Unit - I

Introduction:-

Definition:-

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

* Every embedded system is unique, its hardware and software are highly specialised to the application domain

General purpose computing system	Embedded system
1. A system combination of generic hardware and general purpose operating system (OS) for executing a variety of applications	1. A system combination of special purpose hardware and embedded OS for executing a specific set of applications
2. Applications are programmable by user.	2. The firmware of embedded system is pre-programmed and it is not alterable by end user
3. Performance is primary factor to decide the efficiency [faster is better].	3. Application-specific requirement are key deciding factors
4. Response requirements are not time critical.	4. The response time is highly critical
5. They are designed for more number of applications	5. These are designed for application specific applications

History:-

In olden days Embedded systems were built around Vaccum tube and transistor technology and algorithms used are of "low level languages".

* The first recognised modern embedded system is the "Apollo Guidance computer (AGC)" developed by "MIT instrumentation laboratory" for lunar expedition.
* The clock frequency of first microchip was 1.024 MHz and it is derived from 2.048 MHz crystal clock.
* The first mass-produced embedded system was the guidance computer for the "minute man-1" missile in 1961
* It was "Autonetics D-17" guidance computer built using discrete transistor logic
* The first integrated circuit was produced in 1958 but computers began to used it from 1963.
* Some of the earlier embedded systems are NASA for AGC and by US military in the minute-man II intercontinental ballistic missile

Classification:-

The different classifications of embedded systems are done by using different criteria's such as:

1. Based on generation
2. complexity and performance requirements

3. Based on deterministic behaviour (2)

4. Based on triggering

Classification based on generation:-

a) first generation:- The early embedded systems were built around 8 bit micro processors like 8085 and 780 and 4 bit micro controllers. simple hardware circuits with firmware developed in assembly code.,

Ex:- Digital telephone Keypad, stepper motor control etc--

b) second generation:- These are embedded systems built around 16 bit micro processors and 8 (or) 16 bit micro-controllers. The instruction set for these processors/controllers are much more complex and power full than 1st generation.

Ex:- Data Aquisition systems, SCADA systems e.t.c---

[supervisory control & data Aquisition (SCADA)]

c) Third generation:- These are embedded systems built around 32-bit processors and 16-bit micro controllers for their design. A new concept of application specific and domain specific processors/controllers like digital signal processor (DSP) and Application specific IC's (ASIC's) came into existance. pipelining concepts are introduced making instruction set more complex. processors like Intel, pentium, motorola '68K e.t.c.. gained attention in high performance embedded requirements

Fourth generation:- The advent of system on chips (SoC) reconfigurable processors and multi-core processors increased the speed and miniatured the size of embedded systems in market. System on chip technique implements a total system on chip by integrating different functionalities with a processor core on an IC. They uses high performance RTOs for functioning

Ex:- Smart phones, mobile internet devices (MID's) etc..

Classification based on complexity and performance:-

Small scale embedded system:- These are usually built around low performance and low cost. 8 (or) 16-bit microprocessors | microcontrollers. It may (or) may not contain operating system for its functioning.

Ex:- electronic toy, A battery remote control for TV etc..

Medium-scale embedded system:- Embedded systems which are slightly complex in hard-ware and firmware requirements falls under this category. These are usually built around 16 (or) 32-bit microprocessors (or) controllers It contains an embedded operating system for functioning

large-scale embedded system:- They have high complex hardware and firmware. These are built around 32 (or) 64-bit RISC processors / controllers (or) reconfigurable

System on chip (RSoC) (or) multi-core processors and Programmable logic devices. They have RTOS for task Scheduling, prioritization and management. (3)

Ex:- Decoding | encoding media, cryptographic function implementation e.t.c--

Based on deterministic behaviour :-

It is applicable for 'Real time systems'. Based on execution behaviour embedded systems are classified as hard and soft.

Based on triggering :- Embedded systems which are reactive in nature can be classified based on the trigger. Reactive Systems can be either event triggered (or) time triggered.

Application areas :-

1. Consumer electronics :- camcorders, cameras, voice recorders e.t.c--
2. House hold appliances :- Tv, DVD players, washingmachine, fridge, micro oven e.t.c--
3. Home Automation and security systems :- Air conditioner, sprinklers, intruder detection, alarms, closed circuit Tv cameras, fire alarm e.t.c---
4. Automotive Industry :- Anti-lock breaking system (ABS), engine control, ignition system, Automatic navigation systems e.t.c---

5. Telecom:- cellular telephones, telephone switches, handset multi-media applications e.t.c--

6. computer peripherals:- printers, scanners, fax machines e.t.c--

7. Health care:- Brain tumour scanner, EEG, ECG machines e.t.c--

8. computer networking systems:- Network routers, switches, hubs, firewalls e.t.c--

9. Measurement and instrumentation:- Digital multimeter, Digital CRO's, logic analyzer, PIC systems e.t.c--

10. Banking and retail:- Automatic teller machine (ATM), currency counter, point of sales (POS)

11. Card readers:- Bar code, smart card readers, hand held devices e.t.c--

Purpose of Embedded system:-

Each embedded system is designed to serve the purpose of anyone (or) combination of the following:-

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

1. Data collection | storage | representation :- (ii)

* Data such as video, audio, text, image, electrical signal or any other measurable quantity. It can be analog (or) digital
* The data collection performs aquisition of data from the external world
* Embedded systems which collect the analog (continuous) data, collection mechanism involves in conversion of analog to digital for this purpose a A/D converter and then binary equivalent of the analog data is collected
* Embedded system which collect digital (discrete) data, it can be directly captured with out any additional interface.
* Collected data may (or) may not be stored in the system. It may be transmitted to other system.
* Embedded systems designed for pure measurement without storage, used in control and instrumentation domain.

Ex:- Analog and digital CRO's without storage

* Some systems stores the data collected for processing and analysis

Ex:- LCD, buzzers, alarms, LED display units e.t.c.,

* A digital camera is a typical example for data collection | storage | Representation.
* Images are captured and may be stored with in the memory of camera. The captured image can be ~~processed~~ presented to the user through a graphic LCD unit.

Data communication :-

* The collected data must be transfered to some remote location. The transmission is achieved by a wire-line medium or by a wire-less medium.
* Wire-line medium was used in olden days, wire less systems are of much interest now-a-days.
* Wire-less medium offers cheap connectivity solutions and make the communication link free from wired bundles.
* Data can be either analog / digital
* The data collecting instrument itself may contain data communication units like wireless modules (Bluetooth, Zigbee, Wi-fi, EDGE, GPRS e.t.c--) or wireline modules (RS-232C, USB, TCP/IP etc--)
* Certain embedded systems acts as didicated transmission unit between sending and receiving terminal.

  Ex:- Switches, Hubs, routers e.t.c--

* They act as mediators in data communication and provide various features like data-security, monitoring e.t.c

Data (signal) processing :-

* The data collected by the embedded system might 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, transmissionapplications e.t.c--

Example:- A digital hearing aid is a typical example (5) of an embedded system employing data processing. It improves the hearing capacity of hearing imparied persons.

Monitoring:-

* All the medical embedded product will come under this category.
* They are used for determining state of some variables using input sensors.
* They cannot impose control over the variable.

Ex:- ECG (Electro cardiogram) for monitoring heart beat of a patient

* The ECG is intented to monitor the patients heart beat but it cannot have any control over heart beat.
* The sensors used in ECG are different electrodes connected to the patients body.

Ex:- digital CRO, digital multimeters, logic analyzers e.t.c

* They are used for knowing (monitoring) the status of variables like current, voltage e.t.c--

Control:-

* Embedded systems with control functionalities impose control over some variables according to the changes in input variables
* A system with control functionality has both sensors and actuators

* Sensors are connected to the input port for capturing the changes in measuring variable.

* Actuators connected to the output port are controlled according to the changes in input variable and brings them to specified range.

Application specific user interface :-

* There are embedded systems which are application specific user interfaces like buttons, switches, keypad, lights, bells, display units e.t.c

* Mobile phone is one of the example. In mobile phone the user interface is provided through a keypad, graphical LCD, module, system speaker, vibration alert e.t.c--

* Smart running shoes are another example launched by Adidas

* The shoe constantly adapts its shock absorbing characteristics to customise its value to the individual runner, depending on the running style, pace, body We-ight, and running surface.

Typical Embedded System :-

A typical embedded system contains :-

1. A single controller chip

2. input ports
3. output ports
4. communication interface
5. Embedded firmware
6. Memory

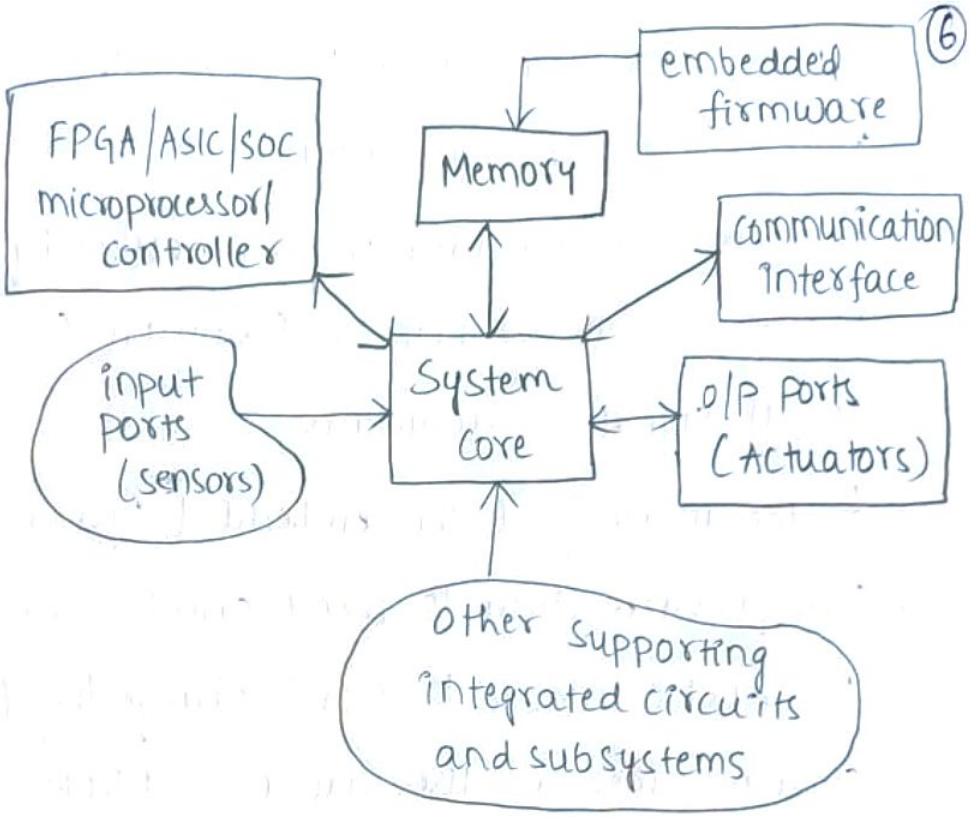


fig:- elements of embedded system

Operation:-

* The controller can be microprocessor (or) microcontroller (or) field programmable Gate Array (FPGA) device (or) a digital signal processor (or) an Application specific IC (ASIC) (Or) an Application specific standard product (ASSP)

Ex:- 8085 up, Atmel AT89C51 uc, xilinx spartan e.t.c

* Embedded systems by using hardware/software control the physical variables by providing control signals to the actuators (or) the devices connected to the output ports of the system.
* The control is achieved by processing information coming from sensors and user interfaces.
* Examples for user interface output devices are LED's, LCD's, picto electric buzzers e.t.c--

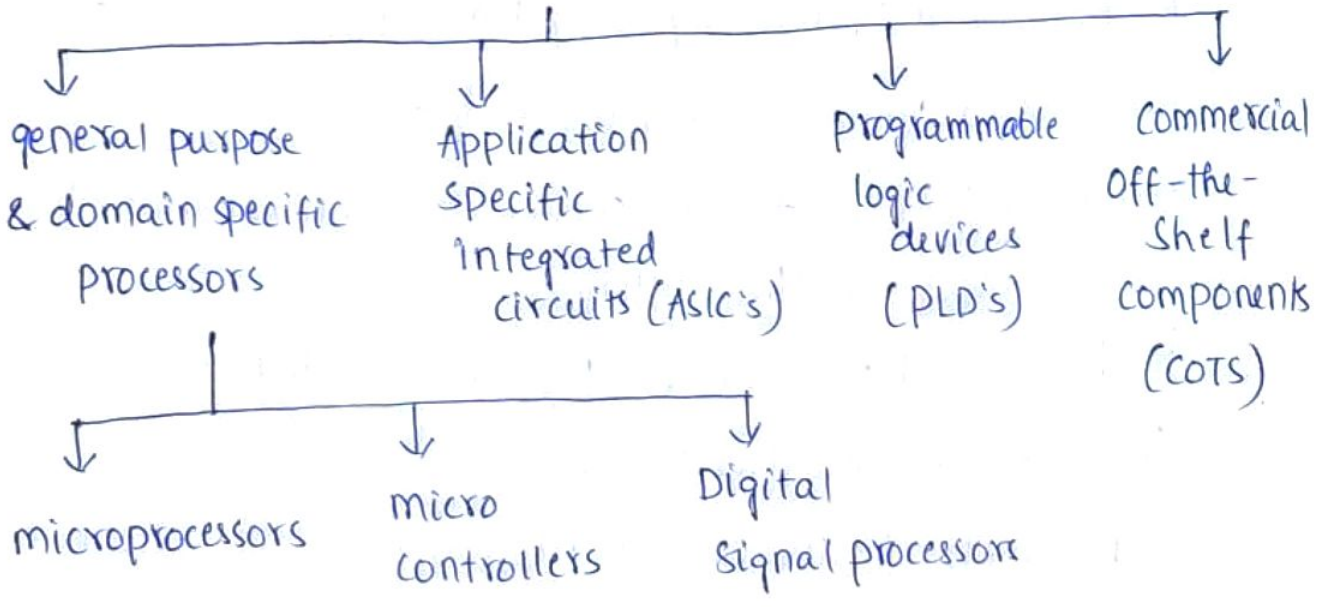
* The sensor information is passed to the processor after conditioning and digitization.

* Upon receiving data, it is processed by the controller (or) Processor with the help of firmware (software) and sends output signals to actuators

* The memory of the embedded system for storage of configuration data is fixed called "Read only memory" (ROM). It is not available to user to change the fixed data. Most common memory uses are OTP, PROM, UVEPROM, EEPROM and Flash.

* Depending upon the application the memory size varies to perform arithmetic operations and control algorithms execution Random access memory (RAM) is used as "working memory". Ex:- SRAM, DRAM, NVRAM e.t.c..

Core of the Embedded system



1. General purpose and domain-specific processors :- (7)

* 80% of embedded systems are processor / controller based systems.
* Most of the industrial applications uses a microprocessor (or) a micro controller for control and monitoring
* But the domains such as speech coding, speech recognisation, e.t.c -- requires a digital signal processor.

a) Microprocessors :-

* A microprocessor is a silicon chip consisting of memory unit, control unit and ALU (Arithmetic & logic unit)
* It works along with the other hardware like memory, timer unit and interrupt controller e.t.c --
* The 1st microprocessor was developed by intel, named intel 4004, a 4-bit processor released on 1971.
* It contains 1K data memory, 12-bit program counter, 4K program memory, 16 4-bit general purpose registers, and 46 instructions, with 740KHz clock frequency.
* In 1972, 14 more instruction were added named Intel 4040.
* Then Intel 8008 with 14 bit wide program counter.
* In 1974, intel launched first 8-bit processor Intel 8080, with 16-bit address bus and program counter and 8-bit registers.
* Then motorola launched processor named motorola - 6800

* In 1976, Intel upgraded processor version of 8080 - 8085, with two namely added instructions, with three interrupt pins and serial I/o clock and bus controllers are built-in with power supply ±5V.

* Today the clock speed of processors is around 2.4 GHz with 16, 32 and 64 bit processors are available with high speed, high performance and low cost.

* Harvard and Von-Neumann are two different architectures for processor design.

* Reduced Instruction set computing (RISC) and complex instruction set computing (CISC) are two common instruction sets architectures (ISA).

* Harvard architecture contains seperate buses for program memory and data memory where as processors based on Von-Neumann architecture shares single system bus for program and data memory.

(6) Micro controller:-

* A micro controller contains cpu, scratch pad RAM, special and general purpose register arrays, on chip ROM / flash memory for program storage, timer, Interrupt control and dedicated I/o ports.

* A microcontroller can be considered as superset of microprocessor.

* Texas instruments 'TMS 1000' is the world's first micro controller. (8)

* It is followed by 4004 / 4040 by adding extra RAM & ROM & I/o ports on single chip with a provision to add custom instructions to CPU, was released in 1974.

* In 1974, intel launched microcontroller family named MCS-48 family. The processors came under it are 8038 HL, 8039 HL, 8040 AHL, 8041 H, 8049 H & 8050 AH

* During 1980-90's, 8051 uC came into existence with low cost, wide availability, memory efficient instruction set e.t.c.

* Intel-8051 uC is an example of both general purpose and application specific instruction set processor. whereas Atmel processor (AVR) is an application specific controller designed to control automotive domain.

Microprocessor	Microcontroller
* It is a silicon chip representing central processing unit (cpu), which is capable of performing arithmetic and logical operations according to the predefined instruction set.	* It 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 with dedicated I/o ports
* Mostly used in general purpose in design and operation	* Mostly application-specific (or) domain specific operation

* It is a dependent unit which requires combination of other units like timers, interrupt controllers e.t.c --	* It is self contained unit and does not require external units for its functioning
* Does not contain I/O ports.	* They contain 8 (or) 16 (or) 32 bit ports (or) as individual port pins.
* Performance is an important factor	* speed and accuracy are important factors
* High cost due to more number of applications.	* Low cost because of application specific usage
* Off-chip memory	* On-chip memory

Digital signal processor (DSP) :-

A DSP is a special purpose 8/16/32 bit micro processor designed specifically to meet the computational demands and power constraints of embedded audio, video and communications applications. DSP's are much faster than general purpose microprocessors in signal processing applications.

Typical digital signal processor contains :-

a) program memory :- memory for storing the program required by DSP to process data.

b) data memory :- working memory for storing temporary

Variables and data / signal to be processed. (a)

c) computational engine:- It performs signal processing in accordance with the stored program memory. It incorporates many specialized arithmetic units and each of them operates simultaneously to increase the execution speed.

d) I/o unit:- It acts as interface between outside world and DSP. It is responsible for capturing signals to process and delivering the processed signal.

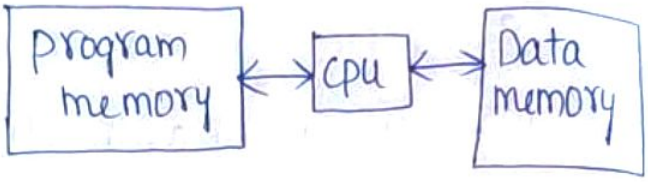
RISC VS CISC:-

(RISC) Reduced Instruction Set Computing	(CISC) Complex Instruction set Computing
* Less no. of instructions	* more no. of instructions
Instruction pipelining and increased execution speed.	No pipelining feature
* Operations are performed on registers only, the only memory operations are load & store.	* operations are performed on registers or memory depending upon instruction.
* orthogonal instruction set.	* It is instruction specific and non-orthogonal
* Large number of registers are available	* limited number of registers are available
* Single, fixed length instruction	* Variable length instruction
* with Harvard architecture	* with Harvard (or) Von neumann

Harvard Vs Von-Neumann Architectures :-

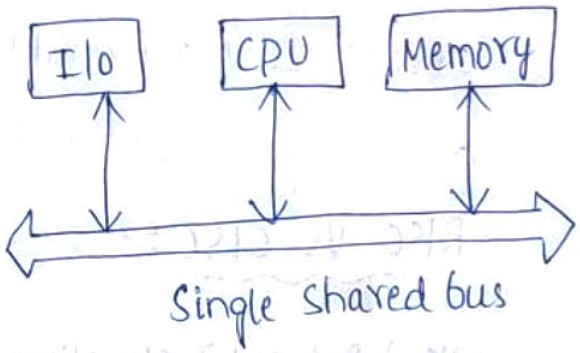
Harvard Architecture :-

It uses seperate buses for instruction and data fetching



Von-Neumann Architecture :-

Single shared bus for instruction and data fetching



Harvard Architecture	Von-Neumann Architecture
1. Seperate buses for instruction and data fetching.	1. Single shared bus for instruction and data fetching.
2. Easier to pipeline, so high performance can be achieved	2. low performance due to sharing of single bus.
3. Comparitively high cost	3. low cost
4. No memory alignment problem	4. Allows self modifying codes.
5. Since data memory and Program memory are stored in different locations no chances for accidental corruption of program memory.	5. Since data and program memory are stored in single chip, chances for accidental corruption of program memory.

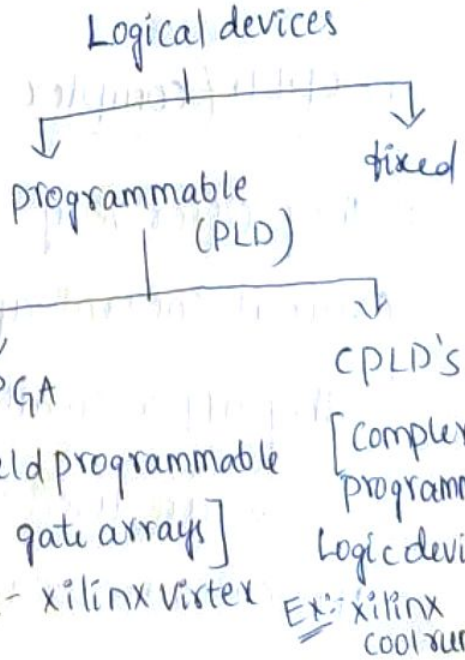
Application Specific Integrated Circuits (ASIC's):- (10)

* It is a micro-chip designed to perform a specific (or) unique application.
* It integrates several functions into a single chip, which reduces the system development cost.
* Asic's are pre-fabricated for a special application (or) they can be custom fabricated by using components from "building block" library of components which are re-usable for particular application.
* For an Asic a non-refundable one time investment called Non-Recurring engineering charges (NRE) requires, hence an application specific IC is used by single manufacturer.
* where as an application specific standard product (Assp) is made openly available in market, which is marketed to multiple customers.

Programmable logic Devices (PLD's):-

* Logical devices provide specific functions including device-to-device interface, data communication, signal processing, data display, timing & control operations.
* fixed logical devices once manufactured, they cannot be changed.

fig:- Logic devices types



Programmable logic devices (PLD):-

* These are re-configurable devices which can perform number of functions at a time with wide range of logic capacity, speed and voltage characteristics.
* A PLD, designers use inexpensive software tools to develop, simulate and test their designs.
* Then the device is programmed by the designs and tested in live circuits.
* There is NO NRE (Non-recurring Engineering charges) cost and the final design is completed much faster.
* During the design of the device the customers can change the circuitary as often as they want until it is satisfactorily designed.
* This is possible by using re-writable memory technology.
* The two major types of PLD's are 1. CPLD
  2. FPGAs

1. CPLD (complex programmable logic devices):-

* They offer very predictable timing which are used in critical control applications
* They offer small amount of logic gates upto 10,000 gates
* These are used in hand held devices and portable applications Ex:- Cool runner

FPGA (Field programmable gate array) :-

* They offer high amount of logic density, the most features and high performance.
* They have large design logic gates of order of 8-100 million gates.
* They offer built-in processors (Ex:- IBM powerpc), enough memory, clock-management systems, and support for many latest, very fast device-to-device signalling technology.
* They are used in wide range of applications, ranging from data processing and storage, to instrumentation, telecommunication and digital signal processing.

Commercial off-the shelf components (COTS) :-

* A commercial off-the shelf product is one in which is used in such a way that it can easily integrate and interoperate with existing system components.
* Example of COTS are control units of atoy car like RF circuitary part, electro-optic IR imaging arrays, UV/IR detectors e.t.c--
* The advantage of COTS are they are readily available in market cheap, and the developer of embedded system need not to develop these components by which development time can be minimized.

Memory:-

It is an important part of a processor/controller based embedded system. Some of the processors/controllers have built in memory and it is called chip memory. External memory connected to store control algorithms is called off-chip memory.

Program Storage Memory (ROM):-

1. Masked ROM (MROM):-

* It is one time programmable device.
* It is used hard wired technology for storing data.
* It is factory programmed by masking and metalization process at the time of production itself, according to the data provided by the end user.

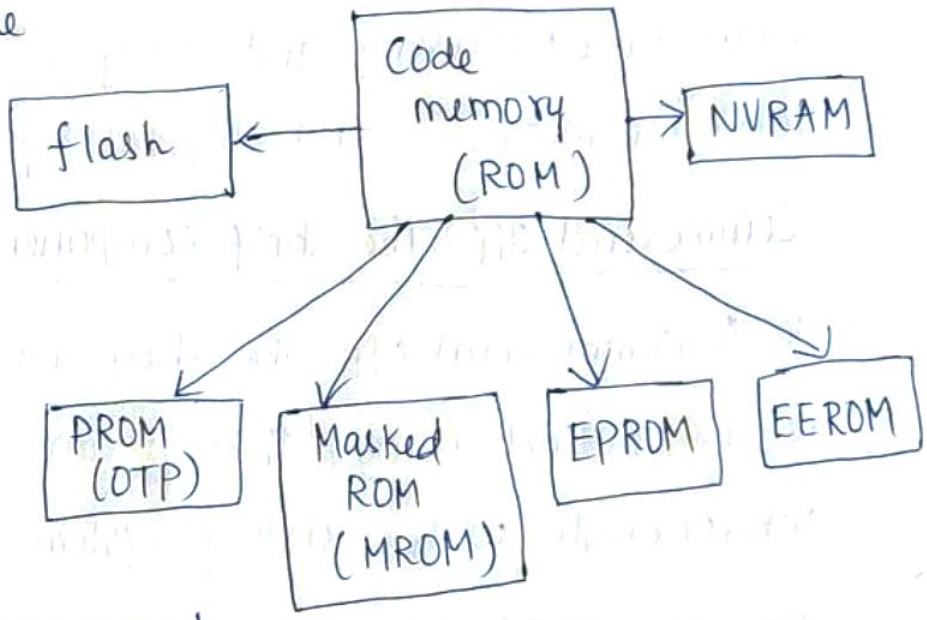


fig:- Classification of program memory

* MROM is good for storing the embedded firmware for low cost embedded devices.

* Its drawback is inability to modify the firmware of device against upgrades. (12)
* once programmed and stored in bit storage, it is not possible to alter the bit information.

2. Programmable Read only memory (PROM) / OTP:-

* It is not pre-programmed by the manufacturers the end user is responsible for programming these devices. One time programmable devices are widely used in commercial purpose of embedded systems whose prototype versions are proven and the code is finalized. we cannot re-program the device once it is programmed.

3. Erasable programmable read only memory (EPROM):-

It gives the flexibility to re-program on the same chip. EPROM chip is flexible in terms of its re-programability, it needs to be taken out of the circuit board and put in UV eraser device for 20 to 30 minutes.

4. EEPROM:-

The information contained in the EEPROM memory can be altered by using electric signals at the register / byte level. They can be erased and reprogrammed in circuit. These chip include a chip erase mode and this mode they can erase in few milli seconds.

5. Flash :-

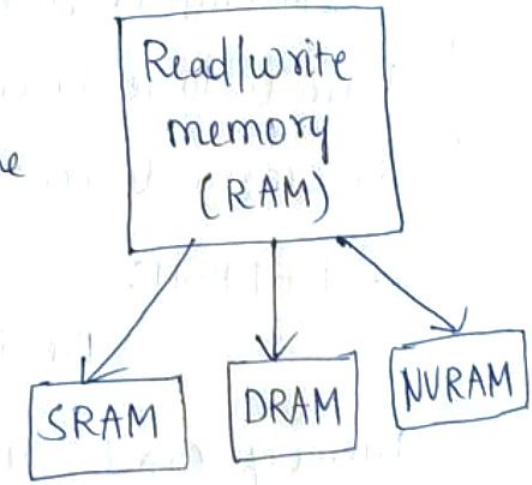
It is the latest ROM technology. Flash memory is a variation of EEPROM technology it combines the reprogramability of EEPROM and the high capacity of Standard ROM's. It stores the information in an array of floating gate MOS-FET transistors.

6. NVRAM (Non-volatile RAM) :-

It is a random access memory with battery backup. It consists of static ram based memory and battery for providing supply to battery in the absence of external power supply. The life of NVRAM is around 10 years.

Read/write Memory (or) random Access memory (RAM) :-

A RAM is a volatile memory i.e. when power is turned-off all the contents are destroyed.



1. Static memory (SRAM) :-

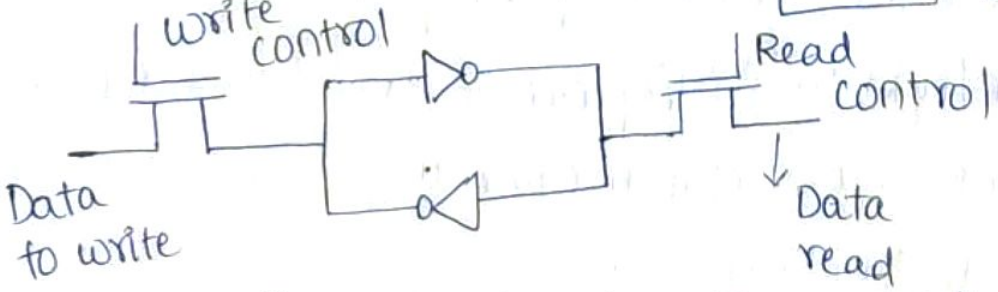


fig:- Visualization of SRAM cell

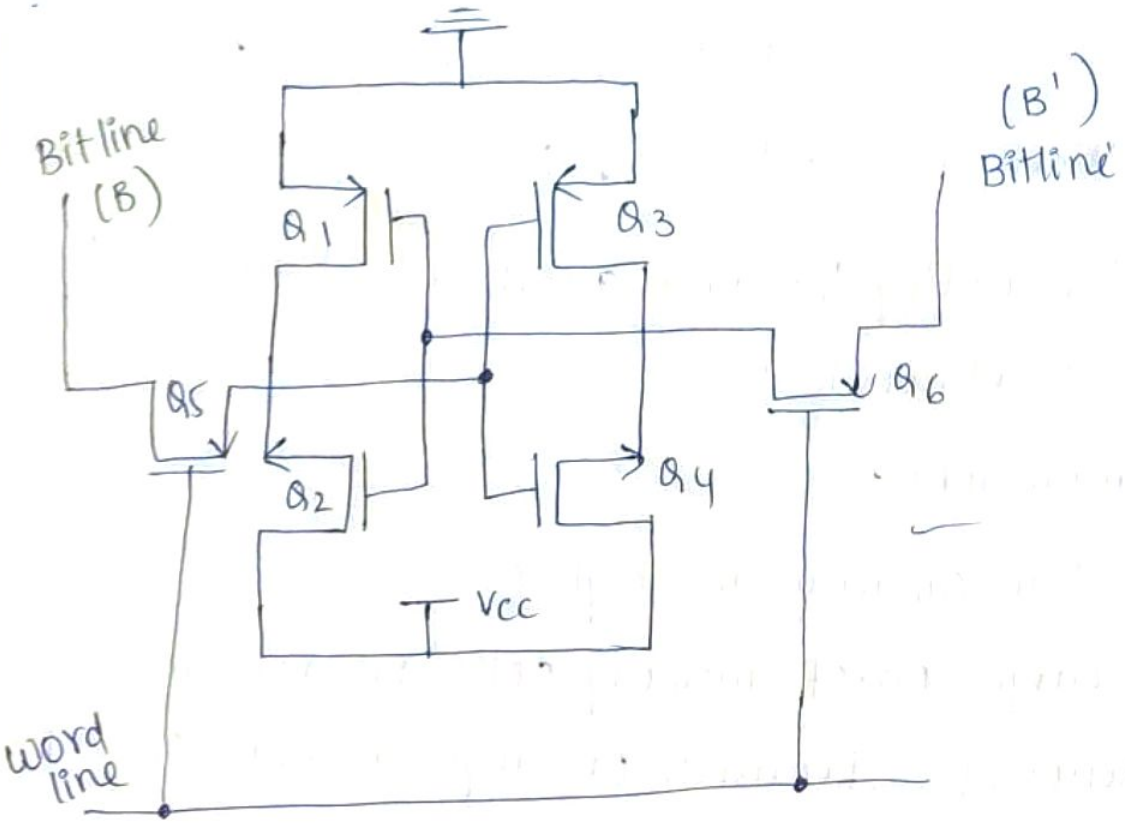


fig b:- SRAM cell implementation

Static RAM stores the data in the form of voltage. They are made up of flip flops. Static form of RAM is faster form of RAM available. An SRAM cell is realized using 6 transistors. 4 of the transistors are used as latch (flip flop) part of the memory cell and two for controlling the access. SRAM has high resistive network and switching capability.

* An SRAM in fig (b) ~~and~~ can be visualized in fig (a), with two cross coupled inverters with read/write control through transistors.
* Q_5 and Q_6 are access control transistors which are

connected to bit line B and B' respectively.

* When B = 1 and B' = 0, for writing '1'.
  When B = 0 and B' = 1, for writing '0'
  } make word line high

* for reading the data assert both B and B' to '1' and word line to '1'.

Limitations :-

* Low capacity and high cost
* Large no. of memory cells are required due to presence of 6 transistr on single memory cell. This increases the size of memory device.

D-RAM (Dynamic RAM) :-

It stores data in the form of charge. They are made up of a MOS transistor and a capacitor. It is high dense and low cost RAM. It requires refreshing of data.

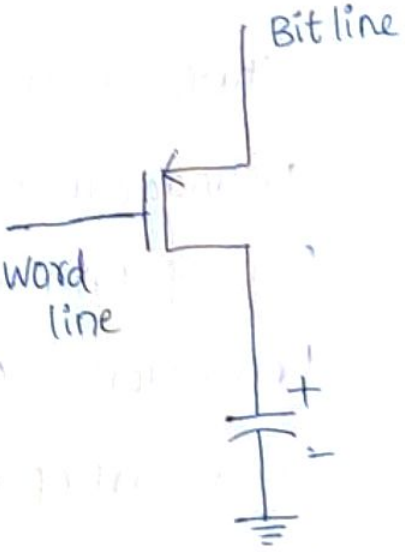


fig:- DRAM cell

* To avoid leakage of charge with time, the memory should be refreshed periodically.
* Special DRAM controllers are used for this purpose.
* The MOSFET acts as a gate ~~capacitor~~ for incoming and out going data and capacitor acts as a storage device.

Sensors & Actuators :-

(14)

Sensor :- It is a transducer which converts one form of energy to another form for measurement purpose (or) for control.

* Sensors are connected to the input ports of embedded systems.

Actuator :- It is a form of transducer which converts signals to corresponding physical action (motion).

* Actuators are connected to the output ports of embedded systems.

The I/O subsystem :-

It facilitates the embedded system interact with external world.

* Sensors cannot be directly connected to input ports, they must be interfaced through signal conditioners and translating systems like ADC, opto couplers etc--

LED (Light emitting diode) :-

It is an important output device for visual indication. It can be used as an indicator for various signals (or) situations

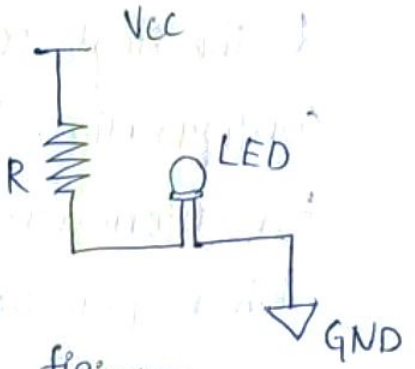


fig:- LED interfacing

7 Segment display :-

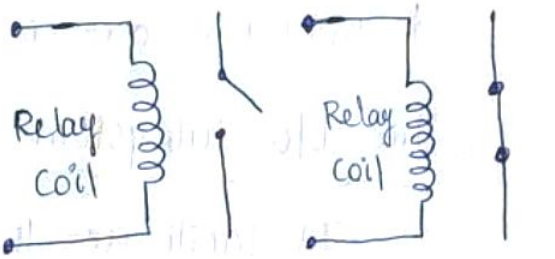
The 7-segment LED display is an output device for displaying alpha numeric characteristics. It contains 8 light emitting diode segments arranged in a special form.

Out of 8 LED's 7 are used for display alpha numeric character and one is used for decimal point.

Relay :-

It is an electro mechanical device. It consists of relay coil made up of insulated wire on a metal core and a metal armature with one (or) more contacts. It works on electromagnetic principle.

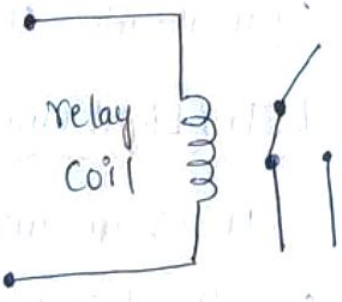
When voltage is applied to relay coil current flows through coil, which inturn generates magnetic field. The magnetic field attracts the armature core and moves the contact point.



a) single pole single throw normally open

b) single pole single throw normally closed.

* The single-pole single throw configuration has only one path for information flow. The path either open (or) closed.



c) single pole double throw

* for single pole double throw relay there are two paths for information flow and they are selected by energising (or) de-energising the relay.

Communication interfaces :-

(15)

These are divided into two types.

1. Device | on board level communication interface
2. product level communication interface

1. On board communication interface :-

Inter integrated circuit (I2C) bus :-

It is a synchronous bi-directional half duplex two wire serial interface bus. The I2C comprises of two bus lines, serial clock - SCL and serial data - SDA

SCL ⟶ To generate synchronized clock pulses.

SDA ⟶ Responsible for transmission of data serially.

* Devices connected to I2C bus can act as master | slave device, the master device is responsible for controlling the communication by initiating | terminating data transfer, sending data (or) clock pulses.
* slave devices for commands from the master.

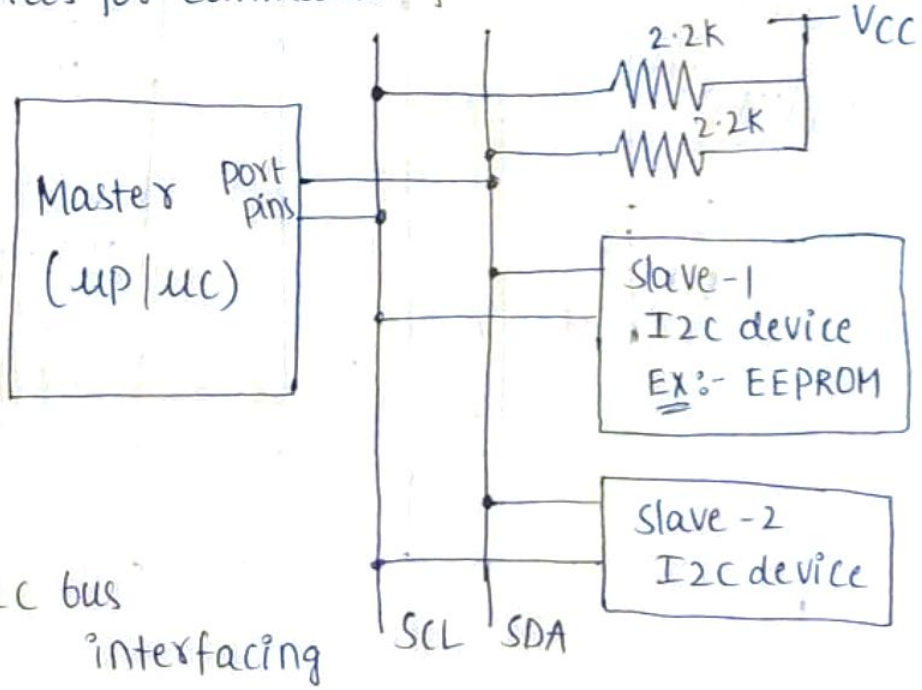


fig:- I2C bus interfacing

Serial peripheral Interface (SPI) Bus:-

* It is synchronous bi-directional, full duplex, four-wire serial interface bus. It is single master and multi-slave system. It requires 4 signal lines for communication.

Master out slave In (MOSI):- Signal line carries the data from master to slave device. It is known as slave In / slave data in (SDI).

Master In slave out (MISO):- Signal line carries the data from slave to master device. It is known as slave out / Slave data out.

Serial clock (SCIK):- Signal line carrying the clock signals.

Slave select (SS):- signal line for slave device select. It is active low.

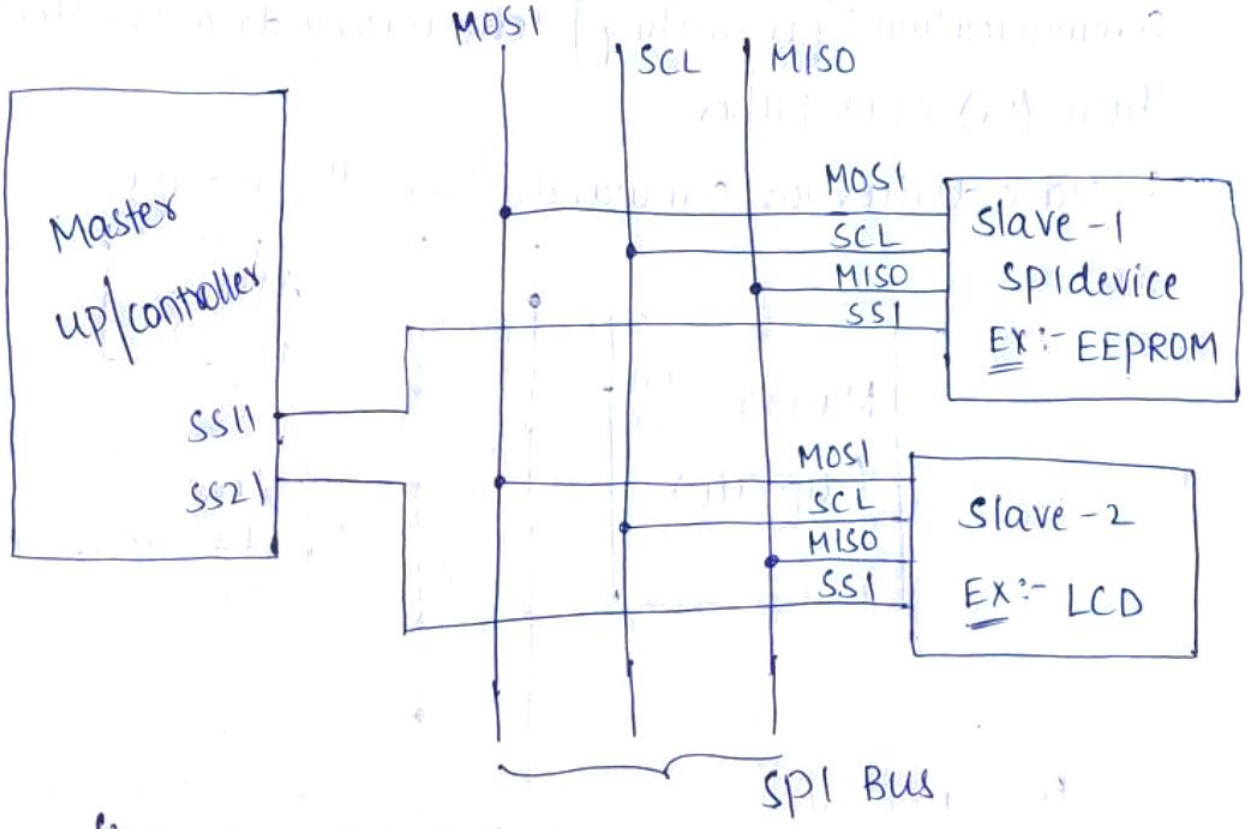


fig:- SPI bus interface

Universal Asynchronous Receiver transmitter UART :- (16)

UART based data transmission is an asynchronous form of serial data transmission. It doesnot require a clock signal to synchronize the transmitting and receiving end for transmission. Sending byte of data, a start bit is added first and a stop bit is added at the end of the bit stream.

The start bit informs the receiver that a data byte is about to arrive. UART of the receiving device calculates the byte received and compares it with the received parity bit for error checking.

External communication Interface :-

1. RS-232 & RS-485 [Recommended Standard-232] :-

It is a legacy, full duplex, wired, asynchronous serial communication interface. RS-232 interface is developed by electronic industries association. RS-232 extend UART communication signals for external data communication.

2. Universal serial bus (USB) :-

It is wired high speed serial bus for data communication. The first USB (USB 1.0) was released in 1995. It follows star topology with a USB host at the centre and one (or) more USB peripherals connected to it.

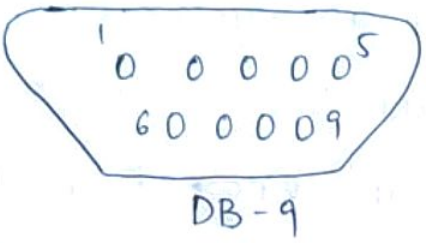


fig:- RS-232 interface.

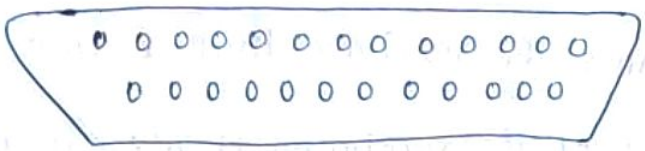


fig:- DB-25, RS-232 interface

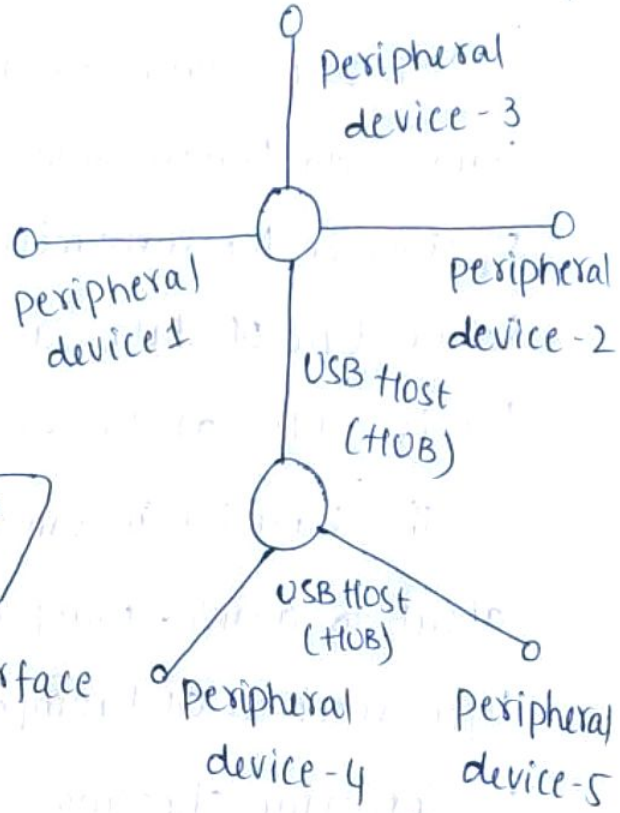


fig:- USB device connection topology

IEEE 1394 (fire wire):-

It is a wired high speed serial communication bus. It is an interface for connecting embedded devices like camera, camcoder, scanner to desktop computer.

Bluetooth (BT):-

It is a low cost, short range wireless technology for data and voices communication. It operates at 2.4GHz of RF spectrum. It has two essential parts

1. physical link 2. protocol link

1. physical link is responsible for physical transmission of data between devices

2. protocol part is responsible for defining the ruler of communication.

Wifi :-

wireless fidility is the popular wireless communication technique for networked communication of devices. It supports internet protocol (IP) based communication. wi-fi operates at 2.4 GHz to 5 GHz radio spectrum. It requires a wifi rater to communicate between the systems

General packed radio service (GPRS) :-

GPRS is a communication technique for transfering data over a mobile communication network like GSM. Data is sent as packets in GPRS communication.

Embedded firmware :-

It refers to control algorithm and configuration settings that an embedded system developer dumps into the code memory of embedded system.

There are various methods available for developing the embedded firmware.

① write a program in high level languages like embedded C/C++ using an IDE (integrated development Environment). IDE's are different for different processors/controllers.

(2) write the program in assembly language using instructions supported by your applications target processor

Characteristics of embedded Systems :-

1. Application and domain specific :-

Embedded systems are used for the specific application only i.e they cannot be used for any other purpose.

Ex:- The embedded control units of Air conditioner (AC) and microwave oven are specifically designed to perform certain specific task.

2. Reactive and real times :-

Due to the presence of sensors connected to input ports any change in the external world is sensed by them and according to the change the real time control algorithms are applied to bring controlled variables to designed level.

Ex:- flight control system, anti breaking system in automobiles.

3. operates in harsh environment :-

An embedded system can be placed in dusty (or) harsh (or) high temperature zone by properly designing all these constrains. Irrespect of cost high graded components are used to handle in worst environment condition.

4. Disturbed :-

An embedded system may be a part of larger systems. It has to work along with other different embedded systems. In such disturbed conditions also an embedded function has to function properly without any variation.

Ex:- Automatic vendor machines contains card reader, a vending unit e.t.c.. ATM (Automatic teller machine) contains card reader, transaction unit, currency count unit and printer unit.

5. Small size and weight :-

Compactness is a significant and dedicating factor in design of an embedded system. Most of the applications demand small size and low weight products.

6. power concerns :-

An embedded system is designed in such a way to minimize the heat dissipation by the system. The production of high amount of heat demands coding requirements like coding fans which inturn occupies additional space and makes system bulky.

Quality Attributes of embedded System :-

These are classified into two types :-

1. operational quality attributes
2. Non-operational quality attributes.

Operational Quality attributes:-

1. Response 2. Throughput 3. Reliability 4. Security 5. maintainability 6. Safety.

1. Response:-

It is measure of Quickness of the system. Any response delay in the system will create potential damages to the safety and embedded systems and humans working on it. It is not necessary that all the embedded systems should be real time in response

Ex:- Electronic toy need not have any time critical response

2. Throughput:-

It is efficiency of the system. It is defined as the rate of production (or) operation of a definedprocess over a stated period of time. In case of a card reader the through put is measured in terms of no. of transistors per minute, It can perform. It is used to compare the embedded system performance with other products.

3. Reliability:-

Mean time between failures (MTBF) and mean time to repair (MTTR) are the terms used to define reliability. It is a measure of how much % we can relay upon an embedded System (or) what % is susceptability of the system to failures.

4. Security:- confidentiality, Integrity and availability are three major measures of information security. Confidentiality deals with the protection of data from un-authorised users. Integrity deals with un-authorised modifications eradication. Availability deals with protection of data from unauthorized.

Ex:- PDA (personal digital Assistant) - Read only access to all users provides Integrity, some of the files are accessible by administrator only which provides confidentiality providing username and password to access is provided by availability.

5. Maintainability:-

It deals with the maintainance and support to the end usermaintainance can be two types. scheduled (or) Periodic maintainance (or) maintainance to unexpected failures. Ex:- printer.

6. Safety:- It deals with the possible damages that can happen to the operators, public and environment due to break down of embedded system (or) due to emission of radio active materials from an embedded system. Break down may occur due to hardware failure (or) software failure. some of safety threats are sudden (like product breakdown) and some are gradual (like hazardous emission of from the product).

Non-operational Quality attributes

1. Testability
2. Evolvability
3. postability
4. Time to prototype and market
5. pre-unit and total cost.

1. Testability (debugability):-

It is a verification of embedded design hardware which can function a specific task for the written firmware init. Debugging can be done at two levels. hardware level debugging which is used to clear the errors in hardware where as firmware level debugging is to clear the errors in programming.

2. Evolvability:-

The ease with which an embedded product can be modified to take advantage of new firmware and hardware technologies.

3. portability:- (system independentness measurement):-

The flexibility of firmware and hardware of the embedded systems to adopt according to the different working environmental conditions. for example an application developed on JAVA can be worked on any operating system that supports JAVA settings. But

microsoft visual c++ uses visual studio which cannot run on other operating systems except microsoft platforms.

4. Time to prototype and market :-

The time between the conceptualization of product and time at which the product is ready for selling. Embedded market is highly competative and time to market the product is critical factor in success of embedded project. The embedded project should have limited prototype time to enter the market long time leads to change in the technology in the market.

5. pre unit and total cost :-

cost is monitoring factor for both end user and product manufacturer. The budget and total system cost must be properly balanced to achieve profit. During design and development there is only investment no return. once product is ready to sell is introduced into market. Initially the sales of the product is low but steadly they increase. After that due to heavy competition again there may be some fluctuations in declining the cost of product.

Application specific embedded systems :- (EX:- washing machine)

An embedded system contains sensors, actuators, control unit and application specific user interfaces like keyboards, displaying units e.t.c.. we can see all these components in washing machine.

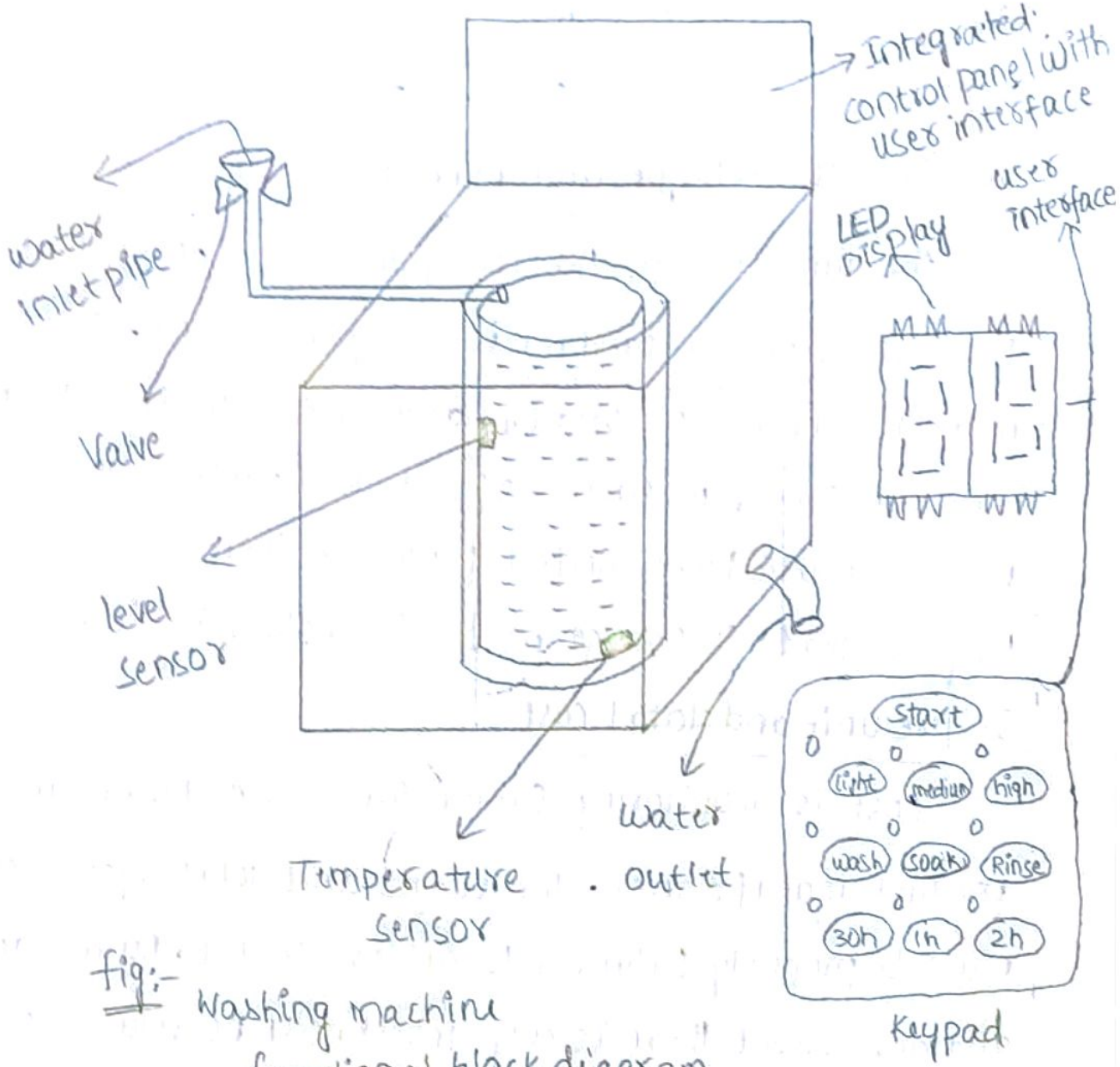


fig:- Washing machine functional block diagram

Actuators:-

A motor connected to agitator with a tub and water drawing pipe and inlet valve to control water flow are few actuator present in washing machine

Sensors:-

water temperature sensor, level sensors are few sensors present in the tub.

Control unit:- A microprocessor / controller based board (21) with interface to sensors and actuators. It also provides connectivity to user interfaces like Keypad for setting washing machine time, Select the type of material to be washed etc. user feed back is reflected through an LED connected to control board.

* washing machine comes in two models ① top loading ② front loading
* In toploading agitator of the machine twists back and pulls the cloths down to bottom of tub.
* on reaching the bottom of the tub the cloths work their way back to the top of tube repeats the same mechanism.
* In front loading machines, the close are tumbled & plunged into water over and over in first phase of washing.
* In second phase, water is pumped out and inner tube uses centrifugal force to rinse out water by spinning fastly.
* By selecting proper key on keyboard such as rinse, wash, spin the machine works according automated.

Automotive communication Buses:- (Domain specific example)

1. Controller Area Network (CAN):-

It supports medium speed (125 Kbps) an high speed (1Mbps) data transfer. It is generally employed in safety systems like train engine control, Antilock break (ABS) systems, Navigation systems like GPS.

2. Local Inner connect network (LIN):-

It is single master multiple share communication interface It is low speed, single wired interface with support of data rates upto 20kbps and is used in sensor/ actuator interface. It is used in applications like mirror control, fan control, Seat position control e.t.c--

3. Media-oriented system transport (MOST):-

It is targeted for automotive audio/video equipment interfacing used in european cars. It is optical fiber cable connected between the electrical optical converter (EOC) and optical electrical converter (OEC). It is multimedia fiber optic point to point network.

Keyplayers of Automotive embedded market:-

1. Silicon providers:- They provide necessary silicon chips which are used in control application development. The chip may be DSP controller (or) ADC/DAC chips.

2. Solution providers:- They provide complete solution for automotive applications by using chips, platforms and different development tools.

EX:- DENSO Automotive, Infosys technologies, Delphi e.t.c.

2. Local Inner connect network (LIN):-

It is single master multiple share communication interface It is low speed, single wired interface with support of data rates upto 20kbps and is used in sensor/ actuator interface. It is used in applications like mirror control, fan control, Seat position control e.t.c--

3. Media-oriented system transport (MOST):-

It is targeted for automotive audio/video equipment interfacing used in european cars. It is optical fiber cable connected between the electrical optical converter (EOC) and optical electrical converter (OEC). It is multimedia fiber optic point to point network.

Key players of Automotive embedded market:-

1. Silicon providers:- They provide necessary silicon chips which are used in control application development. The chip may be DSP controller (or) ADC/DAC chips.

2. Solution providers:- They provide complete solution for automotive applications by using chips, platforms and different development tools.

EX:- DENSO Automotive, Infosys technologies, Delphi e.t.c.

UNIT-II

Embedded Hardware Design

Analog and digital electronic components:-

Analog electronic components:- Various analog electronic components are commonly used in designing the hard ware of an embedded system. The role of some of these components is explained as follows,

1. Resistor:- Resistor is a current limitting device and is commonly used to interface LED's and buzzer's with the port pins of a micro controller in an embedded application.

2. Transistor:- Transistors are popularly used in embedded applications for performing switching and amplification functions. when transistor acts as a switch, it remains either in ON (or) in OFF state whereas in amplification application, a transistor always remains in ON state. for switching and driving circuits in embedded systems NPN transistor in common emitter configuration is used extensively. Examples of such circuits are relays, buzzers and stepper motor driving circuits.

3. Capacitor :- Capacitors are generally used in signal filtering and resonating circuits for performing implementation of reset circuits, matching of circuits for RF designs, decoupling of power supply in embedded applications. Various types of capacitors that are widely used in embedded hardware design are electrolytic capacitors, Ceramic capacitors, and tantalum capacitors.

4. Diode :- the most commonly used diodes in embedded hardware circuits are p-N junction diodes, Schottky diodes and Zener diodes. A schottky diode acts similar to a p-N junction diode, but has low forward voltage drop of the order of 0.15V to 0.45V when compared to a p-N junction diode whose forward voltage drop is of the order of 0.7V to 1.7V, also it has a very small current switching time when compared to a p-N junction diode. A zener diode in forward biased condition acts as p-N junction diode & is used in voltage clamping applications. Various other functions performed by a diode in an embedded system are reverse polarity protection, voltage rectification, free-wheeling of current produced in inductive circuits etc---

* Digital electronic components :- the various digital electronic components used in embedded system development are as follows.

1. Open Collector and tristate output
2. Logic gates
3. Buffer
4. Latch
5. Decoder
6. Encoder
7. Multiplexer
8. De-multiplexer
9. Combinational circuits
10. Sequential circuits.

Open collector and Tristate output :- Open collector is employed in embedded system due to the following reasons,

(i) with the open collector, the additional interface circuits are eliminated.

(ii) The open collector output lines are used to build "wired AND" and "wired OR"

(iii) Multi drop connection (connecting more than one open collector output to a single line) is supported by it.

* The below figure shows open collector o/p configuration.

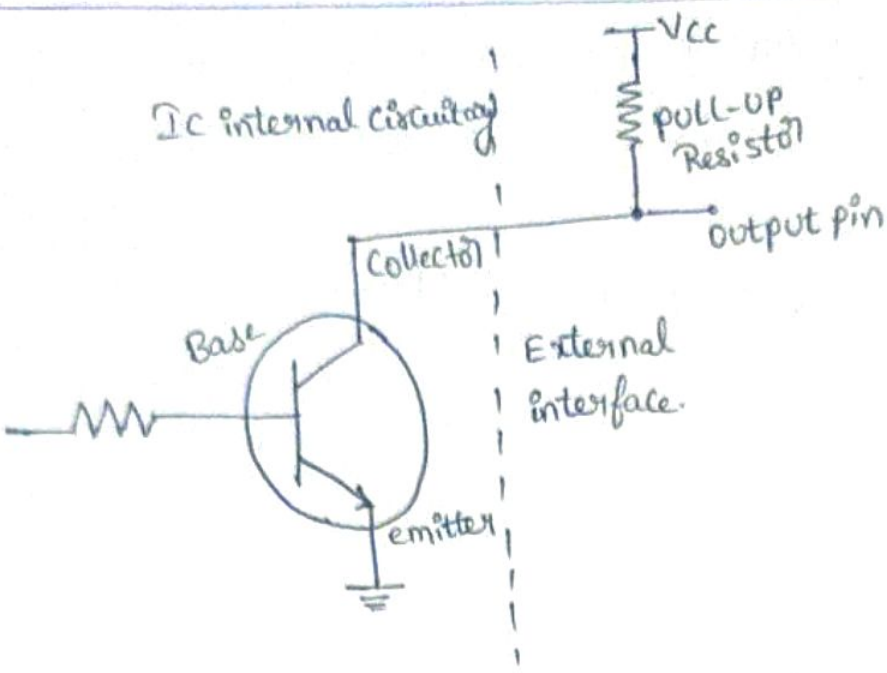


fig:- Configuration of open collector.

* the tristate output allows more devices to share a common bus.

* for the remaining digital electronic components please refer your "DICA material."

Examples of various I/O Devices:-

	I/O Device type	Example
1.	Serial Input	– Audio Input, Video Input, Dial tone, Network Input, transceiver Input, Scanner, Remote i/p and serial I/o bus I/p.

2. Serial output — Audio output, Video o/p, Dialing number, Nlw o/p, Remote TV Contol, transceiver o/p, multi-processor Communication and Serial I/o bus o/p.

3. Serial UART Input — Keypad, mouse, Keyboard, modem, character Inputs on Serial line.

4. serial UART output — modem, printer, character o/p's on serial line.

5. parallel port single bit I/p — (i) Completion of a revolution of a wheel.
(ii) acheiving preset pressure in a boiler.
(iii) exceeding the upper limit of the permitted weight over the pan of an electronic balance.

6. parallel port single bit output — (i) PWM o/p for DAC, which Controls liquid level, temperature, pressure, speed (or) angular position of a rotating shaft (or) a d.c motor Control.

7. parallel port Input – (i) ADC input from liquid level measuring sensor (or) temperature sensor (or) pressure sensor (or) speed sensor.

8. parallel port output – (i) Multiline LCD display matrix unit in a cellular phone to display on the screen the phone number, time, messages, character Inputs (or) pictogram bit-images (or) E-mail (or) webpage.

Serial Communication Devices:- [modes]

* There are three modes of serial Communication devices. they are (i) synchronous mode
(ii) iso synchronous mode
(iii) Asynchronous. mode.

Synchronous Communication:- The Communication in which data byte is transmitted (or) received at fixed intervals of time along with constant phase differences is known as "synchronous Communication". the data bits are transmitted with in a constant maximum time interval.

* An example of synchronous serial communication is frames sent over a LAN. Frames of data communicate with the time interval between each frame remaining constant.

* There are two characteristics of synchronous communication are as follows.

(i) Frames maintain a uniform phase difference i.e; the frames are in complete synchronization with each other. The synchronous communication does not allow hand shaking (source and destination exchange signals before communication of data) among the serial transmitter and receiver port within the communication interval.

(ii) The clock ticking must always be present in order to transmit serial data bytes. Frames sent over a LAN, inter processor communication in a micro processor system are the examples of synchronous serial communication.

Iso-Synchronous Communication:- It is a special type of synchronous communication in which the data bytes are sent (or) received at maximum variable time intervals with uniform phase differences.

* the "voice over Internet protocol" (VOIP) system is an example for Iso- synchronous communication.

Asynchronous Communication: Asynchronous communication is defined as the communication in which data bytes are sent (or) received at variable intervals of time.

Characteristics:-

(i) frames in asynchronous communication doesnot maintain uniform phase difference i.e; frames are not synchronized. It is in between handshaking of txer and receiver ports.

(ii) NO clock rate information is transmitted along with the serial data and thereby the receiver clock does not maintain identical frequency and phase difference with the transmitter clock.

⇒ Examples of asynchronous communication are keypad communication, RS 232C communication b/w UART devices and comm. over a telephone line.

Serial Communication devices:- Serial communication is the process of transmitting data bit by bit over a communication path.

* there are three ways by which serial communication takes place. they are.

(i) Synchronous Communication
(ii) Iso- synchronous Communication
(iii) Asynchronous Communication.

* the various serial Communication devices which supports serial Communication are,

(i) RS232 | RS 485
(ii) USART
(iii) HDLC
(iv) X.25
(v) ATM
(vi) DSL
(vii) ADSL
(viii) UART

UART :- (Universal Asynchronous Receive Transmit):-

* UART stands for universal Asynchronous Receive transmit. It is designed to convert data from serial form to parallel form and viceversa.

* It can be used as a medium of Communication b/w the processor and RS232 port. There are two units in UART namely receiver unit and transmitter unit. A serial data is given as input to the receiver unit that produces parallel data to processor. on the other hand transmitter unit

accepts the processors output data that is in the parallel format, thus producing the data in serial form. In addition the data is attached with a start bit, stop bits and a parity bit. Most of the micro controllers contain UART present on their chip.

* Even though the voltage level of the UART must be converted in order to match the standard RS232 voltage level. Hence a level shifter is employed between UART and RS232 as illustrated in figure.

figure:- Interfacing UART to RS232 Connector.

* The operating voltage of UART is 5 volts, this voltage is then converted to a voltage level that matches RS232 through the level shifter and the signals are transmitted to it.

* The RS 232 standard supports connectivity upto 19.2 meters of distance.

* The rate of data transmission (i.e; data rate) provided by UART varries according to the type of UART chip employed and its clock frequency.
* The Ic's used for level shifter are MAX 3222, MAX 3241. Maximum processors along with Digital Signal processors (DSP) contain UART present on their chip.

Parallel port Devices:-

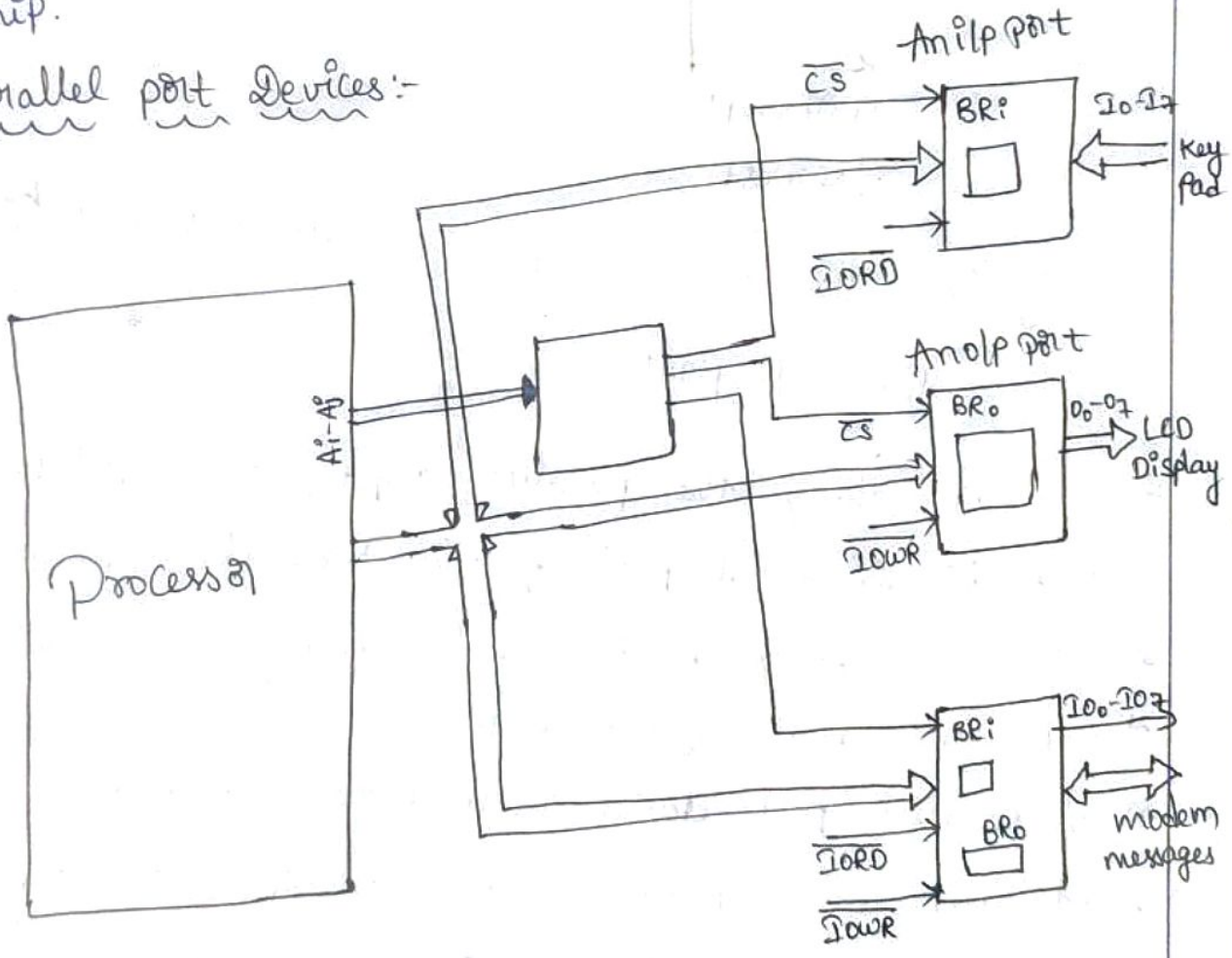


fig (a):- parallel input port, output port and a bidirectional port for Connecting the device.

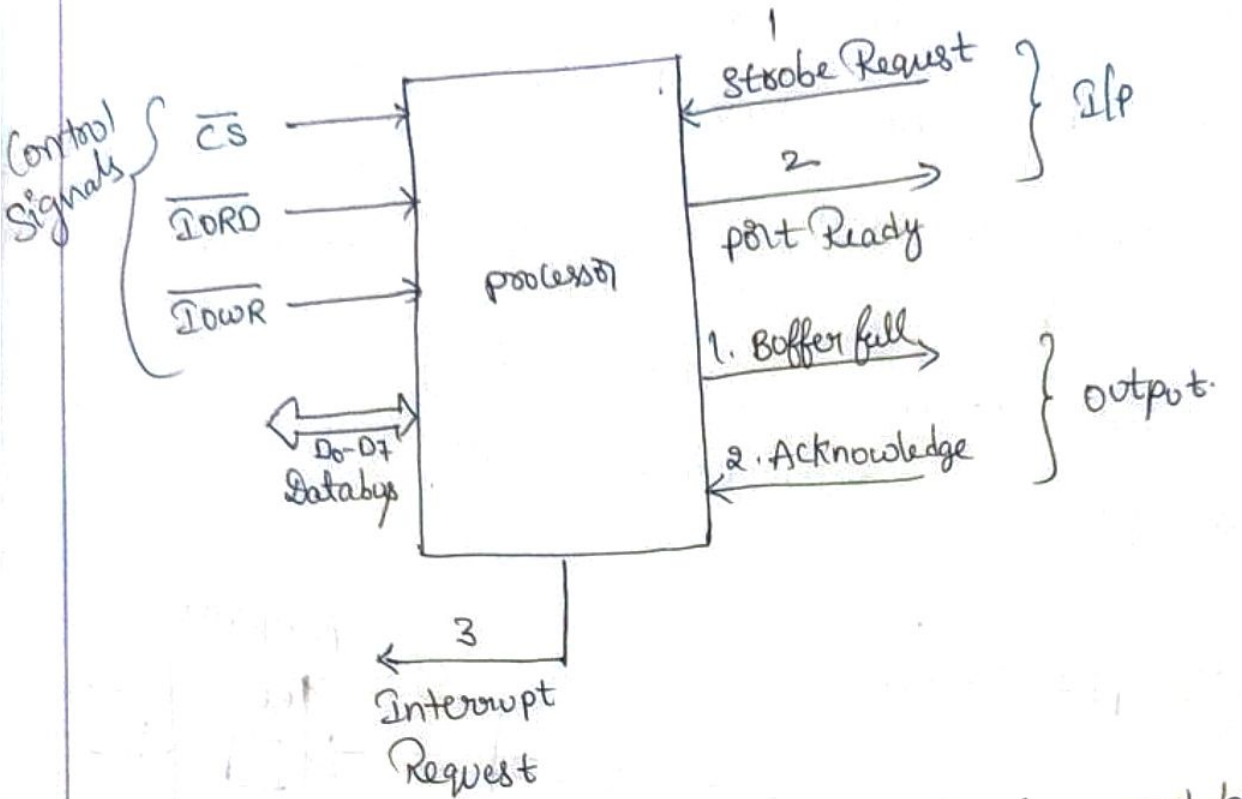


fig (6):- The hand shaking signals when used by the I/o ports.

* fig (a) shows the parallel Input port, output port and a bi-directional port for the three devices, respectively.
* It also shows a device - interfacing circuit with the processor and system buses.
* A parallel port device is used to perform simultaneous read and write operations over multiple bit. The capacitive effect of parallel wires has the following major impacts on the communication.

(i) It tends to decrease the distance of parallel communication.

(ii) The bits undergoing transitions at the other end experience a delay in their transmission due to high capacitance of parallel wires.

(iii) The large amount of capacitance generate noise and cross talk among the parallel wires.

fol these reasons, Parallel ports are used for short distance communication.

* It can be observed from above figure that the port I_0 to I_7 are inputs to the keypad controller, o/p ports O_0 to O_7 are output bits to LCD display output controller. the above figure depicts that.

(i) Status pin → Input status signal to external ckt

(ii) BR_i and BR_0 → Input and output data buffers at bi directional I_0 port.

(iii) A_i and A_j → Address ports connected to device through a port address decoder.

(iv) $\overline{IORD}$ & $\overline{IOWR}$ → Control signal for read and write operation in case of 8086 processor.

(v) $\overline{RD}$ & $\overline{WR}$ → Memory Read and write signals used to map I_0's.

(vi) $\overline{CS}$ → Control signal from o/p to external ckt.

wireless devices (or) wireless communication using different protocals:-

* The wireless devices after suitably modulating the data bits, uses either Infrared (IR) (or) Radio frequencies for their operation.

(i) Infrared frequency (IR):- the IR transmitter communicates over LOS (line of sight) and employs a photo transistor at the receiver side, in order to detect the IR Rays. TV Remote Controller, robotic systems are the major application domains of IR Communication. the IR devices make use of IRDA protocal for communication purposes.

* Frequency Hopping is employed is bluetooth as more number of devices need to communicate in limited number of frequency bands. the data bits are modulated and demodulated as per protocal specification.

(ii) Radio frequencies:-

* These frequencies provide both short and long distance communications. The source and receivers employ antenna to send and receive signals. A modulator and demodulator is also used in order to transmit data

over RF frequencies.

* The most commonly used protocals in wireless devices are bluetooth, IRDA (Infrared Data Association), 802.11, zigbee.

* The below table illustrates the various frequency bands used by radio frequency wireless devices.

wireless Device protocal	Carrier Frequency.
Bluetooth / zigbee	2.4 GHz / 900 MHz
Mobile CDMA	2GHz
Mobile GSM	890-915 / 1710-1785 / 1850-1910 MHz

Advantages of wireless Devices:- The transmission system use wireless devices for data transmission, these wireless devices offer number of advantags. few of them are,

(i) Low cost:- As the wireless devices does not employ cables in their design, the cost required for the implementation of these devices is very much less than when Compared to wired devices.

(ii) Mobility:- the device can be moved easily with in the wireless range. It is more suitable for non-reachable places such as hilly (or) river line (or) rural areas.

(iii) Support more no. of users :- wireless devices support more no. of users in a particular range, as they do not limit the slots available in a system.

(iv) Flexibility :- wireless devices are flexible in adhoc situations where there is a need for additional work station.

(v) Easy Installation :- since no cables are used, the wireless devices are installed without any difficulty.

(vi) Support temperory N/w setup :- wireless devices are ideal for temperory n/w setup.

Timer and Counting Devices :- Timer cum Counting device is used to count both irregular and regular events occuring at the i/p port of the device. It is both time and counting device and performs the following two operations.

(a) It counts the inputs occuring due to irregular time intervals and

(b) It counts clock input pulses occuring at regular intervals of time.

The device functions as either a timer or a counter depending on the input (or) status bit of a timing device register.

* It is necessary to include atleast one timing device in a system which generates interrupts for each clock ticks.

Hard ware timer :- The hard ware timer is used as a system clock to generate "num ticks" number of system clock ticks before system interrupts. A clock -out signal provides inpot from the processor to the hard ware timer and the system clock is activated depending on the "num ticks" defined at the timer.

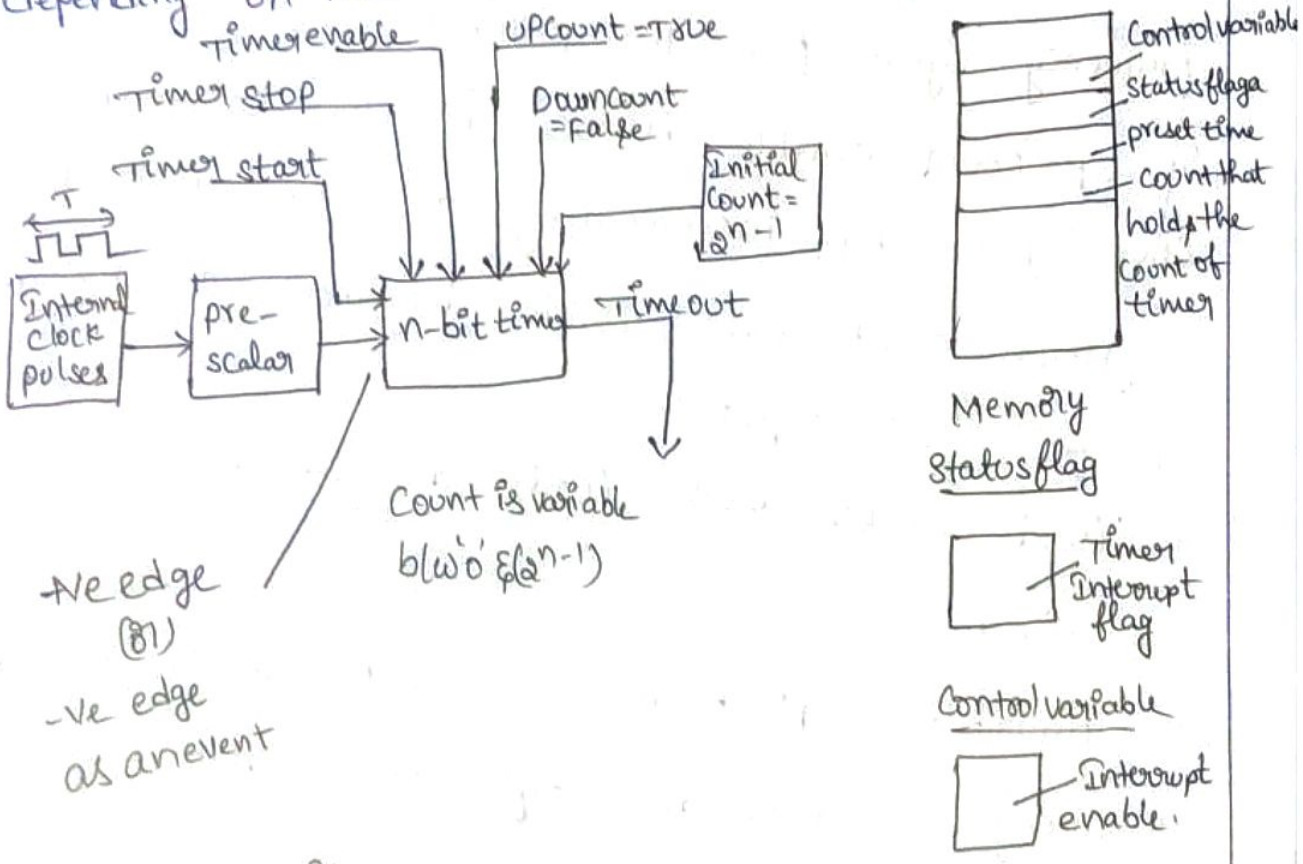


fig:- Hard ware timer Device.

* the above figure depicts that an hardware timer consists of a control bits (based on hardware signals and its associated bit at control register). they are,

1. Timer enable is used to activate a timer device.
2. Timer start which starts counting at each clock Input.
3. Timer stop from the next clock pulse is used in order to stop counting.
4. prescaling bits are used to divide the clock-out frequency signal from the processor.
5. up-count enable is used in order to enable counting up by incrementing clock on every I/P.
6. Down-count enable which decrements count on a specific clock input.
7. Load enable which loads a register value into the timer.
8. Timer-Interrupt enable is used to allow the interrupt servicing whenever an overflow occurs at a timer.

Software Timer :- The input to the software timer (SWT) is the system clock ticks which generate interrupt (a) at periodic intervals of time in accordance with the count - value set. All the software timers present in the activated list employ the same common input of system clock ticks. on reaching a time-out (i.e; count value = 0) all the SWTs set a status flag in the system.

* These timers are also used as virtual timing Devices.

Fig(2) shows the control and status bits in software timer device.

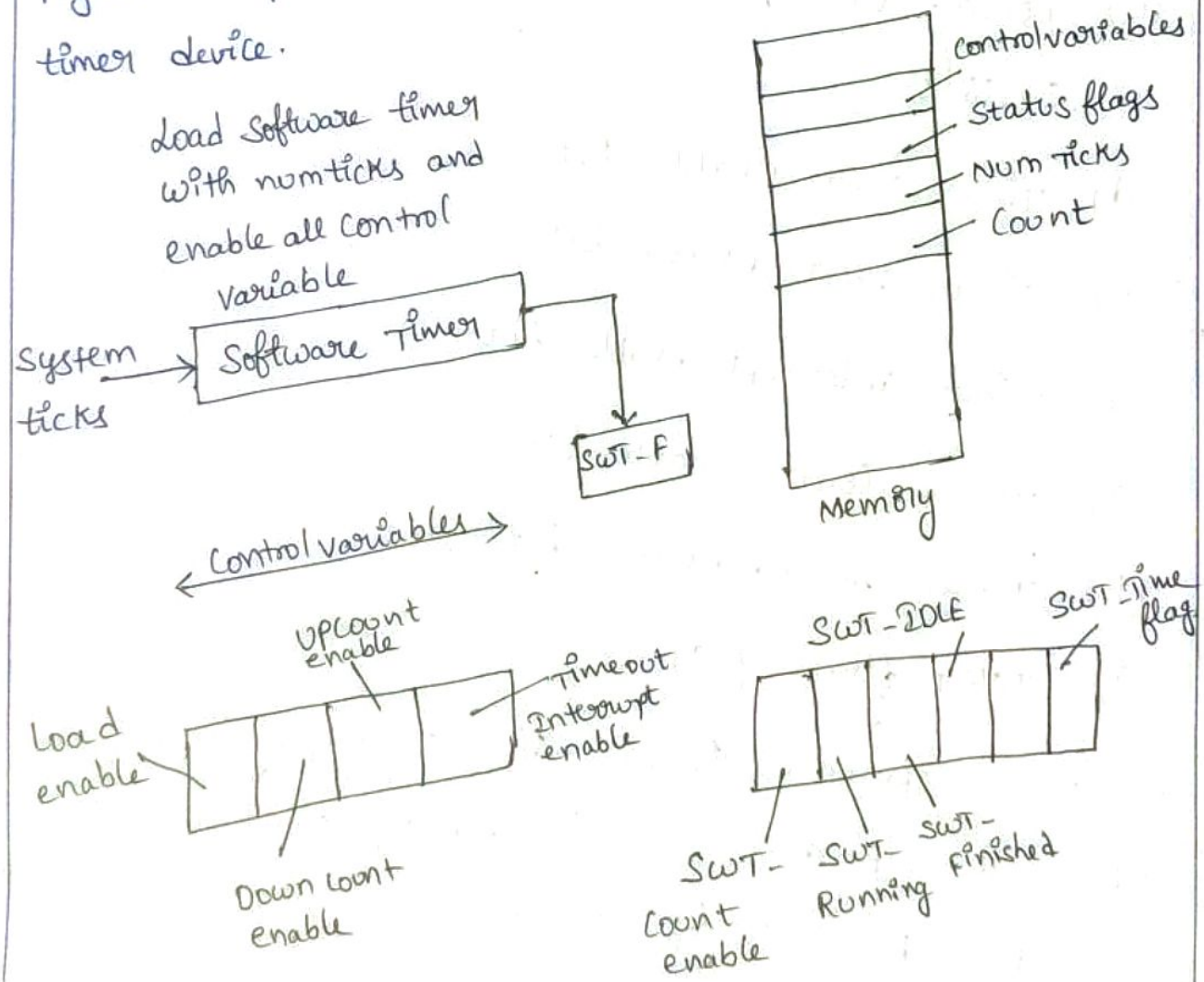


fig:- Software Timer Device.

* Control bits in SWT are set based on the applications and SWT consists of a number of control bits and a time-out status flag in each timer device and functions similar to the hardware timer device. The various available control bit variables and status flag in a software timer are ,.

(a) Reset value 32/16/8.

(b) Initial load value 32/16/8.

(c) Count - value, maximum value and minimum value 32/16/8.

(d) Timer run enable bit.

(e) Timer reset enable bit.

(f) Timer interrupt enable bit.

(g) Timer load enable bit.

(h) Timer reload enable bit and

(i) Over flow - flag.

* The software timers does not employ hardware i/p and o/p and are limited by number of interrupts provided by the user.

Watch Dog Timer (WDT) :- A timing device set for a predefined interval of time for occorance of an event is known as watch dog timer. the device generates the time-out signal if no event occurs in the specified interval of time.

* For instance, Consider that a set of events must occur within a time period of 100ms. If the task is accomplished in the specified time, the WDT (watch Dog Timer) gets disabled and stops Counting. on the other hand, if the system fails to accomplish its tasks the WDT (watch Dog timer) generates interrupts after predefined time (i.e; 100ms) and executes a routine that runs.

1. watch-dog timer can be either a programmed Software task (or) a micro Controller.

2. In mobile phones, WDT's save power when there is no GUI Interaction within a specified time it turnoffs the display. An interval is set to turnoff the display is 15, 20 (or) 25 sec in a mobile phone.

3. In Case of temperature control systems, if the Controller fails to switch off the Current in a given

time, the current automatically gets switched off and a warning signal is issued by indicating failure of the Controller.

4. In mobile phones, beep sound is used to attract attention of user when menu is not selected with in a specified time interval.

* It is a hardware timer for monitoring the firmware execution. If firmware execution doesnot complete due to malfunctioning with in the time required by watch dog to reach maximum Count. The Counter will generate reset pulse and this will reset the processor.

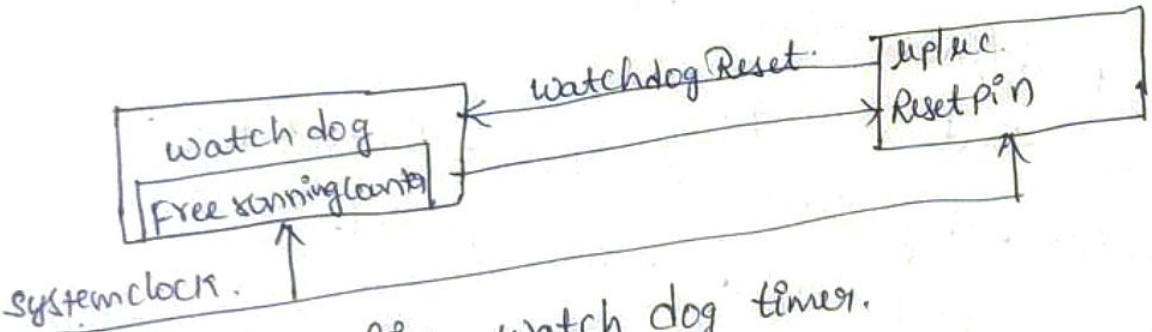


fig:- watch dog timer.

Real time clock:- It is a system Component responsible for keeping the track of time. RTC holds informs like current time 12 hours / 24 hours format.

* It works even if the power supply is off. It contains a micro chip holding time and data related informed and back up battery cell for functioning in the absence of power.
* A clock which generates interrupts continuously at a regular intervals on each clock ticks is known as Real time clock (RTC). After occurance of each time-out (or) overflow of the clock, an interrupt service routine always gets executed in the system. The real time clock once initiated cannot be either stopped (or) modified with another value. use of Real time clock (RTC) in a system preserves both current time and dates. It also enables to initiate return of control to the system after a preset clock period.

Embedded firmware design approaches:

Firmware design depends on the complexity of functions to be performed and speed of operation required. Two basic approaches of Embedded firmware design are:

1. Conventional procedural based firmware design (Super loop model)
2. Embedded operating system (OS) based design.

1. Super loop based model:

* Super loop firmware design approach is adopted for applications that are not time critical and where the response time is not so important.
* These are the embedded systems where the code is executed task by task where mining deadlines are acceptable.
* Super loop based design doesnot require an operating system since there is no need for scheduling and assigning of priority to each task.
* In this design, priority of tasks are fixed and the order of execution is also fixed.

Ex:- Reading/Writing data to and from a card using card reader requires a sequence of operations like checking the presence of card, authentication of the operation, reading/writing etc.

It should follow a sequence and the combination of all these series of tasks constitute a single task.

Advantages:

* Superloop design is simple and straight forward without any OS related overheads.
* No need of special operating system (OS)

Disadvantages:

* Any failure in the part of task will affect the total system.
* Lack of real timeliners brings the probability of missing events.

The firmware execution flow of superloop model will be

① Configure the command parameters and perform initialization of various hardware components, memory, registers etc.
② Start the first task and execute it.
③ Execute second task
④ Execute next task
⑤ ⋮
⑥ ⋮
⑦ Execute the last defined task.
⑧ Jump back to the first task and follow the same flow.

Ex:-
```
Void main()
{
    configurations();
    Initializations();
    while(1)
    {
        Task1();
        Task2();
        ⋮
    }   Taskn();
}
```

② Embedded Operating System (OS) based approach:

There are two types of Operating systems. They are:

(a) General Purpose Operating System (GPOS)

[Ex:- Windows XP, Unix, Linux]

(b) Real time Operating System (RTOS)

[Ex:- Symbian, Elinux, Thread x, Vx Works etc.]

(a) General Purpose Operating System:

This is very similar to conventional personal computer based application development where the device contains an operating system and user applications will run on top of it.

Ex:- Windows, Unix, Linux etc.

Examples of Embedded products which use GPOS are Personal Digital Assistants (PDA's), Hand held devices/ portable devices and Point of sales (POS) terminals.

(b) Real time Operating System:

RTOS Contains real time Kernel responsible for performing pre-emptive multitasking, schedules for scheduling task, multiple threads etc.

RTOS allows flexible scheduling of system resources like the CPU and memory and offers some way to communicate between tasks.

Ex:- Windows CE, PSOS, Thread x, Embedded linux, Symbian Micro C/OS-II etc. Most of the mobile phones are built around the popular RTOS 'symbian'.

Embedded firmware development languages:

Firmware can be in two ways.

① Target Processor/ Controller specific language (low level (or) assembly level language) Ex:- ALP

② Target Processor/ Controller independent language (High level language) Ex:- C, C++, Java, Python etc.

① Assembly level language programming (low level):

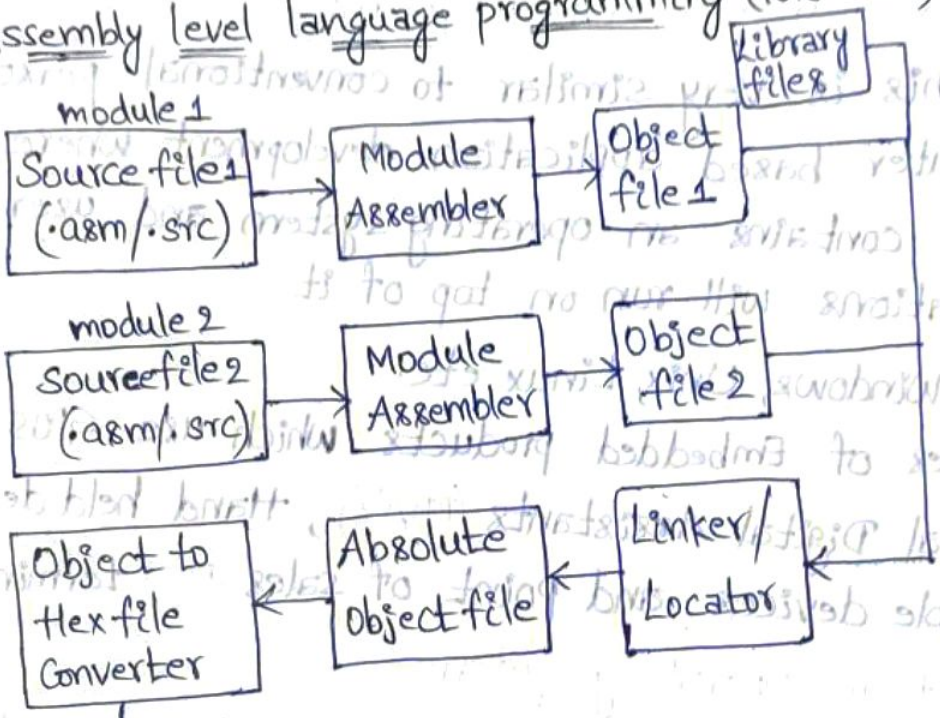


fig: Assembly language to Machine language Conversion Process.

* Assembly language is the human readable notation of "Machine language". They use specific symbols called "Mnemonics".
* A machine level language is processor understandable language. Processor deals with only 1's and 0's.
* Machine language and Assembly language are

processor/controller dependent. A program written for one processor/controller family will not work with others.

* General format for Assembly level language is Opcode followed by operands. Opcode specifies what to do by processor/controller and operands provide the data and information required to perform action specified by opcode.

Some of the opcode is implicitly contains the operand and in such case no operand is required.

Ex:- Consider 8051 ASM instruction `MOV A, #30`
(opcode operand)
This moves the value 30 into Accumulator 'A'. (single)
The same instruction written in Machine language is expressed as 01110100 00011110
(opcode operand)

Ex-2: Inc A : This increments accumulator by value '1'.
(opcode no operand)

Each line of assembly level language is split into 4 fields.

Label | Opcode | Operand | Comments
1 | 2 | 3 | 4

Label: (optional) It represents memory location, address of a program, subroutine, code portion etc. They can contain numbers from 0 to 9 and '_' (under-score). They are always sufixed by colon and begin with valid character.

Ex:- SUBROUTINE FOR GENERATING DELAY
DELAY PARAMETER PASSED THROUGH REGISTER R1
RETURN VALUE NONE
REGISTERS USED : R0, R1

```
DELAY: MOV R0,#255 ; load R0 with 255
       DJNZ R1, Delay ; Decrement R1 & loop till R1=0
       RET ; Return to calling program
```

Operation of assembly level language programming:

* The Assembly level language is written in Assembly code is saved as .asm (assembly file) (or) .src (source) file. Any text editor like wordpad (or), Notepad can provide an IDE (Integrated Development Environment) tool can be used for writing assembly code.
* Modular programming is employed when programming is too complex. Here the entire code is divided into submodules. Each module is made re-usable.
* Source file to object file translation: Translation of source code to assembler code is performed by assembler.
  Ex:- A51 Macro assembler from Keil Software is a popular assembler for 8051 Micro Controllers.
* On successful conversion of .src/.asm file, a corresponding (.obj) object file is generated. Object file is also called relocatable segment because absolute address of generated code is to be placed in program memory.

* Thus a linker/locator is required to place the code memory in the respective absolute address. Each module can share variables/functions among them. Exporting modules/subroutines is done by declaring them as PUBLIC, EXTRN (Extern)
* Library is the collection of pre-defined object modules with (lib) extension.
  Ex:- LIB51 from KEIL Software for A51 Assemblers/ C51 Compiler for 8051 Controller.

Linker/Locator: It assign absolute address to each module. Here all code and data reside in fixed memory locations. This is used for generation of .Hex file.
Ex:- BL51 from Keil Software is example of linker/locator for A51 Assembler/C51 Compiler for 8051 Specific Controller.

Object to Hex file Conversion: This is final stage of conversion of Assembly level language (Mnemonics) to Machine understandable language (Machine code).
Ex:- For Intel processor the hex file will be "Intel Hex".
For Motorola the hex file is "Motorola Hex".
Hex files are ASIC files that contain Hexadecimal representation of target application.
Ex:- OH51 is "absolute object file" to Hex file Converter for Keil Software.

Advantages:

① Efficient code and data memory utilization.
② Provides low level hardware access.
③ Easy for reverse engineering the code memory.

Drawbacks:

① It requires High development time.
② Developer dependency makes code complex and non-productive
③ Non-portable because of re-writing is required for different processors.

② High level language based development:

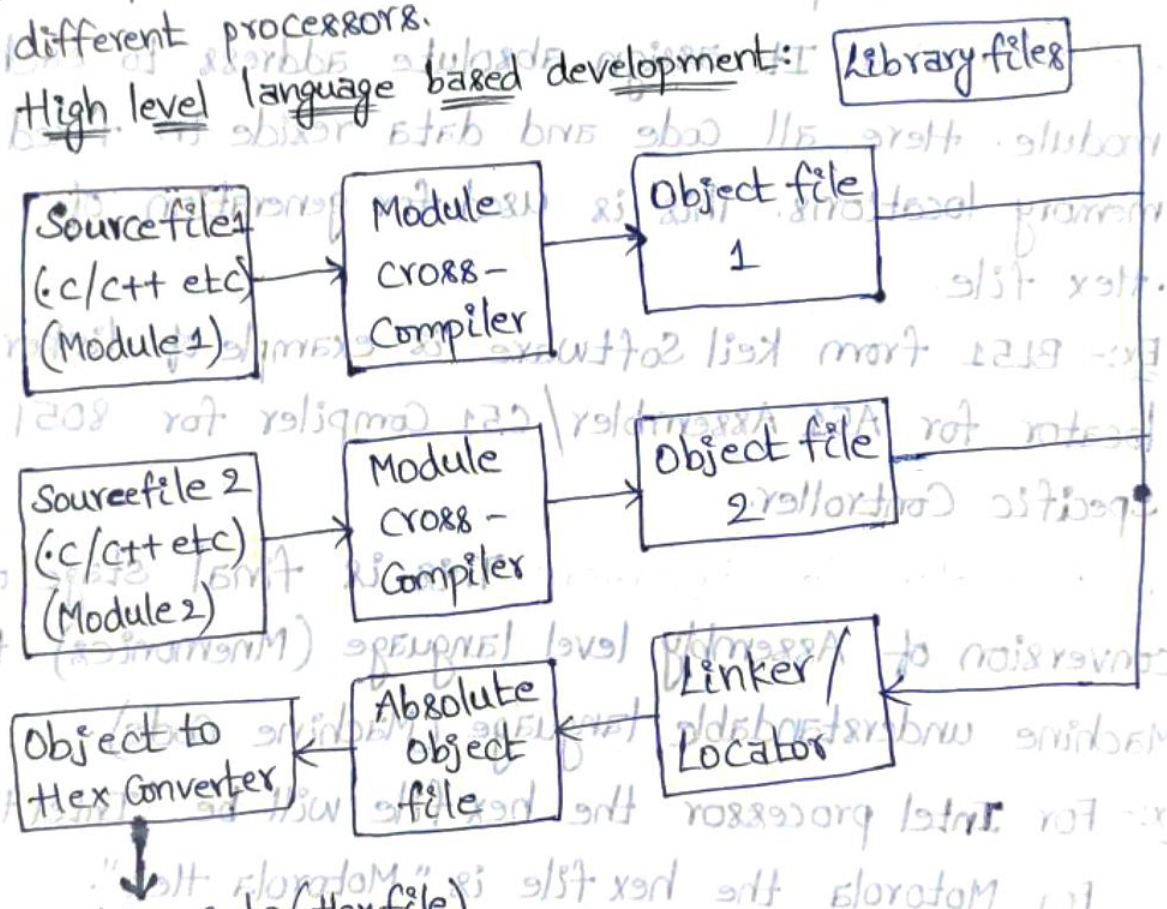


fig: High level language to Machine language Conversion process.

* Any high level language (C, C++, Java, .net, python, oracle, DBMS, R-programming etc.) with a support of cross-compiler (converts application developed in

high level language to target processor specific assembly code for target processor) can be used for embedded firmware development.

* Various steps involved in high level language based embedded systems firmware development is same as that of assembly level language based development except conversion process.
* The program written in any one of the high-level language is saved with corresponding language extension (.c, .cpp etc).
* These files are arranged into modules and given to cross-compiler. Each cross compiler is responsible for converting high level source code into the target processor machine code.

  Ex:- C51 is popular cross compiler available for 'c'-language for 8051 Micro Controller.
* Remaining steps involved are similar to Assembly level language conversion.

Advantages:

① Reduced development time due to less no. of lines of code.
② Developer Independency
③ Portability

Compiler Vs Cross-Compiler:

Compiler	Cross-Compiler
① It is a software tool that converts a source code written in high level language on top of a particular operating system running on a specific target processor architecture.	① These are the software tools used in cross-platform development applications. The Compiler running on particular processor converts source code to machine code of target processor whose architecture & instruction set is different.
② They generate machine code for same machine on which it is running.	② They generate machine code for different machine (processor) on which it is running.
③ They are native Compilers.	③ They are non-native Compilers.
④ Ex:- Intel X86, Pentium Processors etc.	④ Ex:- 8051 Controllers, PIC (Programmable Intelligent Computer) Controller, ARM (Advanced RISC Machine) etc.
⑤ It helps to convert the high level source code into machine level code.	⑤ It can create executable code for different machines other than the machine it runs on.
⑥ It is non-retargeting Compiler.	⑥ It is a retargetable Compiler.

C-language Vs Embedded-C language

C-language	Embedded-C language
① 'C' is a well structured, well defined standardized general purpose language.	① It is a subset of C-language which supports all C-language instructions and incorporates few target specific functions/ instructions.
② C-language is used for desktop computers.	② Embedded-C is for micro-controller based applications.
③ C-language programming uses resources like desktop, memory, OS etc.	③ It has to use limited resources like RAM, ROM, I/Os on Embedded processor
④ Extra features are not provided in C-language.	④ It includes extra features over C such as fixed point, multiple memory areas and I/O register mapping.
⑤ Compilers in C-language generates (ANSI C) OS dependent variables.	⑤ Embedded C requires to generate executable files for micro controllers/processors
⑥ It cannot be associated with real time embedded systems.	⑥ It is more convenient for real time embedded systems.

ISR Concept (Interrupt Service Routine):

There are two types of Interrupt sources.

① Software Interrupt sources
- * Error related sources
- * Instruction related sources

② Hardware Interrupt Sources
- * Internal hardware sources
- * External hardware sources

① Software Error related interrupts:

Each processor has specific instruction set. Any illegal code which doesnot corresponds to the set leads to illegal code interrupts. These are software related hardware interrupts.

Ex:- Division by zero detection (or) trap), overflow by hardware, underflow by hardware, illegal opcode by hardware etc.

② Software Instruction related interrupts:

A program can have computational errors (or) run-time conditions which generate Instruction related interrupts.

Ex:- Programmer defined exceptions, signals from device driver functions, square root of negative numbers etc.

③ Hardware Interrupt related internal devices:

These are processor/controller/internal device hardware specific.

Ex:- Timer overflow, ADC to start conversion, ADC to end Conversion, pulse accumulator overflow, TDRE (Transmit Data Receive Empty) etc.

④ Hardware Interrupt related to external devices:

External hardware interrupts providing vector address (or) Interrupt type through the data bus.

Ex:- NMI, INTO, INT in 8051 etc.

Interrupt Service Mechanism (ISM):

* On an Interrupt, processor vectors to new address

ISR – VECT ADDR

* It means the PC (Program Counter) which contain next

address saves it on stack memory (or) in a temporary register called link register. and processor loads ISR-VECTADDR into the PC.

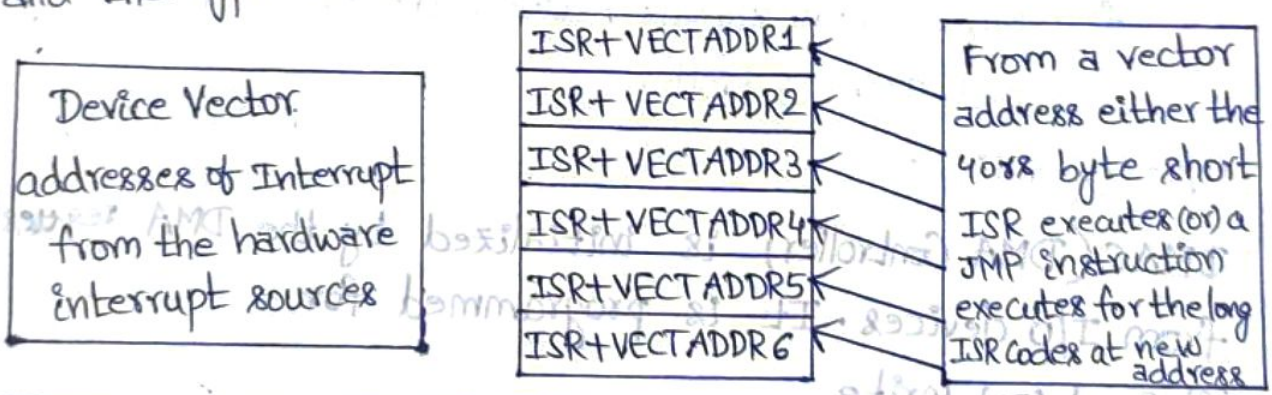
* The stack pointer register of CPU provides the saved address to enable return from the ISR using the stack.
* When PC saves at the link register, it is part of CPU register set. ISR last instruction is RETI.

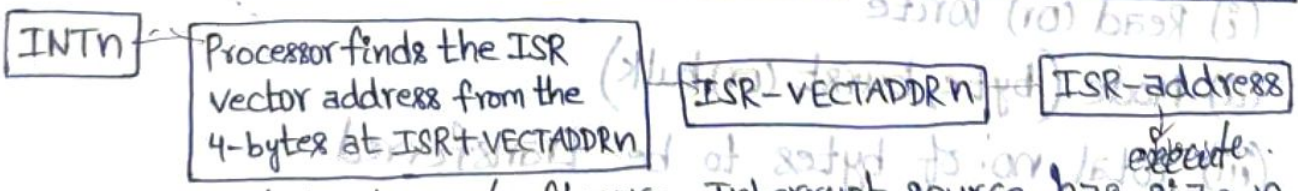
Processor Vector Address:

* A system has internal devices [ext on-chip timers, A/D converter etc.] which generate internal interrupts. Each internal device interrupt source (or) source group has a separate ISR-VECTADDR address each.

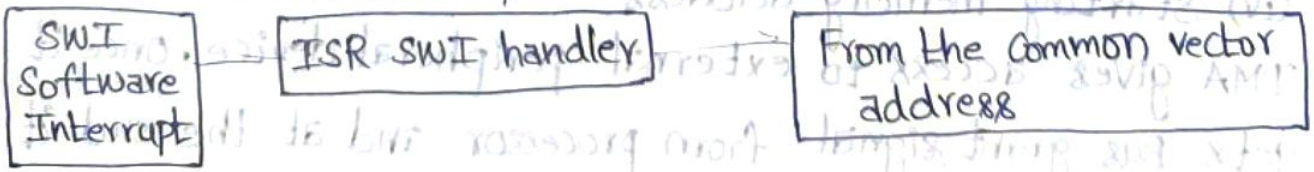
Each external interrupt pin has separate ISR+VECTADDR.

In 8086 Microprocessor architecture, a software instruction for example INTn explicitly also defines the type of interrupt and the type defines the ISR-VECTADDR.





Each hardware/software Interrupt source has size n



Direct Memory Access (DMA):

The transfer of data between fast storage device such as Magnetic disk and memory is often limited by the speed of CPU. Removing the CPU from the path and letting the peripheral devices managing the memory bus directly by using DMA Controller.

→ DMA Controller interface provides I/O transfer of data directly to and from the memory and I/O device.

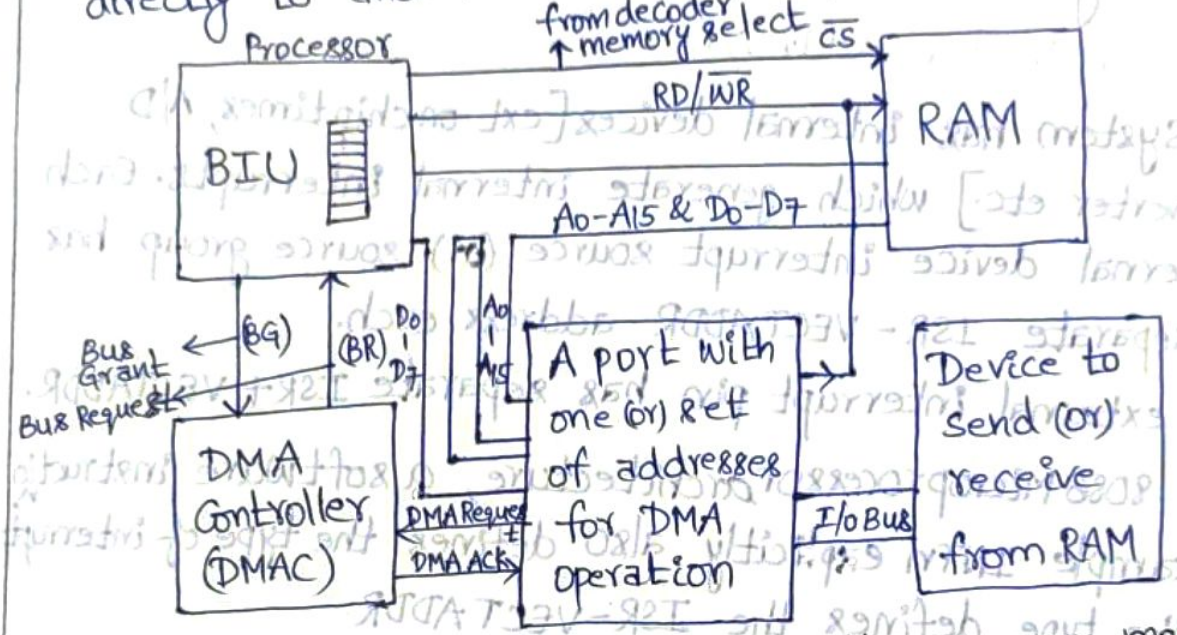


fig: Bus & Control signals between the processor, memory and DMA Controller

DMAC Operation:

→ DMAC (DMA Controller) is initialized by the DMA request from I/O devices. It is programmed for

(i) Read (or) Write

(ii) Mode (byte, burst (or) bulk)

(iii) Total no. of bytes to be transferred

(iv) Starting memory address.

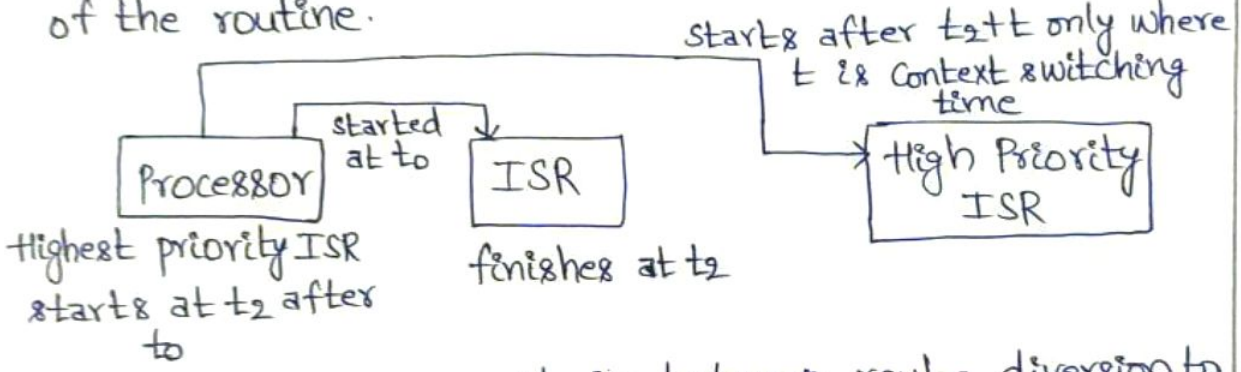
→ DMA gives access to external peripheral device. Once it gets Bus grant signal from processor and at the end it

communicates to processor that the task is completed.

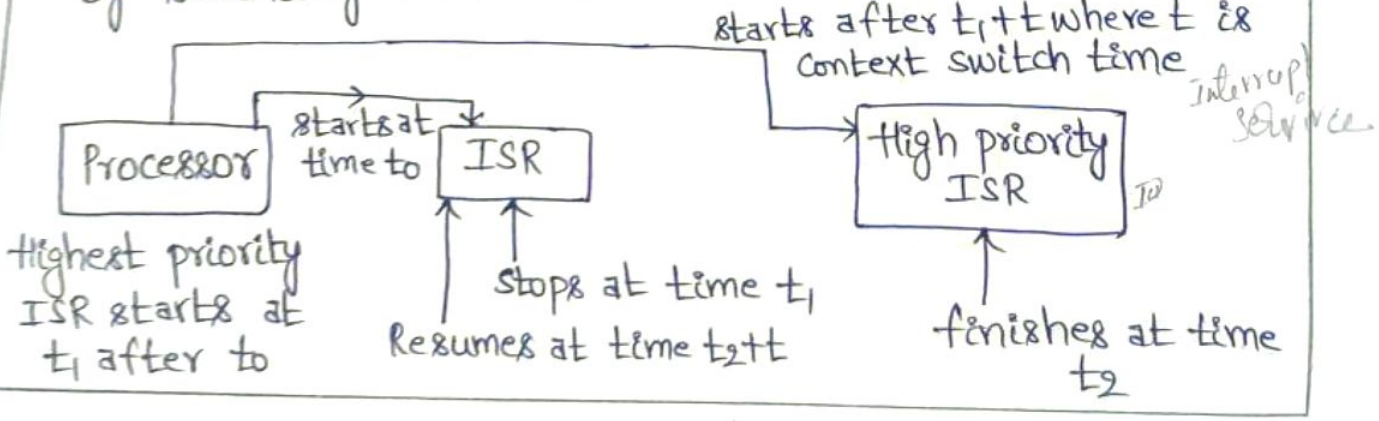
Multiple Interrupts:

When there are multiple interrupt sources, each interrupt is identified by the status register. When multiple interrupts are given then processor services the highest priority interrupt first after it's completion then it returns to lowest priority pending ISR (Interrupt Service Routine).

1. Certain processors do not provide for in-between routine diversion to higher priority interrupts and presume that all interrupts of priority greater than presently running routine are marked till the end of the routine.



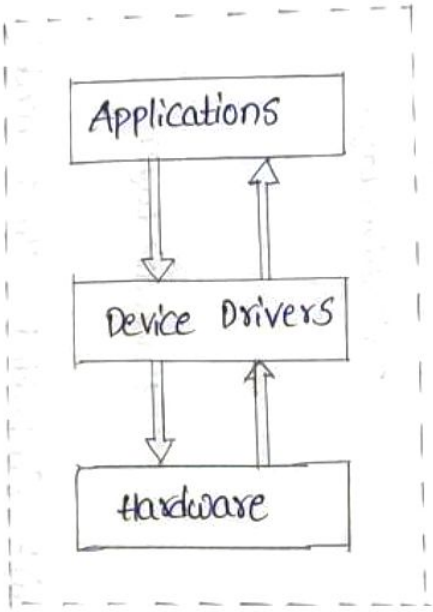
2. Certain processors permit in-between route diversion to highest priority interrupts. These processors provide provision for masking of all interrupts by a primary level bit. These processors also have selective diversion by a provision for masking the interrupt service selectively by secondary-level bits.



Device Drivers:

Most embedded hardware requires some type of Software initialization and management. The software that directly interfaces with and controls this hardware is called a 'Device drivers'.

∴ Device drivers are the software Libraries that initialize the hardware. It is proved between the application and Hardware.



* A driver is a software component that lets the operating system and a device communicate with each other.

* Almost every system operation eventually maps to a physical device, with the exception of the Processor, memory and a very few other entities,

any and all device control operations are performed by code that is specific to the device being addressed that code is called a device driver.

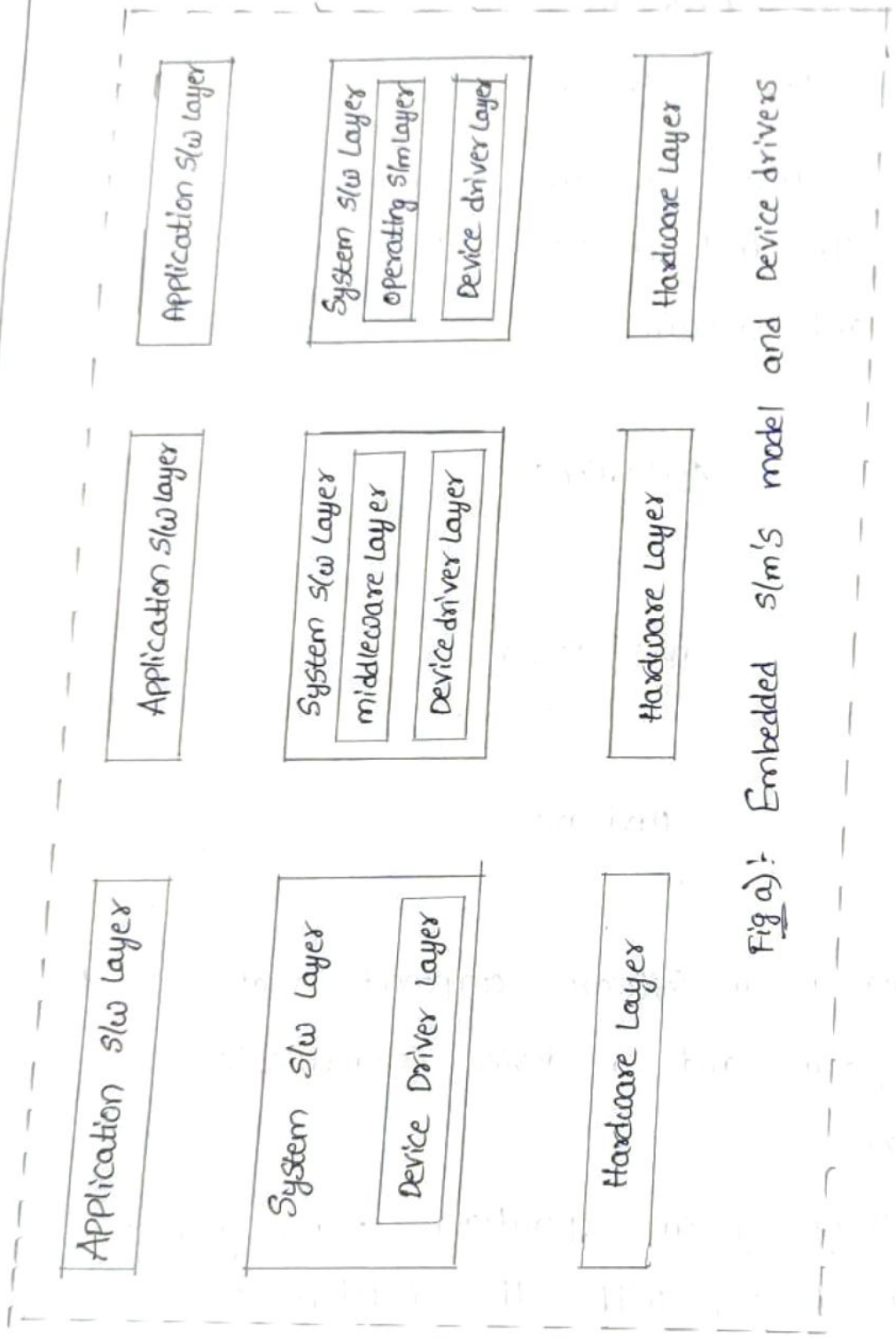


Fig a): Embedded S/m's model and Device drivers

* Device drivers are typically considered either 'architecture-specific' (or) 'generic'.

* Architecture-specific drivers that initialize and enable components with in a master processor include on-chip memory, integrated memory managers and floating Point hardware.

* Generic drivers manages hardware that is located on the board and not integrated on to the master processor.

* Generic driver can be configured to run on a variety of architectures that contain the related board hardware for which the driver is written.

* Regardless of the type of device driver (or) the hardware it manages, all device drivers are generally made up of all (or) some combination of the following functions.

* Hardware Setup
* Hardware Shutdown
* Hardware disable
* Hardware Enable
* Hardware Acquire
* Hardware Release
* Hardware Read
* Hardware write
* Hardware Install
* Hardware Uninstall

3. On-Board Bus device drivers:

* Bus start up → Initialization of the bus upon power-on (or) reset
* Bus shutdown → bus into its power off state
* Bus disable → Allowing other s/w to disable bus on the fly
* Bus Enable → " " " " enable " " " "
* Bus Acquire → locking to bus
* Bus release → unlock
* Bus read → Read data on bus
* Bus write → write data on bus
* Bus Install → Install new bus

4. Board Input-output Drivers:

* I/o startup
* I/o shutdown
* I/o Enable
* I/o Disable
* I/o Acquire
* I/o Release
* I/o Read
* I/o write
* I/o Install
* I/o uninstall

Real time Operating System

Real time Kernel:

The Kernel of real time operating System is referred to real time Kernel. The basic functions of real time Kernel are listed as:

1. Task / process management
2. Task / process Scheduling
3. Task / process Synchronization
4. Error / Exception handling
5. Memory management
6. Interrupt handling
7. I/o management

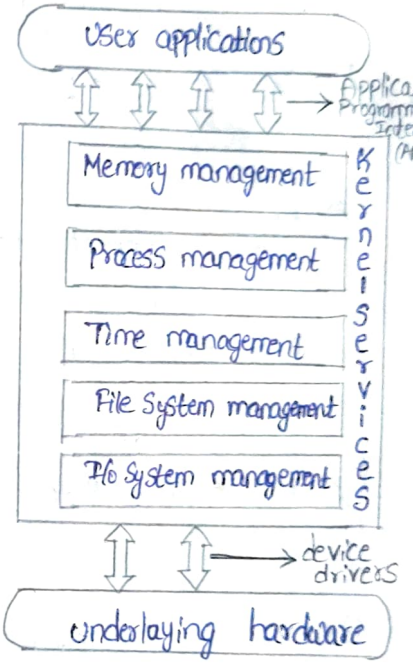


fig:- OS architecture

1. Task / process Management:

It deals with allocation of memory space for the particular task, loading task code into memory space allocating System resources, setting up of task control-Block (TCB) for the termination / deletion of particular task/process.

→ A task control block holds the information corresponding to a task. It consists of information such as

a) Task ID : Task identification number

b) Task type : Task can be hardware real time or Soft real time or baground task

c) Task state : The current state of the task (state = 'Ready') (for a task which is ready to execute)

d) Task priority : Highest priority task is executed (ex: priority=1)

e) Task pointers : Points to other Task control blocks for Proceeding, next and waiting tasks.

f) Task Context pointers : These pointers are used for context saving.

g) Task memory pointers : Points to code memory, data memory and stack memory

h) Task System Resource pointers : Points to System resources (Semophores, mutex, etc) used by the task.

2. Task/process scheduling :-

→ Sharing CPU among various tasks/processes.

→ A Kernel application called scheduler handles the task scheduling.

→ A Schedular is nothing but an algorithm implementation which performs the efficient and optimal scheduling of tasks to provide a deterministic behaviour.

3. Task/process Synchronization :-

It deals with synchronising the concurrent access of a resources, which is shared across multiple tasks and the communication between various tasks.

Error / Exception handling:

It deals with registering and handling the errors occured / exceptions raised during the execution of tasks. Insufficient memory, timeouts, deadlocks, deadline missing, bus error, divide by zero, un-known instruction execution etc. are examples of errors / exceptions.

→ Timeout is an example of a task level exception.

→ OS Kernel gives the information about the error in the form of System call (API).

→ watch dog timer is a mechanism for handling the timeouts for tasks.

5. Memory management:-

→ Instead of dynamic memory allocation which is present in general purpose operating system, RTOS makes use of 'block' based memory allocation technique.

→ RTOS makes use of blocks of fixed size & dynamic memory and the block is allocated for a task on a need bias.

→ A few RTOS implement virtual memory concept for memory allocation if the system supports secondary storage memory (like flash memory)

6. Interrupt handling:-

→ Interrupt inform the processor that an external device or an associated task requires immediate attention of the CPU.

→ Interrupts provide Real-time behaviour to systems.

→ An interrupt can be Synchronous (or) Asynchronous

→ Interrupts which occur in synch with currently running task is an Synchronous interrupts.

→ Software interrupts fall under Synchronous interrupts.

→ Divide by zero, memory segmentation error etc are examples of Synchronous interrupts.

→ Asynchronous interrupts occur at any point of execution of any task and are not Synchronization with current execution of task.

→ Interrupts generated by external devices connected to the processor/controller, timer over-flow interrupts, serial data reception/transmission interrupts etc. are examples of asynchronous interrupts.

7. Time management :-

→ Accurate time management is essential for providing Precise time reference for all applications.

→ The time reference to Kernel is provided by a high resolution real-time clock (RTC) hardware chip (hardware timer)

→ A hardware timer is programmed to interrupt the Processor controllers at a fixed rate.

→ Timer interrupt referred to "timer tick".

If the system register is 32 bit and the timer tick is 1μs the System timer will reset in

$$2^{32} * 10^{6} / (24 * 60 * 60) = 10^{6} \times 49700 \text{ days} = 1.19 \text{ hours}$$

If timer tick in 1msec, the system timer will reset in

$$2^{32} * 10^{-3} / (24 * 60 * 60) = 49.7 \text{days} \simeq 50 \text{days}$$

Task, Process and Threads :-

Task :- It is defined as the program in execution and the related information maintained by the operating system for the program.

Process :- It is a program or part of it in execution. It is also known as instance of a program in execution.

→ multiple instances of same program can execute simultaneously.

→ A process requires various resources like CPU for executing process, memory for storing the code, and associated variables, I/o devices for information exchange etc.

→ A process is sequential in execution

The Structure of process :-

→ A process performs concurrent execution of tasks and there by efficiently utilizing CPU and other system resources.

→ Concurrent execution is achieved by sharing CPU among the processes.

→ A process which inherits all the properties of CPU is called Virtual processor.

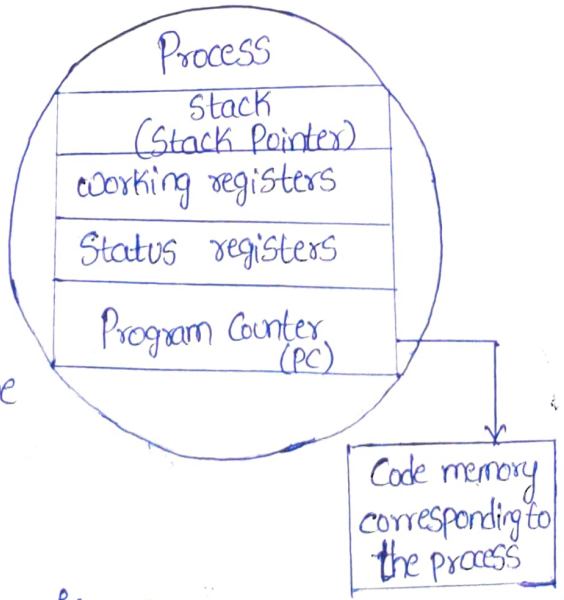


fig: Structure of process

The memory occupied by processor is divided into Stack memory, data memory and code memory.

→ Stack memory holds all temporary data such as variables local to the process.

→ Data memory holds all global data

→ code memory contains the program code (Instructions) corresponding to the process

→ on loading a process into the main memory a specific area of memory is allocated for the process.

→ Stack memory usually starts at the highest memory area allocated for process.

→ For example memory allocated area for the process is 2048 to 2100 the stack memory starts at address 2100 and grows down to accomidate the variables local to the process.

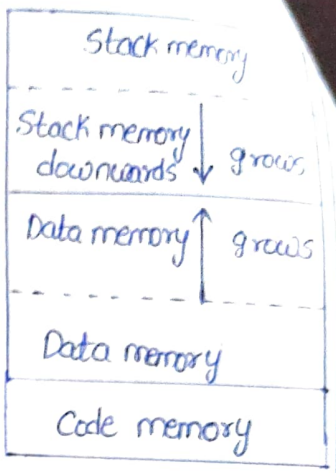


fig: Memory organization of a process

Process State and State transition:-

→ The creation of process to its termination is not a single step operation.

→ The cycle through which a process changes its state from 'newly created' to 'execution completed' is known as "Process life cycle".

→ At create state of process it begins, no resources are allocated to the process.

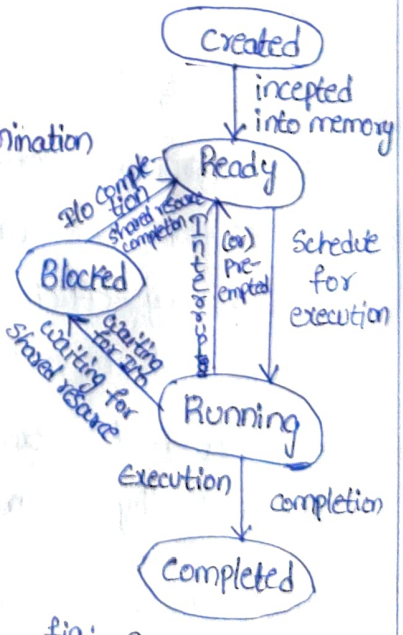


fig: Process States and State transitions representation.

→ The State where a process is incepted into the memory and awaiting Processor time for execution, is known as 'Ready state'.

→ In 'Running State' the source code instructions corresponds to process are executed.

→ The process execution happens in this stage.

→ 'Blocked stage/wait state' in this state the running process is temporarily suspended from execution and does not have immediate access to resources.

→ A State where the process completes its execution is known as 'Completed State'.

→ The transition of process from one state to another is known as 'State transition'.

Threads:-

→ A thread is a primitive that can execute code.

→ It is single sequential flow of control with in a process.

→ A thread is known as light-weight process.

→ Different threads which are part of process, share the same address space i.e, they share data & code memory area.

→ Threads maintain their own thread status.

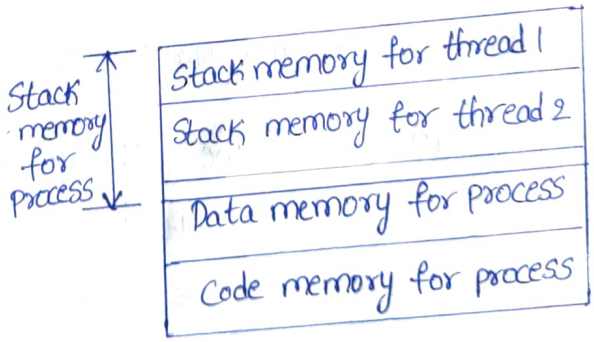


fig: memory organization of a process and its associated threads.

Thread	Process
1. It is single unit of execution and is part of process.	1. Process is a program in execution and contains one (or) more threads.
2. It does not have its own data (or) stack (or) code memory rather they share the memory with other threads of the same process	2. process has its own code, data and stack memory.
3. They are very inexpensive to create	3. They are expensive to create involves many os overheads.
4. Context Switching is fast.	4. context switching is complex and slow.
5. If a thread expires, its stack is reclaimed by the process	5. If a process dies, the resources allocated to it are reclaimed by os and all the associated threads also dies.

Multiprocessing and Multitasking :-

Multitasking :- " The Ability of operating System to hold multiple processes in memory and switch the processor (CPU) from executing one process to another process."

Multiprocessing :- " The ability to execute multiple processes simultaneously by the processor (CPU)".

Context Switching :- The process of switching CPU among the processors (or) changing current execution context is known as context switching.

Context Saving: The process of saving current context which contains details of currently running process at the time of cpu switching is known as context saving.

Context Retrieval: The process of retrieving the saved context details for a process, which is going to be executed due to cpu switching is called context retrieval.

Types of multitasking:

Depending upon type of switching tasks, they are different types:

1. Co-operative multitasking
2. Pre-emptive multitasking
3. Non-preemptive multitasking

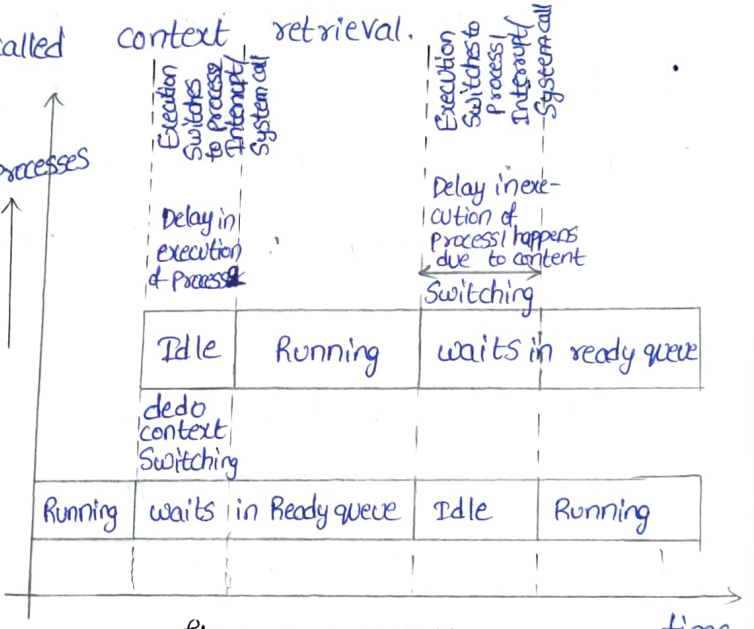


fig: Context Switching

1. Co-operative: A task/process gets a chance to execute only when the currently executing task/process voluntarily relinquishes the CPU. In this method, any task/process can hold the CPU as much time as it wants. If the currently executing task is non-cooperative, the other tasks may have to wait for long time to get the CPU.

2. Pre-emptive: It ensures that every task/process gets a chance to execute depending upon preemptive scheduling time a task/process is given time to execute. The current running task/process is preempted to give chance to other tasks/process.

Non-preemptive: The process/task, which is currently given the CPU time, is allowed to execute until it terminate (or) enters the 'blocked/wait' state, waiting for an I/O (or) System resource. It differs from co-operative multitasking such that the currently executing process/task need not relinquish the CPU when it enters the 'blocked/wait' state, waiting for an I/O, (or) Shared resource access where as in non-preemitive type currently executing tasks relinquishes the CPU when it waits for I/O or system resource (or) event to occur.

Task scheduling:

Task/Process scheduling determines which task/process is to be executed at given point of time. Scheduling policies determines which task is to be executed. The Kernel service/application, which implements the scheduling algorithm, is known as 'Scheduler'. Process scheduling decision is done during process switches its state to

1. Ready state from running state. //scenario 1//
2. Block/wait state from 'Running' state. //scenario 2//
3. Ready state from blocked/wait state // scenario 3//
4. Completed state //Scenario 4//

* A process switches to ready state from running state when it is pre-empted. Scenario 1 is preemptive.
* High priority process is in blocked/wait state completes

I/o and switches to Ready State. It is indicated by Scenario 3 which is Preemptive.

In pre-emptive/non-preemptive multitasking the process relinquishes the CPU when it enters the blocked/wait (or) completed state and switching of CPU happens at this stage. The schedule under scenario 2 can be either preemptive/non-preemptive. Scheduling under Scenario 4 can be Preemptive (or) non-preemptive (or) cooperative.

The selection of scheduling algorithm should consists of following factors

1. CPU utilization: CPU utilization must be high. It is a direct measure of schedule handling efficiency.

2. Through put:
   It indicates number of process executed per unit time. It should be high.

3. Turn around time: Time taken by a process to complete its execution. It includes process waiting time, time spent in ready request, queue time on completing the I/o operations etc. It should be minimum.

4. waiting time: Time spent on waiting queue to get CPU time for execution. It should be minimum.

5. Response time: Time alapsed between submission of process and the first response. Response time should be less.

* Operating System maintains various queues in connection with CPU scheduling. Process passes through these queues during course of its execution until its completion.

Job queue :-

It contains all the processes in the System.

Ready queue :-

Contains all the processes which are ready for execution and waiting for cpu to get their turn. It is empty when no process is ready for running.

Device queue :-

It contains set of processes, which are waiting for an I/o device.

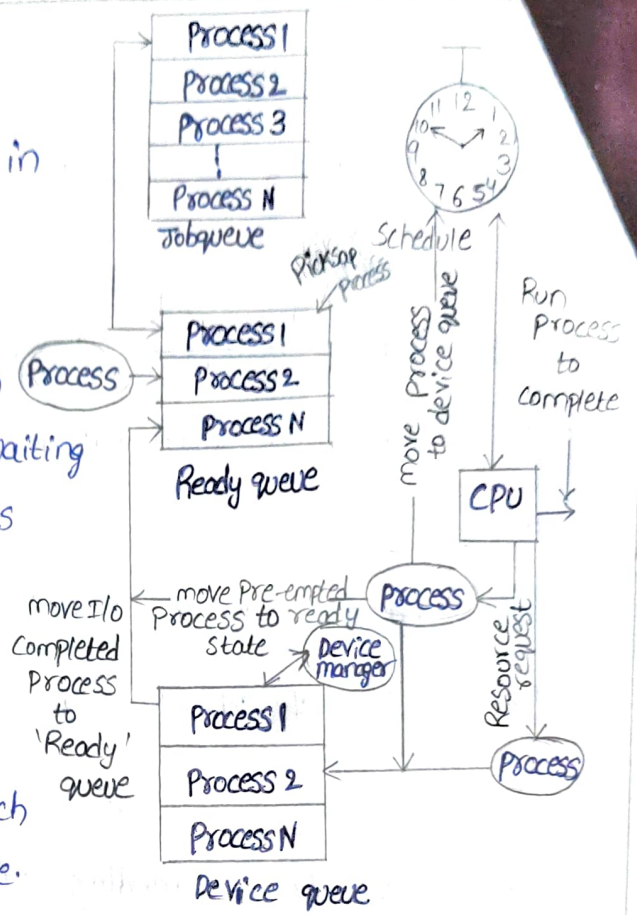


fig:- Illustration of process transition through various queues.

* Based on scheduling algorithm used, the scheduling can be classified into

1. Non-preemptive scheduling
2. Pre-emptive scheduling

1. Non-preemptive Scheduling :-

In this scheduling type, the currently executing task/process is allowed to run until it terminates or enters the wait state waiting for an I/o or system resource. Various types of non-preemptive schedulings are,

a) First come first served (FCFS) / FIFO Scheduling :-

As the name indicates, the first come first served scheduling algorithm allocates cpu time to the processes

based on the order in which they enter the ready queue.

Ex:- Ticket reservation.

Preemptive scheduling:-

Every task in the Ready queue gets a chance to execute. how each process gets a chance to execute (gets cpu time) is dependent on the preemptive scheduling for processes. The scheduler can preempt (temporarily stop) the currently executing task/process and select another task from ready queue for execution, after preempting the current task depends on scheduling algorithm. A task which is preempted is moved to ready queue.

Preemption:- The process of moving a running process/task into the ready queue by the scheduler, without the Processor requesting for it. preemption is implemented in two ways:

a) Time based preemption b) Priority-based preemption

a) Short job first (SJF) scheduling / shortest remaining time (SRT):-

It sorts the 'Ready' queue each time a process enters into wait state (or) execution completion of current process. It checks the execution time of the new process is shorter than the remaining processes in ready queue. It always compares the execution completion time, of new process entered in ready queue.

b) Priority based preemption Scheduling:

Priority based non-preemptive algorithm ensures that the process with high priority is served at earlier compared to other low priority processes in the ready queue.

* Priority based preemptive scheduling algorithm is same as non-preemptive priority based scheduling except for switching of execution between tasks.

* In pre-emptive any high priority process entering into ready queue is immediately served where as in the non-preemptive any high priority process entering ready queue has to wait until the execution of current running task and when the currently executing process voluntarily relinquishes cpu then the highest priority task gets executed.

Task Communication:-

In multitasking system multiple tasks/processes run concurrently (Pseudo Parallelism) and each process may or maynot interact in between. Based on interaction, the processes running on a os are divided as:-

1. Co-operating processes:- one process requires the inputs from other processes to complete the program execution.

2. Competing processes:- They do not share anything among themselves but they share system resources such as file, display device etc.

Co-operative processes exchanges information and communicate through the following methods :-

1. Co-operation through sharing :

The co-operative process exchange data through some shared resources.

2. Co-operation through communication :

No data is shared between the processes. But they communicate for synchronization.

The Mechanism through which task/processes communicate each other is known as Inter process/task communication (IPC). Various types of Inter process communication mechanisms are adopted by process are Kernel (OS) dependent. Some of them are :-

Shared memory :

The information to be communicated by the process is written in Shared memory area. Ex: Notice board.

fig: Concept of shared memory

Some of shared memory mechanisms are :-

1. Pipes : It is a section of the shared memory used by processes for communication.

Pipe management function calls :-

* create Pipe
* open Pipe
* close a Pipe
* Read from the Pipe
* Write to the Pipe

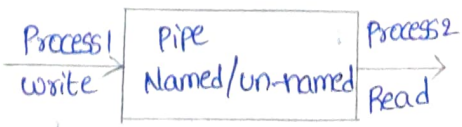


fig: Concept of pipe for IPC

Task Synchronization:

In multitasking environment multiple tasks run concurrently and share the System resources. Each process has its own boundary wall and they communicate with each other with different Inter process Communication (IPC) mechanisms including Shared memory and variables.

When two (or) more processes tries to access same I/o device (or) Shared memory area, where one processes tries to acquire data from it and another reads the data from it.

* The act of making processes aware of the access of Shared resources by each process to avoid conflicts is known as task / process Synchronization.

Task Communication / Synchronization Issues:

1. Racing:

When multiple processes compete (race) each other to access and manipulate shared data concurrently. In Racing condition final value of shared data depends on the process which acted on the data finally.

2. Dead lock:

A Racing condition produces incorrect results where as a dead lock condition creates a Situation where none of the Processes are able to make any progress in their execution, resulting in a set of dead locked processes.

It is a situation very similar to our traffic jam issues in a junction as shown in figure

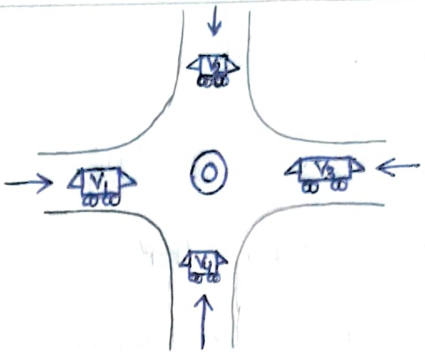


fig:- dead lock situation

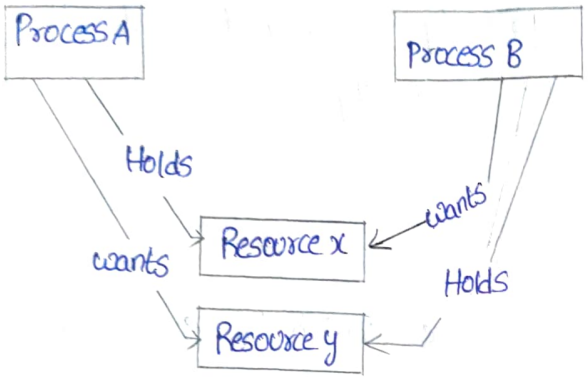


fig:- Scenario leading to dead lock

* Process A holds a resource x and it wants a resource 'y' held by process B.
* Process B holds resource y and it wants the resource 'x' which is currently held by process A.
* Both held the respective resources and they compete each other to get the resources held by respective processes.
* This results in 'Dead lock' situation.

Different conditions favouring a dead lock situation are listed below:-

mutual exclusion:-

The criteria that only one process can hold a resource at a time. Ex:- accessing display hardware in Embedded System device.

Hold and wait:-

The condition in which a process holds a shared resource by acquiring the lock controlling the shared access and waiting

for additional resources held by other processes.

No Resource preemption:

The criteria that operating system cannot take back a resource from a process which is currently holding it and the resource can only be released voluntarily by the process of holding it.

Circular wait:

* A process of waiting for a resource which is currently held by another process which in turn is waiting for a resource held by other process.

Ex: $P_0, P_1, \ldots P_n$ processes are present then if P_1 is waiting for resources held at $P_0, \ldots P_n$ is waiting for resources held at P_0 this forms a circular queue.

Dead lock handling:

There are different ways of handling deadlocks by OS (Operating System) are:

1. Ignore deadlocks:

The cost of removing or solving dead lock is high that it is always assumed that the system design is deadlock free.

Ex: UNIX

2. Detect and recover:

It detects deadlock situation and recovers from it. It is similar to deadlock situation near traffic signalling. The only solution is backup the vehicles and allow the vehicles in only one direction. If traffic is too high then number of

vehicles have to move back, thus this technique is also known as backup cars technique.

3. Avoid dead locks:

In this technique deadlock is avoided by careful resource allocation techniques by operating system. Similar to traffic lights at traffic signalling.

4. Prevent deadlocks:

It prevents dead locks by following certain conditions

1. A process must request all its required resources and they should be allocated before the process begins its execution.

2. Grant resource allocation from processes only if the process does not hold a resource currently.

By following these conditions introduces negative impact like low CPU utilization and starvation of processes.

Starvation: A process does not get its required resource for long time and it has to wait for accessing its resources.

Different types of starvations such as

1. The dining philosophers problem
2. Producer- consumer / Bounded buffer problem
3. Readers - writers problem
4. Priority Inversion

5. Priority Inheritance
6. Priority ceiling

1. Dining philosophers problem:

Let us consider 5 philosophers who want to have their dining and perform three tasks 1) eating 2) waiting 3) brainstorming. If each philosopher needs two forks for completing his dining but only 5 forks (n number of forks) are placed such that they are arranged as shown in figure a). There are various Situations such as

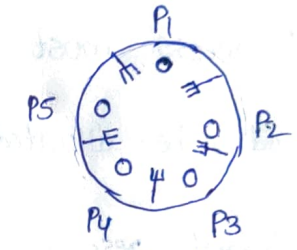
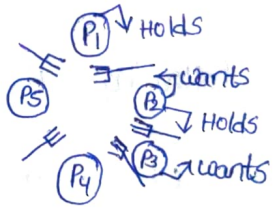


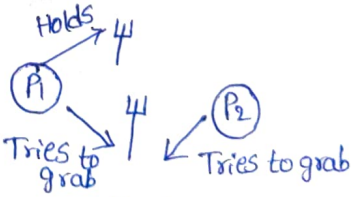
fig a)
Dining philosopher problem

Scenario 1: The starvation occurs when all the philosophers involve in brain storming together and try to eat together. To eat each one needs two forks all of them waits for second fork which leads to dead lock.



b) Starvation & deadlock

Scenario 2: When all of them are hungry and tries to pick up the fork Simultaneously during brain storming leads to Race condition.



c) Racing

Scenario 3: All the philosophers eat together and waits for remaining to put the force down for eating. Here all the philosophers tries to eat with one fork and due to blocking of another fork they cannot complete their dining.

this situation is called livelock. (Ex: two people moving towards each other to cross a Narrow bridge).

Hardware Software co-design:

* In traditional embedded systems hardware, Software partitioning is done before the development of Architecture.

There is less interaction between the two areas and the development happens either serially or in parallel form. Once the hardware Software is ready, integration is performed.

* Due to increased demand and competition and need for reduced 'time-to-market' hardware Software co-design came into existance.

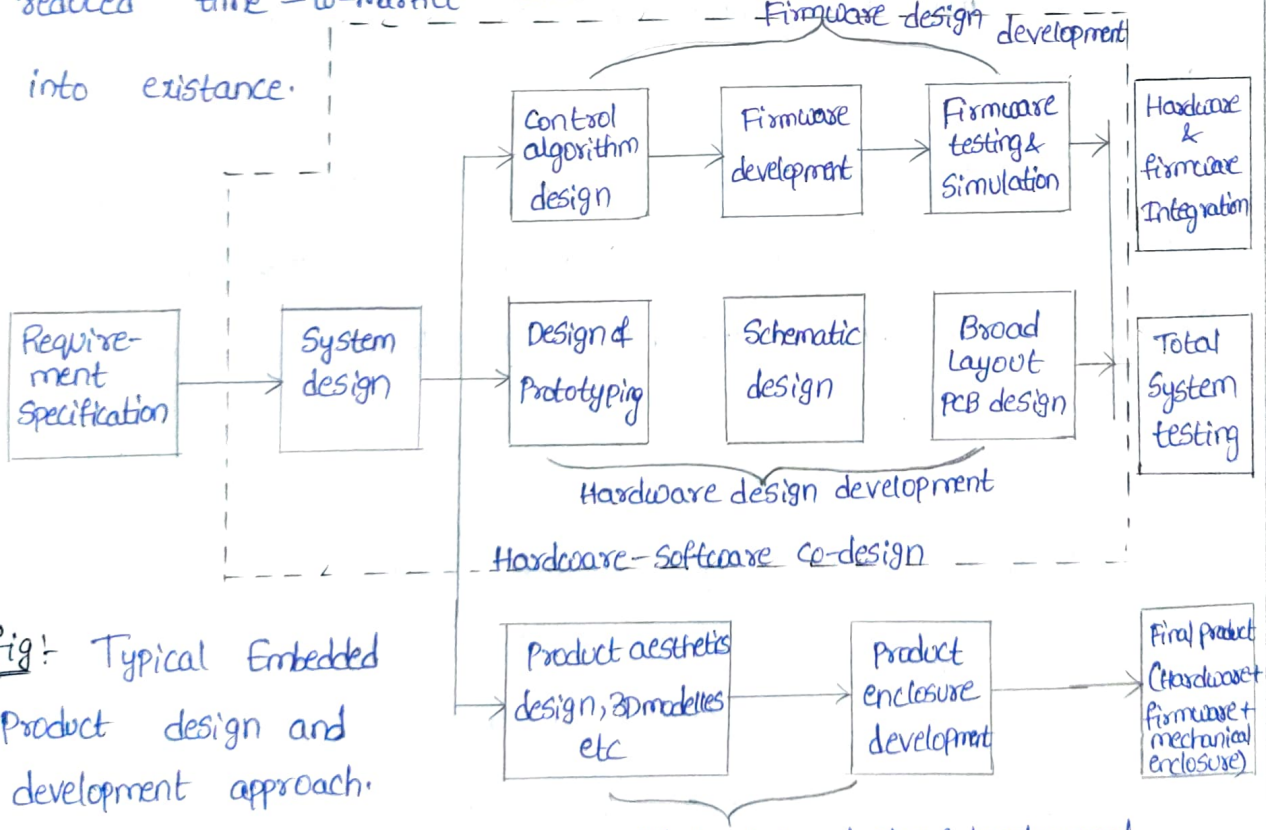


fig: Typical Embedded Product design and development approach.

* Instead of separate sections of implementation. Combining of hardware and Software improves the time to market. During this process it has some issues regarding design.

Fundamental Issues in hardware Software codesign:

1. Selecting the model:

A model is used to capture and describe the System characteristics. It consists of set of rules.

2. Selecting Architecture:

A model only captures the system characteristics and does not provide information on how a system can be manufactured. An architecture defines the different types of components and their interconnections. There are three types of architectures

1. General purpose architecture: Complex Instruction set computing (CISC) and Reduced Instruction set Computing (RISC). The datapath for CISC architecture is complex. RISC architecture supports extensive pipelining.
2. Controller architecture:

It implement finite state machine model using state register and two combinational circuits. The State register holds the Present State and combinational logic circuit holds the logic for the next state and output.

3. Datapath architecture:

This is best suited to implement data flow graph model where the output is generated as a result of set of predefined computations on the input data.

The data path represents a channel between the input and output. The ports in data path connect datapath to multiple buses.

4. Finite state machine datapath architecture (FSMD):

It combines controller architecture with datapath architecture. The controller generates control inputs and data path processes the data. The datapath consists of two types of I/o ports one for external control and another for receiving/sending the control signals from/to the controller unit. The external control I/o port connects the datapath to external world.

5. Very long Instruction world (VLIW) Architecture:

It implements multiple functional units (ALUs, multipliers etc) in the datapath.

6. Parallel Processing Architecture:

It implements multiple concurrent processing elements (PEs) and each processing element may associate a datapath containing register and local memory. Single Instruction multiple data (SIMD) and multiple Instruction multiple data (MIMD) are examples for parallel processing architectures. In SIMD a single instruction is executed in parallel with the help of processing elements. Here single controller is used. SIMD forms the bouse, reconfigurable processor.

MIMD for basic multiprocessor systems. The processing el in multiprocessing system communicates through mechanisms like shared memory and message passing.

3. Selecting the language:-

A programming language captures a computational model and maps it into architecture. A model can be captured by multiple programming languages like C, C++, C#, JAVA etc for software implementation. The languages like VHDL, Verilog, System c etc for hardware implementation.

Computational models in Embedded design:-

1. Data flow Graph model (DFG):-

It translates data processing requirements into data flow graph.

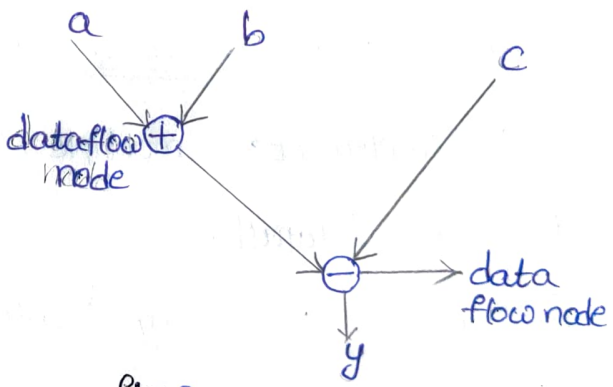


fig:- DFG model

* DFG model is data driven model. In data driven model the program execution is determined by data.
* The circle represents the process and the outward arrow from the process represents output data.
* In the given fig the functional requirements are

$x = a+b$ & $y = x-c$

* Feedback inputs in DFG graph represents cyclic inputs else DFG is a cyclic.

ntrol data flow Graph (CDFG):

A DFG graph with control operations (conditions) is called as control flow graphs (or) CDFG.

* If Flag = 1, x = a+b else y = a-b
* The requirement contains a decision making process.
* The control node is represented by diamond block. The decision of execution depends on control node.

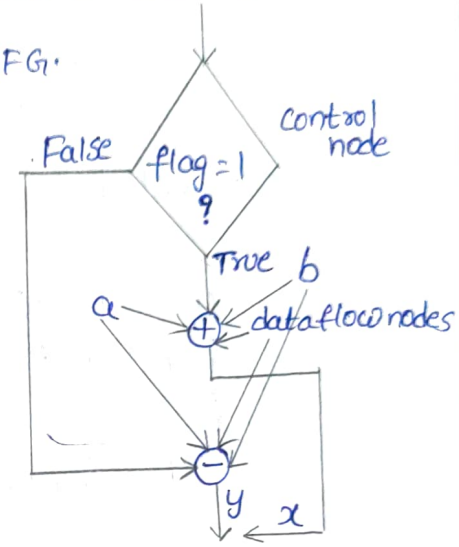
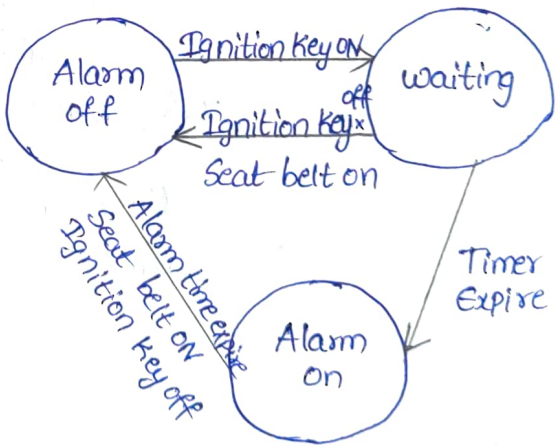


fig: control dataflow graph model

3. State Machine model:

It is an event driven embedded System whose processing behaviour depends upon the State transitions.

* Embedded Systems used in control and industrial applications are examples for event-driven Systems.
* In fig① the vehicle ignition is turned on and the seat belt is not fastened with in 10sec of ignition ON, the System generates on alarm signal of 5sec.
* The alarm is turned off when alarm time expires or seat belt ON.



fig①: FSM model for automatic Seat belt warning system.

Sequential programming model:

The processing requirements are executed Sequentially. The execution of FSM in Sequential program model for 'Seat belt warning' System is illustrated below

Program:

```
#define on 1
#define off 0
#define Yes 1
#define No 0
Void Seat_belt_warning()
{
    wait 10sec();
    if (check_Ignition_Key()==ON)
    {
      if check_Seat_belt()==OFF)
      {
        Set_timer(5);
        Start alarm();
        while ((check_seat_belt()==
                off)
        && (check_ignition_key()==
                off)
        && (timer_expire()==No));
      Stop_alarm();
    }
  }
}
```

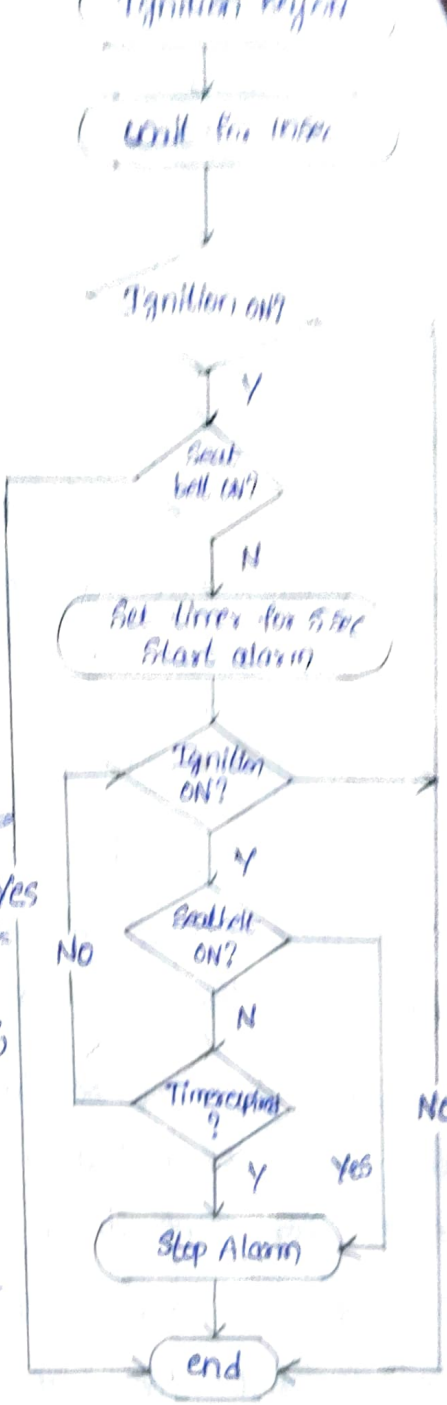


fig: Sequential program model for seat belt warning System.

* We can use a Sequential programming for execution & process else we can also use finite state machine model approach which uses flow chart representation.

Concurrent / communicating process model:

By using concurrent statements it executes tasks / processes. To utilize the processor effectively, but it requires additional overheads for task scheduling, synchronization and communication.

The seat belt model in concurrent process can be implemented as

1. Timer task waiting 10sec (wait timer task)
2. Task for checking the ignition key status (ignition key monitoring task)
3. Seat belt status task checking
4. Alarm control task
5. Alarm timer task

Object Oriented model:-

* It is an object oriented model which is an object based model for modelling system requirements. It brings re-usability, maintainability and productivity in system design. Each object is characterized by a set of unique behaviour and state. A class is considered as blueprint of an object. The concept of object and class brings abstraction, hiding and protection.

Hardware Software trade-offs:-

* The decision of choosing hardware (or) software is based on trade offs and actual system requirements.

Ex:- Embedded System requires multimedia codec requirement it can be required either software (or) hardware chip (ASIC(or)ASSP) Codec developed by hardware chip which is fixed it cannot be changed where as codec developed in software can be reconfigurable but the performance cannot be optimised.

* Memory size is another important as hardware software tradeoff.

* The important Softcoare – Hardcoare tradeoffs in Embe design are :-

1. Processing speed and performance
2. Frequency of change (Re-configurability)
3. memory size and gatecount
4. Reliability
5. Man hours and cost (effort)

In circuit Emulator (ICE) Based firmcoare debugging :-

* The term "Simulator" and "Emulator" are similar coords.
* Both of them are used for "debugging the target firmcoare".
* Simulator is a softcoare application that duplicates (mimics) the target CPU and Simulates various features and instructions supported by target CPU.
* The emulator hardcoare contains necessary emulation logic and it is linked to debugging application running on the development PC on one end and connects to target board through some interface on the other end.
* Pure Softcoare applications which perform the functioning of hardcoare emulator (such as set break points, set internal RAM/CPU registers, halt firmcoare execution etc) is also called as emulators.

Ex: 1) Operation of PDA phone for application development is an example of Software emulator.

2) A hardware emulator is controlled by a debugger application running on a PC.

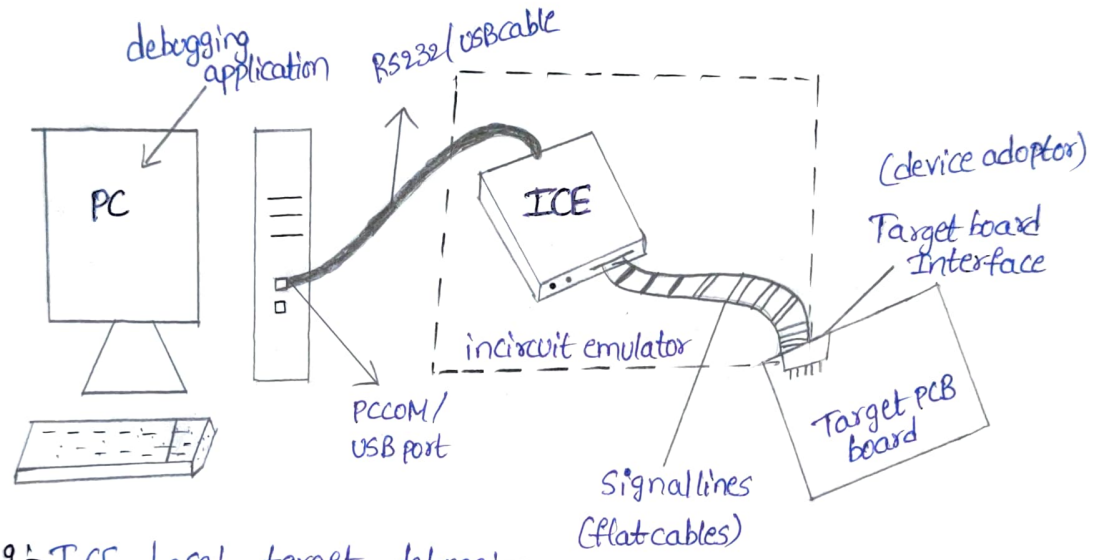


fig:- ICE based target debugging

Emulator device:-

* It is the replica of target CPU which receives various signals from target board through a device adopter, Connected to target board and perform the execution of firmware under the control of debug application.

* An emulator can be a standard chip or a PLD (Programmable logic device) configured to function target CPU.

UNIT-V

Embedded System Development

Integrated development Environment (IDE) :-

It is an integrated environment for developing and debugging the target processor specific embedded firmware. IDE is a software package which bundles a 1) Text editor (Source code editor)

2) cross-compiler (for class platform development and compiler for same platform)

3) Linker

4) Debugger

* An Integrated development Environment can be based on

1) Command line

2) GUI support

* The older version of TURBOC IDE for developing applications in c/c++ for x86 processor on windows platform is an example of generic IDE with command line interface.

* the Command line based IDE's may include little (or) less GUI support

* GUI based IDE provides a visual development environment with mouse click support for each action. Such IDE's are known as Visual IDE's

Ex:- Net Beans, Eclipse, Visual Studio etc---

* IDE's used in embedded firmware are either supplied by target processor / Controller manufacturers (or) third party vendors (or) as open source.

* Keil µ vision 3 from Keil Software is an Example of third party IDE.

* It should be noted that in embedded firmware development applications each IDE is designed for a specific family of Controllers / processors and it may not be possible to develop firmware for all family of Controllers / processor using a single IDE.

* Most of the processor / Control manufacturers and third party IDE providers are trying to build the IDE around the popular Eclipse open source IDE.

* This may lead to a single IDE based on Eclipse for embedded system development in the near future.

Types of files generated on cross-Compilation:-

* Cross Compilation is the process of converting a source code written in high level language like (embedded C) to a target processor / Controller understandable machine code (Ex:- ARM processor (or) 8051 micro controller specific machine code).
* The conversion of code is done by software running on a processor / Controller which is different from target processor.
* The software performing this operation is called cross-Compiler. cross-assembling is similar to cross Compilation the only difference is the code written in target processor/ Controller specific assembly code is converted into corresponding machine code.
* The various files generated during cross Compilation/ cross-assembling process are:

(i) List file (.lst)
(ii) pre-processor output file
(iii) Hex file (.hex)
(iv) Map file (File extension linker dependent)
(v) Object file (.obj)

(i) List file (.lst):-

* List file is generated during cross-compilation process and it contains an abundance of information about cross-compilation process, like cross compiler details, formatted source text ('c' code), symbol tables, errors and warning detected during the cross-compilation process
* List file has following headers.

(a) page header :- Compiler version number, source file name, date, time and page number are indicated on page header.

Ex:- C51 COMPILER V8.02 SAMPLE 05/23/2006 11:29:58 PAGE 1

(b) Command line :- Represents the entire command line that was used for starting compiler.

Ex:- C51 COMPILER V8.02, COMPILATION OF MODULE SAMPLE

OBJECT MODULE PLACED IN SAMPLE.OBJ

COMPILER INVOKED by : C:\Keil\C51\BIN\C51.EXE Sample.C

BROWSE DEBUG OBJECT Extend

Code LIST INCLUDES SYMBOLS.

(c) Source Code:- The source code listing the line numbers as well as source code on the line. It includes comments of source file.

(d) Assembly listing:- It contains assembly code generated by the cross compiler, for the 'c' code. By using compiler directives assembly code can be generated.

(e) Symbol listing:- It contains various symbols present in cross compiler.

(f) warnings and errors:- warning and Errors section of list file records the errors encountered (or) any statement that may create issues in application during cross compilation.

(ii) preprocessor output file:- It generates during cross-compilation and contains the pre-processor output for source files. pre processor output file is used for verification of operation of Macros and conditional pre-processor directives.

* pre-processor o/p file is a valid c source file. File extension of pre-processor output file is cross compiler dependent.

(iii) object file (.OBJ file):-

* It is a specially formated file with data records for symbolic information, object code, debugging information, library references etc---. some of details stored in .OBJ file are;

* Reserved memory for global variables
* public symbol names.
* External symbol references
* Library files with which to link.
* Debugging information to help synchronize source lines with object code.

⇒ The code present in OBJ file is not fixed code, it is the responsibility of linker/locator to assign an absolute memory location to the object code.

(iv) Map file (.MAP):- In a project with multiple source files, the cross compilation of each module generates its own list file.

* Linking and locating of re-locatable object files will also generate a list file called "linker list" file (or) "Map file". different sections listed on map file are "Compiler dependent". A map file contains

(a) page header :- A header on each page of the linker listing (MAP) file which indicates the linker version number, date, time and page number.

Ex:- BL51 Linker/Locater V3.62 02/09/2004 9:59:51 page

(b) Command line :- Command that invokes linker.

(c) CPU details :- It Contains details about target CPU and memory model.

(d) Input modules :- Names of all modules, and library files are present

(e) Memory map :- It lists starting address, length, relocation type and name of each segment in the program.

(f) Symbol table :- It Contains the value, type and name for all symbols from the different input modules.

(g) Inter module cross reference :- the section name, memory type and name of module which is defined by it.

(h) program size :- The size of various memory areas as well as Constants and Code space for the entire application.

(I) Warning and Errors :- these are generated while linking a program debugging link errors are present in this type.

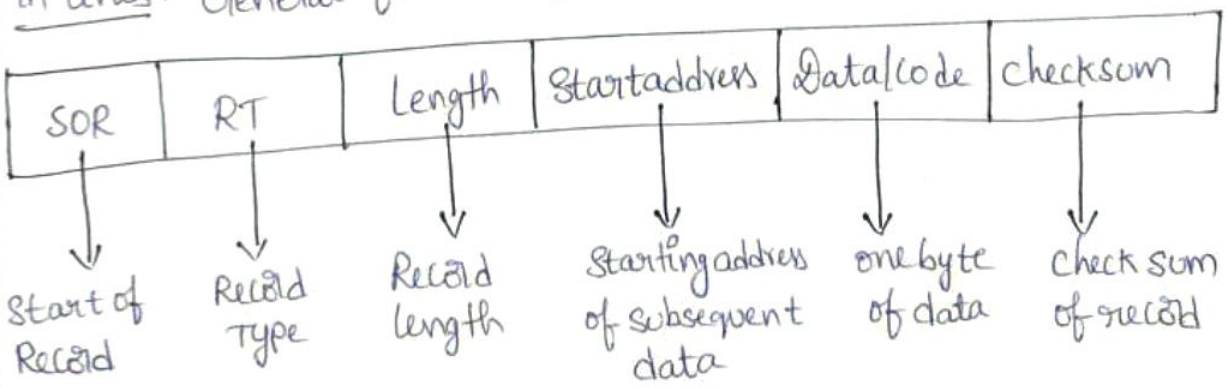
Hex file (.Hex) :-

* Hex file is the binary executable file created from the source code.
* The absolute object file created by the linker / locates is converted into processor understandable binary code.
* The utility used for converting an object file to a hex file is known as object to Hex file converter.
* Hex file format source code in a particular format depending upon the processors / controllers.
* Intel Hex and motorola Hex are two commonly used Hex file formats in embedded applications.
* Intel Hex file is ASCII text file interms of records in hexadecimal numbers that represent machine level language code.

⇒ Hex file for Intel : || aaaattdd ... cc

field	description
:	Start of every hex record
ll:	Record length field no. of data bytes (dd)
aaaa :	Starting address of subsequent data in record
tt	00 Data record 01 End of file record 02 8086 segment address record 04 Extended linear address record
dd :	Data field represents 1 byte of data.
cc :	Check sum field representing check sum of.

Motorola hex file format :- It is also Hex file which represent ASCII Text file where it represent ascii format in lines. General format of motorola Hex file is :



De-assembler / De-compiler :- It is utility program which converts machine codes into target processor specific assembly codes/ instructions.

* The process of converting into assembly coding is called as "De-assembling".

* De-compiling is converting machine codes into corresponding to high level language instructions.

* By performing reverse engineering virus in the code can be easily detected. They can be available free software but they cannot be like original code interms of symbolic contents and comments used. However they generate source code matching to the original source code from which the binary code is generated.

Simulators :- the targeted hardware is simulated for checking firmware execution. the features of Simulators based debugging are :

1. purely software based
2. Does not require a real target system.
3. primitive type (lack of featured Ilo port support)
4. Lack of Real-time behaviour

Advantages:-

① No need of original target board:- IDE's software support simulates the cpu of the target board. user need to know about the memory mapping and device interfacing. since no hardware required. It saves developing time of RTOS.

② Simulate I/o peripherals:- It provides option for various I/o peripherals. we can edit the values for I/o registers according to firmware execution. Hence it ~~erased~~ eliminates connecting I/o devices for debugging the firmware.

③ Simulates abnormal conditions:- we can observe the control flow of firmware by giving various range of i/p values which may lead to abnormal conditions of operation of embedded system and helps to study the behaviour of firmware under abnormal conditions.

Limitations:-

(i) Deviation from real behaviour:- All possible combination of input cannot be debugged virtually which leads

to limitations such that we can't be relay on the simulator based firmware debugging the product may not run in the same way & under production environment

Lack of real timeliness:- Debugging firmware is developer driven in which I/o varying can't be predicted.

* Debugging:- It is the process of checking the signals from various buses of embedded hardware and monitoring target processor memory and registers. It is needed to figure out bug (or) the error

* Debugging can be done in two ways. They are firmware debugging and hardware debugging. In early days of embedded system development, there are no debug tools available and the only way was * "Burn the code in a EEPROM and pray for its proper functing." If firmware doesnot crash the product works fine.

* Due to advancement in science and technology there are vast techniques available in the market

for debugging the firmware. Some of the sophesticated techniques of debugging are.

(i) Incremental EEPROM Burning technique:- In this type of technique to code is seperated into different functional code units the code is burned in an incremental order instead of whole, where the code corresponds to all functionalities are seperated and coded for cross compiling and burning into chip one-by-one. The LED/BUZZER is included for indication of executed code. After burning the code one-by-one if the functionality of the hardware is proper then combine the different codes into single functional unit recompile and burn the system code for total system functioning. It is time consuming process.

(ii) In-line break point type of debugging:- It is also primitive method of firmware debugging with in the firmware where we want to secure the firm-ware execution is reached upto a specified point, insert an inline debug code immediately after the point. The debug code is a printf() function which prints a string given as per the firmware.

(F)

Ex: // first Inline debug code

```
printf ("Start Configuration \n");
Configurations - - - - - -
- - - - - - - - - - - - -
```

// Inline debug code ensuring execution of Configuration Section.

```
printf ("End of Configuration \n");
Code segment 1 - - - -
- - - - - - - - - - - -
```

// Inline debug code ensuring execution of Code segment 1

```
printf ("End of Code Segment \n");
```

* If the firmware is error free and execution occurs properly we can get all the debug messages on "Hyper terminal"

* Based on debug information we can check errors.

3) Monitor program based firmware debugging :-

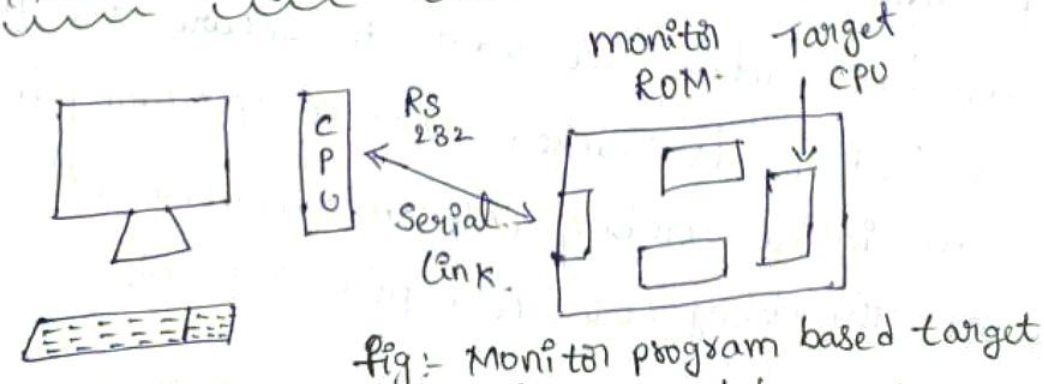


fig :- Monitor program based target firmware debug set up

* It controls the downloading of user code into code memory, inspects and modifies register/memory locations allows single stepping of source code.
* The main set of features of monitor program are:

1. Command set interface to establish communication with debugging application.
2. Firmware download option to code memory.
3. Examine and modify processor registers and working memory (RAM)

In-Circuit Emulator (ICE) based firmware debugging:-

* Emulator is a self contained hardware device which emulates the target CPU. Emulator hardware contains necessary emulation logic and it is hoocked to the debugging application running on the development pc on one end and connects to the target board through some interface and the other end. Now-a-days pure software application can perform the functioning of hardware emulator Ex:- PDA phone.

Target Hardware debugging :- Even though the firmware is bug free, the embedded product need not function as per expected, for various hardware related reasons like dry soldering of components, miss connections in PCB, misplaced components, signal corruption due to noise etc. Thus hardware debugging involves monitoring of various signals of target board (address / data lines, port pins etc), checking interconnection among various components, circuit continuity checking etc. The various hardware debugging tools used are:

(i) Magnifying Glass (Lens):- Magnifying glass (lens) is used to examine the surface of the target board for examining thoroughly for dry soldering of components missing components, short of tracks etc. It is very good tool for produces in interference noise in power supply line and other.

(ii) Multimeter :- It is used to measure various electrical quantities like voltage (both AC and DC), current (DC & AC), Resistance, Capacitance, Continuity checking, transistor checking, Cathode anode identification, diode etc--

(iii) Digital CRO :- CRO is a little more sophisticated tool compared to multimeter CRO is used for waveform capturing and analysis, measurement of signal strength.

(iv) logic analyser :- It is used for capturing digital data (logic 1 and 0) from a digital circuitary where as CRO is employed in capturing all kinds of waves including logical signals. Another major limitation of CRO is total number of logic signals / waveforms that can be captured with a CRO is limited to number of channels. A logic analyser gives an exact reflection of what happens when a particular line of firmware is running. This is acheived by capturing the address line logic and data line logic of target hardware.

(V) Function Generator :- It is input signal simulator tool. It is capable of producing various periodic waveforms like sine wave, square wave, saw-toothe wave etc.. with different frequencies and amplitude. In a debugging environment the function generator serves the purpose of generating and supplying required signals.

Boundary scan :- Boundary Scan is a technique used for testing the interconnection among the various chips, which support JTAG interference present in the board. JTAG port which contains the five signal line namely. TDI, TDO, TCK, TRST and TMS form the test access port (TAP) for a JTAG supported chip. Each device will have its own TAP. PCB also contains a TAP for connecting the JTAG supported chiplines to external world.

⇒ the TDI pin of the TAP of PCB is connected to the TDI pin of the first device. The TDO pin of the first device is connected to the TDI pin of second device. In this way all the pins are connected and the TDO pin of last JTAG device is connected to the TDO pin of the TAP of the PCB. The clock line TCK and Test mode select (TMS) line of the devices are connected to the clock line and test mode select line of text access port of the PCB respectively. This forms a boundary scan path.

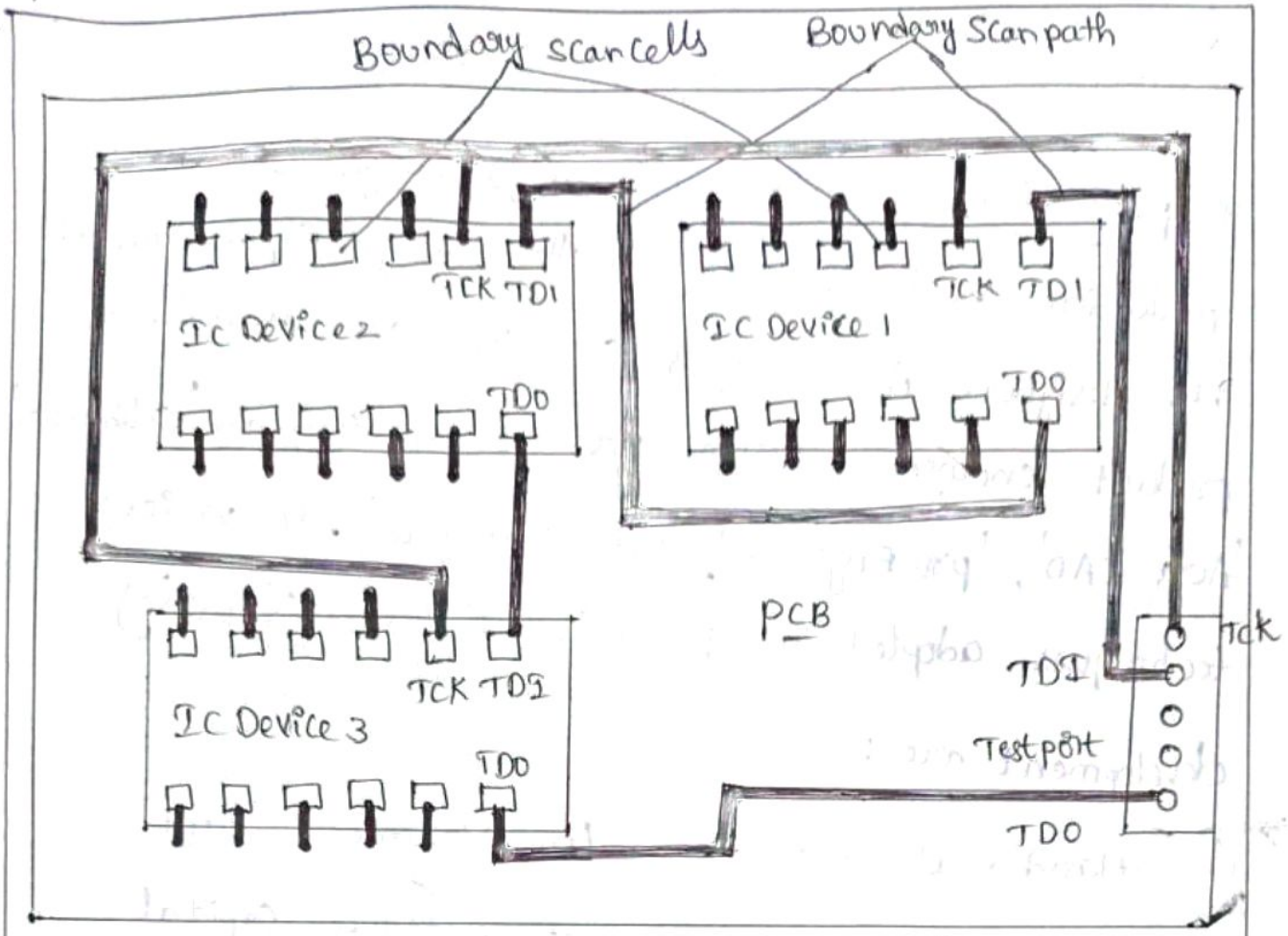


fig:- JTAG based boundary scanning for hardware testing

* The boundary scan cell is multipurpose memory cell the boundary scan cells associated with the input pins of an IC is known as input cells and cells associated with output pins of IC are known as output cells.
* for instructions for different types of boundary scan tests where as Runbit is used for performing a self test on the internal functionality of the chip. It produces pass/fail result.

- Embedded Software development process and tools:-

* production enclosure (housing) design is an art production enclosure reflects the design capabilities of its designer. A variety of CAD tools are available for product enclosure design. Ex:- of CAD tools are 'solid works', 'Auto CAD', 'pro Engineer', 'catia Interface' etc. various techniques adopted in production enclosure (housing) development are:

(i) Hand made enclosures:- low volume product. requirements use this type of prototype. Capital investment required is low commonly used materials hand made enclosures are poly vinyl chloride (PVC) and Acrylic. It take large development time.

(ii) Rapid prototype Development:- It is 3D-computer Assisted manufacturing (3D CAM) technique used. The 3D model designed using any of the 3D CAD tools is used as input for this. Rapid prototyping is also known as generative manufacturing (or) solid free from fabrication (or) layered manufacturing various techniques adopted in rapid prototyping are:

(i) Stereo lithography (SLA) prototyping :- Liquid plastic is solidified in precise patterns by a UV laser beam, resulting in a solid epoxy realisation of a 3D design of the enclosure.

(ii) Selective laser sinter (SLS) prototyping :- SLS is similar to SLA but the 3D prototypes are created from the drawing by fusing (or) sintering powdered thermo plastic materials layer by layer.

(iii) Fused deposition modelling :- (FDM) It is solid based rapid prototyping method. It makes use of melted thermoplastic polymer in the form of paste for building the layers.

Tooling and Moulding :- Tooling and moulding techniques are mainly used for developing plastic and metal housing for high volume production. The initial investment for making a tool (or) mould is too high and hence it is profitable only for high volume productions. The widely adopted tooling and moulding techniques are :-

(i) Room temperature Vulcanised (RTV) Tooling :- RTV technique is used for polyurethane housing.

(ii) Computer Numeric Control (CNC) Tooling:- CNC machines are used for milling the block of plastic according to the input from a design file generated by the 3D modelling tool to develop desired housing.

(iii) Injection moulding:- Injection moulding system consists of a moulding machine and moulding plates (dummy shape of the product to be developed) plastic/resin is fed into injection barrel of the machine through the chopper and in the barrel it is heated to the appropriate melting temperature.

(iv) Sheet metal work:- Sheet metal is the foil form of metal sheet, which can be cut (or) bent into different forms. Copper, brass, mild steel, Aluminium, tin, Nickel, titanium etc. are examples for metals used in sheet metal work. It is suited for products demanding high ruggedness. Sheet metal is not a good choice for products including any RF circuitary. Fancy works like contours, curves are not possible to build on housing using sheet metal work.

Commercial off-the-shelf Enclosures (COE) :- These are available in rectangular, square and circular shapes and can be metal (or) plastic. Ex:- multimeter housing, plastic boxes etc.

* If the product is important only in terms of functionalities are used in COE.
* COE is best suited for concept development projects and product development work with low budget.

UNIT-VI

Embedded system Implementation and testing

The main software utility TOOL :-

* Source code is typically written with a tool such as a Standard ASCII TExt editor (or) an integrated development environment (IDE) located on the host platform.
* An IDE is collection of tools including an ASCII Text Editor, Integrated into one application user Interface.
* while any ASCII Text Editor can be used to write any type of code. Independent of Language and platform.
* An IDE is specific to the platform and is typically provided by the IDE's vendor, a hardware manufacturer, OS vendor (or) Language vendor (java, c... etc).

Computer-Aided Design (CAD) and the Hardware :-

* Computer-Aided Design tools are commonly used by hardware engineers to simulate circuits at the electrical level in order to study a circuits behaviour under various conditions before they actually build the circuit.

* The circuit: Simulation software is a variation of another circuit simulator is called "SPICE" (Simulation program with integrated circuit emphasis).
* pspice is the pc version of SPICE and is an example of a simulator that can do several types of circuit analysis such as nonlinear transient, non-linear dc, linear ac --- etc.
* circuits created in this simulator can be made up of a variety of active and/or passive elements.
* Because of importance of and costs associated with designing hardware, there are many industry techniques in which CAD tools are utilized to simulate a circuit.
* A complex set of circuits in a processor (or) on a board, it is very difficult, if not impossible, to perform a simulation on the whole design, so a heirarchy of simulators and models are typically uses.
* A behavioural model of the entire circuit is created for both analog and digital circuits and is used to study the behaviour of the entire circuit. There are simulating techniques for handling complex circuits.

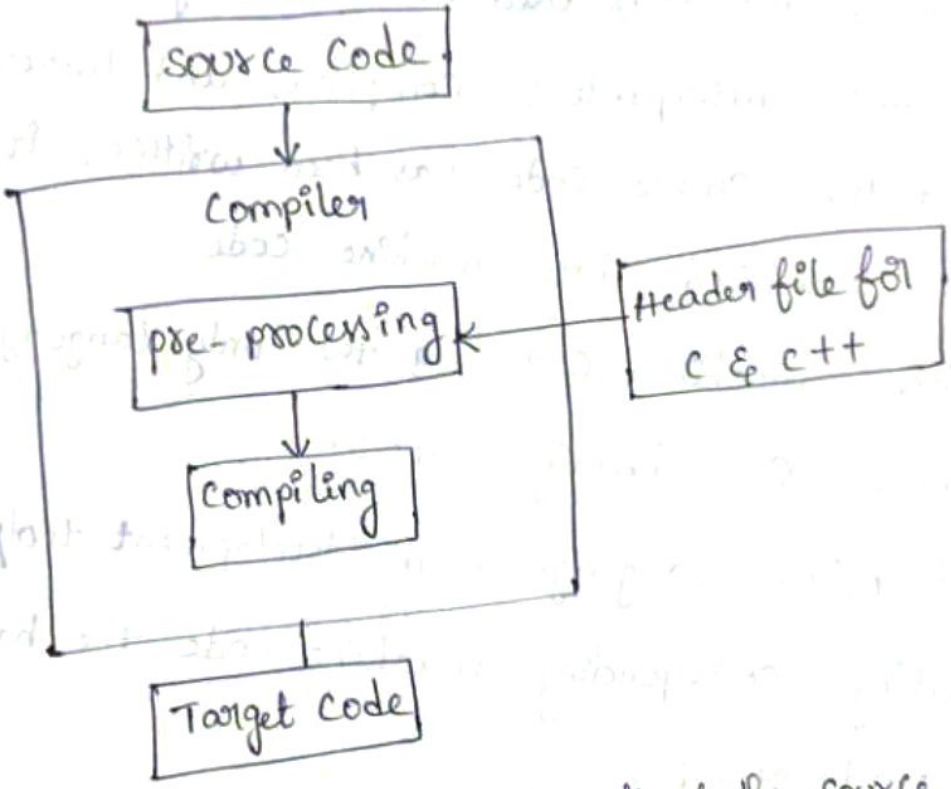
* Dividing more complex circuits into smaller circuits.
* Utilizing special characteristics of certain types of circuits.
* Utilizing vector - high speed and / or parallel components

Translation tools - preprocessors, Interpreters, compilers and linkers :-

* Some of the tools used in translating code, including preprocessors, interpreters, compilers and linkers.
* After the source code has been written, it needs to be translated into machine code.
* Since machine code is the only language the hardware can directly execute.
* All other language need development tools that generating corresponding machine code the hardware will understand
* preprocessing is an optional step that occurs either before translation (or) interpretation of source code and whose functionality is commonly implemented by preprocessor.
* The preprocessor's role is to organize and restructure

the source to make translation (or) interpretation of this code easier.

* Many languages convert source code, either directly (or) after having been preprocessed to target code through the use of a compiler a program which generates some target language such as machine code.



* A compiler typically translates all of the source code to a target code at one time.
* Most compilers are located on the host machine and generate target code for hardware platforms that differ from the platform the compiler is actually running on.
* These compilers are commonly referred to as Cross Compilers.

* In the case of assembles, an assembly Compiler is a Specialized Cross-Compiler referred to as an assembler and will always generate machine code.
* other high level language compilers are commonly referred to by the language name plus "Compiler" (i.e; Java Compiler, C compiler)
* After all the Compilation on the programmer's host machine is completed the remaining. Target code file is Commonly referred to as an object file.
* A linker integrates this object file with any other required system libraries, creating what is commonly referred to as executable binary file. It is to be transferred to the target embedded systems memory by a loader.
* Software placement is the ability to divide the software into modules and relocate these modules of code and data anywhere in memory.
* while the IDE, preprocessors, Compilers, linkers and soon reside on the host development system. some languages such as java and scripting languages have Compilers (or) interpreters located on the target.
* An interpreter generates machine code one source code

line at a time from source code (or) target code generated by - a intermediate compiler on the host system.

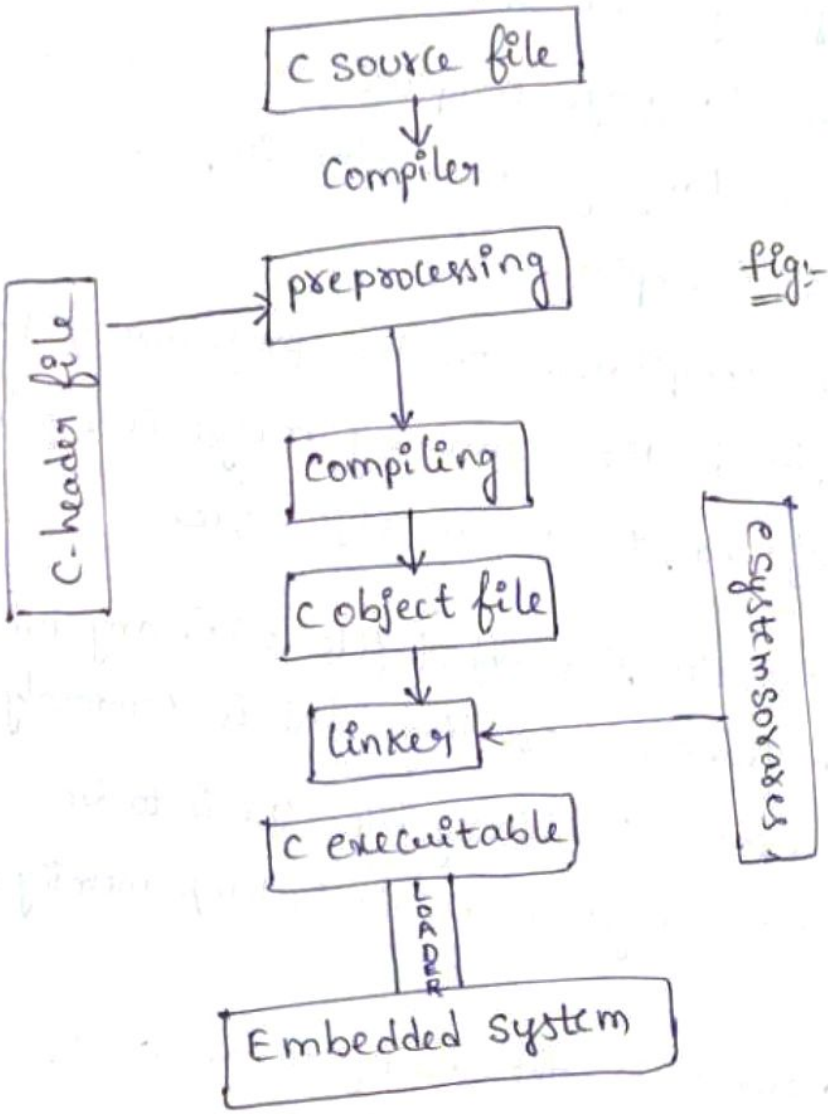
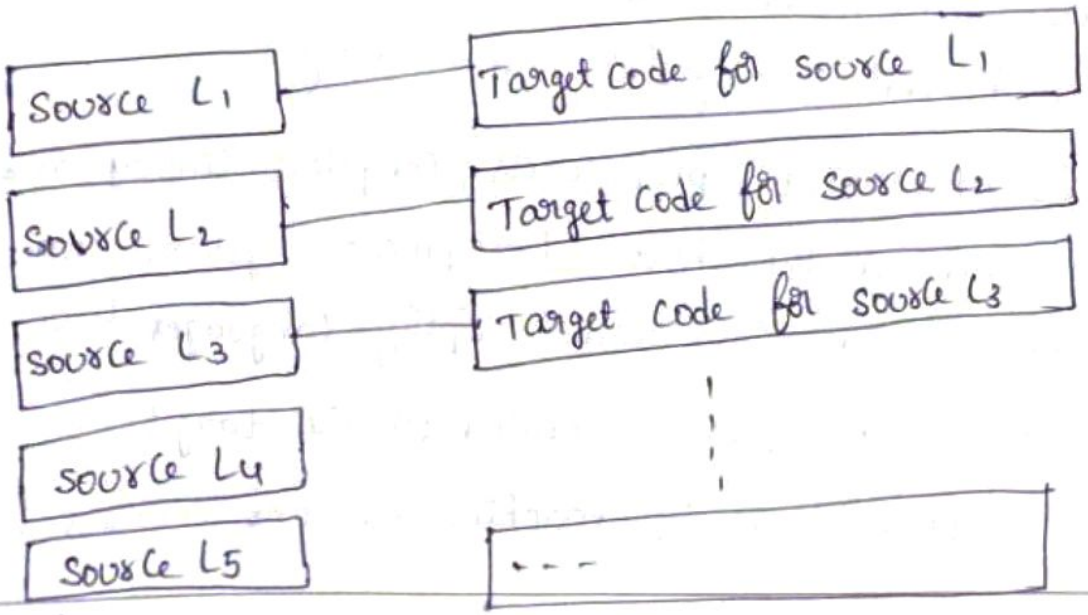


fig:- Header file for C & C++



Debugging tools :-

Hardware type debugging Tools :-

1. InCircuit Emulator (ICE):- Active device replace microprocessor in system.

Examples :- Typically most expensive debug solution but has a lot of debugging capabilities.

* Can operate at the full speed of the processor.
* Allows visibility and modifiability of internal memory, registers, variables etc. real time.

=> process dependent

2. ROM Emulator :- Active tool replaces ROM with Cables connected to dual port RAM within ROM Emulator simulates ROM. It is an intermediate hardware device connected to the target via some Cable and Connected to the host via another port.

Ex:- Allows modification of contents in ROM.

* Can set break point in ROM code and view ROM code real time.
* usually does not support on-chip ROM Custom ASK.

3. Oscilloscope :- passive analog device that graphs voltage Version time, ditecting exact voltage at given time.

Ex:- Monitor upto 2 signals simultaneously.

* Can set a trigger to Capture voltage given specific condition

* used as a voltmeter
* processor independent

4. Logic Analyzer:- passive device that captures and tracks multiple signals simultaneously and can graph them.

Ex:- Can be expensive.

* Typically can only track 2 voltages Vcc and GND signals in-between are graphed as either one (01) the other.
* Can store data.
* logic analyzer can only access data transmitted externally to and from processor not the internal memory, registers etc...
* Some will show assembly code but usually cannot set break point and single step through code using analyzer.

5. Voltmeter :- Measure voltage difference between 2 points on circuit.

Ex:- to measure for particular voltage values.

* TO determine if circuit has any power at all.
* Cheaper than the other hardware tools.

6. Ohm-meter:- Measure resistance between 2 points on ciscuit.

Ex:- Cheaper than other hardware tools.

* TO measure changes in current / voltage interms of resistance (ohm's Law V = IR)

7. Multi meter:- Measure both voltage and resistance.

Ex:- Same as volt and ohm meters.

Software type debugging TOOLS:-

1. Debugger:- [Functional debugging TOOL]

* Implementing break points to stop software execution
* Implement using conditional break points to stop if particular condition is met during condition.
* Can modify contents of RAM, typically cannot modify Contents ROM.

2. profiler:- collects timing history of selected variables, Registers ... etc.

Ex:- Capture time dependent (when) behaviour of executing software.

* TO capture execution pattern (where) of executing Software.

3. Monitor:- Debugging interface similar to ICE. with debug software running to on target and host. software on host and target typically communicate via serial (or) ethernet.

* Testing techniques fall under one of four models. Static black box testing static white box testing dynamic black box testing (or) dynamic white box testing.
* Black box testing occurs with a tester that has no visibility into the internal workings of the system (no schematics, no source codes .--- etc)
* Black box testing is based on general product level requirements documentation.
* In white box testing the tester has access to source code schematics and so on.
* Static testing is done while the system is not running where as dynamic testing is done when the system is running.

Static Testing :-

Blackbox testing :- Testing the product specifications.

1. Looking for high level fundamental problems oversights ommissions.
2. Low level specification testing by insuring completeness,

accuracy, preciseners, cosistences, relavance etc.

White Box testing :- process of methodically reviewing hardware and code for bugs without executing it

Dynamic Testing :- Requires definition of what software and hardware does includes :

Black Box testing :-

* Data testing :- which is checking into of user inputs and outputs.

* Boundary Condition testing :- which is testing simulations at edge of planned operational limits of software.

* Input Testing :- which is testing NULL, invalid data.

* Internal Boundary testing :- which is testing powers of two, ASCII table.

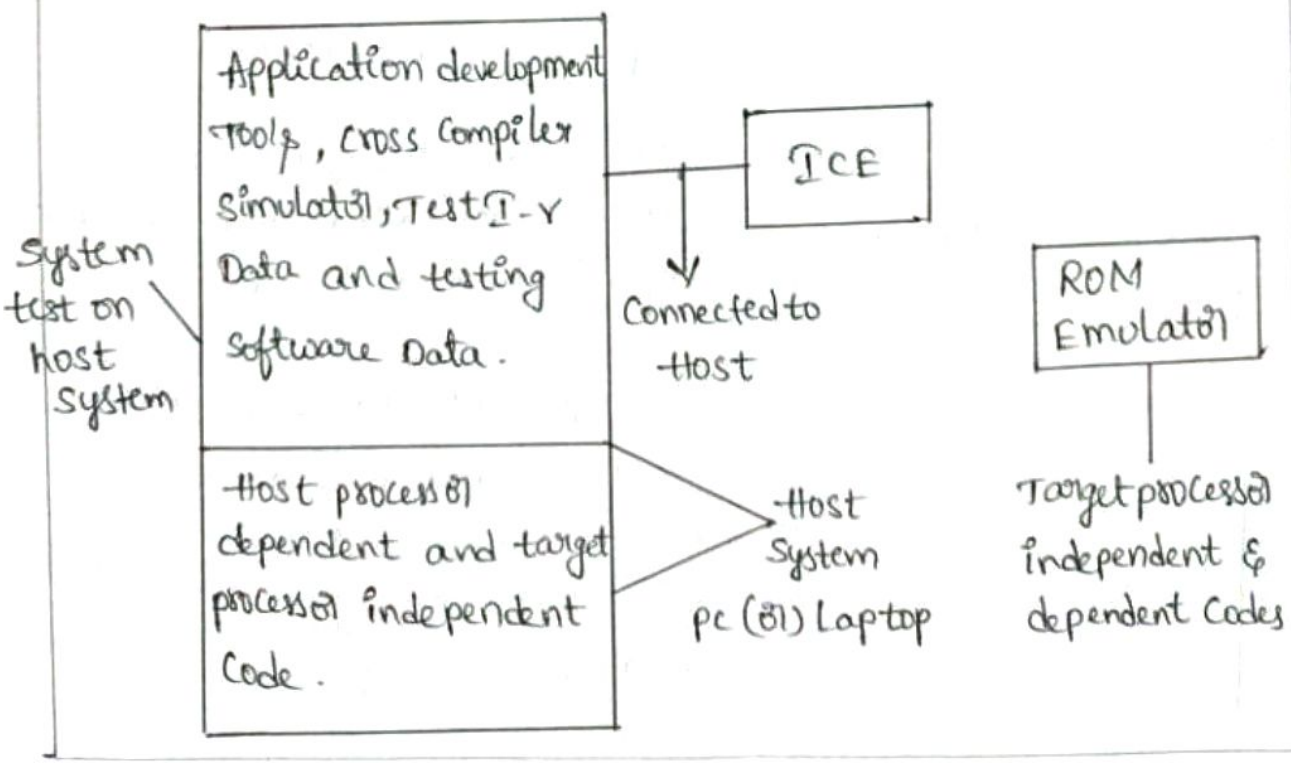
White Box testing :- Testing running system while looking at code, schematics ... etc.

* Directly testing low-level and high-level based on detailed operational knowledge accessing variables and memory dumps.

* Looking for data reference errors, declaration errors, Computation errors, Comparson errors. Control flow errors, parameter errors, I/o errors etc...

Testing on Host machine :-

* we have 2 systems with different cpu's (or) uc and hardware architecture one system in Host and other is the target.
* the Host is typically pc (or) Laptop. Target is actual hardware to used for embedded system under development.
* Testing and debugging have to be there at each stage as well as at the final stage when the modules are put together.
* Test at initial stages is done on the host machine.
* Host machine used to test hard ware - independent Codes. Host machine is also used to run simulator.
* The code has two parts hard ware independent and hardware dependent codes...



Steps	Action
1. Initial test	: Test each module at initial stage itself and on host itself.
2. Test Data	: All possible combinations of data are designed and taken as per data.
3. Exception Conditions Tests	: Consider all possible exceptions for test.
4. Tests - 1	: Test hard ware independent code.
5. Tests - 2	: Test scaffold software (scaf fdd software is software running on the host of target dependent code)
6. Test interrupt servile routines hardware independent part	: Those sections of interrupt servile routines are called, which are H/w independent and tested.
7. Test interrupt servile routines H/w dependent part	: Those sections of interrupt servile routines are called, which are H/w dependent and tested.
8. Timer tests	: H/w dependent code has timing functions and uses as timing device. Timer related routines are clock tick set, counts get, delay are tested.

Simulators:-

* Simulator uses knowledge of target processor (or) micro-controller, and target system architecture on the host processor.
* Simulator first does cross compilation of the codes and places these into the host system RAM.
* The behaviour of the target system processor registers is also simulated in RAM.
* It uses linker and locator to port the cross compiled codes in RAM.
* Simulator software also simulates hardware units such as emulator, peripherals, network and input output devices on a host.
* A simulator remains independent of a particular target system.
* A simulator helps in the development of the system before the final target system is ready with only a PC as the tool for development.
* The below figure shows the detailed design development process using the simulator.

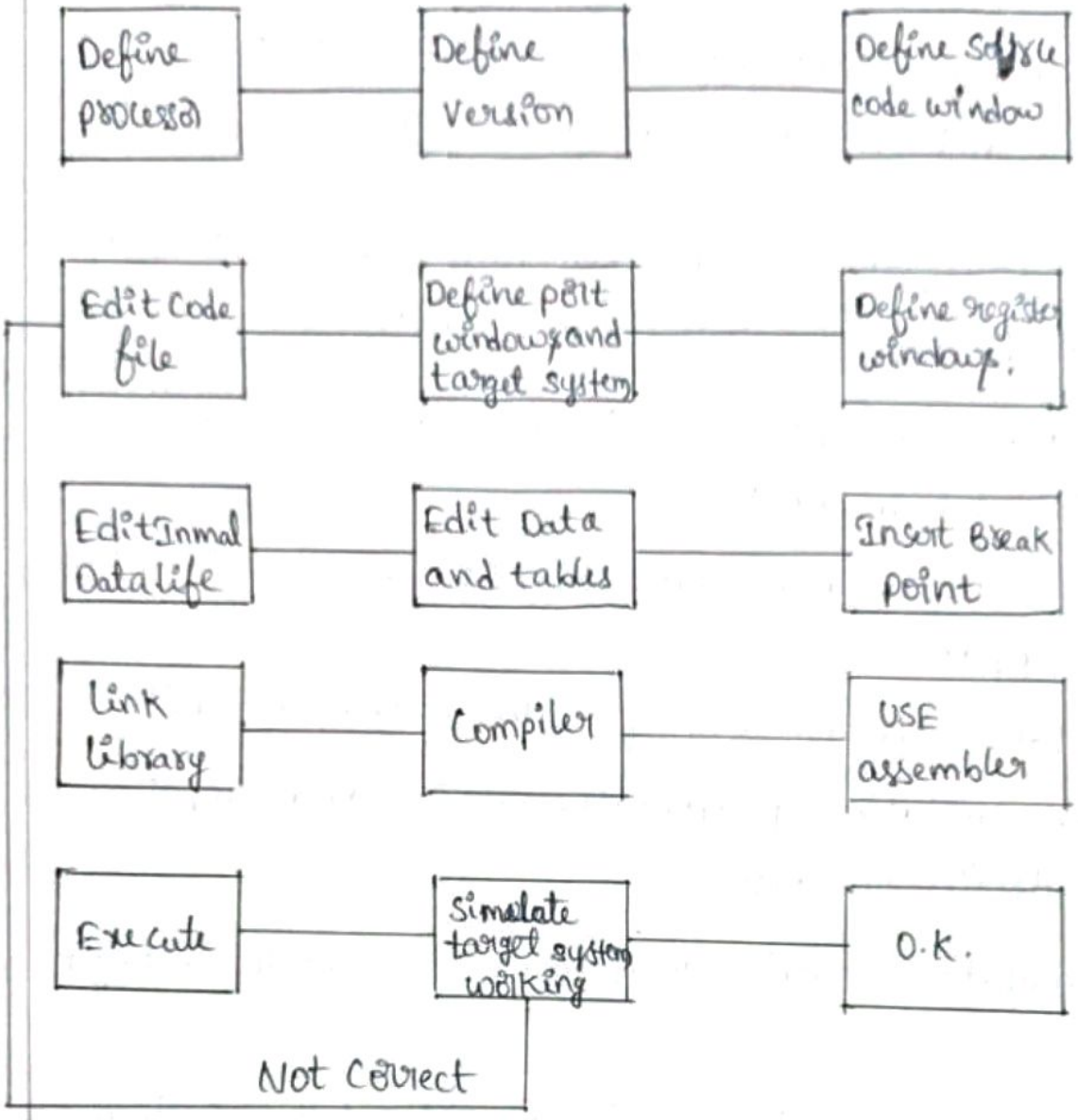


fig:- The detailed design development process using the simulator.

Simulator features:-

* It defines the processor(or) processing device family as well as its various versions for target system.
* It monitors the detailed information of a source code part with labels and symbolic arguments as the

execution goes on for each step.

* It provides the detailed information of the status of peripheral devices with the defined system.
* It provides the detailed information of RAM and ports of the defined target system.
* It provides the detailed information of the registers as the execution goes on for each single step (or) for each single module.
* It monitors the detailed information of the simulator commands as these are entered from the keyboard (or) select from the menu.
* It provides network driver and device driver support.
* It employs preempting RTOS schedules for the high priority tasks.

Laboratory Tools :-

* Simple voltmeter / ohm meter
* Oscilloscope
* Logic analyzer
* In circuit Emulator
* Monitor
* ROM Emulator

The above topics are already discussed in topic "Debugging tools" — Refer this topic.

— o —