Module-4

Embedded systems, Sensors and Interfacing

Embedded Systems – Definition, Embedded systems vs general computing systems, Classification of Embedded Systems, Major application areas of Embedded Systems, Elementsof an Embedded System, Core of the Embedded System, Microprocessor vs Microcontroller, RISC vs CISC.

Sensors and Interfacing –

Instrumentation and control systems, Transducers, Sensors, Actuators, LED, 7-Segment LED Display.

WHAT IS AN EMBEDDED SYSTEM?

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 specialized to the application domain.

Example: Telecommunications, medical equipment, industrial control, consumer products, etc.

EMBEDDED SYSTEMS vs. GENERAL COMPUTING SYSTEM

General Purpose System	Embedded System
A system which is a combination of generic hardware and 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
Contain a General Purpose Operating System (GPOS)	May or may not contain an operating system for functioning
Applications are alterable (programmable) by user (It is possible for the end user to re-install the Operating System, and add or remove user applications)	The firmware of the embedded system is pre-programmed and it is non-alterable by end-user (There may be exceptions for systems supporting OS kernel image flashing through special hardware settings)
Performance is the key deciding factor on 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 hardware and 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 behavior	Execution behavior is deterministic for certain type of embedded systems like „Hard Real Time" systems

CLASSIFICATION OF EMBEDDED SYSTEMS

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:

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

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:

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.

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.

3) *Third Generation* With advances in processor technology, embedded system developers started making use of powerful 32bit processors and I6bit 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.

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 em- bedded 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 SoC"s 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.

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:

1) Small-Scale Embedded Systems Embedded systems which arc 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.

2) Medium-Scale Embedded Systems Embedded systems which arc 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 (cither general purpose or real time operating system) for functioning.

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) fur task scheduling, prioritization and management.

MAJOR APPLICATION AREAS OF EMBEDDED SYSTEMS

The application areas and the products in the embedded domain arc 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, ignitionsystems, 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 multi meters, digital CROs, logic analyzers 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.

The Typical Embedded System

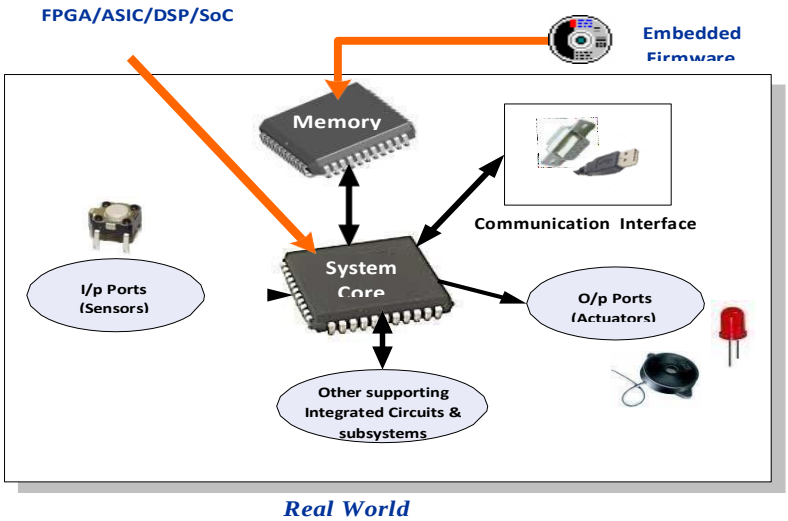


Fig: Elements of an Embedded System

A typical embedded system 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 Metering IC from) Analog Devices for energy metering applications).

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.

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). 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. 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.

CORE OF THE 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 Processor

Microprocessors
Microcontrollers
Digital Signal Processors

2. Application Specific Integrated Circuits (ASICs)
3. Programmable Logic Devices (PLDs)
4. Commercial off-the-shelf Components (COTS)

Microprocessor vs Microcontroller

The following table summarizes 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 pre-defined set of Instructions	A microcontroller is a highly integrated chip that contains a CPU, scratch pad 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
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

RISC vs. CISC Processors/Controllers

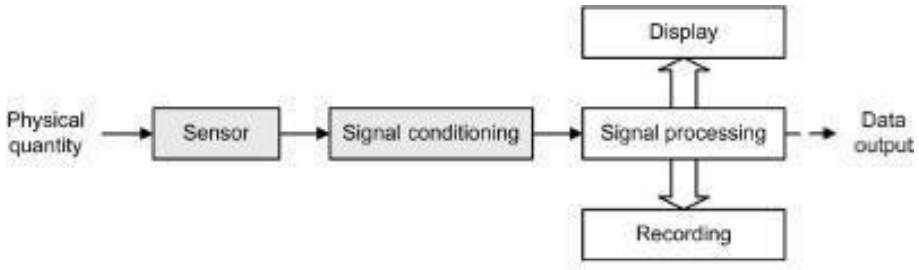
The term RISC stands for Reduced Instruction Set Computing. As the name implies, all RISC processors/controllers possess lesser number of instructions, typically in the range of 30 to 40. CISC stands for Complex Instruction Set Computing. From the definition itself it is clear that the instruction set 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 processorand its instruction set contain only 32 instructions. The original version of 8051 microcontroller (e.g. AT89C51) is a CISC controller and its instruction set contains 255 instructions. 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 instruction	Greater number of Instruction
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 memorydepending 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. Aprogrammer can achieve the desired functionality with a single instruction which in tum 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

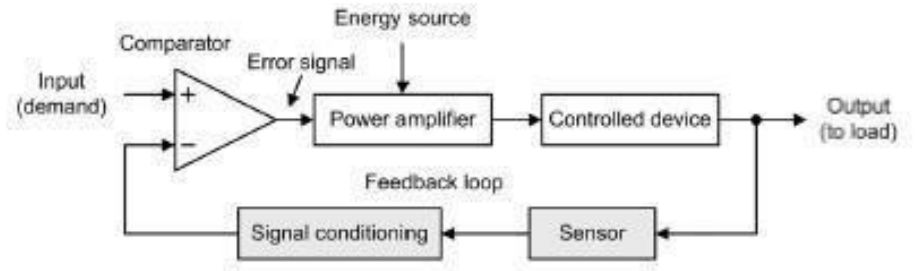
SENSORS AND INTERFACING

Instrumentation and control systems

- Instrumentation: Technology of measurement
- An instrument is a device that measures or manipulates process physical variables such as flow, temperature, level, or pressure etc.
- Fig. (a) 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 signal. 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.



(a) An instrumentation system



(b) A control system

Figure Instrumentation and control systems

- Fig. (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.
- Examples: 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 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.
- **Input transducers examples:** Dynamic microphone, Thermocouple, Rotary potentiometer etc,..
- **output transducer examples:** Loudspeaker, Heating element, Rotary potentiometer etc,.

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", „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.

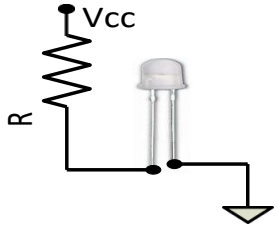
Examples: Resistive rotary position sensor, TachogeneratorToothed rotor tachometer, Rotating vane flow sensor, Photocell Light-dependent resistor (LDR) etc,.

Actuators:

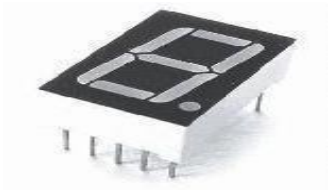
- Actuator is a form of transducer device (mechanical or electrical) which converts signals to corresponding physical action (motion).
- Actuator acts as an output device.
- Example: Looking back to the "Smart" running shoe example, we can see that the actuator used for adjusting the position of the cushioning element is a micro stepper motor.

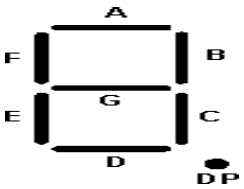
Light Emitting Diode (LED)

- 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 signal or situations. Typical examples are indicating the presence of power conditions like „Device ON" Battery low" or „Charging of battery" for a battery operated hand held embedded devices.
- Light Emitting Diode is a pn 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 Figure.



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.
- 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
- 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

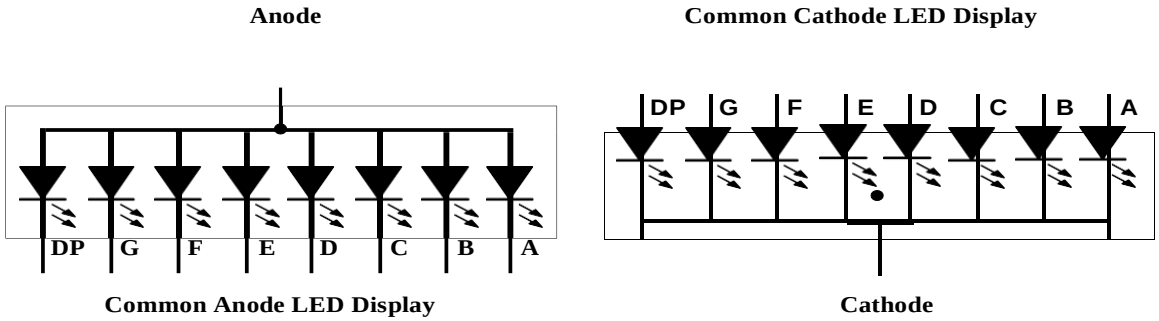


Fig: common anode and cathode configurations of a 7-segment LED Display.