 Introduction to Embedded Systems

LEARNING OBJECTIVES

- LO 1 Know what an embedded system is
- LO 2 Differentiate between embedded and general computing systems
- LO 3 Underline the history of embedded systems
- LO 4 Classify embedded systems based on performance, complexity and the era in which they evolved
- LO 5 Explain the domains and areas of applications of embedded systems
- LO 6 Identify the different purposes of embedded systems
- LO 7 Analyse a real life example on the bonding of embedded technology with human life

Our day-to-day life is becoming more and more dependent on "embedded systems" and digital technologies. Embedded technologies are bonding into our daily activities even without our knowledge. Do you know the fact that the refrigerator, washing machine, microwave oven, air conditioner, television, DVD players, and music systems that we use in our home are built around an embedded system? You may be traveling by a 'Honda' or a 'Toyota' or a 'Ford' vehicle, but have you ever thought of the genius players working behind the special features and security systems offered by the vehicle to you? It is nothing but an intelligent embedded system. In your vehicle itself the presence of specialised embedded systems vary from intelligent head lamp controllers, engine controllers and ignition control systems to complex air bag control systems to protect you from a severe accident. People experience the power of embedded systems and enjoy the features and comfort provided by them. Most of us are totally unaware or ignorant of the intelligent embedded systems giving us so much comfort and security. Embedded systems are like reliable servants–they don't like to reveal their identity and neither they complain about their workloads to their owners or bosses. They are always working behind the scenes and are dedicated to their assigned task till their last breath. This book gives you an overview of embedded systems, the various steps involved in their design and development and the major domains where they are deployed.

1.1 WHAT IS AN EMBEDDED SYSTEM?

LO 1 Know what an embedded system is

An embedded system is an electronic/electro-mechanical system designed to perform a specific function and is a combination of both hardware and firmware (software).

Every embedded system is unique, and the hardware as well as the firmware is highly specialised to the application domain. Embedded systems are becoming an inevitable part of any product or equipment in all fields including household appliances, telecommunications, medical equipment, industrial control, consumer products, etc.

1.2 EMBEDDED SYSTEMS vs. GENERAL COMPUTING SYSTEMS

LO 2 Differentiate between embedded and general computing systems

The computing revolution began with the general purpose computing requirements. Later it was realised that the general computing requirements are not sufficient for the embedded computing requirements. The embedded computing requirements demand 'something special' in terms of response to stimuli, meeting the computational deadlines, power efficiency, limited memory availability, etc.

Let's take the case of your personal computer, which may be either a desktop PC or a laptop PC or a tablet PC. It is built around a general purpose processor like an Intel® Celeron/Core M or a Duo/Quad* core or an AMD A-Series processor and is designed to support a set of multiple peripherals like multiple USB 3.0 ports, Wi-Fi, ethernet, video port, IEEE1394, SD/CF/MMC external interfaces, Bluetooth, etc and with additional interfaces like a DVD read/writer, on-board Hard Disk Drive (HDD), gigabytes of RAM, etc. You can load any supported operating system (like Windows® 8.X/10, or Red Hat Linux/Ubuntu Linux, UNIX etc) into the hard disk of your PC. You can write or purchase a multitude of applications for your PC and can use your PC for running a large number of applications (like printing your dear's photo using a printer device connected to the PC and printer software, creating a document using Microsoft® Office Word tool, etc.) Now, let us think about the DVD player you use for playing DVD movies. Is it possible for you to change the operating system of your DVD? Is it possible for you to write an application and download it to your DVD player for executing? Is it possible for you to add a printer software to your DVD player and connect a printer to your DVD player to take a printout? Is it possible for you to change the functioning of your DVD player to a television by changing the embedded software? The answers to all these questions are 'NO'. Can you see any general purpose interface like Bluetooth or Wi-Fi on your DVD player? Of course 'NO'. The only interface you can find out on the DVD player is the interface for connecting the DVD player with the display screen and one for controlling the DVD player through a remote (May be an IR or any other specific wireless interface). Indeed your DVD player is an embedded system designed specifically for decoding digital video and generating a video signal as output to your TV or any other display screen which supports the display interface supported by the DVD Player. Let us summarise our findings from the comparison of embedded system and general purpose computing system with the help of a table:

General Purpose Computing System	Embedded System
A system which is a combination of a generic hardware and a General Purpose Operating System for executing a variety of applications	A system which is a combination of special purpose hardware and embedded OS for executing a specific set of applications

(Contd.)

*The illustration given here is based on the processor details available till Dec 2015. Since processor technology is undergoing rapid changes, the processor names mentioned here may not be relevant in future.

General Purpose Computing System	Embedded System
Contains a General Purpose Operating System (GPOS)	May or may not contain an operating system for functioning
Applications are alterable (programmable) by the user (It is possible for the end user to re-install the operating system, and also add or remove user applications)	The firmware of the embedded system is pre-programmed and it is non-alterable by the end-user (There may be exceptions for systems supporting OS kernel image flashing through special hardware settings)
Performance is the key deciding factor in the selection of the system. Always, 'Faster is Better'	Application-specific requirements (like performance, power requirements, memory usage, etc.) are the key deciding factors
Less/not at all tailored towards reduced operating power requirements, options for different levels of power management.	Highly tailored to take advantage of the power saving modes supported by the hardware and the operating system
Response requirements are not time-critical	For certain category of embedded systems like mission critical systems, the response time requirement is highly critical
Need not be deterministic in execution behaviour	Execution behaviour is deterministic for certain types of embedded systems like 'Hard Real Time' systems

However, the demarcation between desktop systems and embedded systems in certain areas of embedded applications are shrinking in certain contexts. Smart phones are typical examples of this. Nowadays smart phones are available with RAM 2 to 3 GB and users can extend most of their desktop applications to the smart phones and it invalidates the clause "Embedded systems are designed for a specific application" from the characteristics of the embedded system for the mobile embedded device category. However, smart phones come with a built-in operating system and it is not modifiable by the end user. It makes the clause: "The firmware of the embedded system is unalterable by the end user", still a valid clause in the mobile embedded device category.

1.3 HISTORY OF EMBEDDED SYSTEMS

LO 3 Underline the history of embedded systems

Embedded systems were in existence even before the IT revolution. In the olden days, embedded systems were built around the old vacuum tube and transistor technologies and the embedded algorithm was developed in low level languages. Advances in semiconductor and nano-technology and IT revolution gave way to the development of miniature embedded systems. The first recognised modern embedded system is the Apollo Guidance Computer (AGC) developed by the MIT Instrumentation Laboratory for the lunar expedition. They ran the inertial guidance systems of both the Command Module (CM) and the Lunar Excursion Module (LEM). The Command Module was designed to encircle the moon while the Lunar Module and its crew were designed to go down to the moon surface and land there safely. The Lunar Module featured in total 18 engines. There were 16 reaction control thrusters, a descent engine and an ascent engine. The descent engine was 'designed to' provide thrust to the lunar module out of the lunar orbit and land it safely on the moon. MIT's original design was based on 4K words of fixed memory (Read Only Memory) and 256 words of erasable memory (Random Access Memory). By June 1963, the figures reached 10K of fixed and 1K of erasable memory. The final configuration was 36K words of fixed memory and 2K words of erasable memory. The clock frequency of the first microchip proto model used in AGC

was 1.024 MHz and it was derived from a 2.048 MHz crystal clock. The computing unit of AGC consisted of approximately 11 instructions and 16 bit word logic. Around 5000 ICs (3-input NOR gates, RTL logic) supplied by Fairchild Semiconductor were used in this design. The user interface unit of AGC is known as DSKY (display/keyboard). DSKY looked like a calculator type keypad with an array of numerals. It was used for inputting the commands to the module numerically.

The first mass-produced embedded system was the guidance computer for the Minuteman-I missile in 1961. It was the '*Autonetics D-17*' guidance computer, built using discrete transistor logic and a hard-disk for main memory. The first integrated circuit was produced in September 1958 but computers using them didn't begin to appear until 1963. Some of their early uses were in embedded systems, notably used by NASA for the Apollo Guidance Computer and by the US military in the Minuteman-II intercontinental ballistic missile.

1.4 CLASSIFICATION OF EMBEDDED SYSTEMS

LO 4 Classify embedded systems based on performance, complexity and the era in which they evolved

It is possible to have a multitude of classifications for embedded systems, based on different criteria. Some of the criteria used in the classification of embedded systems are as follows:

(1) Based on generation
(2) Complexity and performance requirements
(3) Based on deterministic behaviour
(4) Based on triggering.

The classification based on deterministic system behaviour is applicable for 'Real Time' systems. The application/task execution behaviour for an embedded system can be either deterministic or non- deterministic. Based on the execution behaviour, Real Time embedded systems are classified into *Hard* and *Soft*. We will discuss about hard and soft real time systems in a later chapter. Embedded Systems which are 'Reactive' in nature (Like process control systems in industrial control applications) can be classified based on the trigger. Reactive systems can be either *event triggered* or *time triggered*.

1.4.1 Classification Based on Generation

This classification is based on the order in which the embedded processing systems evolved from the first version to where they are today. As per this criterion, embedded systems can be classified into the following:

1.4.1.1 First Generation The early embedded systems were built around 8bit microprocessors like 8085 and Z80, and 4bit microcontrollers. Simple in hardware circuits with firmware developed in Assembly code. Digital telephone keypads, stepper motor control units etc. are examples of this.

1.4.1.2 Second Generation These are embedded systems built around 16bit microprocessors and 8 or 16 bit microcontrollers, following the first generation embedded systems. The instruction set for the second generation processors/controllers were much more complex and powerful than the first generation processors/controllers. Some of the second generation embedded systems contained embedded operating systems for their operation. Data Acquisition Systems, SCADA systems, etc. are examples of second generation embedded systems.

1.4.1.3 Third Generation With advances in processor technology, embedded system developers started making use of powerful 32bit processors and 16bit microcontrollers for their design. A new concept of

application and domain specific processors/controllers like Digital Signal Processors (DSP) and Application Specific Integrated Circuits (ASICs) came into the picture. The instruction set of processors became more complex and powerful and the concept of instruction pipelining also evolved. The processor market was flooded with different types of processors from different vendors. Processors like Intel Pentium, Motorola 68K, etc. gained attention in high performance embedded requirements. Dedicated embedded real time and general purpose operating systems entered into the embedded market. Embedded systems spread its ground to areas like robotics, media, industrial process control, networking, etc.

1.4.1.4 Fourth Generation The advent of System on Chips (SoC), reconfigurable processors and multicore processors are bringing high performance, tight integration and miniaturisation into the embedded device market. The SoC technique implements a total system on a chip by integrating different functionalities with a processor core on an integrated circuit. We will discuss about SoCs in a later chapter. The fourth generation embedded systems are making use of high performance real time embedded operating systems for their functioning. Smart phone devices, mobile internet devices (MIDs), etc. are examples of fourth generation embedded systems.

1.4.1.5 What Next? The processor and embedded market is highly dynamic and demanding. So 'what will be the next smart move in the next embedded generation?' Let's wait and see.

1.4.2 Classification Based on Complexity and Performance

This classification is based on the complexity and system performance requirements. According to this classification, embedded systems can be grouped into the following:

1.4.2.1 Small-Scale Embedded Systems Embedded systems which are simple in application needs and where the performance requirements are not time critical fall under this category. An electronic toy is a typical example of a small-scale embedded system. Small-scale embedded systems are usually built around low performance and low cost 8 or 16 bit microprocessors/microcontrollers. A small-scale embedded system may or may not contain an operating system for its functioning.

1.4.2.2 Medium-Scale Embedded Systems Embedded systems which are slightly complex in hardware and firmware (software) requirements fall under this category. Medium-scale embedded systems are usually built around medium performance, low cost 16 or 32 bit microprocessors/microcontrollers or digital signal processors. They usually contain an embedded operating system (either general purpose or real time operating system) for functioning.

1.4.2.3 Large-Scale Embedded Systems/Complex Systems Embedded systems which involve highly complex hardware and firmware requirements fall under this category. They are employed in mission critical applications demanding high performance. Such systems are commonly built around high performance 32 or 64 bit RISC processors/controllers or Reconfigurable System on Chip (RSoC) or multi-core processors and programmable logic devices. They may contain multiple processors/controllers and co-units/hardware accelerators for offloading the processing requirements from the main processor of the system. Decoding/encoding of media, cryptographic function implementation, etc. are examples for processing requirements which can be implemented using a co-processor/hardware accelerator. Complex embedded systems usually contain a high performance Real Time Operating System (RTOS) for task scheduling, prioritisation, and management.

1.5 MAJOR APPLICATION AREAS OF EMBEDDED SYSTEMS

LO 5 Explain the domains and areas of applications of embedded systems

We are living in a world where embedded systems play a vital role in our day-to-day life, starting from home to the computer industry, where most of the people find their job for a livelihood. Embedded technology has acquired a new dimension from its first generation model, the Apollo guidance computer, to the latest radio navigation system combined with in-car entertainment technology and the wearable computing devices (Apple watch, Microsoft Band, Fitbit fitness trackers etc.). The application areas and the products in the embedded domain are countless. A few of the important domains and products are listed below:

(1) *Consumer electronics:* Camcorders, cameras, etc.
(2) *Household appliances:* Television, DVD players, washing machine, fridge, microwave oven, etc.
(3) *Home automation and security systems:* Air conditioners, sprinklers, intruder detection alarms, closed circuit television cameras, fire alarms, etc.
(4) *Automotive industry:* Anti-lock breaking systems (ABS), engine control, ignition systems, automatic navigation systems, etc.
(5) *Telecom:* Cellular telephones, telephone switches, handset multimedia applications, etc.
(6) *Computer peripherals:* Printers, scanners, fax machines, etc.
(7) *Computer networking systems:* Network routers, switches, hubs, firewalls, etc.
(8) *Healthcare:* Different kinds of scanners, EEG, ECG machines etc.
(9) *Measurement & Instrumentation:* Digital multimeters, digital CROs, logic analysers PLC systems, etc.
(10) *Banking & Retail:* Automatic teller machines (ATM) and currency counters, point of sales (POS)
(11) *Card Readers:* Barcode, smart card readers, hand held devices, etc.
(12) *Wearable Devices:* Health and Fitness Trackers, Smartphone Screen extension for notifications, etc.
(13) Cloud Computing and Internet of Things (IOT)

1.6 PURPOSE OF EMBEDDED SYSTEMS

LO 6 Identify the different purposes of embedded systems

As mentioned in the previous section, embedded systems are used in various domains like consumer electronics, home automation, telecommunications, automotive industry, healthcare, control & instrumentation, retail and banking applications, etc. Within the domain itself, according to the application usage context, they may have different functionalities. Each embedded system is designed to serve the purpose of any one or a combination of the following tasks:

(1) Data collection/Storage/Representation
(2) Data communication
(3) Data (signal) processing
(4) Monitoring
(5) Control
(6) Application specific user interface

1.6.1 Data Collection/Storage/Representation

Embedded systems designed for the purpose of data collection performs acquisition of data from the external world. Data collection is usually done for storage, analysis, manipulation, and transmission. The term "data" refers to all kinds of information, viz. text, voice, image, video, electrical signals and any other measurable quantities. Data can be either analog (continuous) or digital (discrete). Embedded systems with analog data capturing techniques collect data directly in the form of analog signals whereas embedded systems with digital data collection mechanism converts the analog signal to corresponding digital signal using analog to digital (A/D) converters and then collects the binary equivalent of the analog data. If the data is digital, it can be directly captured without any additional interface by digital embedded systems.

A typical embedded system (Fig. 2.1) contains a single chip controller, which acts as the master brain of the system. The controller can be a Microprocessor (e.g. *Intel 8085*) or a microcontroller (e.g. Atmel AT89C51) or a Field Programmable Gate Array (FPGA) device (e.g. *Xilinx Spartan*) or a Digital Signal Processor (DSP) (e.g. *Blackfin® Processors* from Analog Devices) or an Application Specific Integrated Circuit (ASIC)/ Application Specific Standard Product (ASSP) (e.g. ADE7760 Single Phase Energy Metreing IC from) Analog Devices for energy metering applications).

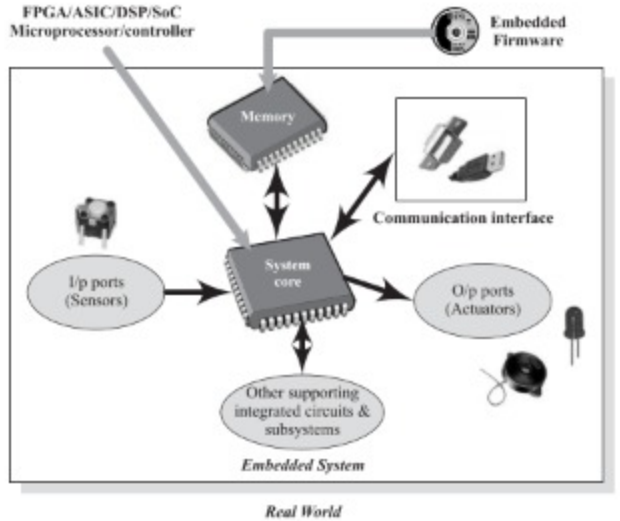


Fig. 2.1 Elements of an embedded system

Embedded hardware/software systems are basically designed to regulate a physical variable or to manipulate the state of some devices by sending some control signals to the Actuators or devices connected to the O/p ports of the system, in response to the input signals provided by the end users or Sensors which are connected to the input ports. Hence an embedded system can be viewed as a reactive system. The control is achieved by processing the information coming from the sensors and user interfaces, and controlling some actuators that regulate the physical variable.

Key boards, push button switches, etc. are examples for common user interface input devices whereas LEDs, liquid crystal displays, piezoelectric buzzers, etc. are examples for common user interface output devices for a typical embedded system. It should be noted that it is not necessary that all embedded systems should incorporate these I/O user interfaces. It solely depends on the type of the application for which the embedded system is designed. For example, if the embedded system is designed for any handheld application, such as a mobile handset application, then the system should contain user interfaces like a keyboard for performing input operations and display unit for providing users the status of various activities in progress.

Some embedded systems do not require any manual intervention for their operation. They automatically sense the variations in the input parameters in accordance with the changes in the real world, to which they are interacting through the sensors which are connected to the input port of the system. The sensor information is passed to the processor after signal conditioning and digitisation. Upon receiving the sensor data the processor or brain of the embedded system performs some predefined operations with the help of the firmware embedded in the system and sends some actuating signals to the actuator connected to the output port of the embedded system, which in turn acts on the controlling variable to bring the controlled variable to the desired level to make the embedded system work in the desired manner.

The Memory of the system is responsible for holding the control algorithm and other important configuration details. For most of embedded systems, the memory for storing the algorithm or configuration data is of fixed type, which is a kind of Read Only Memory (ROM) and it is not available for the end user for modifications, which means the memory is protected from unwanted user interaction by implementing some kind of memory protection mechanism. The most common types of memories used in embedded systems for control algorithm storage are OTP, PROM, UVEPROM, EEPROM and FLASH. Depending on the control application, the memory size may vary from a few bytes to megabytes. We will discuss them in detail in the coming sections. Sometimes the system requires temporary memory for performing arithmetic operations or control algorithm execution and this type of memory is known as "working memory". Random Access Memory (RAM) is used in most of the systems as the working memory. Various types of RAM like SRAM, DRAM, and NVRAM are used for this purpose. The size of the RAM also varies from a few bytes to kilobytes or megabytes depending on the application. The details given under the section "Memory" will give you a more detailed description of the working memory.

An embedded system without a control algorithm implemented memory is just like a new born baby. It is having all the peripherals but is not capable of making any decision depending on the situational as well as real world changes. The only difference is that the memory of a new born baby is self-adaptive, meaning that the baby will try to learn from the surroundings and from the mistakes committed. For embedded systems it is the responsibility of the designer to impart intelligence to the system.

In a controller-based embedded system, the controller may contain internal memory for storing the control algorithm and it may be an EEPROM or FLASH memory varying from a few kilobytes to megabytes. Such controllers are called controllers with on-chip ROM, e.g. Atmel AT89C51. Some controllers may not contain on-chip memory and they require an external (off-chip) memory for holding the control algorithm, e.g. Intel 8031AH.

2.1 CORE OF THE EMBEDDED SYSTEM

LO 1 Identify the building blocks of a typical Embedded System

Embedded systems are domain and application specific and are built around a central core. The core of the embedded system falls into any one of the following categories:

(1) General Purpose and Domain Specific Processors
  1.1 Microprocessors
  1.2 Microcontrollers
  1.3 Digital Signal Processors
(2) Application Specific Integrated Circuits (ASICs)
(3) Programmable Logic Devices (PLDs)
(4) Commercial off-the-shelf Components (COTS)

If you examine any embedded system you will find that it is built around any of the core units mentioned above.

100 devices by more than 20 independent manufacturers like Maxim, NXP, Atmel, etc. under the license from Intel. Due to the low cost, wide availability, memory efficient instruction set, mature development tools and Boolean processing (bit manipulation operation) capability, *8051 family* derivative microcontrollers are much used in high-volume consumer electronic devices, entertainment industry and other gadgets where cost-cutting is essential.

Another important family of microcontrollers used in industrial control and embedded applications is the **PIC** family micro controllers from Microchip Technologies (It will be discussed in detail in a later section of this book). It is a high performance RISC microcontroller complementing the CISC (Complex Instruction Set Computing) features of *8051*. The terms RISC and CISC will be explained in detail in a separate heading.

Some embedded system applications require only 8bit controllers whereas some embedded applications requiring superior performance and computational needs demand 16/32bit microcontrollers. Infineon, Freescale, Philips, Atmel, Maxim, Microchip etc. are the key suppliers of 16bit microcontrollers. Philips tried to extend the *8051* family microcontrollers to use for 16bit applications by developing the Philips XA (eXtended Architecture) microcontroller series.

8bit microcontrollers are commonly used in embedded systems where the processing power is not a big constraint. As mentioned earlier, more than 20 companies are producing different flavours of the *8051* family microcontroller. They try to add more and more functionalities like built in SPI, I2C serial buses, USB controller, ADC, Networking capability, etc. So the competitive market is driving towards a one-stop solution chip in microcontroller domain. High processing speed microcontroller families like ARM Cortex M Series are also available in the market, which provides solution to applications requiring hardware acceleration and high processing capability.

Freescale, Renesas, Zilog, Cypress (Spansion), Infineon, ST Micro Electronics, EPSON, Texas Instruments, Toshiba, NXP, Microchip, Analog Devices, Daewoo, Intel, Maxim, Sharp, Silicon Laboratories, HOLTEK, LAPIS, CYROD, Atmel, etc. are the key players in the microcontroller market. Of these Atmel has got special significance. They are the manufacturers of a variety of Flash memory based microcontrollers. They also provide In-System Programmability (which will be discussed in detail in a later section of this book) for the controller. The Flash memory technique helps in fast reprogramming of the chip and thereby reduces the product development time. Atmel also provides another special family of microcontroller called AVR (it will be discussed in detail in a later chapter), an 8bit RISC Flash microcontroller, fast enough to execute powerful instructions in a single clock cycle and provide the latitude you need to optimise power consumption.

The instruction set architecture of a microcontroller can be either RISC or CISC. Microcontrollers are designed for either general purpose application requirement (general purpose controller) or domain- specific application requirement (application specific instruction set processor). The *Intel 8051* microcontroller is a typical example for a general purpose microcontroller, whereas the automotive AVR microcontroller family from Atmel Corporation is a typical example for ASIP specifically designed for the automotive domain.

2.1.1.4 Microprocessor vs Microcontroller The following table summarises the differences between a microcontroller and microprocessor.

Microprocessor	Microcontroller
A silicon chip representing a central processing unit (CPU), which is capable of performing arithmetic as well as logical operations according to a predefined set of instructions	A microcontroller is a highly integrated chip that contains a CPU, scratchpad RAM, special and general purpose register arrays, on chip ROM/FLASH memory for program storage, timer and interrupt control units and dedicated I/O ports

(*Contd.*)

Microprocessor	Microcontroller
It is a dependent unit. It requires the combination of other chips like timers, program and data memory chips, interrupt controllers, etc. for functioning	It is a self-contained unit and it doesn't require external interrupt controller, timer, UART, etc. for its functioning
Most of the time general purpose in design and operation	Mostly application-oriented or domain-specific
Doesn't contain a built in I/O port. The I/O port functionality needs to be implemented with the help of external programmable peripheral interface chips like 8255	Most of the processors contain multiple built-in I/O ports which can be operated as a single 8 or 16 or 32 bit port or as individual port pins
Targeted for high end market where performance is important	Targeted for embedded market where performance is not so critical (At present this demarcation is invalid)
Limited power saving options compared to microcontrollers	Includes lot of power saving features

2.1.1.5 Digital Signal Processors Digital Signal Processors (DSPs) are powerful special purpose 8/16/32 bit microprocessors designed specifically to meet the computational demands and power constraints of today's embedded audio, video, and communications applications. Digital signal processors are 2 to 3 times faster than the general purpose microprocessors in signal processing applications. This is because of the architectural difference between the two. DSPs implement algorithms in hardware which speeds up the execution whereas general purpose processors implement the algorithm in firmware and the speed of execution depends primarily on the clock for the processors. In general, DSP can be viewed as a microchip designed for performing high speed computational operations for 'addition', 'subtraction', 'multiplication' and 'division'. A typical digital signal processor incorporates the following key units:

Program Memory Memory for storing the program required by DSP to process the data.

Data Memory Working memory for storing temporary variables and data/signal to be processed.

Computational Engine Performs the signal processing in accordance with the stored program memory. Computational Engine incorporates many specialised arithmetic units and each of them operates simultaneously to increase the execution speed. It also incorporates multiple hardware shifters for shifting operands and thereby saves execution time.

I/O Unit Acts as an interface between the outside world and DSP. It is responsible for capturing signals to be processed and delivering the processed signals.

Audio video signal processing, telecommunication, and multimedia applications are typical examples where DSP is employed. Digital signal processing employs a large amount of real-time calculations. Sum of products (SOP) calculation, convolution, fast fourier transform (FFT), discrete fourier transform (DFT), etc, are some of the operations performed by digital signal processors.

Blackfin®[†] processors from Analog Devices is an example of DSP which delivers breakthrough signal-processing performance and power efficiency while also offering a full 32-bit RISC MCU programming model. Blackfin processors present high-performance, homogeneous software targets, which allows flexible resource allocation between hard real-time signal processing tasks and non real-time control tasks. System control tasks can often run in the shadow of demanding signal processing and multimedia tasks.

2.1.1.6 RISC vs. CISC Processors/Controllers The term RISC stands for Reduced Instruction Set Computing. As the name implies, all RISC processors/controllers possess lesser number of instructions,

[†]Blackfin® is a Registered trademark of Analog Devices Inc.

typically in the range of 30 to 40. CISC stands for Complex Instruction Set Computing. From the definition itself it is clear that the instruction set is complex and instructions are high in number. From a programmers point of view RISC processors are comfortable since s/he needs to learn only a few instructions, whereas for a CISC processor s/he needs to learn more number of instructions and should understand the context of usage of each instruction (This scenario is explained on the basis of a programmer following Assembly Language coding. For a programmer following C coding it doesn't matter since the cross-compiler is responsible for the conversion of the high level language instructions to machine dependent code). Atmel AVR microcontroller is an example for a RISC processor and its instruction set contains only 32 instructions. The original version of *8051* microcontroller (e.g. *AT89C51*) is a CISC controller and its instruction set contains 255 instructions. Remember it is not the number of instructions that determines whether a processor/controller is CISC or RISC. There are some other factors like pipelining features, instruction set type, etc. for determining the RISC/CISC criteria. Some of the important criteria are listed below:

RISC	CISC
Lesser number of instructions	Greater number of Instructions
Instruction pipelining and increased execution speed	Generally no instruction pipelining feature
Orthogonal instruction set (Allows each instruction to operate on any register and use any addressing mode)	Non-orthogonal instruction set (All instructions are not allowed to operate on any register and use any addressing mode. It is instruction-specific)
Operations are performed on registers only, the only memory operations are load and store	Operations are performed on registers or memory depending on the instruction
A large number of registers are available	Limited number of general purpose registers
Programmer needs to write more code to execute a task since the instructions are simpler ones	Instructions are like macros in C language. A programmer can achieve the desired functionality with a single instruction which in turn provides the effect of using more simpler single instructions in RISC
Single, fixed length instructions	Variable length instructions
Less silicon usage and pin count	More silicon usage since more additional decoder logic is required to implement the complex instruction decoding.
With Harvard Architecture	Can be Harvard or Von-Neumann Architecture

I hope now you are clear about the terms RISC and CISC in the processor technology. Isn't it?

2.1.1.7 Harvard vs. Von-Neumann Processor/Controller Architecture The terms Harvard and Von-Neumann refers to the processor architecture design.

Microprocessors/controllers based on the **Von-Neumann** architecture shares a single common bus for fetching both instructions and data. Program instructions and data are stored in a common main memory. Von-Neumann architecture based processors/controllers first fetch an instruction and then fetch the data to support the instruction from code memory. The two separate fetches slows down the controller's operation. Von-Neumann architecture is also referred as **Princeton** architecture, since it was developed by the Princeton University.

Microprocessors/controllers based on the **Harvard** architecture will have separate data bus and instruction bus. This allows the data transfer and program fetching to occur simultaneously on both buses. With Harvard architecture, the data memory can be read and written while the program memory is being accessed. These separated data memory and code memory buses allow one instruction to execute while the next instruction is fetched ("prefetching"). The prefetch theoretically allows much faster execution than Von-Neumann

architecture. Since some additional hardware logic is required for the generation of control signals for this type of operation it adds silicon complexity to the system. Figure 2.2 explains the Harvard and Von-Neumann architecture concept.

Fig. 2.2 Harvard vs Von-Neumann architecture

The following table highlights the differences between Harvard and Von-Neumann architecture.

Harvard Architecture	Von-Neumann Architecture
Separate buses for instruction and data fetching	Single shared bus for instruction and data fetching
Easier to pipeline, so high performance can be achieved	Low performance compared to Harvard architecture
Comparatively high cost	Cheaper
No memory alignment problems	Allows self modifying codes*
Since data memory and program memory are stored physically in different locations, no chances for accidental corruption of program memory	Since data memory and program memory are stored physically in the same chip, chances for accidental corruption of program memory

2.1.1.8 Big-Endian vs. Little-Endian Processors/Controllers Endianness specifies the order in which the data is stored in the memory by processor operations in a multibyte system (Processors whose word size is greater than one byte). Suppose the word length is two byte then data can be stored in memory in two different ways:

(1) Higher order of data byte at the higher memory and lower order of data byte at location just below the higher memory.
(2) Lower order of data byte at the higher memory and higher order of data byte at location just below the higher memory.

Little-endian **Little-endian** (Fig. 2.3) means the lower-order byte of the data is stored in memory at the lowest address, and the higher-order byte at the highest address. (The little end comes first.) For example, a 4 byte long integer **Byte3 Byte2 Byte1 Byte0** will be stored in the memory as shown below:

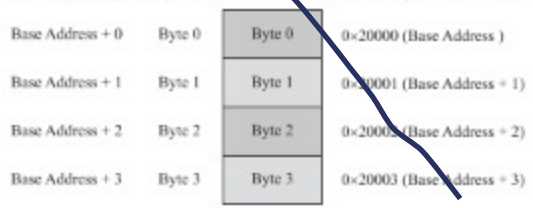


Fig. 2.3 Little-Endian operation

*Self-modifying code is a code/instruction which modifies itself while execution.

CHAPTER 15

Sensors and interfacing

Chapter summary

Sensors provide us with a means of generating signals that can be used as inputs to electronic circuits. The things that we might want to sense include physical parameters such as temperature, light level and pressure. Being able to generate an electrical signal that accurately represents these quantities allows us not only to measure and record these values but also to control them.

Sensors are, in fact, a subset of a larger family of devices known as **transducers** so we will consider these before we look at sensors and how we condition the signals that they produce in greater detail. We begin, however, with a brief introduction to the instrumentation and control systems in which sensors, transducers and signal conditioning circuits are used.

Instrumentation and control systems

Fig. 15.1 shows the arrangement of an instrumentation system. The physical quantity to be measured (e.g. temperature) acts upon a sensor that produces an electrical output signal. This signal is an electrical analogue of the physical input but note that there may not be a linear relationship between the physical quantity and its electrical equivalent. Because of this and since the output produced by the sensor may be small or may suffer from the presence of noise (i.e. unwanted signals) further signal conditioning will be required before the signal will be at an acceptable level and in an acceptable form for signal processing, display and recording. Furthermore, because the signal processing may use digital rather than analogue signals an additional stage of analogue-to-analogue conversion may be required.

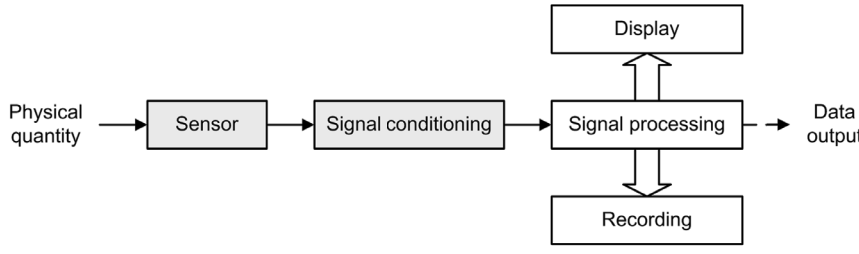
Fig. 15.1(b) shows the arrangement of a control system. This uses **negative feedback** in order to regulate and stabilize the output. It thus becomes possible to set the input or **demand** (i.e. what we desire the output to be) and leave the system to regulate itself by comparing it with a signal derived from the output (via a sensor and appropriate signal conditioning).

A **comparator** is used to sense the difference in these two signals and where any discrepancy is detected the input to the power amplifier is adjusted accordingly. This signal is referred to as an **error signal** (it should be zero when the output exactly matches the demand). The input (demand) is often derived from a simple potentiometer connected across a stable d.c. voltage source while the controlled device can take many forms (e.g. a d.c. motor, linear actuator, heater, etc.).

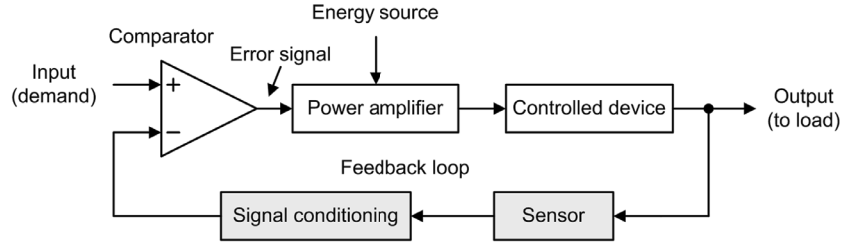
Transducers

Transducers are devices that convert energy in the form of sound, light, heat, etc., into an equivalent electrical signal, or vice versa.

Before we go further, let's consider a couple of examples that you will already be familiar with. A loudspeaker is a transducer that converts low-frequency electric current into audible sounds. A microphone, on the other hand, is a transducer that performs the reverse function, i.e. that of



(a) An instrumentation system



(b) A control system

Figure 15.1 Instrumentation and control systems

converting sound pressure variations into voltage or current. Loudspeakers and microphones can thus be considered as complementary transducers.

Transducers may be used both as inputs to electronic circuits and outputs from them. From the two previous examples, it should be obvious that a loudspeaker is an **output transducer** designed for use in conjunction with an audio system. A microphone is an **input transducer** designed for use with a recording or sound reinforcing system.

There are many different types of transducer and Tables 15.1 and 15.2 provide some examples of transducers that can be used to input and output three important physical quantities; sound, temperature and angular position.

Sensors

A *sensor* is a special kind of transducer that is used to generate an input signal to a measurement, instrumentation or control system. The signal produced by a sensor is an **electrical analogy** of a physical quantity, such as distance, velocity, acceleration, temperature, pressure, light level, etc. The signals returned from a sensor, together with control inputs from the user or controller (as appropriate) will subsequently be used to determine the output from the system. The choice of sensor is governed by a number of factors including accuracy, resolution, cost and physical size.

Sensors can be categorized as either **active** or **passive**. An active sensor *generates* a current or voltage output. A passive transducer *requires a source of current or voltage* and it modifies this in some way (e.g. by virtue of a change in the sensor's resistance). The result may still be a voltage or current *but it is not generated by the sensor on its own*.

Sensors can also be classed as either **digital** or **analogue**. The output of a digital sensor can exist in only two discrete states, either 'on' or 'off',

Table 15.1 Some examples of input transducers

Physical quantity	Input transducer	Notes
Sound (pressure change)	Dynamic microphone (see Fig. 15.3)	Diaphragm attached to a coil is suspended in a magnetic field. Movement of the diaphragm causes current to be induced in the coil.
Temperature	Thermocouple (see Fig. 15.2)	Small e.m.f. generated at the junction between two dissimilar metals (e.g. copper and constantan). Requires reference junction and compensated cables for accurate measurement.
Angular position	Rotary potentiometer	Fine wire resistive element is wound around a circular former. Slider attached to the control shaft makes contact with the resistive element. A stable d.c. voltage source is connected across the ends of the potentiometer. Voltage appearing at the slider will then be proportional to angular position.

Table 15.2 Some examples of output transducers

Physical quantity	Output transducer	Notes
Sound (pressure change)	Loudspeaker (see Fig. 15.3)	Diaphragm attached to a coil is suspended in a magnetic field. Current in the coil causes movement of the diaphragm which alternately compresses and rarefies the air mass in front of it.
Temperature	Heating element (resistor)	Metallic conductor is wound onto a ceramic or mica former. Current flowing in the conductor produces heat.
Angular position	Rotary potentiometer	Multi-phase motor provides precise rotation in discrete steps of 15° (24 steps per revolution), 7.5° (48 steps per revolution) and 1.8° (200 steps per revolution).

Figure 15.2 A selection of thermocouple probes

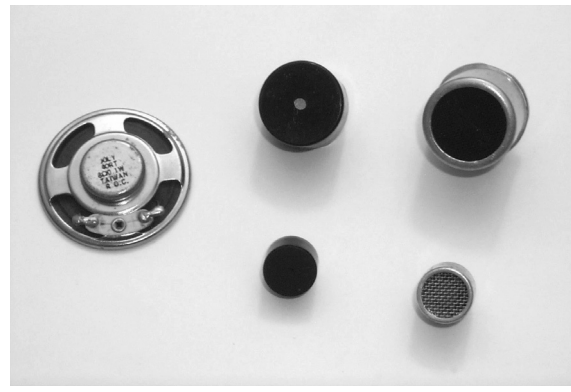


Figure 15.3 A selection of audible transducers

'low' or 'high', 'logic 1' or 'logic 0', etc. The output of an analogue sensor can take any one of an infinite number of voltage or current levels. It is thus said to be *continuously variable*. Table 15.3 provides details of some common types of sensor.

Table 15.3 Some examples of input transducers

Physical quantity	Input transducer	Notes
Angular position	Resistive rotary position sensor (see Fig. 15.5)	Rotary track potentiometer with linear law produces analogue voltage proportional to angular position.
	Optical shaft encoder	Encoded disk interposed between optical transmitter and receiver (infra-red LED and photodiode or photo-transistor).
Angular velocity	Tachogenerator	Small d.c. generator with linear output characteristic. Analogue output voltage proportional to shaft speed.
	Toothed rotor tachometer	Magnetic pick-up responds to the movement of a toothed ferrous disk. The pulse repetition frequency of the output is proportional to the angular velocity.
Flow	Rotating vane flow sensor (see Fig. 15.9)	Turbine rotor driven by fluid. Turbine interrupts infra-red beam. Pulse repetition frequency of output is proportional to flow rate.
Linear position	Resistive linear position sensor	Linear track potentiometer with linear law produces analogue voltage proportional to linear position. Limited linear range.
	Linear variable differential transformer (LVDT)	Miniature transformer with split secondary windings and moving core attached to a plunger. Requires a.c. excitation and phase-sensitive detector.
	Magnetic linear position sensor	Magnetic pick-up responds to movement of a toothed ferrous track. Pulses are counted as the sensor moves along the track.
Light level	Photocell	Voltage-generating device. The analogue output voltage produced is proportional to light level.
	Light-dependent resistor (LDR) (see Fig. 15.8)	An analogue output voltage results from a change of resistance within a cadmium sulphide (CdS) sensing element. Usually connected as part of a potential divider or bridge.
	Photodiode (see Fig. 15.8)	Two-terminal device connected as a current source. An analogue output voltage is developed across a series resistor of appropriate value.
	Phototransistor (see Fig. 15.8)	Three-terminal device connected as a current source. An analogue output voltage is developed across a series resistor of appropriate value.

Table 15.3 (*Continued*)

Physical quantity	Input transducer	Notes
Liquid level	Float switch (see Fig. 15.7)	Simple switch element which operates when a particular level is detected.
	Capacitive proximity switch	Switching device which operates when a particular level is detected. Ineffective with some liquids.
	Diffuse scan proximity switch	Switching device which operates when a particular level is detected. Ineffective with some liquids.
Pressure	Microswitch pressure sensor (see Fig. 15.4)	Microswitch fitted with actuator mechanism and range-setting springs. Suitable for high-pressure applications.
	Differential pressure vacuum switch	Microswitch with actuator driven by a diaphragm. May be used to sense differential pressure. Alternatively, one chamber may be evacuated and the sensed pressure applied to a second input.
	Piezo-resistive pressure sensor	Pressure exerted on diaphragm causes changes of resistance in attached piezo-resistive transducers. Transducers are usually arranged in the form of a four active element bridge which produces an analogue output voltage.
Proximity	Reed switch (see Fig. 15.4)	Reed switch and permanent magnet actuator. Only effective over short distances.
	Inductive proximity switch	Target object modifies magnetic field generated by the sensor. Only suitable for metals (non-ferrous metals with reduced sensitivity).
	Capacitive proximity switch	Target object modifies electric field generated by the sensor. Suitable for metals, plastics, wood and some liquids and powders.
	Optical proximity switch (see Fig. 15.4)	Available in diffuse and through scan types. Diffuse scan types require reflective targets. Both types employ optical transmitters and receivers (usually infra-red emitting LEDs and photo-diodes or photo-transistors). Digital input port required.
Strain	Resistive strain gauge	Foil type resistive element with polyester backing for attachment to body under stress. Normally connected in full bridge configuration with temperature-compensating gauges to provide an analogue output voltage.
	Semiconductor strain gauge	Piezo-resistive elements provide greater outputs than comparable resistive foil types. More prone to temperature changes and also inherently non-linear.
Temperature	Thermocouple (see Fig. 15.2)	Small e.m.f. generated by a junction between two dissimilar metals. For accurate measurement, requires compensated connecting cables and specialized interface.
	Thermistor (see Fig. 15.6)	Usually connected as part of a potential divider or bridge. An analogue output voltage results from resistance changes within the sensing element.
	Semiconductor temperature sensor (see Fig. 15.6)	Two-terminal device connected as a current source. An analogue output voltage is developed across a series resistor of appropriate value.
Weight	Load cell	Usually comprises four strain gauges attached to a metal frame. This assembly is then loaded and the analogue output voltage produced is proportional to the weight of the load.
Vibration	Electromagnetic vibration sensor	Permanent magnet seismic mass suspended by springs within a cylindrical coil. The frequency and amplitude of the analogue output voltage are respectively proportional to the frequency and amplitude of vibration.

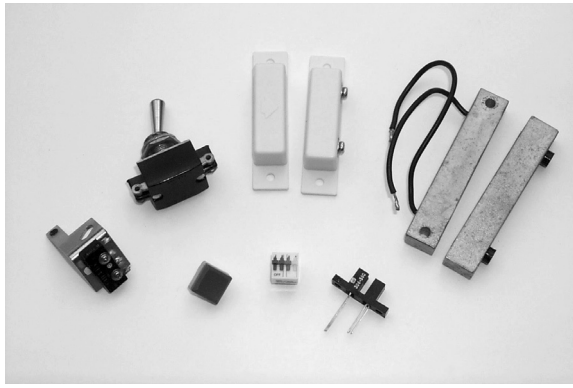


Figure 15.4 Various switch sensors

Figure 15.7 Liquid level float switch

Figure 15.5 Resistive linear position sensor

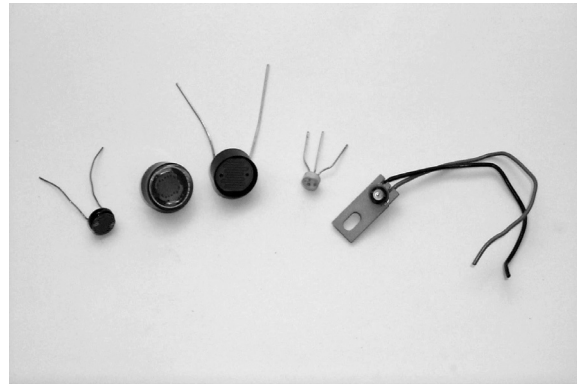


Figure 15.8 Various optical and light sensors

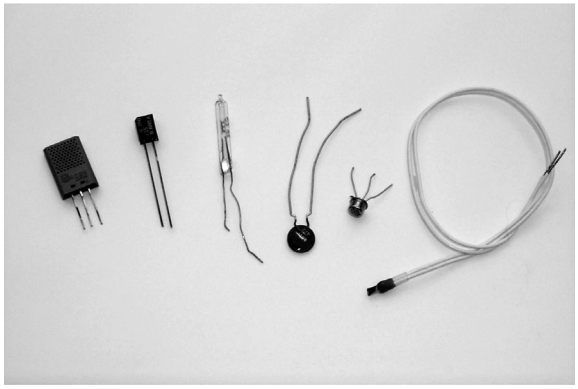


Figure 15.6 Various temperature and gas sensors

Figure 15.9 Liquid flow sensor (digital output)

The EEPROM data storage memory is available as either serial interface or parallel interface chip. If the processor/controller of the device supports serial interface and the amount of data to write and read to and from the device is less, it is better to have a Serial EEPROM chip. The Serial EEPROM saves the address space of the total system. The memory capacity of the serial EEPROM is usually expressed in bits or kilobits. 512 bits, 1Kbits, 2Kbits, 4Kbits, etc. are examples for serial EEPROM memory representation. For embedded systems with low power requirements like portable devices, choose low power memory devices. Certain embedded devices may be targeted for operating at extreme environmental conditions like high temperature, high humid area, etc. Select an industrial grade memory chip in place of the commercial grade chip for such devices.

2.3 SENSORS AND ACTUATORS

LO 3 Analyse the role of sensors, actuators, and their interfacing with the I/O subsystems of an embedded system

At the very beginning of this chapter it is already mentioned that an embedded system is in constant interaction with the Real world and the controlling/monitoring functions executed by the embedded system is achieved in accordance with the changes happening to the Real world. The changes in system environment or variables are detected by the sensors connected to the input port of the embedded system. If the embedded system is designed for any controlling purpose, the system will produce some changes in the controlling variable to bring the controlled variable to the desired value. It is achieved through an actuator connected to the output port of the embedded system. If the embedded system is designed for monitoring purpose only, then there is no need for including an actuator in the system. For example, take the case of an ECG machine. It is designed to monitor the heart beat status of a patient and it cannot impose a control over the patient's heart beat and its order. The sensors used here are the different electrode sets connected to the body of the patient. The variations are captured and presented to the user (may be a doctor) through a visual display or some printed chart.

2.3.1 Sensors

A sensor is a transducer device that converts energy from one form to another for any measurement or control purpose. This is what I "by-hearted" during my engineering degree from the transducers paper.

Looking back to the 'Wearable devices' example given at the end of Chapter 1, we can identify that the sensor which counts steps for pedometer functionality is an Accelerometer sensor and the sensor used in some of the smartwatch devices to measure the light intensity is an Ambient Light Sensor (ALS)

2.3.2 Actuators

Actuator is a form of transducer device (mechanical or electrical) which converts signals to corresponding physical action (motion). Actuator acts as an output device.

Looking back to the 'Wearable devices' example given at the end of Chapter 1, we can see that certain smartwatches use Ambient Light Sensor to detect the surrounding light intensity and uses an electrical/electronic actuator circuit to adjust the screen brightness for better readability.

2.3.3 The I/O Subsystem

The I/O subsystem of the embedded system facilitates the interaction of the embedded system with the external world. As mentioned earlier the interaction happens through the sensors and actuators connected to the input and output ports respectively of the embedded system. The sensors may not be directly interfaced to the input ports, instead they may be interfaced through signal conditioning and translating systems like ADC,

optocouplers, etc. This section illustrates some of the sensors and actuators used in embedded systems and the I/O systems to facilitate the interaction of embedded systems with external world.

2.3.3.1 Light Emitting Diode (LED) Light Emitting Diode (LED) is an important output device for visual indication in any embedded system. LED can be used as an indicator for the status of various signals or situations. Typical examples are indicating the presence of power conditions like 'Device ON', 'Battery low' or 'Charging of battery' for a battery operated handheld embedded devices.

Light Emitting Diode is a *p-n* junction diode (Refer Analog Electronics fundamentals to refresh your memory for *p-n* junction diode ☺) and it contains an anode and a cathode. For proper functioning of the LED, the anode of it should be connected to +ve terminal of the supply voltage and cathode to the –ve terminal of supply voltage. The current flowing through the LED must be limited to a value below the maximum current that it can conduct. A resister is used in series between the power supply and the LED to limit the current through the LED. The ideal LED interfacing circuit is shown in Fig. 2.13.

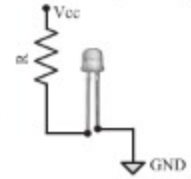


Fig. 2.13 LED interfacing

LEDs can be interfaced to the port pin of a processor/controller in two ways. In the first method, the anode is directly connected to the port pin and the port pin drives the LED. In this approach the port pin 'sources' current to the LED when the port pin is at logic High (Logic '1'). In the second method, the cathode of the LED is connected to the port pin of the processor/controller and the anode to the supply voltage through a current limiting resistor. The LED is turned on when the port pin is at logic Low (Logic '0'). Here the port pin 'sinks' current. If the LED is directly connected to the port pin, depending on the maximum current that a port pin can source, the brightness of LED may not be to the required level. In the second approach, the current is directly sourced by the power supply and the port pin acts as the sink for current. Here we will get the required brightness for the LED.

2.3.3.2 7-Segment LED Display The 7-segment LED display is an output device for displaying alpha numeric characters. It contains 8 light-emitting diode (LED) segments arranged in a special form. Out of the 8 LED segments, 7 are used for displaying alpha numeric characters and 1 is used for representing 'decimal point' in decimal number display. Figure 2.14 explains the arrangement of LED segments in a 7-segment LED display.

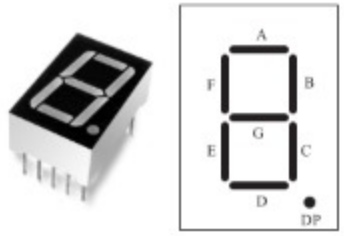


Fig. 2.14 7-Segment LED Display

The LED segments are named A to G and the decimal point LED segment is named as DP. The LED segments A to G and DP should be lit accordingly to display numbers and characters. For example, for displaying the number 4, the segments F, G, B and C are lit. For displaying 3, the segments A, B, C, D, G and DP are lit. For displaying the character 'd', the segments B, C, D, E and G are lit. All these 8 LED segments need to be connected to one port of the processor/controller for displaying alpha numeric digits. The 7-segment LED displays are available in two different configurations, namely; Common Anode and Common Cathode. In the common anode configuration, the anodes of the 8 segments are connected commonly whereas in the common cathode configuration, the 8 LED segments share a common cathode line. Figure 2.15 illustrates the Common Anode and Cathode configurations.

Based on the configuration of the 7-segment LED unit, the LED segment's anode or cathode is connected to the port of the processor/controller in the order 'A' segment to the least significant port pin and DP segment to the most significant port pin.

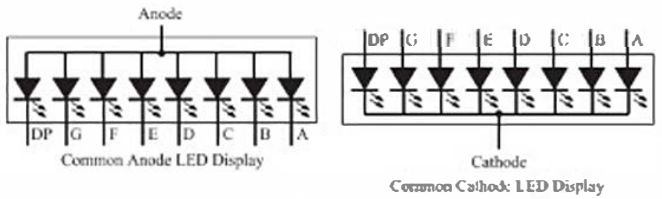


Fig. 2.15 Common anode and cathode configurations of a 7-segment LED Display

The current flow through each of the LED segments should be limited to the maximum value supported by the LED display unit. The typical value for the current falls within the range of 20mA. The current through each segment can be limited by connecting a current limiting resistor to the anode or cathode of each segment. The value for the current limiting resistors can be calculated using the current value from the electrical parameter listing of the LED display.

For common cathode configurations, the anode of each LED segment is connected to the port pins of the port to which the display is interfaced. The anode of the common anode LED display is connected to the 5V supply voltage through a current limiting resistor and the cathode of each LED segment is connected to the respective port pin lines. For an LED segment to lit in the Common anode LED configuration, the port pin to which the cathode of the LED segment is connected should be set at logic 0.

7-segment LED display is a popular choice for low cost embedded applications like. Public telephone call monitoring devices, point of sale terminals, etc.

2.3.3.3 Optocoupler Optocoupler is a solid state device to isolate two parts of a circuit. Optocoupler combines an LED and a photo-transistor in a single housing (package). Figure 2.16 illustrates the functioning of an optocoupler device.

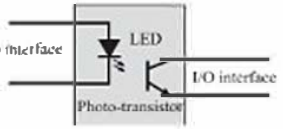


Fig. 2.16 An optocoupler device

In electronic circuits, an optocoupler is used for suppressing interference in data communication, circuit isolation, high voltage separation, simultaneous separation and signal intensification, etc. Optocouplers can be used in either input circuits or in output circuits. Figure 2.17 illustrates the usage of optocoupler in input circuit and output circuit of an embedded system with a microcontroller as the system core.

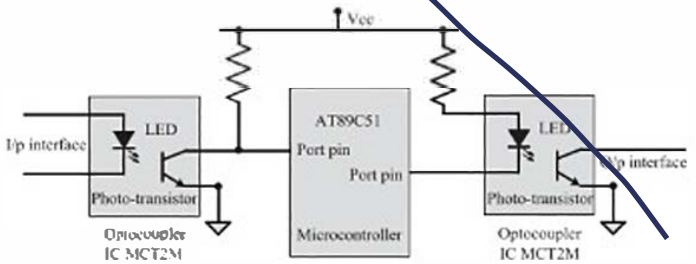


Fig. 2.17 Optocoupler in Input and Output circuit

Optocoupler is available as ICs from different semiconductor manufacturers. The MCT2M IC from Fairchild semiconductor (http://www.fairchildsemi.com/) is an example for optocoupler IC.

2.3.3.4 Stepper Motor A stepper motor is an electro-mechanical device which generates discrete displacement (motion) in response to dc electrical signals. It differs from the normal dc motor in its operation. The dc motor produces continuous rotation on applying dc voltage whereas a stepper motor produces discrete rotation in response to the dc voltage applied to it. Stepper motors are widely used in industrial embedded applications, consumer electronic products and robotics control systems. The paper feed mechanism of a printer/fax makes use of stepper motors for its functioning.

Based on the coil winding arrangements, a two-phase stepper motor is classified into two. They are:

(1) Unipolar
(2) Bipolar

(1) Unipolar A unipolar stepper motor contains two windings per phase. The direction of rotation (clockwise or anticlockwise) of a stepper motor is controlled by changing the direction of current flow. Current in one direction flows through one coil and in the opposite direction flows through the other coil. It is easy to shift the direction of rotation by just switching the terminals to which the coils are connected. Figure 2.18 illustrates the working of a two-phase unipolar stepper motor.

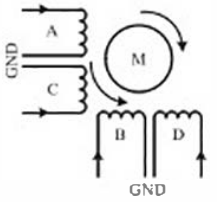


Fig. 2.18 2-Phase unipolar stepper motor

The coils are represented as A, B, C and D. Coils A and C carry current in opposite directions for phase 1 (only one of them will be carrying current at a time). Similarly, B and D carry current in opposite directions for phase 2 (only one of them will be carrying current at a time).

(2) Bipolar A bipolar stepper motor contains single winding per phase. For reversing the motor rotation the current flow through the windings is reversed dynamically. It requires complex circuitry for current flow reversal. The stator winding details for a two phase unipolar stepper motor is shown in Fig. 2.19.

The stepping of stepper motor can be implemented in different ways by changing the sequence of activation of the stator windings. The different stepping modes supported by stepper motor are explained below.

Full Step In the full step mode both the phases are energised simultaneously. The coils A, B, C and D are energised in the following order:

Step	Coil A	Coil B	Coil C	Coil D
1	H	H	L	L
2	L	H	H	L
3	L	L	H	H
4	H	L	L	H

It should be noted that out of the two windings, only one winding of a phase is energised at a time.