MODULE-2: Introduction to Embedded Systems

VSIT Vidyalankar School of Information Technology NAAC Accredited College	**Vidyalankar School of Information Technology** **Wadala (E), Mumbai** **www.vsit.edu.in**

Umesh Koyande

Asst. Professor, VSIT

VSIT | Vidyalankar School of Information Technology

Certificate

This is to certify that the e-book titled "EMBEDDED SYSTEMS" comprises all elementary learning tools for a better understating of the relevant concepts. This e-book is comprehensively compiled as per the predefined eight parameters and guidelines.

Signature

Date: 30-01-2021

Ms. Umesh Koyande

Assistant Professor Department of IT

⚠ *DISCLAIMER: The information contained in this e-book is compiled and distributed for educational purposes only. This e-book has been designed to help learners understand relevant concepts with a more dynamic interface. The compiler of this e-book and*

Vidyalankar Institute of Technology give full and due credit to the authors of the contents, developers and all websites from wherever information has been sourced. We acknowledge our gratitude towards the websites YouTube, Wikipedia, and Google search engine. No

commercial benefits are being drawn from this project.

UNIT – II

Embedded Systems – Application and Domain Specific.

SYLLABUS

- **Embedded Systems: Application specific** washing machine, **domain specific -** automotive.
- Embedded Hardware: Memory map, i/o map, interrupt map,processor family, external peripherals, memory RAM , ROM, types of RAM and ROM, memory testing, CRC ,Flash memory.
- Peripherals: Control and Status Registers, Device Driver, Timer
- Driver - Watchdog Timers.

Application Specific Domain:

➢ Working of Washing Machine :

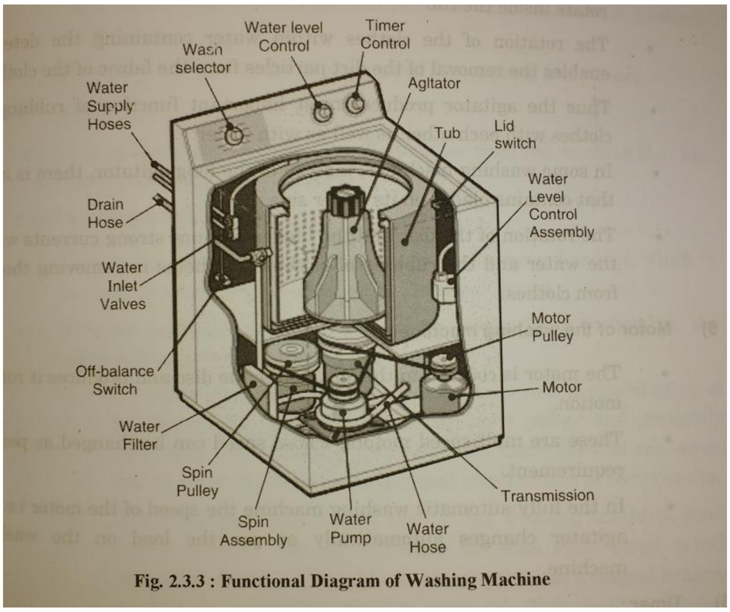


Fig. 2.3.3 : Functional Diagram of Washing Machine

(i) User-Interface :

1) The user-interface consists of user panel and a keypad.

2) It is a touch panel screen to control all the operation of the machine.

3) The display panel is used to display the information to the user such as total wash-time or remaining time or type of washing modes.

(ii) Sensor :

1) Sensors are used to sense different physical quantities.

2) It measures the water-level and adds appropriate amount of detergent.

3) Input devices of automatic washing machines are sensors for water-flow, water-level, temperature and door-switch.

(iii) Water inlet control valve :

1) Near the water inlet point of the washing machine, there is a water inlet control valve.

2) When you load the clothes in washing machine, this valve opens automatically, and it closes automatically depending on the total quantity of the water required.

3) The water control valve is actually a solenoid valve.

(iv) Water Pump :

1) The water pump circulates water through the washing machine.

2) It works in two directions, i.e. recirculating the water during wash cycle and draining the water during spin-cycle.

(v) Tub / Drum :

1) There are two types of tubs in washing machine, i.e. inner & outer tub.

2) The clothes are loaded in the inner drum where it is washed, rinsed and dried.

3) The inner drum has small holes, for draining the water out.

4) The external/outer drum covers the inner drum and supports it during various cycles of clothe washing.

(vi) Agitator / Rotating disk :

1) The agitator is located inside the tub of the washing machine.

2) It is the important part of washing machine that actually performs the cleaning operation of the clothes.

3) During the wash cycle, the agitator rotates continuously and produces strong rotating currents within the water due to which the clothes also rotate inside the tub.

4) The rotation of the clothes within the water containing the detergent enables the removal of the dirt particles from the fabrics of the clothes.

5) Thus the agitator produces the most important function of rubbing the clothes with each other as well as with water.

(vii) Motor of the Washing Machine :

1) The motor is coupled *(connected)* to the agitator or the disk and produces its rotation motion.

2) These are multispeed motors whose speed can be changed as per the requirement.

3) In fully automatic washing machine, the speed of the motor, i.e. the agitator changes automatically as per the load on the washing machine.

(viii) Timer / Timer Control :

1) The timer helps in setting the wash time for the clothes manually.

2) In the automatic mode, time is set automatically depending upon the number of clothes inside the washing machine.

(ix) Drain Pipe : The drain-pipe enables removing the dirty water from the washing machine that has been used for washing purpose.

Automotive – Domain-specific Examples Of Embedded System

- Automotive embedded systems are normally built around microcontrollers or DSPs or a hybrid of the two and are generally known as **Electronic Control Units** (ECUs).
- The number of embedded controllers in an ordinary vehicle varies from 20 to 40 whereas a luxury vehicle may contain 75 to 100 numbers of embedded controllers
- **High–speed Electronic Control Units (HECUs)**

High–speed electronic control units (HECUs) are deployed in critical control units requiring fast response.

e.g.fuel injection systems, antilock brake systems, engine control

- **Low–speed Electronic Control Units (LECUs)**

Low–Speed Electronic Control Units (LECUs) are deployed in applications where response time is not so critical.

They generally are built around low cost microprocessors / microcontrollers and digital signal processors.

e.g. Audio controllers, passenger and driver door locks

- **Controller Area Network (CAN)**

It supports medium speed (125 Kbps) and high speed (1Mbps) data transfer.

CAN is an event–driven protocol interface with support for error handling in data transmission

e.g. airbag control; Antilock Brake System (ABS); navigation systems like GPS

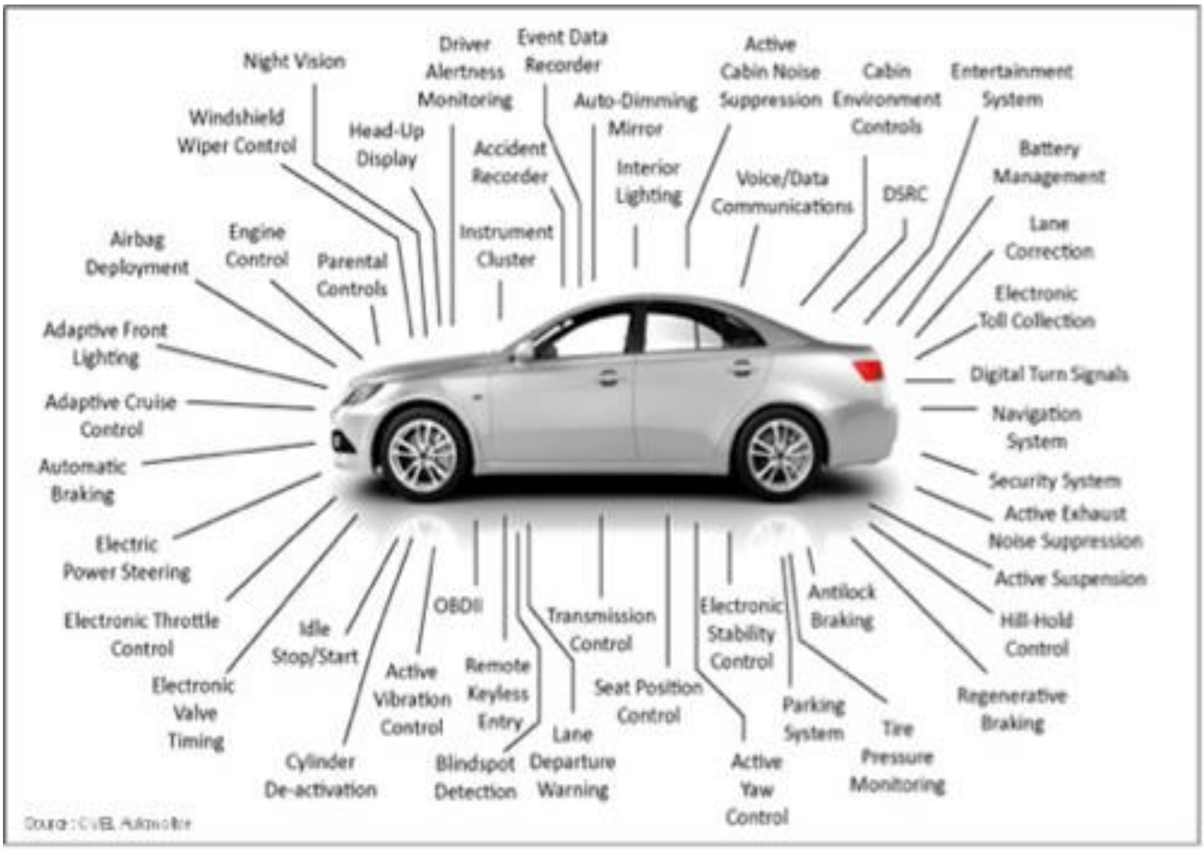
- **Local Interconnect Network (LIN)**

LIN bus is a single master multiple slave (up to 16 independent slave nodes) communication interface.

LIN is a low speed, single wire communication interface with support for data rates up to 20 Kbps and is used for sensor / actuator interfacing.

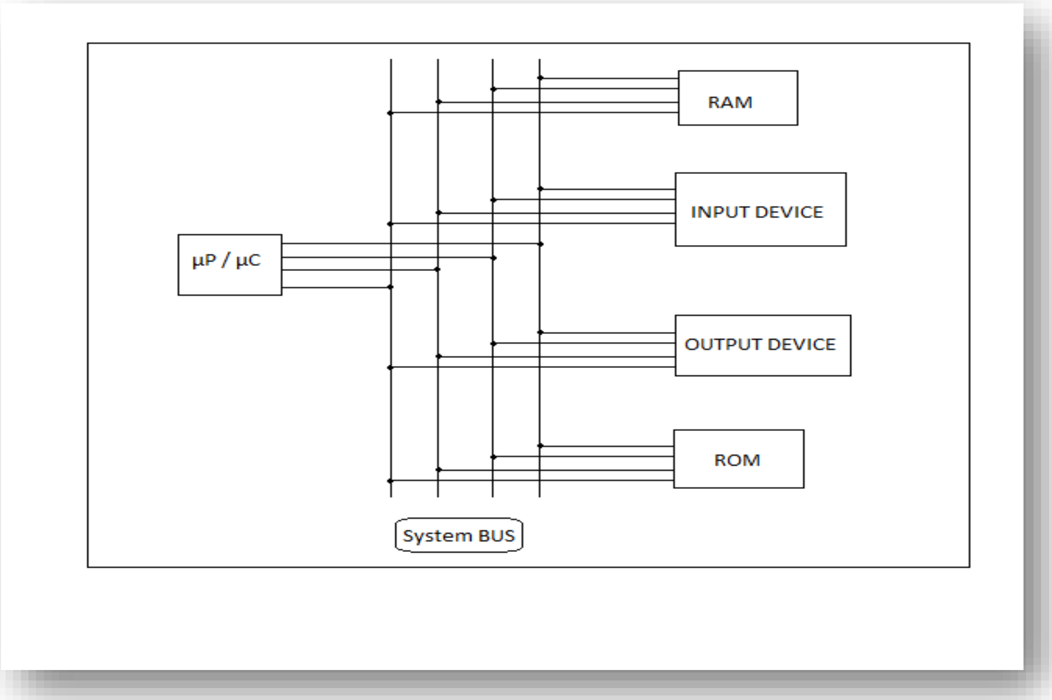
LIN bus follows the master communication triggering technique to eliminate the possible bus arbitration problem

e.g. mirror controls, fan controls, seat positioning controls, window controls and position controls where response time is not a critical issue



Source : CVEL Automotive

Memory-Map Input-Output :



1) In *memory map input-output* scheme, a part of the memory space is allocated to the I/O (input-output) devices.
2) All the I/O devices are treated as memory location.
3) This technique does not require any special instruction.
4) The microprocessor/microcontroller can access the I/O devices using the signals

$$\overline{\text{MEM R}} \ \ \& \ \ \overline{\text{MEM W}}$$

Advantages of Memory Map I/O

1) This scheme simplifies as well as increases the data transfer rate for peripheral operation.
2) All the memory related instruction can be used in read and write operation.
3) This scheme provides more than (2^8) 256 I/O ports.
4) Using this scheme CPU register can exchange data with the I/O devices without accumulator.

- **Disadvantage of Memory Map I/O**

1) Memory map I/O requires complex programming.
2) Due to wide port addresses, the interface of the hardware is also complicated.

3) Memory map I/O instruction utilizes memory reference; these instructions are longer as compared to I/O instructions.

I/O Mapped I/O

1) In *input/output map input/output* scheme, I/O device cannot be considered as a memory location.
2) This scheme requires special instruction such as IN/OUT to access the I/O devices.
3) Microprocessor 8085 can access (2^8) 256 I/O port using 8-address lines out of 16-address lines.
4) The I/O read & write operations are controlled by control signal

$\overline{\text{IOR}}$ & $\overline{\text{IOW}}$.

5) The port address can be from 00 H to FF H.

- **Advantages**
  1) A total of 256 address spaces are available for I/O devices for 8085 microprocessor.
  2) Writing program for this scheme is quite simple as compared to memory mapped I/O.
  3) To access any I/O device, simple instructions are used which is very short, hence take less memory space.
  4) Debugging of the program is also very easy.
- **Disadvantages**
  1) It requires additional instruction to distinguish between peripheral and memory related operation.
  2) In data transfer with I/O devices and the microprocessor, the data must be transferred to the accumulator.
  3) This scheme does not support various addressing modes.

Interrupt Mapped :

1) By using interrupts, advanced programming can be done; there are varieties of interrupts available in microprocessor and microcontroller.
2) Every interrupt has an ***ISR*** (Interrupt Service Routine) which executes necessary instructions so as to deal with interrupts.

3) In order that the processor executes correct ISR, a mapping must exist between the interrupt source and ISR function.

4) This mapping is generally controlled by an interrupt vector table.

The vector table has an array of pointers to the memory location.

5) The processor uses interrupt numbers to index the array. (Eg.: RST 6.5, 7.5, 5.5)

6) An ISR is the code executed when an interrupt occurs after finishing the ISR program, execution returns to the original program, (i.e. to the main program) & continues execution exactly from where it was interrupted.

https://www.youtube.com/watch?v=qdkxXygc3rE

Processor Family

1) A set of related processors, usually successive generations from the same manufacturer is called processor family.
2) Example: Intel 80x family began with 8085, followed by 8086and now includes 80286, 80386, 80486, Pentium and many others.
3) The term processor refers to any of the following 3 devices:-

   (i) Microprocessor (µP)

   a) The name µP usually reserved for a chip that contains a powerful CPU that has not been designed with any particular computation or for some specific function.
   b) The most common processors are members of Motorola i.e. 68k found in older Macintosh computers.

   (ii) Microcontroller (µC)

   a) A µC is like µP except that it has been designed specifically for use in embedded system.
   b) µC typically includes CPU, memory and other peripherals.
   c) The most famous/popular µC is 8051, which has many imitators such as Motorola’s 68 HCXX series.

   (iii) DSP – Digital Signal Processor

   a) The CPU within a DSP is specially designed to perform discrete time signal processing calculation. Example:- processor for audio-video communication.
   b) The DSP performs such calculations much faster, than other processors.
   c) They offer a powerful low cost µP alternative for designer of modems and other telecommunication , two of the most common DSP families are TMS 320CXX and 5600X series from Motorola respectively.

External Peripherals:

- Embedded System contains different hardware devices called peripherals.
- The peripherals which include within the same chip as the processor is called as internal peripherals, while the peripherals which reside outside the chip is called as external peripherals.
- Peripherals can be classified on the following basis:

i. Simplex, Duplex or Semi-duplex:

Simplex communication involves unidirectional data transfer while duplex communication involves bidirectional data transfer. Full duplex interfaces have different channels for transmission and reception. Semi-duplex interfaces uses the same communication channels for both transmission and reception.

ii. Serial & Parallel:

Serial peripherals communicate over a single data line. Hence this type of communication requires less hardware as well as reduce cost. Parallel peripherals communicate using many data lines, they are much faster than serial devices.

iii. Synchronous & Asynchronous:

Synchronous data transfer are synchronous by a reference clock which may be on chip or may be provided by an external device. Asynchronous data transfers are not synchronised with the clock.

iv. Data Throughput:

Peripherals devices can also be classified on the basis of their data transfer rate. Generally parallel interfaces provide much more data throughput as compared to serial interfaces.

Types of Memory :

A computer memory can be viewed as a list of cells into which numbers can be placed or read. Each cell has a numbered address and can store a single number. The information stored in memory may represent practically anything.

Memory can be further classified as :

1) RAM
2) ROM

1) RAM

- It is called as Random Access Memory because the processor or computer can access any location in the memory at any given time.
- RAM are made from relays, transistors, ICs, magnetic core or anything that can hold and store binary values,+,- sign or positive, negative or high, low.
- RAM are further classified as
  (i) Static RAM (S-RAM)
  (ii) Dynamic RAM (D-RAM)

(i) Static RAM is called static because it will continue to hold and store information even when the power is removed.

Magnetic cores and relays are examples of static S-RAM.

(ii) Dynamic RAM is called as dynamic because it loses all data when the power is removed. Transistors and ICs are examples of D-RAM.

2) <u>ROM</u>

- ROM is read only memory, which is typically used to store things that will never change for the life time of the computer.
- ROM is also referred as ‘mask ROM’, which is fabricated with the desired data permanently stores in it and this, can never be modified.
- However, more modern types of ROMs such as EPROM(Erasable Programmable Read Only Memory) & EEPROM (Electronically Erasable Programmable Read Only Memory) can be erased and reprogrammed multiple times and yet they are described as ROM, because the reprogramming process is generally infrequent and often does not permit random access rights to individual memory location. There are three types of ROM:

    (i) PROM (Programmable Read Only Memory)

    (ii) EPROM (Erasable Programmable Read Only Memory)

    (iii) EEPROM (Electronically (Erasable Programmable Read Only Memory)

(i) PROM:-

PROM is one time programmable non-volatile memory which is in a form of digital memory where the setting of each bit is locked by a fuse or antifuse. Such PROMs are used to store programs permanently. The key difference between ROM and PROM is that the programming is applied after the device is constructed. These types of memories are frequently used in video game consoles, mobile phones, medical devices, etc. The memory can e programmed just once after manufacturing.

(ii) EPROM:-

EPROM is a type of memory chip that can retain its data, even when the power supply is switched off. Once programmed and EPROM can be erased by exposing it to strong ultra violet lights from a mercury vapour light source.

(iii) EEPROM:-

E^{2}PROM is a non-volatile memory used in computers and other electronic devices to store small amount of data that can be saved when power is removed. E^{2}PROM is user modifiable READ ONLY MEMORY that can be erased and reprogrammed repeatedly through the application of higher than normal electrical voltage generated externally or internally in case of modern E^{2}PROM.

https://www.youtube.com/watch?v=qdkxXygc3rE

Memory Testing :

1) Testing the memory in the present microcontroller based system is quite simple.
2) Generally, microcontroller based system have a keypad and a 7-segment display as an output device.
3) To check the given memory chip, we can store a specific sequence of data using keypad and verifying the same by displaying them on output device.
4) Using the same procedure, we can check the memory location from start to end.
5) If during verification of memory, it is observed that some bit within the data byte are always corrupted, then we can conclude that there is problem within the data-bus.
6) Following are the frequent problems associated with memory :-

   (i) Physical/electronic damage to memory chip.

   (ii) Memory chip not present in memory slot.

   (iii) Improperly inserted memory modules.

   (iv) A short circuit which generally occurs at the time of soldering.

   (v) Another problem is open circuit in which the given trace is floating (i.e. not connected).

   (vi) Problem may also arise due to address BUS leading to selection of wrong addresses.

Developing a Memory Test Strategy

- It is best to have three individual memory tests: a data bus test, an address bus test, and a device test.
- The first two test for electrical wiring problems and improperly inserted chips; the third is intended to detect missing chips and catastrophic failures.
- The order in which you execute these three tests is important. The proper order is: data bus test first, followed by the address bus test, and then the device test. That's because the address bus test assumes a working data bus, and the device test results are meaningless unless both the address and data buses are known to be good.

Data bus test:

- We need to confirm that any value placed on the data bus by the processor is correctly received by the memory device at the other end.
- The most obvious way to test that is to write all possible data values and verify that the memory device stores each one successfully. However, that is not the most efficient test available.
- A faster method is to test the bus one bit at a time.
- The data bus passes the test if each data bit can be set to 1 an d independently of the other data bits.
- A good way to test each bit independently is to perform the so-called "walking 1's test." Table below shows the data patterns used in an 8-bit version of this test.
- The name, walking 1's, comes from the fact that a single data bit is set to 1 and "walked" through the entire data word. The number of data values to test is the same as the width of the data bus. This reduces the number of test patterns from 2n to n, where n is the width of the data bus.

Consecutive Data Values for the Walking 1's Test
00000001
00000010
00000100
00001000
00010000
00100000
01000000

10000000

- Because we are testing only the data bus at this point, all of the data values can be written to the same address. Any address within the memory device will do. However, if the data bus splits as it makes its way to more than one memory chip, you will need to perform the data bus test at multiple addresses, one within each chip.

Address bus test:

- After confirming that the data bus works properly, you should next test the address bus. Remember that address bus problems lead to overlapping memory locations.
- There are many possible addresses that could overlap. However, it is not necessary to check every possible combination. You should instead follow the example of the previous data bus test and try to isolate each address bit during testing.
- You simply need to confirm that each of the address pins can be set to and 1 without affecting any of the others.
- The smallest set of addresses that will cover all possible combinations is the set of

"power-of-two" addresses.

- These addresses are analogous to the set of data values used in the walking 1's test. The corresponding memory locations are 00001h, 00002h, 00004h, 00008h, 00010h, 00020h, and so forth. In addition, address 00000h must also be tested.
- The possibility of overlapping locations makes the address bus test harder to implement. After writing to one of the addresses, you must check that none of the others has been overwritten.
- It is important to note that not all of the address lines can be tested in this way. Part of the address-the leftmost bits-selects the memory chip itself. Another part-the rightmost bits-might not be significant if the data bus width is greater than 8 bits. These extra bits will remain constant throughout the test and reduce the number of test addresses.
- For example, if the processor has 20 address bits, as the 80188EB does, then it can address up to 1 megabyte of memory. If you want to test a 128-kilobyte block of memory, the three most significant address bits will remain constant. In that case, only the 17 rightmost bits of the address bus can actually be tested.

Device test:

- Once you know that the address and data bus wiring are correct, it is necessary to test the integrity of the memory device itself.
- The thing to test is that every bit in the device is capable of holding both 0 and 1. This is a fairly straightforward test to implement, but it takes significantly longer to execute than the previous two.
- For a complete device test, you must visit (write and verify) every memory location twice. You are free to choose any data value for the first pass, so long as you invert that value during the second. And because there is a possibility of missing memory chips, it is best to select a set of data that changes with (but is not equivalent to) the address. A simple example is an increment test.
- The data values for the increment test are shown in the first two columns of Table given below ,the third column shows the inverted data values used during the second pass of this test.
- The second pass represents a decrement test. There are many other possible choices of data, but the incrementing data pattern is adequate and easy to compute.

Table 6-3. Data Values for an Increment Test

Memory Offset	**Binary Value**	**Inverted Value**
000h	00000001	11111110
001h	00000010	11111101
002h	00000011	11111100
003h	00000100	11111011
...	...	...
0FEh	11111111	00000000
0FFh	00000000	11111111

Cyclic Redundancy Codes:

- A cyclic redundancy code (CRC) is a specific checksum algorithm that is designed to detect the most common data errors.

- Cyclic redundancy codes are frequently useful in embedded applications that require the storage or transmission of large blocks of data.
- When computing a CRC, you consider the set of data to be a very long string of 1's and 0's (called the *message*).
- This binary string is divided-in a rather peculiar way-by a smaller fixed binary string called the *generator polynomial*. The *remainder* of this binary long division is the CRC checksum.
- By carefully selecting the generator polynomial for certain desirable mathematical properties, you can use the resulting checksum to detect most (but never all) errors within the message.
- The strongest of these generator polynomials are able to detect all single and double bit errors, and all odd-length strings of consecutive error bits.

- In addition, greater than 99.99% of all burst errors-defined as a sequence of bits that has one error at each end-can be detected. Together, these types of errors account for a large percentage of the possible errors within any stored or transmitted binary message.

	CCITT	CRC16	CRC32
Checksum size (width)	16 bits	16 bits	32 bits
Generator Polynomial	$\mathbf{X^{16} + x^{12} + x^{5} + 1}$	$\mathbf{X^{16} + x^{12} + x^{5} + 1}$	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} +$ $\mathbf{x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1}$
Divisor (polynomial)	0x1021	0x8005	0x04C11DB7
Initial remainder	0xFFFF	0x0000	0xFFFFFFFF
Final XOR value	0x0000	0x0000	0xFFFFFFFF

- Those generator polynomials with the very best error-detection capabilities are frequently adopted as international standards. Three such international standard are as follows:

Flash Memory:

- Flash memory is the most recent advancement in memory technology. It combines all the best features of the memory devices described thus far.
- Flash memory devices are high density, low cost, nonvolatile, fast (to read, but not to write), and electrically reprogrammable.
- These advantages are overwhelming and the use of Flash memory has increased dramatically in embedded systems as a direct result.
- From a software viewpoint, Flash and EEPROM technologies are very similar. The major difference is that Flash devices can be erased only one sector at a time, not byte by byte.
- Typical sector sizes are in the range of 256 bytes to 16 kilobytes. Despite this disadvantage, Flash is much more popular than EEPROM and is rapidly displacing many of the ROM devices as well.
- Flash is arguably the most complicated memory device ever invented. The hardware interface has improved somewhat since the original devices were introduced in 1988, but there is still a long way to go.
- Reading from Flash memory is fast and easy, as it should be. In fact, reading data from a Flash is not all that different from reading from any other memory device.
- The processor simply provides the address, and the memory device returns the data stored at that location. Most Flash devices enter this type of "read" mode automatically whenever the system is reset; no special initialization sequence is required to enable reading.
- There is one small difference worth noting here. The erase and write cycles take longer than the read cycle. So if a read is attempted in the middle of one of those operations, the result will be either delayed or incorrect, depending on the device.
- Writing data to a Flash is much harder. Three factors make writes difficult. First, each memory location must be erased before it can be rewritten. If the old data is not erased, the result of the write operation will be some logical combination of the old and new values, and the stored value will usually be something other than what you intended.

- The second thing that makes writes to a Flash difficult is that only one sector, or block, of the device can be erased at a time; it is impossible to erase a single byte.

The size of an individual sector varies by device, but it is usually on the order of several thousand bytes. For example, the Flash device on the Arcom board-an AMD 29F010-has eight sectors, each containing 16 kilobytes.

- Finally, the process of erasing the old data and writing the new varies from one manufacturer to another and is usually rather complicated. These device programming interfaces are so awkward that it is usually best to add a layer of software to make the Flash memory easier to use. If implemented, this hardware-specific layer of software is usually called the Flash driver.

Control and Status Register:

- The basic interface between an embedded processor and a peripheral device is a set of control and status registers.
- These registers are part of the peripheral hardware, and their locations, size, and individual meanings are features of the peripheral. For example, the registers within a serial controller are very different from those in a timer/counter.
- Depending upon the design of the processor and board, peripheral devices are located either in the processor's memory space or within the I/O space.
- In fact, it is common for embedded systems to include some peripherals of each type. These are called memory-mapped and I/O-mapped peripherals, respectively. Of the two types, memory-mapped peripherals are generally easier to work with and are increasingly popular.
- Memory-mapped control and status registers can be made to look just like ordinary variables. To do this, you need simply declare a pointer to the register, or block of registers, and set the value of the pointer explicitly.
- There is one very important difference between device registers and ordinary variables. The contents of a device register can change without the knowledge or intervention of program. That's because the register contents can also be modified by the peripheral hardware.
- By contrast, the contents of a variable will not change unless program modifies them explicitly. For that reason, we say that the contents of a device register are volatile, or subject to change without notice.
- The primary disadvantage of the other type of device registers, I/O-mapped registers, is that there is no standard way to access them from C or C++. Such registers are accessible only with the help of special machine-language instructions.

Device Drivers:

- Device drivers for embedded systems are quite different from their workstation counterparts.

- In a modern computer workstation, device drivers are most often concerned with satisfying the requirements of the operating system. For example, workstation operating systems generally impose strict requirements on the software interface between themselves and a network card.
- The device driver for a particular network card must conform to this software interface, regardless of the features and capabilities of the underlying hardware.
- Application programs that want to use the network card are forced to use the networking API provided by the operating system and don't have direct access to the card itself. In this case, the goal of hiding the hardware completely is easily met.
- By contrast, the application software in an embedded system can easily access hardware. In fact, because all of the software is linked together into a single binary image, there is rarely even a distinction made between application software, operating system, and device drivers.
- The benefits of good device driver design are threefold. First, because of the modularization, the structure of the overall software is easier to understand.
- Second, because there is only one module that ever interacts directly with the peripheral's registers, the state of the hardware can be more accurately tracked.
- And, last but not least, software changes that result from hardware changes are localized to the device driver.
- Each of these benefits can and will help to reduce the total number of bugs in your embedded software.
- If you agree with the philosophy of hiding all hardware specifics and interactions within the device driver, it will usually consist of the five components in the following list and to make driver implementation as simple and incremental as possible, these elements should be developed in the order in which they are presented.

1. A data structure that overlays the memory-mapped control and status registers of the device
2. A set of variables to track the current state of the hardware and device driver
3. A routine to initialize the hardware to a known state
4. A set of routines that, taken together, provide an API for users of the device driver
5. One or more interrupt service routines

Watchdog timer :

https://www.youtube.com/watch?v=qdkxXygc3rE

i) Most of the embedded systems need to be self-relying. It is not usually possible to wait for someone to reboot them if the system hangs.

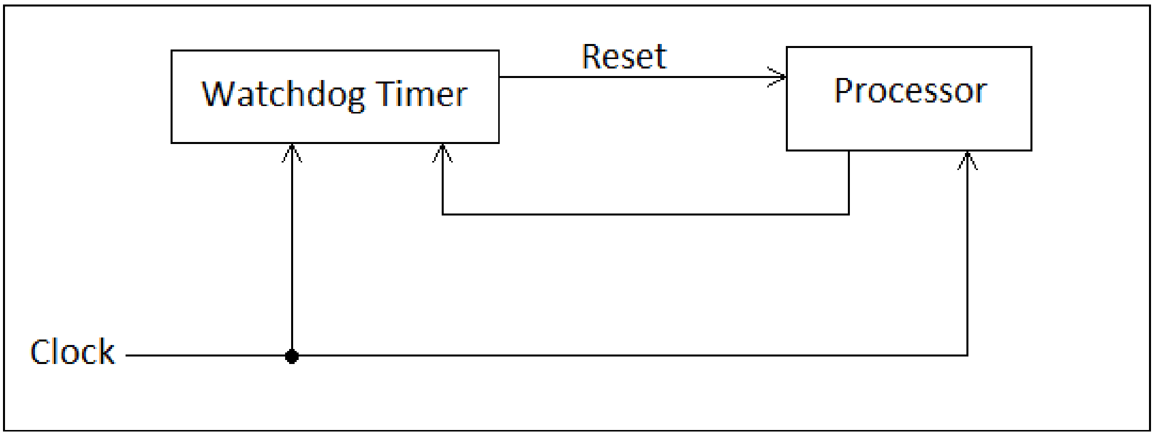
ii) Some embedded designs such as space probes are simply not accessible to human operator.

iii) If their software ever hangs, such systems are permanently disabled.

iv) In other cases, the speed with which a human operator might reset the system would be too slow to meet the uptime requirements of the product.

v) A watchdog timer is a piece of hardware that can be used to automatically detect software abnormalities and reset the processor if required.

vi) The following figure shows a typical arrangement of watchdog timer



vii) In the diagram it is seen that watchdog timer is a chip external to the processor however it could also be included within the same chip as the CPU.This is done in many microcontrollers.In either cases , the output from watchdog timer is tied/connected directly to the processor’s reset signal.

viii) *Kicking the dog* is nothing but the process of restarting the watchdog timer counter by software.

ix) The software must restart the watchdog timer at regular rate otherwise, there is a risk if the system being restarted.

x) A watchdog timer is a useful tool in helping the system to recover from failures.

Multiple Choice Questions-

1. Which one of following memories is volatile?
   a. NVRAM
   b. Flash
   c. SRAM
   d. EPROM

2. Watchdog timer is used -
   a. To prevent software hangs
   b. To delay the operations
   c. To generate interrupts
   d. To halt the system

3. Increment test is used for
   a. Memory testing
   b. Data testing
   c. Device testing
   d. none of the mentioned

4. Periodic refreshing must be carried out in-
   a. SRAM
   b. EPROM
   c. DRAM
   d. Masked ROM

5. EPROM is

a. Electrical Programmable Read only memory
b. External Programmable Read only memory
c. Electrically Partial Read only memory
d. Erasable Programmable Read only memory

Graded Questions:

1. Explain in detail application specific embedded system with an example.
2. Explain different automotive communication buses.
3. Define and explain ROM and it types.
4. Differentiate between SRAM and DRAM.
5. Write short note on DMA.
6. What do you mean by memory testing. Explain data bus test in detail.
7. Explain address bus test in detail.
8. Explain device test in detail.
9. Write short not on checksum in embedded system.
10. Write short not on CRC.
11. Why writing to flash memory is difficult.
12. Enlist step to develop device driver.
13. Write short note on watchdog timer.