MODULE – 3

Embedded System Components

Introduction

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 specialised to the application domain.

Embedded Systems vs. 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 capability, etc.

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
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 system 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

History of Embedded Systems

- Embedded systems were in existence even before the IT revolution.
  - Built around the old vacuum tube and transistor technologies.
- Advances in semiconductor and nanotechnology 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.
  - It had 36K words of fixed memory and 2K words of erasable memory.
  - The clock frequency of was 1.024 MHz and it was derived from a 2.048 MHz crystal clock.
- The first mass-produced embedded system was the Autonetics D-17 guidance computer for the Minuteman-I missile in 1961.
  - It was built using discrete transistor logic and a hard-disk for main memory.
- The first integrated circuit was produced in September 1958 and computers using them began to appear in 1963.

Classification of Embedded Systems

- 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

Classification Based on Generation

- First Generation
- Second Generation
- Third Generation
- Fourth Generation
- Next Generation

Classification Based on Generation (continued)

- First Generation
  - Early embedded systems were built around 8-bit microprocessors like 8085 and Z80 and 4-bit microcontrollers.
  - Simple in hardware circuits with firmware developed in assembly code.
  - E.g.: Digital telephone keypads, stepper motor control units, etc.

Classification Based on Generation (continued)

- Second Generation
  - Embedded systems built around 16-bit microprocessors and 8-bit or 16-bit microcontrollers.
  - Instruction set were much more complex and powerful than the first generation.
  - Some of the second generation embedded systems contained embedded operating systems for their operation.
  - E.g.: Data acquisition systems, SCADA systems, etc.

Classification Based on Generation (continued)

- Third Generation
  - Embedded systems built around 32-bit microprocessors and 16-bit microcontrollers.
  - Application and domain specific processors/controllers like Digital Signal Processors (DSP) and Application Specific Integrated Circuits (ASICs) came into picture.
  - The instruction set of processors became more complex and powerful and the concept of instruction pipelining also evolved.
  - 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.

Classification Based on Generation (continued)

- 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 implementing different functionalities with a processor core on an integrated circuit.
  - They make use of high performance real time embedded operating systems for their functioning.
  - E.g.: Smart phone devices, Mobile Internet Devices (MIDs), etc.

Classification Based on Generation (continued)

- Next Generation
  - The processor and embedded market is highly dynamic and demanding.
  - The next generation embedded systems are expected to meet growing demands in the market.

Classification Based on Complexity and Performance

- Small-Scale Embedded Systems
- Medium-Scale Embedded Systems
- Large-Scale Embedded Systems

Classification Based on Complexity and Performance (continued)

- Small-Scale Embedded Systems
  - Simple in application needs and the performance requirements are not time critical.
  - E.g.: An electronic toy
  - Usually built around low performance and low cost 8-bit or 16-bit microprocessors/microcontrollers.
  - May or may not contain an operating system for its functioning.

Classification Based on Complexity and Performance (continued)

- Medium-Scale Embedded Systems
  - Slightly complex in hardware and firmware (software) requirements.
  - Usually built around medium performance, low cost 16-bit or 32-bit microprocessors/microcontrollers or digital signal processors.
  - Usually contain an embedded operating system (either general purpose or real time operating system) for functioning.

Classification Based on Complexity and Performance (continued)

- Large-Scale Embedded Systems
  - Highly complex in hardware and firmware (software) requirements.
  - They are employed in mission critical applications demanding high performance.
  - Usually built around high performance 32-bit or 64-bit RISC processors/controllers or Reconfigurable System on Chip (RSoC) or multi-core processors and programmable logic devices.
  - 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 of processing requirements which can be implemented using a co-processor/hardware accelerator.
  - Usually contain a high performance real time operating system (RTOS) for task scheduling, prioritization and management.

Classification Based on Deterministic Behaviour

- Applicable for ‘Real Time’ systems.
- The application/task execution behaviour can be either deterministic or non-deterministic.
- Based on the execution behaviour, real time embedded systems are classified into Hard Real Time and Soft Real Time systems.

Classification Based on Triggering

- 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.

Major Application Areas of Embedded Systems

1. Consumer electronics: Camcorders, cameras, etc.
2. Household appliances: Television, DVD players, washing machine, refrigerators, microwave oven, etc.
3. Home automation and security systems: Air conditioners, sprinklers, intruder detection alarms, closed circuit television (CCTV) cameras, fire alarms, etc.
4. Automotive industry: Anti-lock braking systems (ABS), engine control, ignition systems, automatic navigation systems, etc.
5. Telecom: Cellular telephones, telephone switches, handset multimedia applications, etc.

Major Application Areas of Embedded Systems (continued)

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. Measurements & Instrumentation: Digital multimeters, digital CROs, logic analyzers, PLC systems, etc.
10. Banking & Retail: Automated teller machines (ATM) and currency counters, point of sales (POS), etc.
11. Card readers: Barcode, smart card readers, hand held devices, etc.

Purpose of Embedded Systems

- 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

Purpose of Embedded Systems (continued)

- 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 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).
  - The collected data may be stored or transmitted or it may be processed or it may be deleted instantly after giving a meaningful representation.

- A digital camera is a typical example of an embedded system with data collection/storage/representation of data.
- Images are captured and the captured image may be stored within the memory of the camera.
- The captured image can also be presented to the user through a graphic LCD unit.

Purpose of Embedded Systems (continued)

- Data Communication
  - Embedded data communication systems are deployed in applications ranging from complex satellite communication systems to simple home networking systems.
  - The transmission is achieved either by a wire-line medium or by a wireless medium.
  - The data collecting embedded terminal itself can incorporate data communication units like wireless modules (Bluetooth, ZigBee, Wi-Fi, EDGE, GPRS, etc.) or wire-line modules (RS-232C, USB, TCP/IP, PS2, etc.).

Fig: A wireless network router for data communication

- Network hubs, routers, switches, etc. are typical examples of dedicated data transmission embedded systems.
- They act as mediators in data communication and provide various features like data security, monitoring etc.

Purpose of Embedded Systems (continued)

- Data (Signal) Processing
  - The data (voice, image, video, electrical signals and other measurable quantities) collected by embedded systems may be used for various kinds of data processing.
  - Embedded systems with signal processing functionalities are employed in applications demanding signal processing like speech coding, synthesis, audio video codec, transmission applications, etc.

- A digital hearing aid is a typical example of an embedded system employing data processing.
- Digital hearing aid improves the hearing capacity of hearing impaired persons.

Purpose of Embedded Systems (continued)

- Monitoring
  - Almost embedded products coming under the medical domain are used for monitoring.
- A very good example is the electro cardiogram (ECG) machine for monitoring the heartbeat of a patient.
  - The machine is intended to do the monitoring of the heartbeat.
  - It cannot impose control over the heartbeat.
  - The sensors used in ECG are the different electrodes connected to the patient's body.
  - Some other examples of embedded systems with monitoring function are measuring instruments like digital CRO, digital multimeters, logic analyzers, etc. used in Control & Instrumentation applications.

Fig: A patient monitoring system for monitoring heartbeat

Purpose of Embedded Systems (continued)

- Control
  - Embedded systems with control functionalities impose control over some variables according to the changes in input variables.
  - A system with control functionality contains both sensors and actuators.
  - Sensors are connected to the input port for capturing the changes in environmental variable or measuring variable.
  - The actuators connected to the output port are controlled according to the changes in input variable to put an impact on the controlling variable to bring the controlled variable to the specified range.

- An Air Conditioner System used to control the room temperature to a specified limit is a typical example for embedded system for control purpose.
- An air conditioner contains a room temperature-sensing element (sensor) which may be a thermistor and a handheld unit for setting up (feeding) the desired temperature.

Purpose of Embedded Systems (continued)

- Application Specific User Interface
  - These are embedded systems with application-specific user interfaces like buttons, switches, keypad, lights, bells, display units, etc.
  - Mobile phone is an example for this.
  - In mobile phone the user interface is provided through the keypad, graphic LCD module, system speaker, vibration alert, etc.

- A mobile phone is an example for embedded system with an application-specific user interfaces.

A Typical Embedded System

A Typical Embedded System

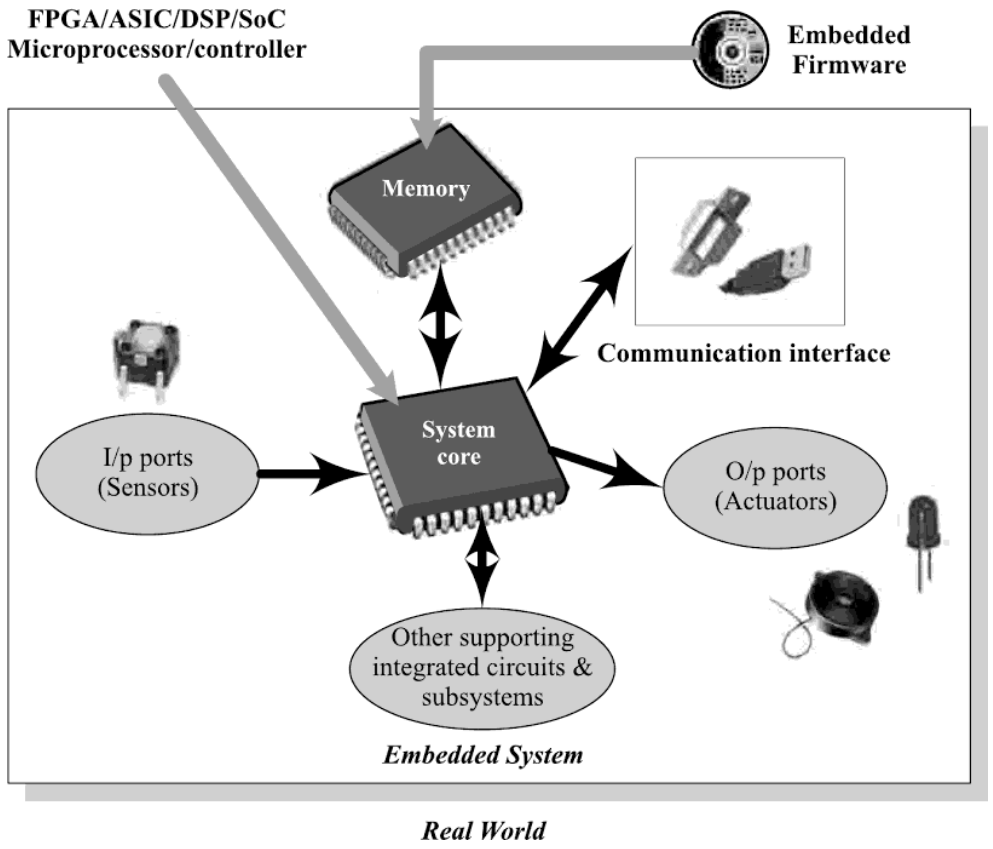


Fig: Elements of an Embedded System

A Typical Embedded System (continued)

- It contains a single chip controller, which acts as the master brain of the system.
- The controller can be
  - ✓ A microprocessor or
  - ✓ A microcontroller or
  - ✓ A Field Programmable Gate Array (FPGA) device or
  - ✓ A Digital Signal Processor (DSP) or
  - ✓ An Application Specific Integrated Circuit (ASIC)/Application Specific Standard Product (ASSP)

A Typical Embedded System (continued)

- 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.
- LEDs, liquid crystal displays, piezoelectric buzzers, etc. are examples for common user interface output devices for a typical embedded system.

A Typical Embedded System (continued)

- 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).
  - It is not available for the end user for modifications
  - 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.
- 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.

A Typical Embedded System (continued)

- Apart from these, communication interface is essential for communicating with various subsystems of the embedded system and with the external world.
- The communication interfaces may be used to achieve onboard (I2C, SPI, UART, parallel bus interface, etc.) or external communication (wireless interfaces like Infrared, Bluetooth, Wi-Fi, etc.)

Core of the Embedded System

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 Processors
     1. Microprocessors
     2. Microcontrollers
     3. Digital Signal Processors
  2. Application Specific Integrated Circuits (ASICs)
  3. Programmable Logic Devices (PLDs)
  4. Commercial off-the-shelf Components (COTS)

General Purpose and Domain Specific Processors

- Almost 80% of the embedded systems are processor/controller based.
- The processor may be a microprocessor or a microcontroller or a digital signal processor, depending on the domain and application.
- Most of the embedded systems in the industrial control and monitoring applications make use of the commonly available microprocessors or microcontrollers.
- Domains which require signal processing such as speech coding, speech recognition, etc. make use of special kind of digital signal processors.

Microprocessors

- A Microprocessor is 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.
- In general the CPU contains the Arithmetic and Logic Unit (ALU), control unit and working registers.
- A microprocessor is a dependent unit and it requires the combination of other hardware like memory, timer unit, and interrupt controller, etc. for proper functioning.
- Intel, AMD, Freescale, IBM, TI, Cyrix, Hitachi, NEC, LSI Logic, etc. are the key players in the processor market.

General Purpose Processor (GPP) vs. Application-Specific Instruction Set Processor (ASIP)

General Purpose Processor (GPP)

- A General Purpose Processor or GPP is a processor designed for general computational tasks.
  - The processor running inside laptop or desktop is a typical example for general purpose processor.
- Due to the high volume production, the per unit cost for a chip is low.
- A typical general purpose processor contains an Arithmetic and Logic Unit (ALU) and Control Unit (CU).

Application-Specific Instruction Set Processor (ASIP)

- Application Specific Instruction Set Processors (ASIPs) are processors with architecture and instruction set optimised to specific-domain/application requirements like network processing, automotive, telecom, media applications, digital signal processing, control applications, etc.
- Most of the embedded systems are built around application specific instruction set processors.
  - Some microcontrollers (like automotive AVR, USB AVR from Atmel), system on chips, digital signal processors, etc. are examples for application specific instruction set processors (ASIPs).
- ASIPs incorporate a processor and on-chip peripherals, demanded by the application requirement, program and data memory.

Microcontrollers

- 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.
- A microcontroller contains all the necessary functional blocks for independent working.
  - Have greater place in embedded domain in place of microprocessors.
  - They are cheap, cost effective and are readily available in the market.
- Atmel, Texas Instruments, Toshiba, Philips, Freescale, NEC, Zilog, Hitachi, Mitsubishi, Infineon, ST Micro Electronics, National, Microchip, Analog Devices, Daewoo, Intel, Maxim, Sharp, Silicon Laboratories, TDK, Triscend, Winbond, etc. are the key players in the microcontroller market.

Microprocessor vs. Microcontroller

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, 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
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
Limited power saving options compared to microcontrollers	Includes lot of power saving features

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.
- 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.

Digital Signal Processors (continued)

- 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.
  - It 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.

RISC vs. CISC Processors/Controllers

- RISC stands for Reduced Instruction Set Computing.
  - All RISC processors/controllers possess lesser number of instructions, typically in the range of 30 to 40.
  - E.g.: Atmel AVR microcontroller – its instruction set contains only 32 instructions.
- CISC stands for Complex Instruction Set Computing.
  - The instruction set is complex and instructions are high in number.
  - E.g.: 8051 microcontroller – its instruction set contains 255 instructions.

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

Harvard vs. Von-Neumann Processor/Controller Architecture

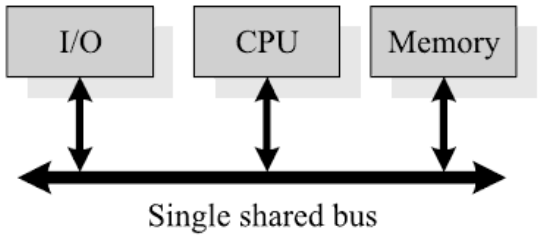
- Von-Neumann Architecture
  - Microprocessors/controllers based on the Von-Neumann architecture share a single common bus for fetching both instructions and data.
  - Program instructions and data are stored in a common main memory.
  - They first fetch an instruction and then fetch the data to support 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.

Harvard vs. Von-Neumann Processor/Controller Architecture (continued)

- Harvard Architecture
  - 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.
  - 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 ("pre-fetching").
    - The pre-fetch theoretically allows much faster execution than Von-Neumann architecture.

Harvard vs. Von-Neumann Processor/Controller Architecture (continued)

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

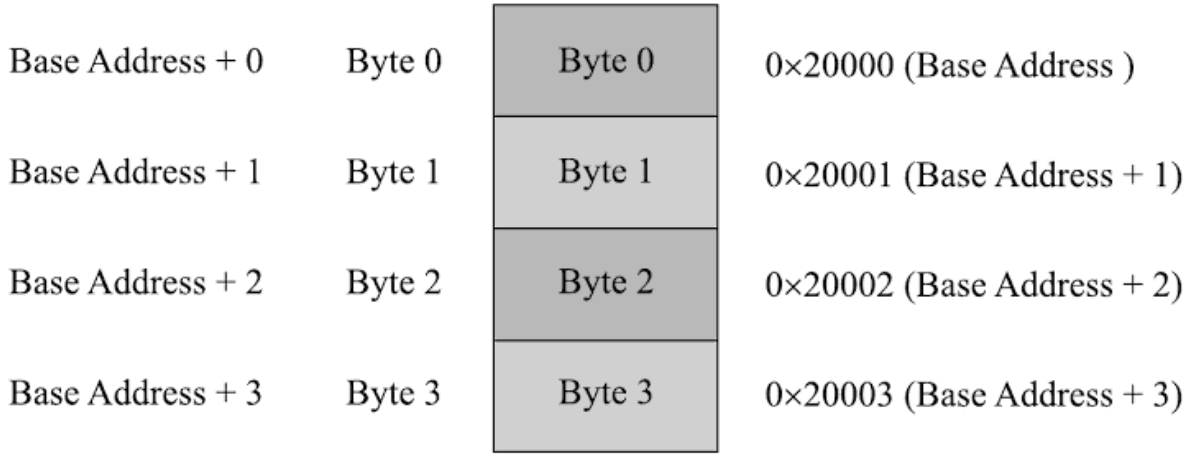


Big-Endian vs. Little-Endian Processors/Controllers

- Endianness specifies the order in which the data is stored in the memory by processor operations in a multi byte system.
- 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 – Little-Endian
     - E.g.: Intel x86 Processors
  2. Lower order of data byte at the higher memory and higher order of data byte at location just below the higher memory – Big-Endian
     - E.g.: Motorola 68000 Series Processors

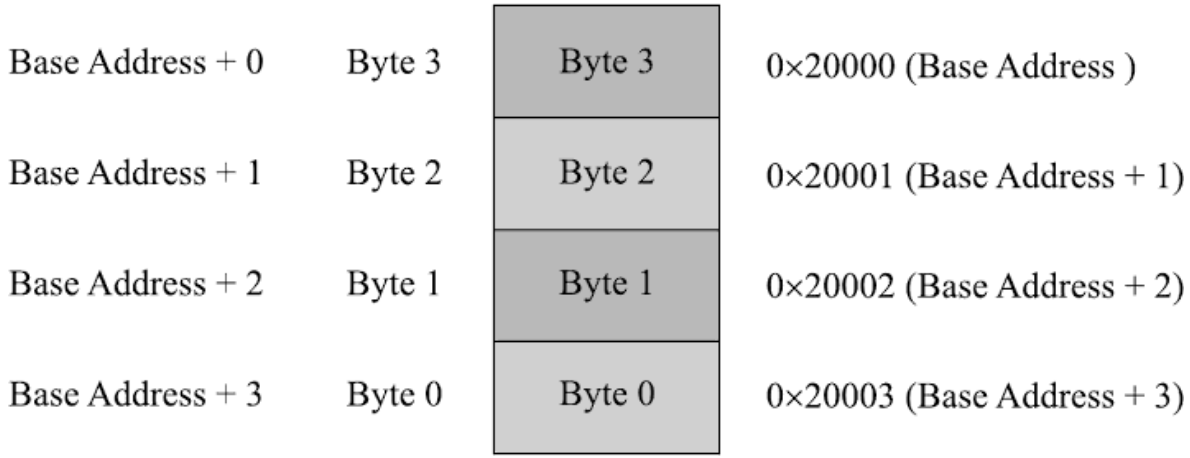
Big-Endian vs. Little-Endian Processors/Controllers (continued)

- Little-endian 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:



Big-Endian vs. Little-Endian Processors/Controllers (continued)

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



Load Store Operation and Instruction Pipelining

- The memory access related operations are performed by the special instructions load and store.
  - If the operand is specified as memory location, the content of it is loaded to a register using the load instruction.
  - The instruction store stores data from a specified register to a specified memory location.
- The concept of Load Store Architecture is illustrated with the following example:
  - Suppose *x*, *y* and *z* are memory locations and we want to add the contents of *x* and *y* and store the result in location z. Under the load store architecture the same is achieved with 4 instructions as shown:

Load Store Operation and Instruction Pipelining (continued)

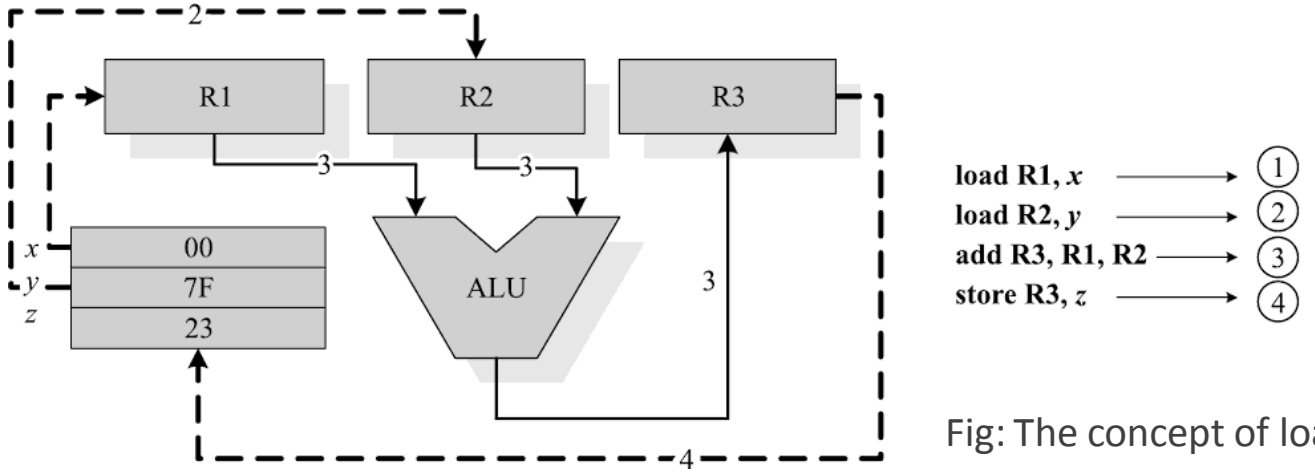


Fig: The concept of load store architecture

- The first instruction *load R1, x* loads the register R1 with the content of memory location *x*.
- The second instruction *load R2, y* loads the register R2 with the content of memory location *y*.
- The instruction *add R3, R1, R2* adds the content of registers R1 and R2 and stores the result in register R3.
- The next instruction *store R3,z* stores the content of register R3 in memory location *z*.

Load Store Operation and Instruction Pipelining (continued)

- The conventional instruction execution by the processor follows the fetch-decode-execute sequence.
  - The fetch part fetches the instruction from program memory or code memory.
  - The decode part decodes the instruction to generate the necessary control signals.
  - The execute stage reads the operands, perform ALU operations and stores the result.
- In conventional program execution, the fetch and decode operations are performed in sequence. For simplicity let's consider decode and execution together.

Load Store Operation and Instruction Pipelining (continued)

- During the decode operation, the memory address bus is available and if it is possible to effectively utilise it for an instruction fetch, the processing speed can be increased.
- Instruction pipelining refers to the overlapped execution of instructions – i.e., while the current instruction is being decoded and executed, the next instruction will be fetched.
- If the current instruction in progress is a program control flow transfer instruction like jump or call instruction, the instruction fetched is flushed and a new instruction fetch is performed to fetch the instruction.
- Whenever the current instruction is executing the program counter will be loaded with the address of the next instruction.
- In case of jump or branch instruction, the new location is known only after completion of the jump or branch instruction.

Load Store Operation and Instruction Pipelining (continued)

- Depending on the stages involved in an instruction (fetch, read register and decode, execute instruction, access an operand in data memory, write back the result to register, etc.), there can be multiple levels of instruction pipelining.
- Figure illustrates the concept of instruction pipelining for single stage pipelining.

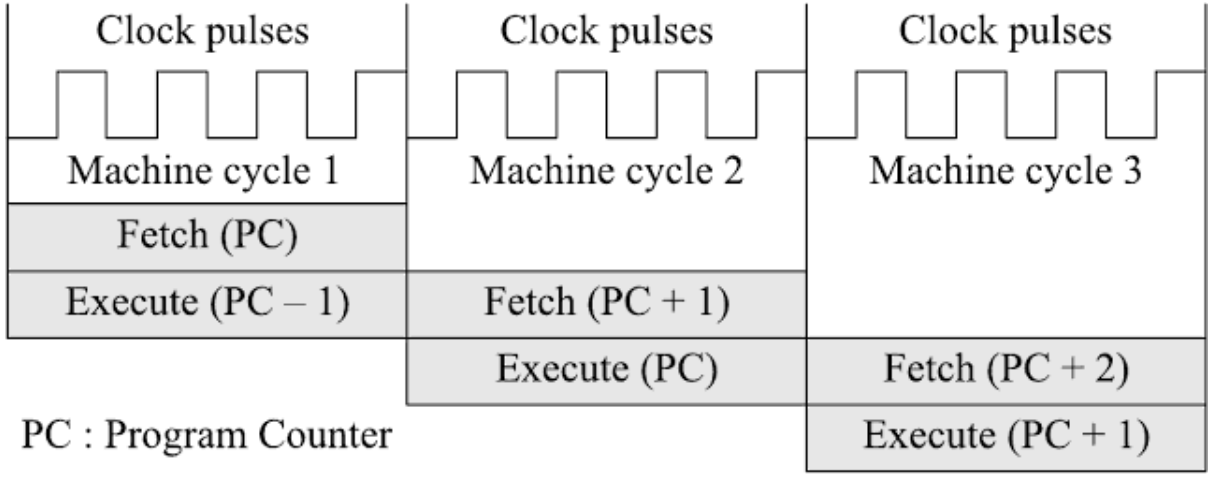


Fig: The single-stage pipelining concept

Application Specific Integrated Circuits (ASICs)

- Application Specific Integrated Circuit (ASIC) is a microchip designed to perform a specific or unique application.
  - Used as replacement to conventional general purpose logic chips.
- It integrates several functions into a single chip and there by reduces the system development cost.
- ASIC consumes a very small area in the total system.
  - Helps in the design of smaller systems with high capabilities/functionalities.
- Fabrication of ASICs requires a non-refundable initial investment for the process technology and configuration expenses. This investment is known as Non-Recurring Engineering Charge (NRE) and it is a one time investment.
- If the Non-Recurring Engineering Charges (NRE) is borne by a third party and the ASIC is made openly available in the market, the it is referred as Application Specific Standard Product (ASSP).
  - E.g.: ADE7760 Energy Meter ASIC developed by Analog Devices for Energy metering applications

Programmable Logic Devices

- Logic devices provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, timing and control operations, and almost every other function a system must perform.
- Logic devices can be classified into two broad categories—fixed and programmable.
- The circuits in a fixed logic device are permanent, they perform one function or set of functions—once manufactured, they cannot be changed.
- Programmable Logic Devices (PLDs) offer customers a wide range of logic capacity, features, speed, and voltage characteristics and these devices can be re-configured to perform any number of functions at any time.
  - Designers use inexpensive software tools to quickly develop, simulate, and test their designs.
  - Then, a design can be quickly programmed into a device, and immediately tested in a live circuit.

Programmable Logic Devices (continued)

- There are no NRE costs and the final design is completed much faster than that of a custom, fixed logic device.
- Another key benefit of using PLDs is that during the design phase customers can change the circuitry as often as they want until the design operates to their satisfaction.
  - PLDs are based on re-writable memory technology to change the design, the device is simply reprogrammed.
- Once the design is final, customers can go into immediate production by simply programming as many PLDs as they need with the final software design file.
- The two major types of programmable logic devices are Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs).

CPLDs and FPGAs

FPGAs

- The FPGAs offer the highest amount of logic density, the most features, and the highest performance.
- The largest FPGA now shipping, part of the Xilinx Virtex line of devices, provides eight million "system gates" (the relative density of logic).
- These advanced devices also offer features such as built-in hardwired processors (such as the IBM power PC), substantial amounts of memory, clock management systems, and support for many of the latest, very fast device-to-device signaling technologies.
- FPGAs are used in a wide variety of applications ranging from data processing and storage, to instrumentation, telecommunications, and digital signal processing.

CPLDs

- CPLDs offer much smaller amounts of logic—up to about 10,000 gates.
- But CPLDs offer very predictable timing characteristics and are therefore ideal for critical control applications.
- CPLDs such as the Xilinx CoolRunner series also require extremely low amounts of power and are very inexpensive, making them ideal for cost-sensitive, battery-operated, portable applications such as mobile phones and digital handheld assistants.

Advantages of PLD

- PLDs offer customers much more flexibility during the design cycle because design iterations are simply a matter of changing the programming file, and the results of design changes can be seen immediately in working parts.
- PLDs do not require long lead times for prototypes or production parts—the PLDs are already on a distributor's shelf and ready for shipment.
- PLDs do not require customers to pay for large NRE costs and purchase expensive mask sets—PLD suppliers incur those costs when they design their programmable devices and are able to amortize those costs over the multi-year lifespan of a given line of PLDs.
- PLDs allow customers to order just the number of parts they need, when they need them, allowing them to control inventory.
- PLDs can be reprogrammed even after a piece of equipment is shipped to a customer. The manufacturers can add new features or upgrade products that already are in the field. To do this, they simply upload a new programming file to the PLD, via the Internet, creating new hardware logic in the system.

Commercial Off-the-Shelf Components (COTS)

- A Commercial Off-the-Shelf (COTS) product is one which is used 'as-is’.
- COTS products are designed in such a way to provide easy integration and interoperability with existing system components.
- The COTS component itself may be developed around a general purpose or domain specific processor or an Application Specific Integrated Circuit or a Programmable Logic Device.
- Advantages:
  - They are readily available in the market
  - Cheap
  - Developer can cut down his development time to a great extent
  - Reduces the time to market

Commercial Off-the-Shelf Components (COTS) (continued)

- Typical examples of COTS hardware unit are remote controlled toy car control units including the RF circuitry part, high performance, high frequency microwave electronics (2—200 GHz), high bandwidth analog-to-digital converters, devices and components for operation at very high temperatures, electro-optic IR imaging arrays, UV/IR detectors, etc.
- E.g.: The TCP/IP plug-in module available from various manufactures like 'WIZnet', 'Freescale', 'Dynalog', etc. are very good examples of COTS product.

Fig: An example of a COTS product for TCP/IP plug-in from WIZnet (WIZnet NM7010A Plug in Module)

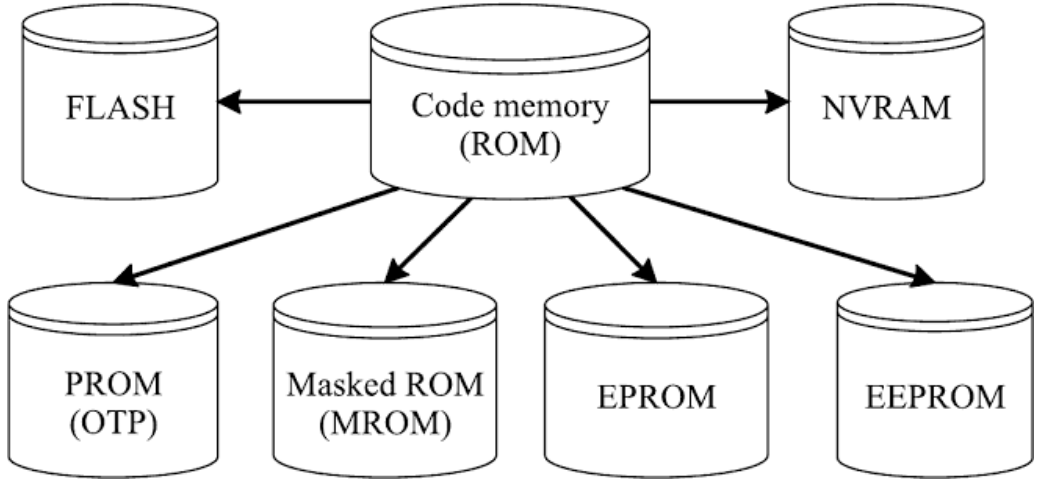
Memory

Memory

- Memory is an important part of a processor/controller based embedded systems.
- Some of the processors/controllers contain built in memory and this memory is referred as on-chip memory.
- Others do not contain any memory inside the chip and requires external memory to be connected with the controller/processor to store the control algorithm. It is called off-chip memory.
- Also some working memory is required for holding data temporarily during certain operations.

Program Storage Memory (ROM)

- The program memory or code storage memory of an embedded system stores the program instructions.
- The code memory retains its contents even after the power is turned off. It is generally known as non-volatile storage memory.
- It can be classified into different types as shown:



Masked ROM (MROM)

- Masked ROM is a one-time programmable device.
- Masked ROM makes use of the hardwired technology for storing data.
- The device is factory programmed by masking and metallisation process at the time of production itself, according to the data provided by the end user.
- Advantage – low cost for high volume production.
- Limitation – inability to modify the device firmware against firmware upgrades.
  - Since the MROM is permanent in bit storage, it is not possible to alter the bit information.

Masked ROM (MROM) (continued)

- Different mechanisms are used for the masking process of the ROM, like
  1. Creation of an enhancement or depletion mode transistor through channel implant.
  2. By creating the memory cell either using a standard transistor or a high threshold transistor.
     - In the high threshold mode, the supply voltage required to turn ON the transistor is above the normal ROM IC operating voltage.
     - This ensures that the transistor is always off and the memory cell stores always logic 0.

Programmable Read Only Memory (PROM) / (OTP)

- One Time Programmable Memory (OTP) or PROM is not pre-programmed by the manufacturer.
  - The end user is responsible for programming these devices.
- This memory has nichrome or polysilicon wires arranged in a matrix. These wires can be functionally viewed as fuses.
  - It is programmed by a PROM programmer which selectively burns the fuses according to the bit pattern to be stored.
  - Fuses which are not blown/burned represents a logic “1" whereas fuses which are blown/burned represents a logic 0 .
  - The default state is logic "1".
- OTP is widely used for commercial production of embedded systems whose prototyped versions are proven and the code is finalised.
  - It is a low cost solution for commercial production.
- OTPs cannot be reprogrammed.

Erasable Programmable Read Only Memory (EPROM)

- Erasable Programmable Read Only Memory (EPROM) gives the flexibility to re-program the same chip.
- EPROM stores the bit information by charging the floating gate of an FET.
- Bit information is stored by using an EPROM programmer, which applies high voltage to charge the floating gate.
- EPROM contains a quartz crystal window for erasing the stored information.
  - If the window is exposed to ultraviolet rays for a fixed duration, the entire memory will be erased.
- Even though the EPROM chip is flexible in terms of re-programmability, it needs to be taken out of the circuit board and put in a UV eraser device for 20 to 30 minutes.
  - It is a tedious and time-consuming process.

Electrically Erasable Programmable Read only Memory (EEPROM)

- The information contained in the EEPROM memory can be altered by using electrical signals at the register/byte level.
- They can be erased and reprogrammed in-circuit.
- These chips include a chip erase mode and in this mode they can be erased in a few milliseconds.
- It provides greater flexibility for system design.
- The only limitation is their capacity is limited (a few kilobytes) when compared with the standard ROM.

FLASH

- FLASH memory is a variation of EEPROM technology – It combines the re-programmability of EEPROM and the high capacity of standard ROMs.
- FLASH is the latest ROM technology.
  - Most popular ROM technology used in today's embedded designs.
- FLASH memory is organised as sectors (blocks) or pages.
- FLASH memory stores information in an array of floating gate MOSFET transistors.
- The erasing of memory can be done at sector level or page level without affecting the other sectors or pages.
- Each sector/page should be erased before re-programming.
- The typical erasable capacity of FLASH is 1000 cycles.
- E.g.: W27C512 from WINBOND is an example of 64KB FLASH memory.

Non-Volatile RAM (NVRAM)

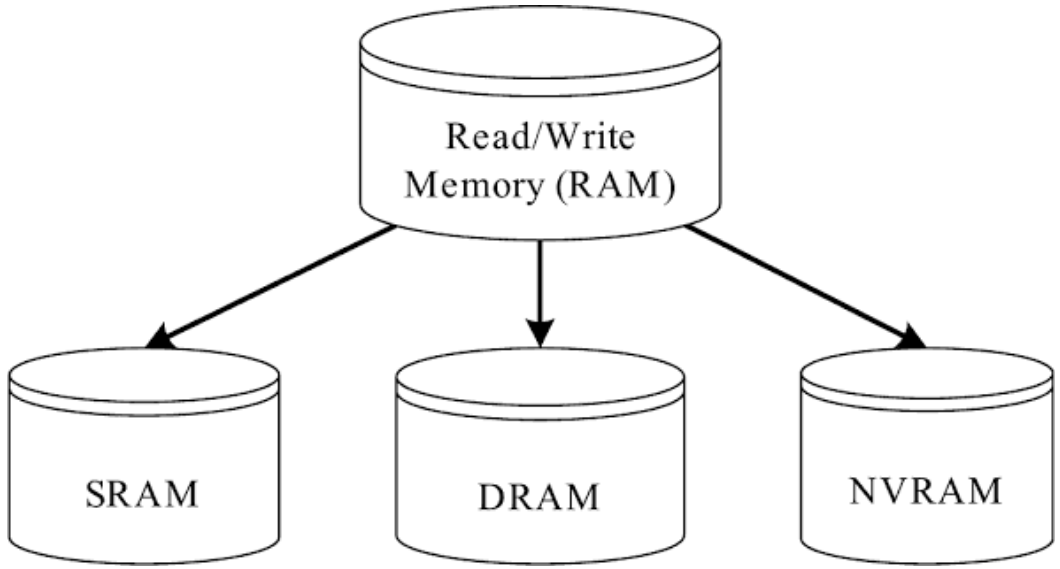
- Non-volatile RAM is a random access memory with battery backup.
- It contains static RAM based memory and a minute battery for providing supply to the memory in the absence of external power supply.
- The memory and battery are packed together in a single package.
- The life span of NVRAM is expected to be around 10 years.
- E.g.: DS1644 from Maxim/Dallas is an example of 32KB NVRAM.

Read-Write Memory/Random Access Memory (RAM)

- RAM is the data memory or working memory of the controller/processor.
- Controller/processor can read from it and write to it.
- RAM is volatile – when the power is turned off, all the contents are destroyed.
- RAM is a direct access memory – we can access the desired memory location directly without the need for traversing through the entire memory locations to reach the desired memory position (i.e. random access of memory location).
  - This is in contrast to the Sequential Access Memory (SAM), where the desired memory location is accessed by either traversing through the entire memory or through a 'seek' method. Magnetic tapes, CD ROMs, etc. are examples of sequential access memories.

Read-Write Memory/Random Access Memory (RAM) (continued)

- RAM generally falls into three categories: Static RAM (SRAM), Dynamic RAM (DRAM) and Non-Volatile RAM (NVRAM).



Static RAM (SRAM)

- Static RAM stores data in the form of voltage.
- They are made up of flip-flops.
- Static RAM is the fastest form of RAM available.
  - Fast due to its resistive networking and switching capabilities.
- In typical implementation, an SRAM cell (bit) is realised using six transistors (or 6 MOSFETs).
  - Four of the transistors are used for building the latch (flip-flop) part of the memory cell and two for controlling the access.

Static RAM (SRAM) (continued)

- In its simplest representation an SRAM cell can be visualised as shown in the figure below:

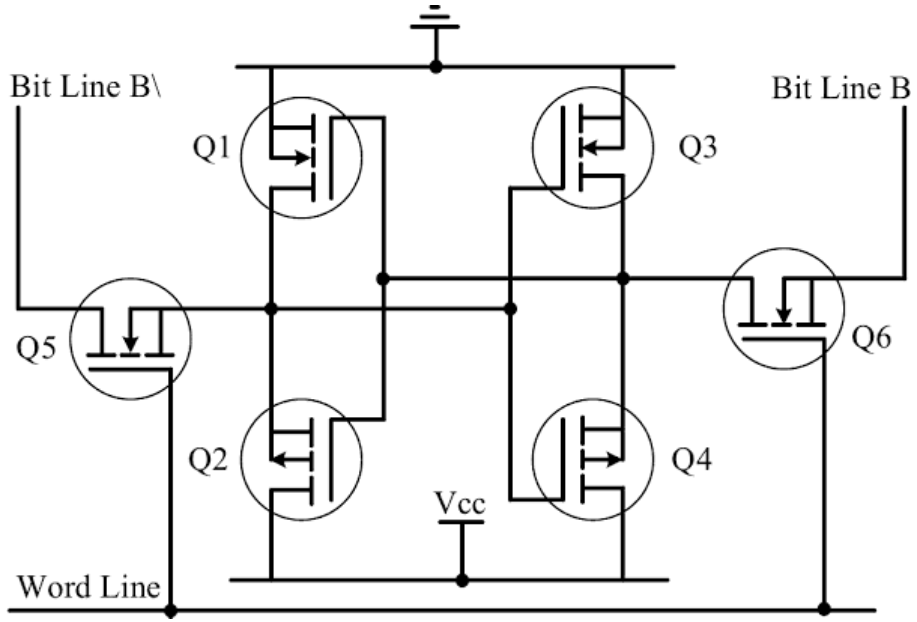


Fig: SRAM cell implementation

Static RAM (SRAM) (continued)

- This implementation in its simpler form can be visualised as two cross-coupled inverters with read/write control through transistors.
  - The four transistors in the middle form the cross-coupled inverters.
- This can be visualised as shown in the figure below:

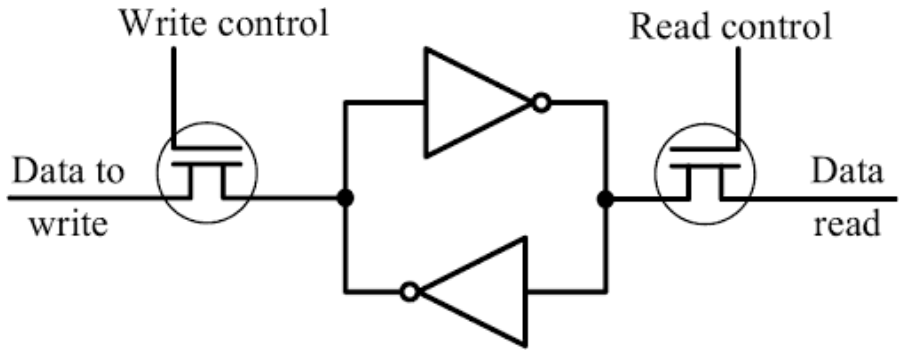


Fig: Visualisation of SRAM cell

Static RAM (SRAM) (continued)

- The access to the memory cell is controlled by Word Line, which controls the access transistors (MOSFETs) Q5 and Q6.
  - The access transistors control the connection to bit lines B & B\.
- In order to write a value to the memory cell, apply the desired value to the bit control lines (For writing 1, make B = 1 and B\ = 0; For writing 0, make B = 0 and B\ = 1) and assert the Word Line (Make Word line high).
  - This operation latches the bit written in the flip-flop.
- For reading the content of the memory cell, assert both B and B\ bit lines to 1 and set the Word line to 1.
- The major limitations of SRAM are low capacity and high cost.

Dynamic RAM (DRAM)

- Dynamic RAM stores data in the form of charge.
- They are made up of MOS transistor gates.
- Advantages – high density and low cost compared to SRAM.
- Disadvantage – since the information is stored as charge it gets leaked off with time and to prevent this they need to be refreshed periodically.
- Special circuits called DRAM controllers are used for the refreshing operation.
- The refresh operation is done periodically in milliseconds interval.

Dynamic RAM (DRAM) (continued)

- Figure below illustrates the typical implementation of a DRAM cell.
- The MOSFET acts as the gate for the incoming and outgoing data whereas the capacitor acts as the bit storage unit.

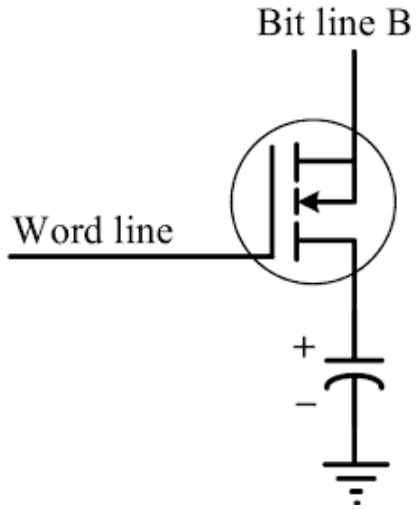


Fig: DRAM cell implementation

SRAM vs DRAM

- Table given below summarises the relative merits and demerits of SRAM and DRAM technology.

SRAM Cell	DRAM Cell
Made up of 6 CMOS transistors (MOSFET)	Made up of a MOSFET and a capacitor
Doesn't require refreshing	Requires refreshing
Low capacity (Less dense)	High capacity (Highly dense)
More expensive	Less expensive
Fast in operation. Typical access time is 10ns	Slow in operation due to refresh requirements. Typical access time is 60ns. Write operation is faster than read operation.

Memory According to the Type of Interface

- The interface (connection) of memory with the processor/controller can be of various types.
  - Parallel interface
  - Serial interface like I2C
  - SPI (Serial peripheral interface)
  - Single wire interconnection (like Dallas 1-Wire interface)
- Serial interface is commonly used for data storage memory like EEPROM.
- The memory density of a serial memory is usually expressed in terms of kilobits, whereas that of a parallel interface memory is expressed in terms of kilobytes.
- Atmel Corporations AT24C512 is an example for serial memory with capacity 512 kilobits and 2-wire interface.

Memory Shadowing

- Generally the execution of a program or a configuration from a ROM is very slow (120 to 200 ns) compared to the execution from a RAM(40 to 70 ns).
  - RAM access is about three times as fast as ROM access.
- Memory Shadowing is a technique adopted to solve the execution speed problem in processor-based systems.
- In computer systems and video systems there will be a configuration holding ROM called Basic Input Output Configuration ROM or simply BIOS.
  - BIOS stores the hardware configuration information like the address assigned for various serial ports and other non-plug 'n' play devices, etc.
  - Usually it is read and the system is configured according to it during system boot up and it is time consuming.

Memory Shadowing (continued)

- In memory shadowing, a RAM is included behind the logical layer of BIOS at its same address as a shadow to the BIOS.
- The first step that happens during the boot up is copying the BIOS to the shadowed RAM and write protecting the RAM then disabling the BIOS reading.
  - RAM is volatile and it cannot hold the configuration data which is copied from the BIOS when the power supply is switched off. Only a ROM can hold it permanently.
  - But for high system performance it should be accessed from a RAM instead of accessing from a ROM.

Memory Selection for Embedded Systems

- Embedded systems require a
  - Program memory for holding the control algorithm or embedded OS and the applications designed to run on top of it
  - Data memory for holding variables and temporary data during task execution
  - Memory for holding non-volatile data (like configuration data, look up table etc) which are modifiable by the application
- The memory requirement for an embedded system in terms of RAM and ROM (EEPROM/FLASH/NVRAM) is solely dependent on the type of the embedded system and the applications for which it is designed.
  - Lot of factors need to be considered when selecting the type and size of memory for embedded system.

Memory Selection for Embedded Systems (continued)

- For example, if the embedded system is designed using SOC or a microcontroller with on-chip RAM and ROM (FLASH/EEPROM), depending on the application need the on-chip memory may be sufficient for designing the total system.
- Consider a simple electronic toy design as an example.
  - As the complexity of requirements are less and data memory requirement are minimal, we can think of a microcontroller with a few bytes of internal RAM, a few bytes or kilobytes of FLASH memory and a few bytes of EEPROM (if required) for designing the system. Hence there is no need for external memory at all.
  - A PIC microcontroller device which satisfies the I/O and memory requirements can be used in this case.

Memory Selection for Embedded Systems (continued)

- If the embedded design is based on an RTOS, the RTOS requires certain amount of RAM for its execution and ROM for storing the RTOS image.
- Normally the binary code for RTOS kernel containing all the services is stored in a non-volatile memory (Like FLASH) as either compressed or non-compressed data.
- During boot-up of the device, the RTOS files are copied from the program storage memory, decompressed if required and then loaded to the RAM for execution.
- A smart phone device with Windows mobile operating system is a typical example for embedded device with OS.
  - Say 64MB RAM and 128MB ROM are the minimum requirements for running the Windows mobile device, extra RAM and ROM are needed for running user applications.

Memory Selection for Embedded Systems (continued)

- There are two parameters for representing a memory:
  1. Size of the memory chip – Memory density expressed in terms of number of memory bytes per chip.
     - Memory chips come in standard sizes like 512 bytes, 1024 bytes (1 kilobyte), 2048 bytes (2 kilobytes), 4KB, 8KB, 16KB, 32KB, 64KB, 128KB, 256KB, 512KB, 1024KB(1 megabytes), etc.
     - While selecting a memory size, the address range supported by the processor must also be considered. For example, for a processor/controller with 16 bit address bus, the maximum number of memory locations that can be addressed is 2^{16} = 65536 bytes = 64KB
     - The entire memory range supported by the processor/controller may not be available to the memory chip alone. It may be shared between I/O, other ICs and memory.
  2. Word size of the memory – The number of memory bits that can be read/written together at a time.
     - Word size can be 4, 8, 12, 16, 24, 32, etc.
     - The word size supported by the memory chip must match with the data bus width of the processor/controller.

Memory Selection for Embedded Systems (continued)

- FLASH memory is the popular choice for ROM in embedded applications.
- It is a powerful and cost-effective solid-state storage technology for mobile electronics devices and other consumer applications.
- FLASH memory comes in two major variants – NAND FLASH and NOR FLASH.
- NAND FLASH is a high-density low cost non-volatile storage memory, while NOR FLASH is less dense and slightly expensive.
- NOR FLASH supports the Execute in Place (XIP) technique for program execution.
  - The XIP technology allows the execution of code memory from ROM itself without the need for copying it to the RAM as in the case of conventional execution method.
- It is a good practice to use a combination of NOR and NAND memory for storage memory requirements, where NAND can be used for storing the program code and/or data like the data captured in a camera device.
  - NAND FLASH doesn't support XIP and if NAND FLASH is used for storing program code, a DRAM can be used for copying and executing the program code.
  - NOR FLASH supports XIP and it can be used as the memory for bootloader or for even storing the complete program code.

Memory Selection for Embedded Systems (continued)

- 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.

Memory Selection for Embedded Systems (continued)

- 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.

Sensors and Actuators

Sensors and Actuators

- 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.

Sensors and Actuators (continued)

- A sensor is a transducer device that converts energy from one form to another for any measurement or control purpose.
  - E.g.: Temperature sensor, magnetic hall effect sensor, humidity sensor, etc.
- An actuator is a form of transducer device (mechanical or electrical) which converts signals to corresponding physical action (motion).
  - Actuator acts as an output device.
  - E.g.: Stepper motor

The I/O Subsystem

- The I/O subsystem of the embedded system facilitates the interaction of the embedded system with the external world.
- 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.

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.
  - E.g.: 'Device ON', 'Battery low' or 'Charging of battery' conditions
- Light Emitting Diode is a p-n junction diode and it contains an anode and a cathode.
- For proper functioning of the LED, the anode is 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 to limit the current through the LED.
- The ideal LED interfacing circuit is shown in the figure.

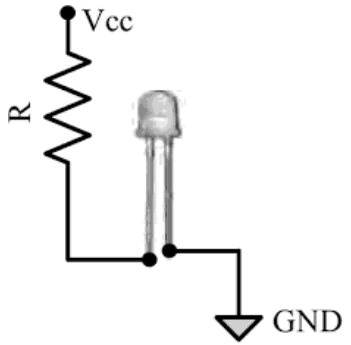


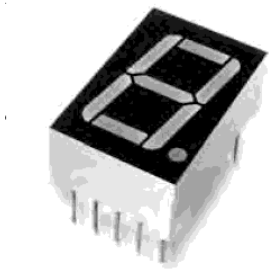
Fig: LED interfacing

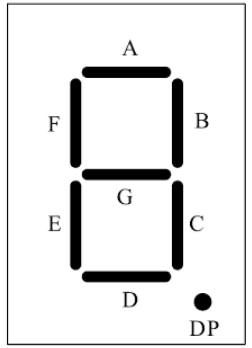
Light Emitting Diode (LED) (continued)

- 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.
    - 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.

7-Segment LED Display

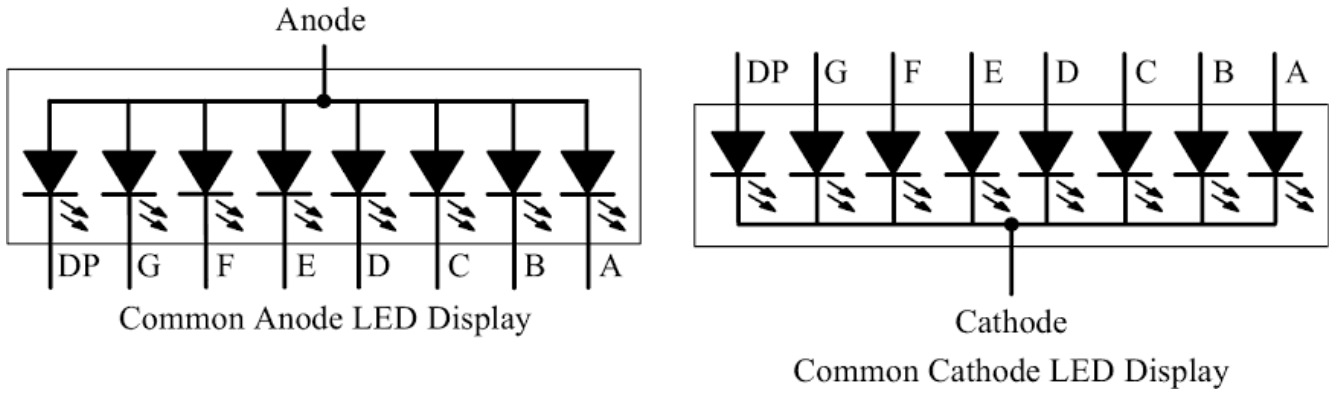
- The 7-segment LED display is an output device for displaying alpha numeric characters.
- It contains 7 LED segments arranged in a special form used for displaying alpha numeric characters and 1 LED used for representing 'decimal point' in decimal number display.
- The LED segments are named A to G 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.





7-Segment LED Display (continued)

- 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 cathodes of 8 LED segments are connected commonly.
- Figure illustrates the Common Anode and Cathode configurations.



7-Segment LED Display (continued)

- 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.
- 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 is 20mA.
  - The current can be limited by connecting a current limiting resistor to the anode or cathode of each segment.
- 7-segment LED display is used in low cost embedded applications like Public telephone call monitoring devices, point of sale terminals, etc.

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.
- Figure illustrates the functioning of an optocoupler device.

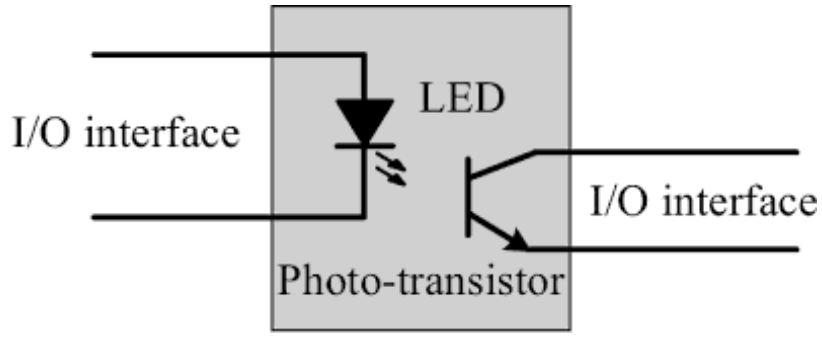


Fig: An optocoupler device

Optocoupler (continued)

- 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.
- Optocoupler is available as ICs from different semiconductor manufacturers.
  - The MCT2M IC from Fairchild semiconductor is an example for optocoupler IC.

Optocoupler (continued)

- Figure illustrates the usage of optocoupler in input circuit and output circuit of an embedded system with a microcontroller as the system core.

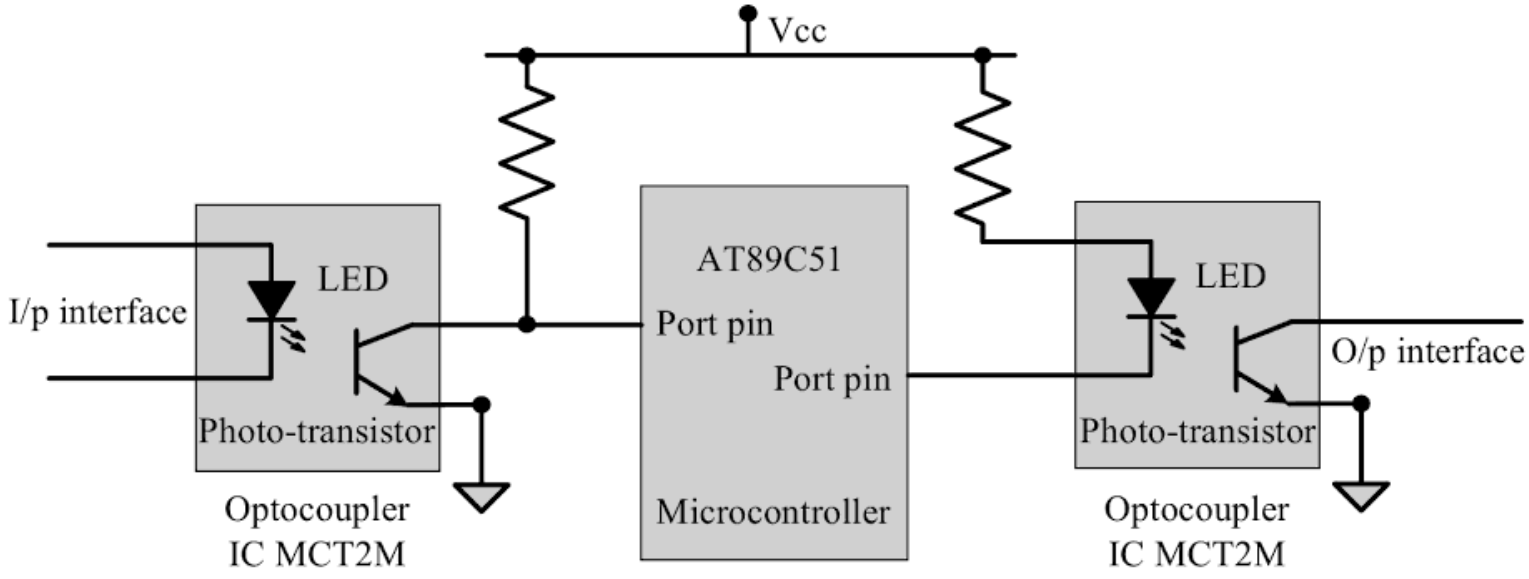


Fig: Optocoupler in Input and Output circuit

Relay

- Relay is an electro-mechanical device.
- In embedded application, the Relay unit acts as dynamic path selector for signals and power.
- The Relay unit contains a relay coil made up of insulated wire on a metal core and a metal armature with one or more contacts.
- Relay works on electromagnetic principle.
  - When a voltage is applied to the relay coil, current flows through the coil, which in turn generates a magnetic field.
  - The magnetic field attracts the armature core and moves the contact point.
  - The movement of the contact point changes the power/signal flow path.

Relay (continued)

- Relays are available in different configurations.
- Figure given below illustrates the widely used relay configurations for embedded applications.

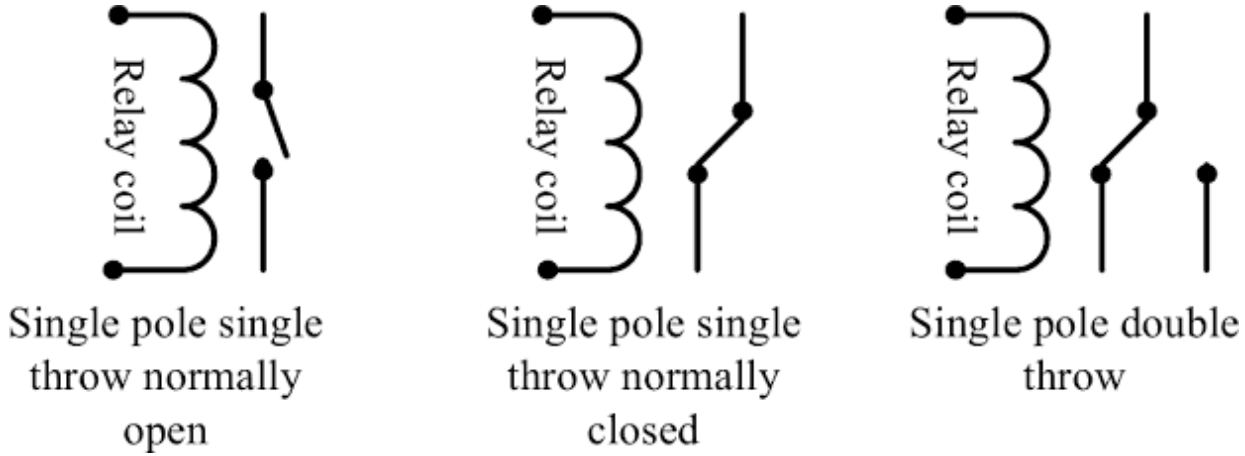


Fig: Relay configurations

Relay (continued)

- The Single Pole Single Throw configuration has only one path for information flow.
- The path is either open or closed in normal condition.
  - For Normally Open Single Pole Single Throw relay, the circuit is normally open and it becomes closed when the relay is energised.
  - For Normally Closed Single Pole Single Throw relay, the circuit is normally closed and it becomes open when the relay is energised.
- For Single Pole Double Throw configuration, there are two paths for information flow and they are selected by energising or de-energising the relay.

Relay (continued)

- The Relay is normally controlled using a relay driver circuit connected to the port pin of the processor/controller.
- A transistor is used for building the relay driver circuit as shown in the figure.
- A free-wheeling diode is used for free-wheeling the voltage produced in the opposite direction when the relay coil is de-energised.
- The freewheeling diode is essential for protecting the relay and the transistor.

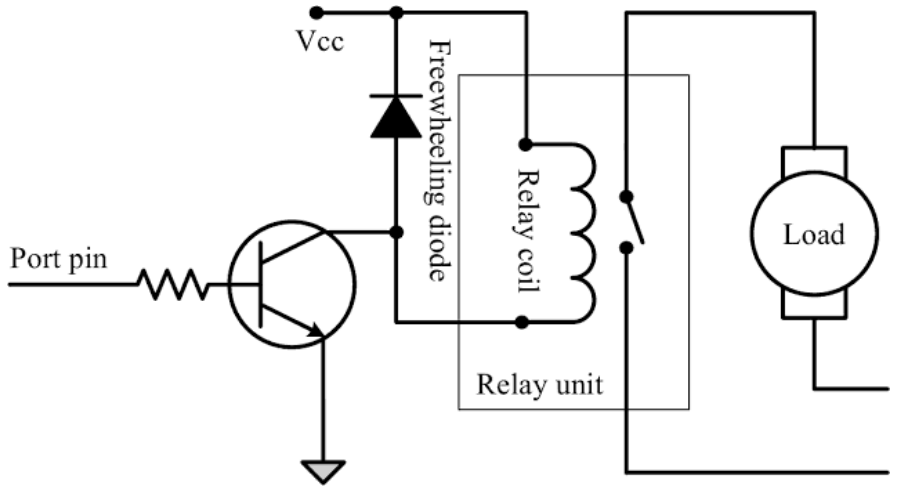


Fig: Transistor based Relay driving circuit

- Most of the industrial relays are bulky and require high voltage to operate.
- Special relays called 'Reed' relays are available for embedded application requiring switching of low voltage DC signals.

Piezo Buzzer

- Piezo buzzer is a piezoelectric device for generating audio indications in embedded application.
- A piezoelectric buzzer contains a piezoelectric diaphragm which produces audible sound in response to the voltage applied to it.
- Piezoelectric buzzers are available in two types – 'Self-driving' and 'External driving'.
- The Self-driving circuit contains all the necessary components to generate sound at a predefined tone.
  - It will generate a tone on applying the voltage.
- External driving piezo buzzers support the generation of different tones.
  - The tone can be varied by applying a variable pulse train to the piezoelectric buzzer.
- A piezo buzzer can be directly interfaced to the port pin of the processor/control.
- Depending on the driving current requirements, the piezo buzzer can also be interfaced using a transistor based driver circuit as in the case of a 'Relay'.

Push Button Switch

- It is an input device.
- Push button switch comes in two configurations, namely 'Push to Make' and 'Push to Break’.
- In the 'Push to Make' configuration, the switch is normally in the open state and it makes a circuit contact when it is pushed or pressed.
- In the 'Push to Break' configuration, the switch is normally in the closed state and it breaks the circuit contact when it is pushed or pressed.
- The push button stays in the 'closed' (For Push to Make type) or 'open' (For Push to Break type) state as long as it is kept in the pushed state and it breaks/makes the circuit connection when it is released.

Push Button Switch (continued)

- Push button is used for generating a momentary pulse.
- In embedded applications, push button is generally used as reset and start switch and pulse generator.
- The Push button is normally connected to the port pin of the host processor/controller.
- Depending on the way in which the push button interfaced to the controller, it can generate either a 'HIGH' pulse or a 'LOW' pulse.
- Figure illustrates how the push button can be used for generating 'LOW' and 'HIGH' pulses.

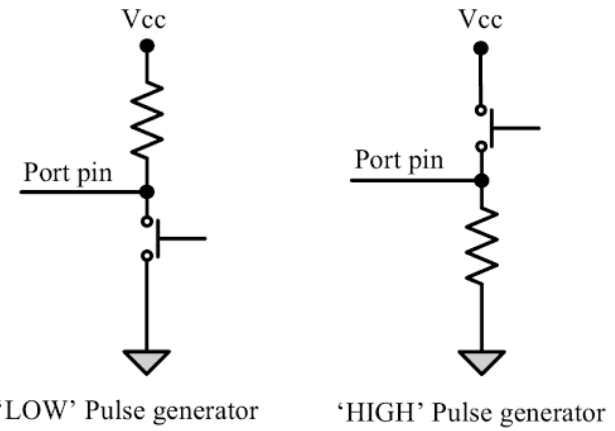


Fig: Push button switch configurations

Communication Interface

Communication Interface

- Communication interface is essential for communicating with various subsystems of the embedded system and with the external world.
- For an embedded product, the communication interface can be viewed in two different perspectives:
  - Onboard Communication Interface (Device/board level communication interface)
    - E.g.: Serial interfaces like I2C, SPI, UART, 1-Wire, etc and parallel bus interface.
  - External Communication Interface (Product level communication interface)
    - E.g.: Wireless interfaces like Infrared (IR), Bluetooth (BT), Wireless LAN (Wi-Fi), Radio Frequency waves (RF), GPRS, etc. and wired interfaces like RS-232C/RS-422/RS-485, USB, Ethernet IEEE 1394 port, Parallel port, CF-II interface, SDIO, PCMCIA, etc.

Onboard Communication Interfaces

- An embedded system is a combination of different types of components (chips/devices) arranged on a printed circuit board (PCB).
- Onboard Communication Interface refers to the different communication channels/buses for interconnecting the various integrated circuits and other peripherals within the embedded system.
- E.g.: Serial interfaces like I2C, SPI, UART, 1-Wire, etc and parallel bus interface

Inter Integrated Circuit (I2C) Bus

- The Inter Integrated Circuit Bus (I2C or I^2C Pronounced 'I square C') is a synchronous bi-directional half duplex two wire serial interface bus.
  - (Half duplex - one-directional communication at a given point of time)
- The concept of I2C bus was developed by Philips Semiconductors in the early 1980s.
- The original intention of I2C was to provide an easy way of connection between a microprocessor/microcontroller system and the peripheral chips in television sets.
- The I2C bus comprise of two bus lines:
  - Serial Clock (SCL line) – responsible for generating synchronisation clock pulses
  - Serial Data (SDA line) – responsible for transmitting the serial data across devices

Inter Integrated Circuit (I2C) Bus (continued)

- I2C bus is a shared bus system to which many number of I2C devices can be connected.
- Devices connected to the I2C bus can act as either 'Master' or 'Slave'.
  - The 'Master' device is responsible for controlling the communication by initiating/terminating data transfer, sending data and generating necessary synchronisation clock pulses.
  - 'Slave' devices wait for the commands from the master and respond upon receiving the commands.
- 'Master' and 'Slave' devices can act as either transmitter or receiver.
- Regardless whether a master is acting as transmitter or receiver, the synchronisation clock signal is generated by the 'Master' device only.
- I2C supports multi masters on the same bus.

Inter Integrated Circuit (I2C) Bus (continued)

- The following bus interface diagram illustrates the connection of master and slave devices on the I2C bus.

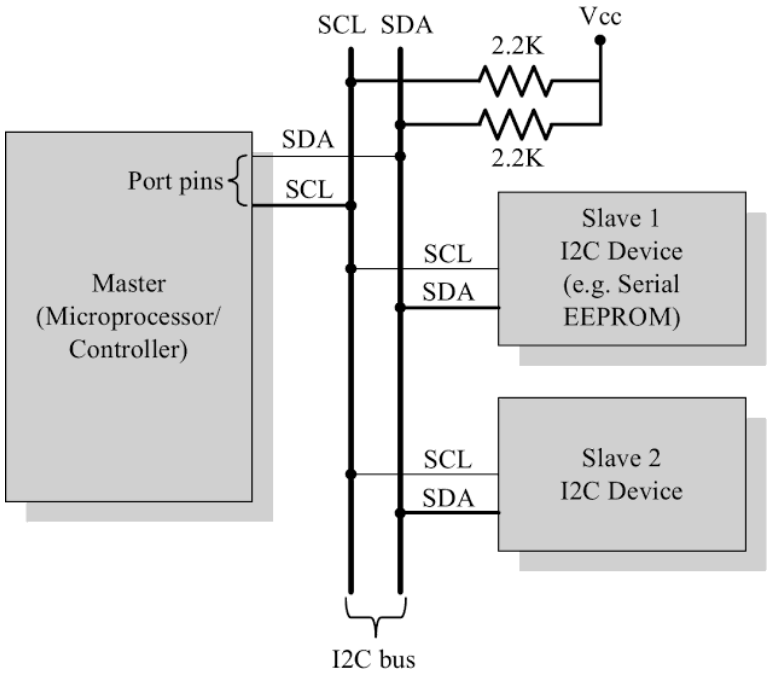


Fig: I2C Bus Interfacing

Inter Integrated Circuit (I2C) Bus (continued)

- The I2C bus interface is built around an input buffer and an open drain or collector transistor.
- When the bus is in the idle state, the open drain/collector transistor will be in the floating state and the output lines (SDA and SCL) switch to the 'High Impedance' state.
- For proper operation of the bus, both the bus lines should be pulled to the supply voltage (+5 V for TTL family and +3.3V for CMOS family devices) using pull-up resistors.
  - The typical value of resistors used in pull-up is 2.2K.
  - With pull-up resistors, the output lines of the bus in the idle state will be 'HIGH'
- The address of a I2C device is assigned by hardwiring the address lines of the device to the desired logic level.
  - Done at the time of designing the embedded hardware.

Inter Integrated Circuit (I2C) Bus (continued)

- The sequence of operations for communicating with an I2C slave device is listed below:
  1. The master device pulls the clock line (SCL) of the bus to 'HIGH'
  2. The master device pulls the data line (SDA) 'LOW', when the SCL line is at logic 'HIGH' (This is the 'Start' condition for data transfer)
  3. The master device sends the address (7 bit or 10 bit wide) of the 'slave' device to which it wants to communicate, over the SDA line.
     - Clock pulses are generated at the SCL line for synchronising the bit reception by the slave device.
     - The MSB of the data is always transmitted first.
     - The data in the bus is valid during the 'HIGH' period of the clock signal

Inter Integrated Circuit (I2C) Bus (continued)

4. The master device sends the Read or Write bit (Bit value = 1 Read operation; Bit value = 0 Write operation) according to the requirement
5. The master device waits for the acknowledgement bit from the slave device whose address is sent on the bus along with the Read/ Write operation command.
   - Slave devices connected to the bus compares the address received with the address assigned to them
6. The slave device with the address requested by the master device responds by sending an acknowledge bit (Bit value 1) over the SDA line
7. Upon receiving the acknowledge bit, the Master device sends the 8 bit data to the slave device over SDA line, if the requested operation is 'Write to device'.
   - If the requested operation is 'Read from device', the slave device sends data to the master over the SDA line
8. The master device waits for the acknowledgement bit from the device upon byte transfer complete for a write operation and sends an acknowledge bit to the Slave device for a read operation
9. The master device terminates the transfer by pulling the SDA line 'HIGH' when the clock line SCL is at logic 'HIGH' (Indicating the 'STOP' condition)

Inter Integrated Circuit (I2C) Bus (continued)

- I2C bus supports three different data rates:
  - Standard mode (Data rate up to 100kbits/sec (100 kbps))
  - Fast mode (Data rate up to 400kbits/sec (400 kbps))
  - High speed mode (Data rate up to 3.4Mbits/sec (3.4 Mbps))

Serial Peripheral Interface (SPI) Bus

- The Serial Peripheral Interface Bus (SPI) is a synchronous bi-directional full duplex four-wire serial interface bus.
- The concept of SPI was introduced by Motorola.
- SPI is a single master multi-slave system.
  - There can be more than one masters, but only one master device can be active at any given point of time.
- SPI requires four signal lines for communication. They are:
  - Master Out Slave In (MOSI) – Signal line carrying the data from master to slave device. It is also known as Slave Input/Slave Data In (SI/SDI)
  - Master In Slave Out (MISO) – Signal line carrying the data from slave to master device. It is also known as Slave Output (SO/SDO)
  - Serial Clock (SCLK) – Signal line carrying the clock signals
  - Slave Select (SS) – Signal line for slave device select. It is an active low signal

Serial Peripheral Interface (SPI) Bus (continued)

- The bus interface diagram shown in the figure illustrates the connection of master and slave devices on the SPI bus.

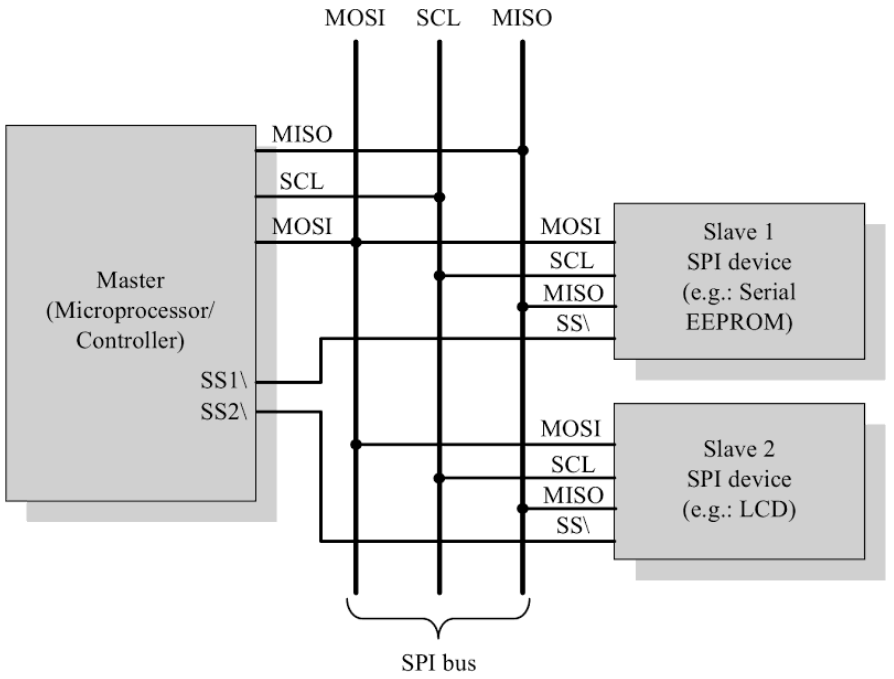


Fig: SPI Bus Interfacing

Serial Peripheral Interface (SPI) Bus (continued)

- The master device is responsible for generating the clock signal.
- It selects the required slave device by asserting the corresponding slave device's slave select signal 'LOW'.
- The data out line (MISO) of all the slave devices when not selected floats at high impedance state.
- The serial data transmission through SPI bus is fully configurable.
  - SPI devices contain a certain set of registers for holding these configurations.
  - The control register holds the various configuration parameters like master/slave selection for the device, baud rate selection for communication, clock signal control, etc.
  - The status register holds the status of various conditions for transmission and reception.

Serial Peripheral Interface (SPI) Bus (continued)

- SPI works on the principle of 'Shift Register’.
- The master and slave devices contain a special shift register for the data to transmit or receive.
  - The size of the shift register is device dependent. Normally it is a multiple of 8.
- During transmission from the master to slave, the data in the master's shift register is shifted out to the MOSI pin and it enters the shift register of the slave device through the MOSI pin of the slave device.
- At the same time the shifted out data bit from the slave device's shift register enters the shift register of the master device through MISO pin.
- In summary, the shift registers of 'master' and 'slave' devices form a circular buffer.
- When compared to I2C, SPI bus is most suitable for applications requiring transfer of data in ' streams '.
- The only limitation is SPI doesn't support an acknowledgement mechanism.

Universal Asynchronous Receiver Transmitter (UART)

- Universal Asynchronous Receiver Transmitter (UART) based data transmission is an asynchronous form of serial data transmission.
- It doesn't require a clock signal to synchronise the transmitting end and receiving end for transmission.
- Instead it relies upon the pre-defined agreement between the transmitting device and receiving device.

Universal Asynchronous Receiver Transmitter (UART) (continued)

- The serial communication settings (Baudrate, number of bits per byte, parity, number of start bits and stop bit and flow control) for both transmitter and receiver should be set as identical.
- The start and stop of communication is indicated through inserting special bits in the data stream.
- While sending a byte of data, a start bit is added first and a stop bit is added at the end of the bit stream.
- The least significant bit of the data byte follows the 'start' bit.

Universal Asynchronous Receiver Transmitter (UART) (continued)

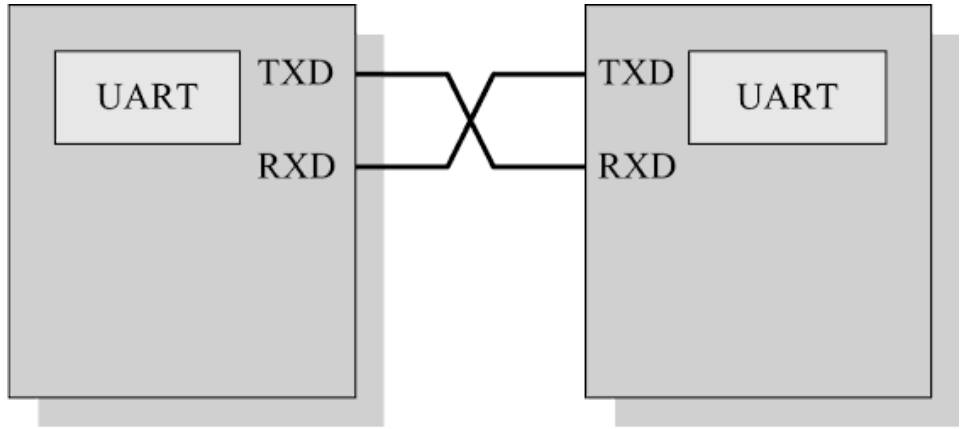
- The 'start' bit informs the receiver that a data byte is about to arrive.
- The receiver device starts polling its 'receive line' as per the baud rate settings.
  - If the baud rate is 'x' bits per second, the time slot available for one bit is 1/x seconds.
- The receiver unit polls the receiver line at exactly half of the time slot available for the bit.
- If parity is enabled for communication, the UART of the transmitting device adds a parity bit (bit value is 1 for odd number of 1s in the transmitted bit stream and 0 for even number of 1s).
- The UART of the receiving device calculates the parity of the bits received and compares it with the received parity bit for error checking.
- The UART of the receiving device discards the 'Start', 'Stop' and 'Parity' bit from the received bit stream and converts the received serial bit data to a word
  - In the case of 8 bits/byte, the byte is formed with the received 8 bits with the first received bit as the LSB and last received data bit as MSB.

Universal Asynchronous Receiver Transmitter (UART) (continued)

- For proper communication, the 'Transmit line' of the sending device should be connected to the 'Receive line' of the receiving device.
- In addition to the serial data transmission function, UART provides hardware handshaking signal support for controlling the serial data flow.
- UART chips are available from different semiconductor manufacturers.
  - National Semiconductor's 8250 UART chip is considered as the standard setting UART. It was used in the original IBM PC.
- Nowadays most of the microprocessors/controllers are available with integrated UART functionality and they provide built-in instruction support for serial data transmission and reception.

Universal Asynchronous Receiver Transmitter (UART) (continued)

- Figure illustrates the UART interfacing.



TXD: Transmitter line
RXD: Receiver line

Fig: UART Interfacing

1-Wire Interface

- 1-wire interface is an asynchronous half-duplex communication protocol developed by Maxim Dallas Semiconductor.
- It is also known as Dallas 1-Wire protocol.
- It makes use of only a single signal line (wire) called DQ for communication and follows the master-slave communication model.
- One of the key feature of 1-wire bus is that it allows power to be sent along the signal wire as well.
- The slave devices incorporate internal capacitor (typically of the order of 800 pF) to power the device from the signal line.
- The 1-wire interface supports a single master and one or more slave devices on the bus.

1-Wire Interface (continued)

- The bus interface diagram shown in the figure illustrates the connection of master and slave devices on the 1-wire bus.

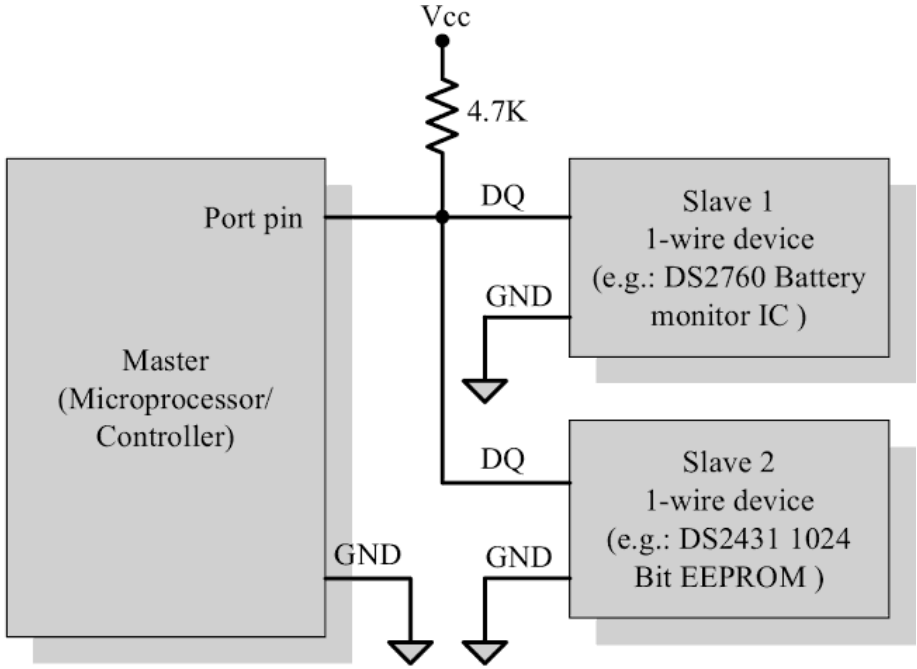


Fig: 1-Wire Interface Bus

1-Wire Interface (continued)

- Every 1-wire device contains a globally unique 64bit identification number stored within it.
- This unique identification number can be used for addressing individual devices present on the bus in case there are multiple slave devices connected to the 1-wire bus.
- The identifier has three parts: an 8 bit family code, a 48 bit serial number and an 8 bit CRC computed from the first 56 bits.

1-Wire Interface (continued)

- The sequence of operation for communicating with a 1-wire slave device is listed below:
  1. The master device sends a 'Reset' pulse on the 1-wire bus.
  2. The slave device(s) present on the bus respond with a 'Presence' pulse.
  3. The master device sends a ROM command (Net Address Command followed by the 64 bit address of the device).
     - This addresses the slave device(s) to which it wants to initiate a communication.
  4. The master device sends a read/write function command to read/write the internal memory or register of the slave device.
  5. The master initiates a Read data/Write data from the device or to the device.

1-Wire Interface (continued)

- All communication over the 1 -wire bus is master initiated.
- The communication over the 1-wire bus is divided into timeslots of 60 microseconds.
- The 'Reset' pulse occupies 8 time slots. For starting a communication, the master asserts the reset pulse by pulling the 1-wire bus 'LOW' for at least 8 time slots (480 µs).
- If a 'slave' device is present on the bus and is ready for communication it should respond to the master with a 'Presence' pulse, within 60 µs of the release of the 'Reset' pulse by the master.
- The slave device(s) responds with a 'Presence' pulse by pulling the 1-wire bus 'LOW' for a minimum of 1 time slot (60 µs).

1-Wire Interface (continued)

- For writing a bit value of 1 on the 1-wire bus, the bus master pulls the bus for 1 to 15 µs and then releases the bus for the rest of the time slot.
  - A bit value of ‘0' is written on the bus by master pulling the bus for a minimum of 1 time slot (60 µs) and a maximum of 2 time slots (120 µs).
- To Read a bit from the slave device, the master pulls the bus 'LOW' for 1 to 15 µs.
  - If the slave wants to send a bit value ‘1' in response to the read request from the master, it simply releases the bus for the rest of the time slot.
  - If the slave wants to send a bit value '0', it pulls the bus 'LOW' for the rest of the time slot.

Parallel Interface

- The on-board parallel interface is normally used for communicating with peripheral devices which are memory mapped to the host of the system.
- The host processor/controller of the embedded system contains a parallel bus and the device which supports parallel bus can directly connect to this bus system.
- The communication through the parallel bus is controlled by the control signal interface between the device and the host.
  - The Control Signals for communication includes Read/Write signal and device select signal.
- The device normally contains a device select line and the device becomes active only when this line is asserted by the host processor.
- The direction of data transfer (Host to Device or Device to Host) can be controlled through the control signal lines for 'Read' and 'Write'.
- Only the host processor has control over the 'Read' and 'Write' control signals.

Parallel Interface (continued)

- The device is normally memory mapped to the host processor and a range of address is assigned to it.
- An address decoder circuit is used for generating the chip select signal for the device.
- When the address selected by the processor is within the range assigned for the device, the decoder circuit activates the chip select line and thereby the device becomes active.
- The processor then can read or write from or to the device by asserting the corresponding control line (RD\ and WR\ respectively).

Parallel Interface (continued)

- The bus interface diagram shown in the figure illustrates the interfacing of devices through parallel interface.

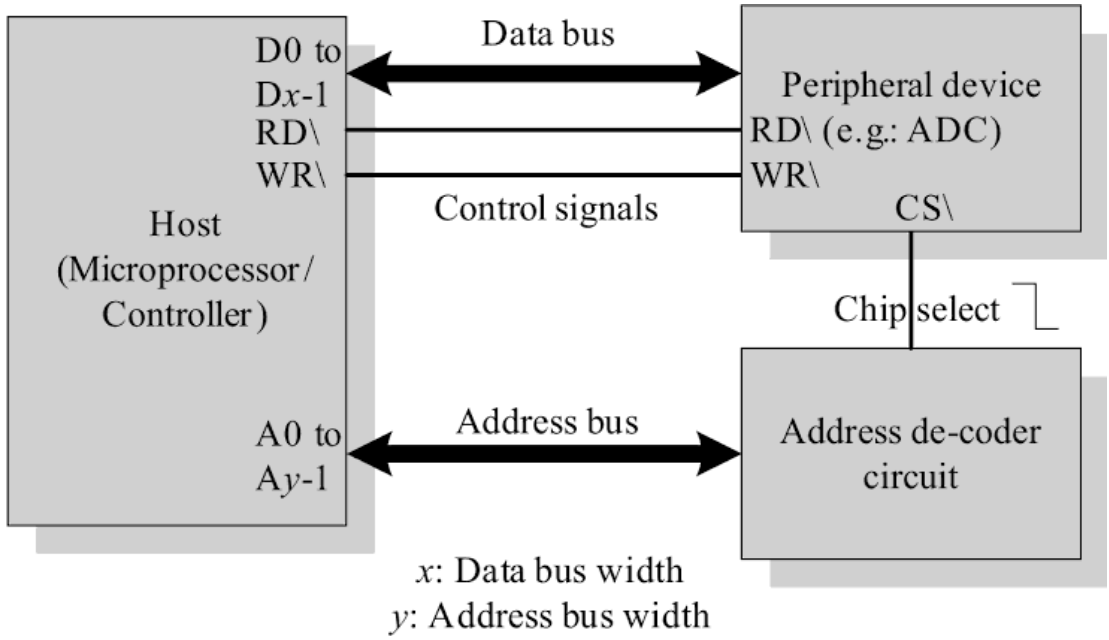


Fig: Parallel Interface Bus

Parallel Interface (continued)

- Parallel communication is host processor initiated.
- If a device wants to initiate the communication, it can inform the same to the processor through interrupts.
  - For this, the interrupt line of the device is connected to the interrupt line of the processor and the corresponding interrupt is enabled in the host processor.
- The width of the parallel interface is determined by the data bus width of the host processor.
  - It can be 4 bit, 8 bit, 16 bit, 32 bit or 64 bit etc.
  - The bus width supported by the device should be same as that of the host processor.
- Parallel data communication offers the highest speed for data transfer.

External Communication Interfaces

- External Communication Interface refers to the different communication channels/buses used by the embedded system to communicate with the external world.
- E.g.: RS-232 C & RS-485, Universal Serial Bus (USB), IEEE 1394 (Firewire), Infrared (IR), Bluetooth (BT), Wi-Fi, ZigBee, GPRS, etc.

RS-232 C & RS-485

- RS-232 C (Recommended Standard number 232, revision C) is a legacy, full duplex, wired, asynchronous serial communication interface.
- The RS-232 interface was developed by the Electronics Industries Association (EIA) during the early 1960s.
- RS-232 extends the UART communication signals for external data communication.
- UART uses the standard TTL/CMOS logic (Logic ‘High’ corresponds to bit value 1 and Logic ‘Low’ corresponds to bit value 0) for bit transmission whereas RS-232 follows the EIA standard for bit transmission.
  - As per the EIA standard, a logic ‘0’ is represented with voltage between +3 and +25V and a logic ‘1’ is represented with voltage between -3 and -25 V.
  - In EIA standard, logic ‘0’ is known as 'Space' and logic ‘1’ as ‘Mark’.

RS-232 C & RS-485 (continued)

- The RS-232 interface defines various handshaking and control signals for communication apart from the 'Transmit' and 'Receive' signal lines for data communication.
- RS-232 supports two different types of connectors:
  - DB-9: 9-Pin connector
  - DB-25: 25-Pin connector.

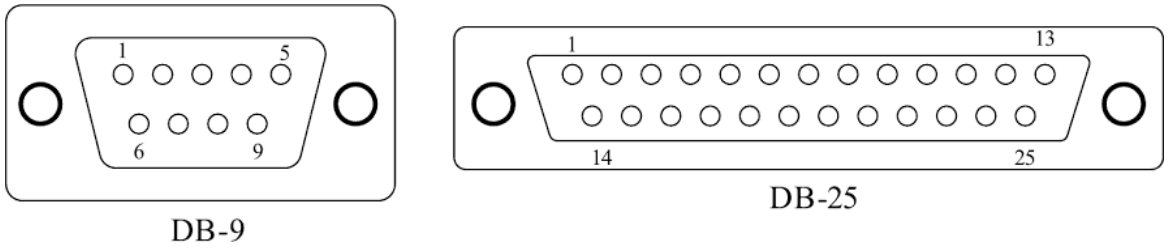


Fig: DB-9 and DB-25 RS-232 Connector Interface

RS-232 C & RS-485 (continued)

- The pin details for the two connectors are explained in the following table:

Pin Name	Pin no: (For DB-9 Connector)	Pin no: (For DB-25 Connector)	Description
TXD	3	2	Transmit Pin for Transmitting Serial Data
RXD	2	3	Receive Pin for Receiving Serial Data
RTS	7	4	Request to send.
CTS	8	5	Clear To Send
DSR	6	6	Data Set Ready
GND	5	7	Signal Ground
DCD	1	8	Data Carrier Detect
DTR	4	20	Data Terminal Ready
RI	9	22	Ring Indicator
FG		1	Frame Ground
SDCD		12	Secondary DCD
SCTS		13	Secondary CTS
STXD		14	Secondary TXD
TC		15	Transmission Signal Element Timing
SRXD		16	Secondary RXD
RC		17	Receiver Signal Element Timing
SRTS		19	Secondary RTS
SQ		21	Signal Quality detector
NC		9	No Connection
NC		10	No Connection
NC		11	No Connection
NC		18	No Connection
NC		23	No Connection
NC		24	No Connection
NC		25	No Connection

Pin Name	Pin no: (For DB-9 Connector)	Pin no: (For DB-25 Connector)	Description
TXD	3	2	Transmit Pin for Transmitting Serial Data
RXD	2	3	Receive Pin for Receiving Serial Data
RTS	7	4	Request to send.
CTS	8	5	Clear To Send
DSR	6	6	Data Set Ready
GND	5	7	Signal Ground
DCD	1	8	Data Carrier Detect
DTR	4	20	Data Terminal Ready
RI	9	22	Ring Indicator
FG		1	Frame Ground
SDCD		12	Secondary DCD
SCTS		13	Secondary CTS
STXD		14	Secondary TXD
TC		15	Transmission Signal Element Timing
SRXD		16	Secondary RXD
RC		17	Receiver Signal Element Timing
SRTS		19	Secondary RTS
SQ		21	Signal Quality detector
NC		9	No Connection
NC		10	No Connection
NC		11	No Connection
NC		18	No Connection
NC		23	No Connection
NC		24	No Connection
NC		25	No Connection

RS-232 C & RS-485 (continued)

- RS-232 is a point-to-point communication interface and the devices involved in RS-232 communication are called 'Data Terminal Equipment (DTE)' and 'Data Communication Equipment (DCE)’.
- If no data flow control is required, only TXD and RXD signal lines and ground line (GND) are required for data transmission and reception.
  - The RXD pin of DCE should be connected to the TXD pin of DTE and vice versa for proper data transmission.
- If hardware data flow control is required for serial transmission, various control signal lines of the RS-232 connection are used appropriately.

RS-232 C & RS-485 (continued)

- The Request To Send (RTS) and Clear To Send (CTS) signals co-ordinate the communication between DTE and DCE.
  - Whenever the DTE has a data to send, it activates the RTS line and if the DCE is ready to accept the data, it activates the CTS line.
- The Data Terminal Ready (DTR) signal is activated by DTE when it is ready to accept data.
- The Data Set Ready (DSR) is activated by DCE when it is ready for establishing a communication link.
  - DTR should be in the activated state before the activation of DSR.
- The Data Carrier Detect (DCD) control signal is used by the DCE to indicate the DTE that a good signal is being received.
- Ring Indicator (RI) is a modem specific signal line for indicating an incoming call on the telephone line.

RS-232 C & RS-485 (continued)

- As per the EIA standard RS-232 C supports baudrates up to 20Kbps (Upper limit 19.2 Kbps)
  - The commonly used baudrates by devices are 300bps, 1200bps, 2400bps, 9600bps, 11.52Kbps and 19.2Kbps.
  - 9600 is the popular baudrate setting used for PC communication.
- The maximum operating distance supported by RS-232 is 50 feet at the highest supported baudrate.
- Embedded devices contain a UART for serial communication and they generate signal levels conforming to TTL/CMOS logic.
  - A level translator IC like MAX 232 from Maxim Dallas semiconductor is used for converting the signal lines from the UART to RS-232 signal lines for communication.

RS-232 C & RS-485 (continued)

- RS-232 supports only point-to-point communication and not suitable for multi-drop communication.
  - It uses single ended data transfer technique for signal transmission and thereby more susceptible to noise and it greatly reduces the operating distance.
- RS-422 is another serial interface standard from EIA for differential data communication.
  - It supports data rates up to 100Kbps and distance up to 400 ft.
- RS-422 supports multi-drop communication with one transmitter device and receiver devices up to 10.
- RS-485 is the enhanced version of RS-422 and it supports multi-drop communication with up to 32 transmitting devices (drivers) and 32 receiving devices on the bus.
  - The communication between devices in the bus uses the 'addressing' mechanism to identify slave devices.

Universal Serial Bus (USB)

- Universal Serial Bus (USB) is a wired high speed serial bus for data communication.
- The first version of USB (USB 1.0) was released in 1995.
- The USB communication system follows a star topology with a USB host at the centre and one or more USB peripheral devices/USB hosts connected to it.
- A USB host can support connections up to 127, including slave peripheral devices and other USB hosts.

Universal Serial Bus (USB) (continued)

- Figure illustrates the star topology for USB device connection.

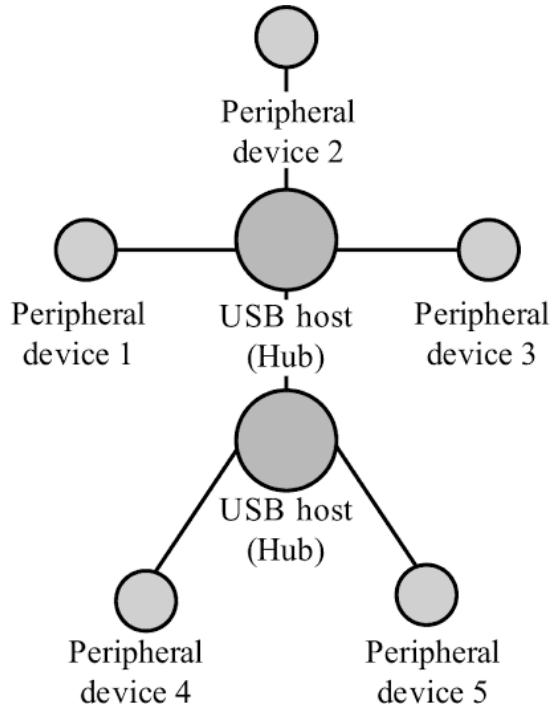


Fig: USB Device Connection topology

Universal Serial Bus (USB) (continued)

- USB transmits data in packet format.
- Each data packet has a standard format.
- The USB communication is a host initiated one.
- The USB host contains a host controller which is responsible for controlling the data communication, including establishing connectivity with USB slave devices, packetizing and formatting the data.
- There are different standards for implementing the USB Host Control interface:
  - Open Host Control Interface (OHCI)
  - Universal Host Control Interface (UHCI)

Universal Serial Bus (USB) (continued)

- The physical connection between a USB peripheral device and master device is established with a USB cable.
- The USB cable supports communication distance of up to 5 metres.
- The USB standard uses two different types of connector at the ends of the USB cable for connecting the USB peripheral device and host device.
- 'Type A' connector is used for upstream connection (connection with host) and Type B connector is used for downstream connection (connection with slave device).
- The USB connector present in desktop PCs or laptops are examples for 'Type A' USB connector.

Universal Serial Bus (USB) (continued)

- Both Type A and Type B connectors contain 4 pins for communication.
- The Pin details for the connectors are listed in the table given below.

Pin no:	Pin name	Description
1	V_{BUS}	Carries power (5V)
2	D–	Differential data carrier line
3	D+	Differential data carrier line
4	GND	Ground signal line

Universal Serial Bus (USB) (continued)

Type A Connector

Type B Connector

Type C Connector

Universal Serial Bus (USB) (continued)

- USB uses differential signals for data transmission.
  - It improves the noise immunity.
- USB interface has the ability to supply power to the connecting devices.
  - Two connection lines (Ground and Power) of the USB interface are dedicated for carrying power.
  - It can supply power up to 500 mA at 5 V.
  - It is sufficient to operate low power devices.
- Mini and Micro USB connectors are available for small form factor devices like portable media players.
- Each USB device contains a Product ID (PID) and a Vendor ID (VID).
  - Embedded into the USB chip by the USB device manufacturer.
  - The VID for a device is supplied by the USB standards forum.
  - PID and VID are essential for loading the drivers corresponding to a USB device for communication.

Universal Serial Bus (USB) (continued)

- USB supports four different types of data transfers:
- Control transfer : Used by USB system software to query, configure and issue commands to the USB device.
- Bulk transfer : Used for sending a block of data to a device.
  - Supports error checking and correction.
  - Transferring data to a printer is an example for bulk transfer.
- Isochronous data transfer : Used for real-time data communication.
  - Data is transmitted as streams in real-time.
  - Doesn't support error checking and re-transmission of data in case of any transmission loss.
  - All streaming devices like audio devices and medical equipment for data collection make use of the isochronous transfer.
- Interrupt transfer : Used for transferring small amount of data.
  - Interrupt transfer mechanism makes use of polling technique to see whether the USB device has any data to send.
  - The frequency of polling is determined by the USB device and it varies from 1 to 255 milliseconds.
  - Devices like Mouse and Keyboard, which transmits fewer amounts of data, uses Interrupt transfer.

Universal Serial Bus (USB) (continued)

- USB.ORG is the standards body for defining and controlling the standards for USB communication.
- Presently USB supports different data rates:
  - Low-Speed (LS) - 1.5Mbps – USB 1.0
  - Full-Speed (FS) - 12Mbps – USB 1.0
  - High-Speed (HS) - 480Mbps – USB 2.0
  - SuperSpeed (SS) - 5Gbps – USB 3.0
  - SuperSpeed+ (SS+) - 10Gbps – USB 3.1, 20 Gbps – USB 3.2

IEEE 1394 (Firewire)

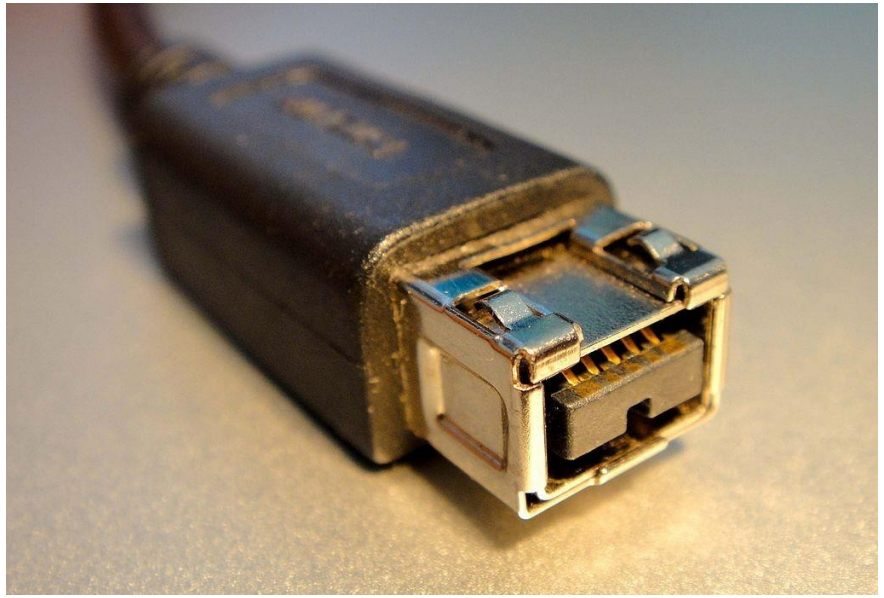
- IEEE 1394 is a wired, isochronous high speed serial communication bus.
- It is also known as High Performance Serial Bus (HPSB).
- The research on 1394 was started by Apple Inc. in 1985 and the standard for this was coined by IEEE.
- The implementation of 1394 is available from various players with different names:
  - Firewire is the implementation from Apple Inc
  - i.LINK is the implementation from Sony Corporation
  - Lynx is the implementation from Texas Instruments

IEEE 1394 (Firewire) (continued)

- 1394 supports peer-to-peer connection and point-to-multipoint communication allowing 63 devices to be connected on the bus in a tree topology.
- 1394 is a wired serial interface and it can support a cable length of up to 15 feet for interconnection.
- The 1394 standard supports a data rate of 400 to 3200 Mbits/second.
- The IEEE 1394 uses differential data transfer.
  - It increases the noise immunity.
- The interface cable supports 3 types of connectors, namely; 4-pin connector, 6-pin connector (alpha connector) and 9 pin connector (beta connector).
- The 6 and 9 pin connectors carry power also to support external devices.
  - It can supply unregulated power in the range of 24 to 30V.

IEEE 1394 (Firewire) (continued)

4-pin and 6-pin Connectors



9-pin Connector

IEEE 1394 (Firewire) (continued)

- The table given below illustrates the pin details for 4, 6 and 9 pin connectors.

Pin name	Pin no: (4 Pin Connector)	Pin no: (6 Pin Connector)	Pin no: (9 Pin Connector)	Description
Power		1	8	Unregulated DC supply. 24 to 30V
Signal Ground		2	6	Ground connection
TPB–	1	3	1	Differential Signal line for Signal line B
TPB+	2	4	2	Differential Signal line for Signal line B
TPA–	3	5	3	Differential Signal line for Signal line A
TPA+	4	6	4	Differential Signal line for Signal line A
TPA(S)			5	Shield for the differential signal line A. Normally grounded
TPB(S)			9	Shield for the differential signal line B. Normally grounded
NC			7	No connection

IEEE 1394 (Firewire) (continued)

- There are two differential data transfer lines A and B per connector.
- In a 1394 cable, normally the differential lines of A are connected to B (TPA+ to TPB+ and TPA– to TPB–) and vice versa.
- 1394 is a popular communication interface for connecting embedded devices like Digital Camera, Camcorder, Scanners to desktop computers for data transfer and storage.
- IEEE 1394 doesn't require a host for communicating between devices.
  - For example, you can directly connect a scanner with a printer for printing.
- The data rate supported by 1394 is far higher than the one supported by USB2.0 interface.
- The 1394 hardware implementation is much costlier than USB implementation.

Infrared (IrDA)

- Infrared (IrDA) is a serial, half duplex, line of sight based wireless technology for data communication between devices.
- It is in use from the olden days of communication and you may be very familiar with it.
  - E.g.: The remote control of TV, VCD player, etc. works on Infrared.
- Infrared communication technique uses infrared waves of the electromagnetic spectrum for transmitting the data.
- It supports point-point and point-to-multipoint communication, provided all devices involved in the communication are within the line of sight.
- The typical communication range for IrDA lies in the range 10 cm to 1 m.
- The range can be increased by increasing the transmitting power of the IR device.

Infrared (IrDA) (continued)

- IR supports data rates ranging from 9600bits/second to 16Mbps.
- Depending on the speed of data transmission IR is classified into:
  - Serial IR (SIR) – supports data rates ranging from 9600bps to 115.2kbps.
  - Medium IR (MIR) – supports data rates of 0.576Mbps and 1.152Mbps.
  - Fast IR (FIR) – supports data rates up to 4Mbps.
  - Very Fast IR (VFIR) – supports high data rates up to 16Mbps.
  - Ultra Fast IR (UFIR) – targeted to support a data rate up to 100Mbps.

Infrared (IrDA) (continued)

- IrDA communication involves a transmitter unit for transmitting the data over IR and a receiver for receiving the data.
- Infrared Light Emitting Diode (LED) is the IR source for transmitter and at the receiving end a photodiode acts as the receiver.
- Both transmitter and receiver unit will be present in each device supporting IrDA communication for bidirectional data transfer.
  - Such IR units are known as 'Transceiver’.
- Certain devices like a TV remote control always require unidirectional communication and so they contain either the transmitter or receiver unit.
  - The remote control unit contains the transmitter unit and TV contains the receiver unit.

Infrared (IrDA) (continued)

- Infrared Data Association (IrDA) is the regulatory body responsible for defining and licensing the specifications for IR data communication.
- IR communication has two essential parts: a physical link part and a protocol part.
  - The physical link is responsible for the physical transmission of data between devices supporting IR communication
  - Protocol part is responsible for defining the rules of communication.
- The physical link works on the wireless principle making use of Infrared for communication.
- The IrDA specifications include the standard for both physical link and protocol layer.
- The IrDA control protocol contains implementations for Physical Layer (PHY), Media Access Control (MAC) and Logical Link Control (LLC).

Infrared (IrDA) (continued)

- IrDA is a popular interface for file exchange and data transfer in low cost devices.
- IrDA was the prominent communication channel in mobile phones before Bluetooth's existence.

Bluetooth (BT)

- Bluetooth is a low cost, low power, short range wireless technology for data and voice communication.
- Bluetooth was first proposed by Ericsson in 1994.
- Bluetooth operates at 2.4GHz of the Radio Frequency spectrum and uses the Frequency Hopping Spread Spectrum (FHSS) technique for communication.
- It supports a data rate of up to 1Mbps and a range of approximately 30 feet for data communication.

Bluetooth (BT) (continued)

- Bluetooth communication has two essential parts – a physical link part and a protocol part.
  - The physical link is responsible for the physical transmission of data between devices supporting Bluetooth communication
  - The protocol part is responsible for defining the rules of communication.
- The physical link works on the wireless principle making use of RF waves for communication.
- Bluetooth enabled devices essentially contain a Bluetooth wireless radio for the transmission and reception of data.

Bluetooth (BT) (continued)

- The rules governing the Bluetooth communication is implemented in the 'Bluetooth protocol stack'.
  - The Bluetooth communication IC holds the stack.
- Each Bluetooth device will have a 48 bit unique identification number.
- Bluetooth communication follows packet based data transfer.
- Bluetooth supports point-to-point (device to device) and point-to-multipoint (device to multiple device broadcasting) wireless communication.
- The point-to-point communication follows the master-slave relationship.
- A Bluetooth device can function as either master or slave.
- When a network is formed with one Bluetooth device as master and more than one device as slaves, it is called a Piconet.
  - A Piconet supports a maximum of seven slave devices.

Bluetooth (BT) (continued)

- Bluetooth is the favourite choice for short range data communication in handheld embedded devices.
- Bluetooth technology is very popular among cell phone users as they are the easiest communication channel for transferring ringtones, music files, pictures, media files, etc. between neighbouring Bluetooth enabled phones.
- The Bluetooth standard specifies the minimum requirements that a Bluetooth device must support for a specific usage scenario.
- The Generic Access Profile (GAP) defines the requirements for detecting a Bluetooth device and establishing a connection with it.
  - All other specific usage profiles are based on GAP.
  - Serial Port Profile (SPP) for serial data communication, File Transfer Profile (FTP) for file transfer between devices, Human Interface Device (HID) for supporting human interface devices like keyboard and mouse are examples for Bluetooth profiles.
- The specifications for Bluetooth communication is defined and licensed by the standards body 'Bluetooth Special Interest Group (SIG)'.

Wi-Fi

- Wi-Fi or Wireless Fidelity is the popular wireless communication technique for networked communication of devices.
- Wi-Fi follows the IEEE 802.11 standard.
- Wi-Fi is intended for network communication and it supports Internet Protocol (IP) based communication.
- It is essential to have device identities in a multipoint communication to address specific devices for data communication.
- In an IP based communication each device is identified by an IP address, which is unique to each device on the network.

Wi-Fi (continued)

- Wi-Fi based communications require an intermediate agent called Wi-Fi router/Wireless Access point to manage the communications.
- The Wi-Fi router is responsible for restricting the access to a network, assigning IP address to devices on the network, routing data packets to the intended devices on the network.
- Wi-Fi enabled devices contain a wireless adaptor for transmitting and receiving data in the form of radio signals through an antenna.
- The hardware part of it is known as Wi-Fi Radio.
- Wi-Fi operates at 2.4 GHz or 5 GHz of radio spectrum and they co-exist with other ISM band devices like Bluetooth.

Wi-Fi (continued)

- Figure illustrates the typical interfacing of devices in a Wi-Fi network.

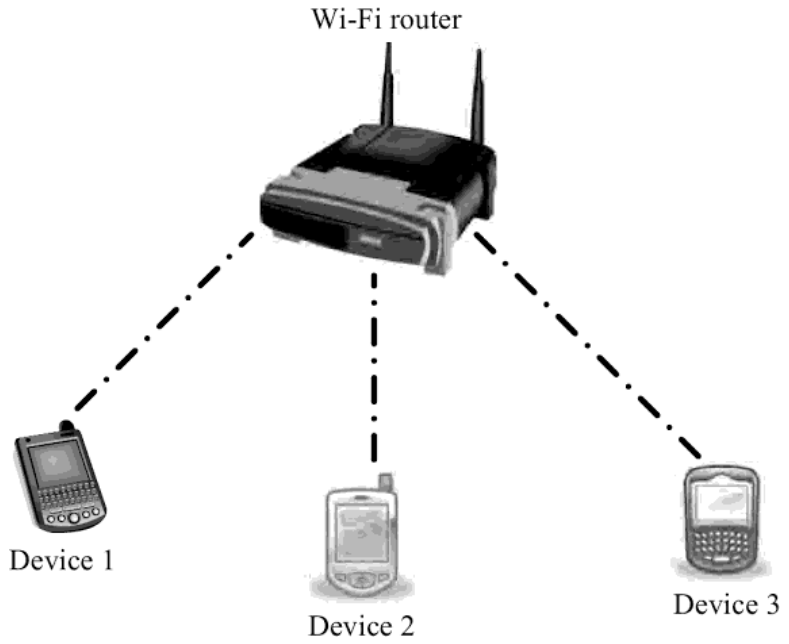


Fig: Wi-Fi Network

Wi-Fi (continued)

- For communicating with devices over a Wi-Fi network, the device when its Wi-Fi radio is turned ON, searches the available Wi-Fi network in its vicinity and lists out the Service Set Identifier (SSID) of the available networks.
- If the network is security enabled, a password may be required to connect to a particular SSID.
- Wi-Fi employs different security mechanisms like Wired Equivalency Privacy (WEP), Wireless Protected Access (WPA), etc. for securing the data communication.
- Wi-Fi supports data rates ranging from 1 Mbps to 1.73 Gbps depending on the standards (802.11a/b/g/n) and access/modulation method.
- Depending on the type of antenna and usage location (indoor/outdoor), Wi-Fi offers a range of 100 to 300 feet.

ZigBee

- ZigBee is a low power, low cost, wireless network communication protocol based on the IEEE 802.15.4-2006 standard.
- ZigBee is targeted for low power, low data rate and secure applications for Wireless Personal Area Networking (WPAN).
- The ZigBee specifications support a robust mesh network containing multiple nodes.
- This networking strategy makes the network reliable by permitting messages to travel through a number of different paths to get from one node to another.
- ZigBee operates worldwide at the unlicensed bands of Radio spectrum, mainly at 2.400 to 2.484 GHz, 902 to 928 MHz and 868.0 to 868.6 MHz.
- ZigBee supports an operating distance of up to 100 metres and a data rate of 20 to 250 Kbps.

ZigBee (continued)

- In the ZigBee terminology, each ZigBee device falls under any one of the following ZigBee device category:
- ZigBee Coordinator (ZC)/Network Coordinator
  - The ZigBee coordinator acts as the root of the ZigBee network.
  - The ZC is responsible for initiating the ZigBee network and it has the capability to store information about the network.
- ZigBee Router (ZR)/Full function Device (FFD)
  - Responsible for passing information from device to another device or to another ZR.
- ZigBee End Device (ZED)/Reduced Function Device (RFD):
  - End device containing ZigBee functionality for data communication.
  - It can talk only with a ZR or ZC and doesn't have the capability to act as a mediator for transferring data from one device to another.

ZigBee (continued)

- The diagram shown in figure gives an overview of ZC, ZED and ZR in a ZigBee network.

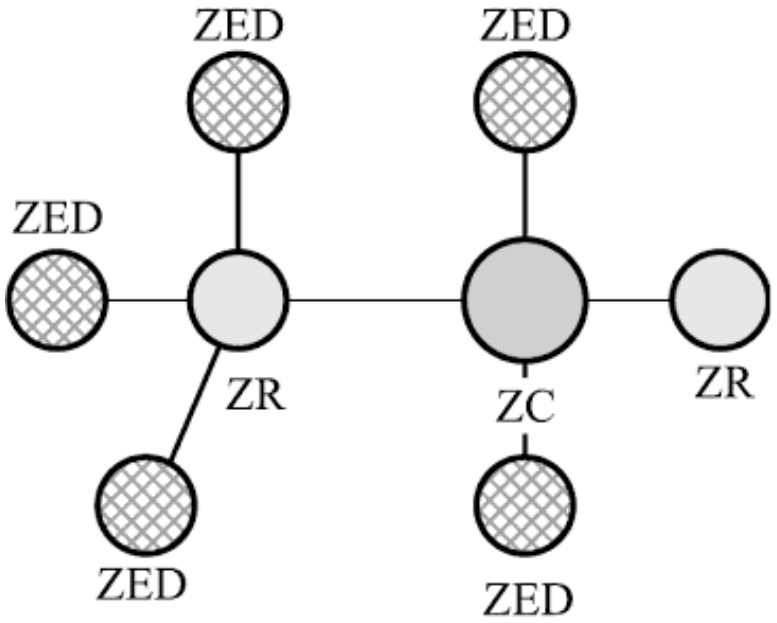


Fig: A ZigBee network model

ZigBee (continued)

- ZigBee is primarily targeting application areas like home & industrial automation, energy management, home control/security, medical/patient tracking, logistics & asset tracking and sensor networks & active RFID.
- Automatic Meter Reading (AMR), smoke detectors, wireless telemetry, HVAC control, heating control, lighting controls, environmental controls, etc. are examples for applications which can make use of the ZigBee technology.
- The specifications for ZigBee is developed and managed by the ZigBee Alliance, a non-profit consortium of leading semiconductor manufacturers, technology providers, OEMs and end-users worldwide.

General Packet Radio Service (GPRS)

- General Packet Radio Service (GPRS) is a communication technique for transferring data over a mobile communication network like GSM.
- Data is sent as packets in GPRS communication.
- The transmitting device splits the data into several related packets.
- At the receiving end the data is re-constructed by combining the received data packets.
- GPRS supports a theoretical maximum transfer rate of 171.2 kbps.
- In GPRS communication, the radio channel is concurrently shared between several users instead of dedicating a radio channel to a cell phone user.

General Packet Radio Service (GPRS) (continued)

- The GPRS communication divides the channel into 8 timeslots and transmits data over the available channel.
- GPRS supports Internet Protocol (IP), Point to Point Protocol (PPP) and X.25 protocols for communication.
- GPRS is mainly used by mobile enabled embedded devices for data communication.
- The device should support the necessary GPRS hardware like GPRS modem and GPRS radio.
- To accomplish GPRS based communication, the carrier network also should have support for GPRS communication.
- GPRS is an old technology and it is being replaced by new generation data communication techniques like EDGE, High Speed Downlink Packet Access (HSDPA), Long Term Evolution (LTE), etc. which offers higher bandwidths for communication.

Embedded Firmware

Embedded Firmware

- Embedded firmware refers to the control algorithm (Program instructions) and or the configuration settings that an embedded system developer dumps into the code (Program) memory of the embedded system.
- It is an un-avoidable part of an embedded system.
- There are various methods available for developing the embedded firmware:
  1. Write the program in high level languages like Embedded C/C++ using an Integrated Development Environment (IDE).
     - The IDE will contain an editor, compiler, linker, debugger, simulator, etc. IDES are different for different family of processors/controllers.
     - For example, Keil µVision 4 IDE is used for all family members of 8051 microcontroller, since it contains the generic 8051 compiler C51.
  2. Write the program in Assembly language using the instructions supported by your application's target processor/controller.

Embedded Firmware (continued)

- The program written in high level language or assembly code should be converted into a processor understandable machine code before loading it into the program memory.
- The process of converting the program written in either a high level language or processor/controller specific Assembly code to machine readable binary code is called 'HEX File Creation'.
- The methods used for 'HEX File Creation' is different depending on the programming techniques used.
  - If the program is written in Embedded C/C++ using an IDE, the cross compiler included in the IDE converts it into corresponding processor/controller understandable 'HEX File'.
- If Assembly language based programming technique is used, the utilities supplied by the processor/controller vendors can be used to convert the source code into 'HEX File'.
  - Also third party tools are available, which may be of free of cost, for this conversion.

Embedded Firmware (continued)

- For a beginner in the embedded software field, it is strongly recommended to use the high level language based development technique.
  - Writing codes in a high level language is easy
  - The code written in high level language is highly portable
    - The same code can be used to run on different processor/controller with little or less modification.
    - The only thing you need to do is re-compile the program with the required processor's IDE, after replacing the include files for that particular processor.
  - The programs written in high level languages are not developer dependent.
    - Any skilled programmer can trace out the functionalities of the program by just having a look at the program.
    - It will be much easier if the source code contains necessary comments and documentation lines.
  - It is very easy to debug and the overall system development time will be reduced to a greater extent.

Embedded Firmware (continued)

- The embedded software development process in assembly language is tedious and time consuming.
- The developer needs to know about all the instruction sets of the processor/controller or at least he should carry an instruction set reference manual with him.
- A programmer using assembly language technique writes the program according to his view and taste.
- Often he may be writing a method or functionality which can be achieved through a single instruction as an experienced person's point of view, by two or three instructions in his own style.
- So the program will be highly dependent on the developer.
- It is very difficult for a second person to understand the code written in Assembly even if it is well documented.

Embedded Firmware (continued)

- Two types of control algorithm design exist in embedded firmware development:
  - The first type of control algorithm development is known as the infinite loop or 'super loop' based approach, where the control flow runs from top to bottom and then jumps back to the top of the program in a conventional procedure.
    - It is similar to the *while (1) { };* based technique in C.
  - The second method deals with splitting the functions to be executed into tasks and running these tasks using a scheduler which is part of a General Purpose or Real Time Embedded Operating System (GPOS/RTOS).

Other System Components

Other System Components

- The other system components refer to the components/circuits/ICs which are necessary for the proper functioning of the embedded system.
- Some of these circuits may be essential for the proper functioning of the processor/controller and firmware execution.
- E.g.: Watchdog timer, Reset IC (or passive circuit), brown-out protection IC (or passive circuit), etc.
- Some of the controllers or SoCs integrate these components within a single IC and doesn't require such components externally connected to the chip for proper functioning.

Reset Circuit

- The reset circuit is essential to ensure that the device is not operating at a voltage level where the device is not guaranteed to operate, during system power ON.
- The reset signal brings the internal registers and the different hardware systems of the processor/controller to a known state and starts the firmware execution from the reset vector
  - Normally from vector address 0x0000 for conventional processors/controllers.
- The reset signal can be either active high or active low.
- Since the processor operation is synchronised to a clock signal, the reset pulse should be wide enough to give time for the clock oscillator to stabilise before the internal reset state starts.

Reset Circuit (continued)

- The reset signal to the processor can be applied at power ON through an external passive reset circuit comprising a Capacitor and Resistor or through a standard Reset IC like MAX810 from Maxim Dallas.
- Select the reset IC based on the type of reset signal and logic level (CMOS/TTL) supported by the processor/controller in use.
- Some microprocessors/controllers contain built-in internal reset circuitry and they don't require external reset circuitry.

Reset Circuit (continued)

- Figure illustrates a resistor capacitor based passive reset circuit for active high and low configurations.
- The reset pulse width can be adjusted by changing the resistance value R and capacitance value C.

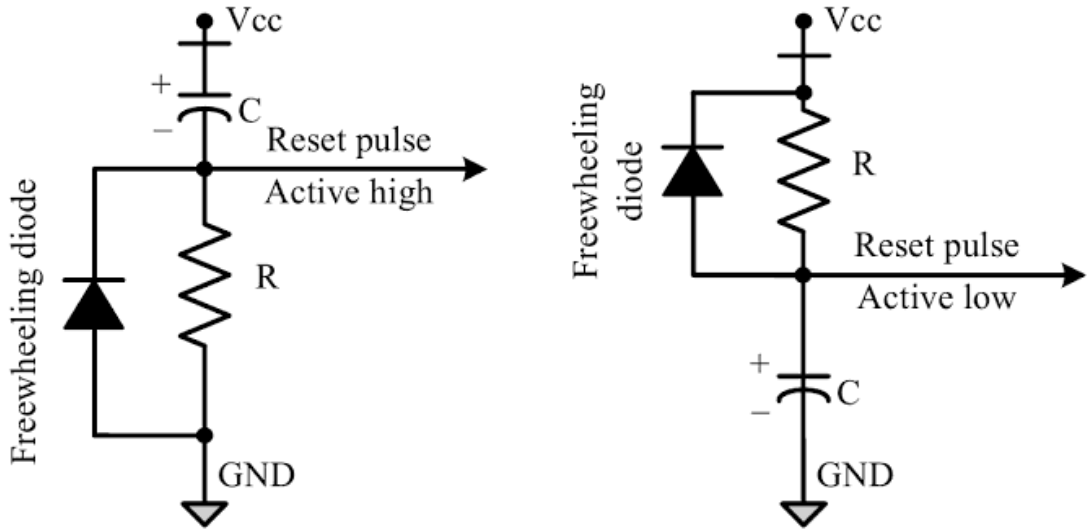


Fig: RC based reset circuit

Brown-out Protection Circuit

- Brown-out protection circuit prevents the processor/controller from unexpected program execution behaviour when the supply voltage to the processor/controller falls below a specified voltage.
- It is essential for battery powered devices since there are greater chances for the battery voltage to drop below the required threshold.
  - The processor behaviour may not be predictable if the supply voltage falls below the recommended operating voltage.
  - It may lead to situations like data corruption.

Brown-out Protection Circuit (continued)

- A brown-out protection circuit holds the processor/controller in reset state, when the operating voltage falls below the threshold, until it rises above the threshold voltage.
- Certain processors/controllers support built in brown-out protection circuit which monitors the supply voltage internally.
- If the processor/controller doesn't integrate a built-in brown-out protection circuit, the same can be implemented using external passive circuits or supervisor ICs.

Brown-out Protection Circuit (continued)

- Figure illustrates a brown-out circuit implementation using Zener diode and transistor for processor/controller with active low Reset logic.
- The Zener diode D_Z and transistor Q forms the heart of this circuit.
- The transistor conducts always when the supply voltage V_{CC} is greater than that of the sum of V_{BE} and V_Z (Zener voltage).
- The transistor stops conducting when the supply voltage falls below the sum of V_{BE} and V_Z .
- Select the Zener diode with required voltage for setting the low threshold value for V_{CC}.
- The values of R1, R2, and R3 can be selected based on the electrical characteristics of the transistor in use.
- Microprocessor Supervisor ICs like DS1232 from Maxim also provides Brown-out protection.

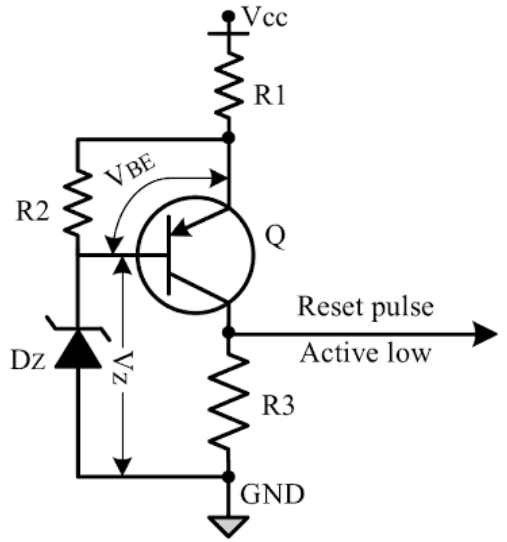


Fig: Brown-out protection circuit with Active low output

Oscillator Unit

- A microprocessor/microcontroller is a digital device made up of digital combinational and sequential circuits.
- The instruction execution of a microprocessor/controller occurs in sync with a clock signal.
- The oscillator unit of the embedded system is responsible for generating the precise clock for the processor.
  - Analogous to the heart in living beings which produces heart beats.
- Certain processors/controllers integrate a built-in oscillator unit and simply require an external ceramic resonator/quartz crystal for producing the necessary clock signals.

Oscillator Unit (continued)

- Quartz crystals and ceramic resonators are equivalent in operation, however they possess physical difference.
- A quartz crystal is normally mounted in a hermetically sealed metal case with two leads protruding out of the case.
- Certain devices may not contain a built-in oscillator unit and require the clock pulses to be generated and supplied externally.
  - Quartz crystal Oscillators are available in the form of chips and they can be used for generating the clock pulses in such cases.
- The speed of operation of a processor is primarily dependent on the clock frequency.
  - However we cannot increase the clock frequency blindly for increasing the speed of execution.
  - The logical circuits lying inside the processor always have an upper threshold value for the maximum clock at which the system can run, beyond which the system becomes unstable and non functional.

Oscillator Unit (continued)

- The total system power consumption is directly proportional to the clock frequency.
  - The power consumption increases with increase in clock frequency.
- The accuracy of program execution depends on the accuracy of the clock signal.
- The accuracy of the crystal oscillator or ceramic resonator is normally expressed in terms of +/-ppm (Parts per million).

Oscillator Unit (continued)

- Figure illustrates the usage of quartz crystal/ceramic resonator and external oscillator chip for clock generation.

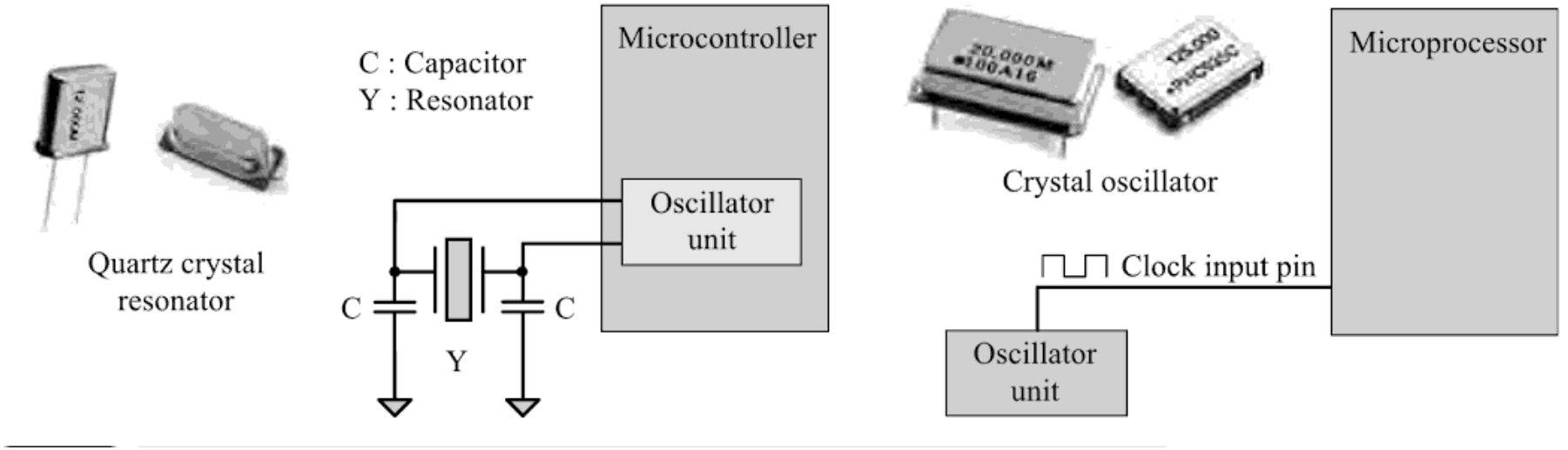


Fig: Oscillator circuitry using quartz crystal and quartz crystal oscillator

Real-Time Clock (RTC)

- Real-Time Clock (RTC) is a system component responsible for keeping track of time.
- RTC holds information like current time (In hours, minutes and seconds) in 12 hour/24 hour format, date, month, year, day of the week, etc. and supplies timing reference to the system.
- RTC is intended to function even in the absence of power.
- RTCs are available in the form of Integrated Circuits from different semiconductor manufacturers like Maxim/Dallas, ST Microelectronics etc.
- The RTC chip contains a microchip for holding the time and date related information and backup battery cell for functioning in the absence of power, in a single IC package.
- The RTC chip is interfaced to the processor or controller of the embedded system.

Real-Time Clock (RTC) (continued)

- For Operating System based embedded devices, a timing reference is essential for synchronising the operations of the OS kernel.
- The RTC can interrupt the OS kernel by asserting the interrupt line of the processor/controller to which the RTC interrupt line is connected.
- The OS kernel identifies the interrupt in terms of the Interrupt Request (IRQ) number generated by an interrupt controller.
- One IRQ can be assigned to the RTC interrupt and the kernel can perform necessary operations like system date time updation, managing software timers, etc. when an RTC timer tick interrupt occurs.
- The RTC can be configured to interrupt the processor at predefined intervals or to interrupt the processor when the RTC register reaches a specified value (used as alarm interrupt).

Watchdog Timer

- A watchdog timer, or simply a watchdog, is a hardware timer for monitoring the firmware execution and resetting the system processor/microcontroller when the program execution hangs up.
- Depending on the internal implementation, the watchdog timer increments or decrements a free running counter with each clock pulse and generates a reset signal to reset the processor if the count reaches zero for a down counting watchdog, or the highest count value for an up counting watchdog.

Watchdog Timer (continued)

- If the watchdog counter is in the enabled state, the firmware can write a zero (for up counting watchdog implementation) to it before starting the execution of a piece of code (which is susceptible to execution hang up) and the watchdog will start counting.
- If the firmware execution doesn't complete due to malfunctioning, within the time required by the watchdog to reach the maximum count, the counter will generate a reset pulse and this will reset the processor.
- If the firmware execution completes before the expiration of the watchdog timer you can reset the count by writing a 0 (for an up counting watchdog timer) to the watchdog timer register.

Watchdog Timer (continued)

- Most of the processors implement watchdog as a built-in component and provides status register to control the watchdog timer (like enabling and disabling watchdog functioning) and watchdog timer register for writing the count value.
- If the processor/controller doesn't contain a built in watchdog timer, the same can be implemented using an external watchdog timer IC circuit.
- The external watchdog timer uses hardware logic for enabling/disabling, resetting the watchdog count, etc. instead of the firmware based 'writing' to the status and watchdog timer register.
- The Microprocessor supervisor IC DS1232 integrates a hardware watchdog timer in it.
- In modern systems running on embedded operating systems, the watchdog can be implemented in such a way that when a watchdog timeout occurs, an interrupt is generated instead of resetting the processor.
- The interrupt handler for this handles the situation in an appropriate fashion.

Watchdog Timer (continued)

- Figure illustrates the implementation of an external watchdog timer based microprocessor supervisor circuit for a small scale embedded system.

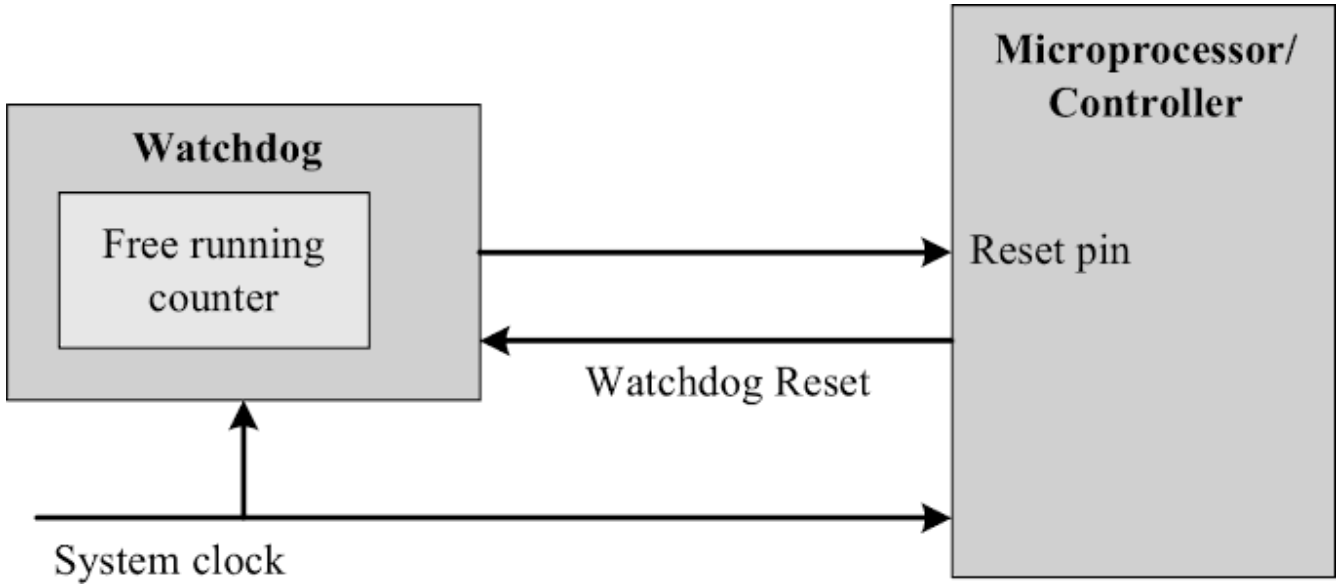


Fig: Watchdog timer for firmware execution supervision

PCB and Passive Components

- Printed Circuit Board (PCB) is the backbone of every embedded system.
- After finalising the components and the inter-connection among them, a schematic design is created and according to the schematic, the PCB is fabricated.
- PCB acts as a platform for mounting all the necessary components as per the design requirement.
- Also it acts as a platform for testing the embedded firmware.
- Apart from the subsystems mentioned already, passive electronic components like resistor, capacitor, diodes, etc. are also found on the board.
  - They are the co-workers of various chips contained in the embedded hardware.
  - They are very essential for the proper functioning of your embedded system.
  - For example for providing a regulated ripple-free supply voltage to the system, a regulator IC and spike suppressor filter capacitors are very essential.

References

1. Shibu K V, ***"Introduction to Embedded Systems"***, Tata McGraw Hill, 2009.
2. Raj Kamal, ***"Embedded Systems: Architecture and Programming",*** Tata McGraw Hill, 2008.