UNIT-1 - INTRODUCTION TO EMBEDDED SYSTEMS

PART A (2 marks)

1. ***Define Embedded System. What are the components of embedded system?***

An Embedded system is one that has computer hardware with software embedded in it as one of its most important component. The three main components of an embedded system are (i) Hardware (ii) Main application software (iii) RTOS

2. What are the types of embedded system?

- Small scale embedded systems
- Medium scale embedded systems
- Sophisticated embedded systems

3. Name some of the hardware parts of embedded systems.

- Power source
- Clock oscillator circuit
- Timers
- Memory units
- DAC and ADC
- LCD and LED displays
- Keyboard/Keypad

4. Give the characteristics of embedded system.

- Single-functioned
- Tightly constrained
- Reactive and real time

5. What are the steps involved in the build process.

The steps involved are

- Preprocessing
- Compiling
- Linking
- Locating

6. ***What is a compiler and a cross compiler?***

Compiler: Compiler is a program that creates an object file from the source codes. It checks the language grammar/semantics. For eg: tcc. It converts high level language codes into machine codes.

Cross Compiler: A compiler that runs on one computer platform and produces code for another is called a cross-compiler. The use of a cross-compiler is one of the defining features of embedded software development.

7. List the important considerations when selecting a processor.

- Instruction set
- Maximum bits in an operand
- Clock frequency
- Processor ability

8. Classify the processors in embedded system.

a. General purpose processor

- Microprocessor
- Microcontroller
- Embedded processor
- Digital signal processor
- Media processor

b. Application specific system processor

c. Multiprocessor system using GPP and ASSP GPP core or ASIP core integrated into either an ASIC or a VLSI circuit or an FPGA core integrated with processor unit in a VLSI chip.

9. ***What are the various types of memory in embedded systems?***

- RAM (internal External)
- ROM/PROM/EEPROM/Flash
- Cache memory

10. What are the two essential units of a processor on a embedded system?

(i) Program Flow control Unit. (ii) Execution Unit

11. What are the different modes of DMA transfer? Which one is suitable for embedded system?

- Single transfer at a time and then release of the hold on the system bus.
- Burst transfer at a time and then release of the hold on the system bus. A burst may be of a few kB.
- Bulk transfer and then release of the hold on the system bus after the transfer is completed.

12. Define ROM image and RAM.

ROM image: Final stage software is also called as ROM image .The final implement able software for a product embeds in the ROM as an image at a frame. Bytes at each address must be defined for creating the image.

RAM: RAM refers Random Access Memory. It is a memory location that can be accessed without touching the other locations.

13. What are the uses of timers?

- The time intervals between two events can be computed
- Initiating an event after a preset time delay.
- Capturing the count value
- Watch dog timer

14. What is a watch dog timer?

The watch dog timer is a timing device that resets the system after a predefined timeout. It is activated within the first few clock cycles after power-up. It has a number of applications. Example: Mobile phone display turnoff in case of no interactions takes place within a specified time.

15. Define Real Time Clock.

Real time clock is a clock which once the system starts does not stop and cannot be reset and its count value cannot be reloaded.

UNIT – 1

PART B (16 marks)

1. Explain the build process for embedded system.

The build is often referred either to the process of converting source code files into stand alone software artifacts that can be run on a computer or the result of doing so. The process of converting the source code representation of the embedded software into an executable binary image involves three distinct steps.

- Each of the source files must be compiled or assembled into an object file.
- All of the object files that result from the first step must be linked together to produce a single object file called the re-locatable program.
- Physical memory addresses must be assigned to the relative offsets within the re- locatable program in a process called relocation.

The result of the final step is a file containing an executable binary image that is ready to run on the embedded system.

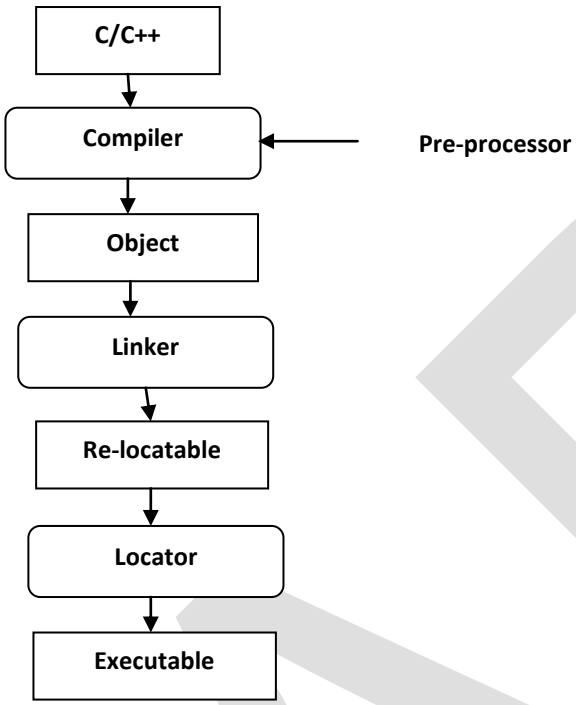


Figure: The Build Process Flowchart Chart

GCC (GNU compiler collection) is a compiler takes care of the compilation stages or builds process by calling appropriate programs. The compiler, assembler, linker and locator run on a host computer rather than on the embedded system itself. These tools combine their efforts to produce an executable binary image that will execute properly only on the target embedded system.

Pre-processing: A C program has the following pre-processor structural elements

1. *Include* directive for the file inclusion
2. *Definitions for preprocessor global variables* (global means throughout the program module)
3. *Definitions* of constants
4. *Declarations* for global data type, type declaration and data structures, macros and functions.

Preprocessor constants, variables and inclusion of configuration files, text files, header files and library functions are used in embedded C programs. A pre-processor directive starts with a sharp hash (#) sign.

Line begins with # symbol are called commands of pre-processor. GCC includes a pre-processor called CPP (C pre-processor) Example: *#define, #include* etc. during compilation, it is started with the pre-processing directives. The pre-processor is a separate program in reality but it is invoked by a compiler.

Example:

livesimple.c Program

```
#include<stdio.h>    //Preprocessor directives
int main()
{
        Printf("LIVESIMPLE!");
}
```

Both pre-processor directives and livesimple.c combined together to form livesimple.i file.

Syntax: [root@host ~] # CPP livesimple.c> livesimple.i

Compilation: Compiler uses the complete set of codes. It may also include codes, functions and expressions from the library routines. It creates a file called object file.

Compilation involves both gcc compiler and assembler. First, the gcc compiler converts the livesimple.i file into livesimple.s file. The file livesimple.s contains assembly code. The gcc conversion command is

Syntax: [root@host~]#gcc –s livesimple.i

The option – s tells the gcc compiler to convert the preprocessed code to assembly code without creating the object file.

Second, the assembler translates the livesimple.s file into machine language format and creating an object file called livesimple.o

Syntax: [root@host`]# as livesimple.s – o livesimple.o

The option – o converts the assembly file into object file.

Linking: A linker links the compiled codes of application software, object codes from library and OS kernel. Linking is necessary because there are number of codes to be linked for the final binary file. The linker file in the binary for run on a computer is commonly known as executable file or simply '.exe' file.

This process converts the object file into a new object file that is a special relocatable copy of the program. No memory addresses have been assigned to the

code until this process. An object file needs to be linked with many C run time library files, system functions etc., to form an executable object file. For example: in livesimple.c program, printf statement is needed. So printf.o file must be linked. The linker (Id) will perform all those tasks.

Syntax: [root@host~] # Id – dynamic-linker/

Locating: The tool that performs the conversion from relocatable program to executable binary image is called a locator. In embedded systems, the next step after linking is the use of locator for the program – codes and data in place of the loader. The features of locator are,

- The locator is specified by the programmer the available addresses at the RAM and ROM in target. The programmer has to define the available addresses to load and create file for permanently locating codes using a device programmer.
- It uses cross-assembler output, a memory allocation map and provides the locator program output file. It is the final step of software design process for the embedded system.
- The locator locates the I/O tasks and hardware device – driver codes at the addresses without reallocation. This is because the port and device addresses for these are fixed for a given system.
- The locator program reallocates the linked file and creates a file for permanent location of codes in a standard format.

Output:

Syntax : [root@host~] # ./ livesimple

The output is

LIVESIMPLE!

In windows, the executable file is denoted as livesimple.exe but there is no need of .exe extension file in Linux environment.

2. Discuss in detail about the structural units in embedded processor.

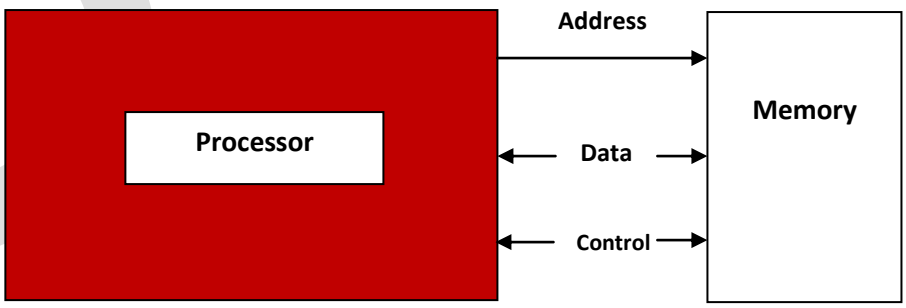


Figure: Bus Interface between Processor and Memory

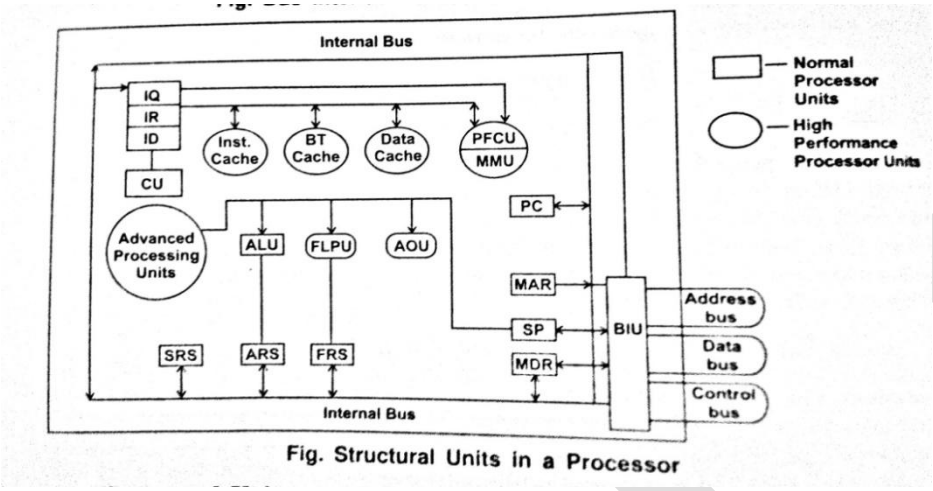


Fig. Structural Units in a Processor

STRUCTURAL UNIT IN A PROCESSOR:

1. **Memory Address Register (MAR):** It holds the address of the byte or word to be fetched from external memories. Processor issues the address of instruction or data to MAR before it initiates fetch cycle.
2. **Memory Data Register (MDR):** It holds a byte or word fetched (or to sent) from an external memory or IO address.
3. **System Buses:**
   a. *Internal Bus:* It internally connects all the structural units inside the processor. Its width can be 8, 18, 32, 48 or 64 bits.
   b. *Address Bus:* An external bus that carries the address from MAR to memory as well as to IO devices and other units of system.
   c. *Data Bus*: An external bus that carries, during a read or write operation, the bytes for instruction or data from or to an address. The address is determined by MAR.
   d. *Control Bus:* An external set of signals to carry control signals to processor or memory or device.
4. **Bus Interface Unit (BIU):** An interface unit between processor's internal units and external buses.
5. **Instruction Register (IR):** It sequentially takes instruction codes (opcode) to execution unit of processor.
6. **Instruction Decoder (ID):** It decodes the instruction received at the IR and passes it to processor CU.
7. **Control Unit (CU):** It controls all the bus activities and unit functions needed for processing.
8. **Application Register Set (ARS)**

a. A set of on chip registers used during processing of instructions of an application program.
b. A register window.
c. A subset of registers with each subset storing static variables of a software routine or
d. A register file associated to a unit such as ALU or FLPU.

9. **Arithmetic Logic Unit (ALU):** A unit to execute arithmetic or logic instructions according to the current instruction present at IR.
10. **Program Counter (PC):**

    It generates an instruction cycle by sending the address defined by it to memory through MAR. It auto increments as the instructions are fetched regularly and sequentially. It is called instruction pointer in 80 x 86 processors.
11. **Stack Pointer (SP):** A pointer for an address, which corresponds to a stack-top in memory.

ADVANCED PROCESSOR'S STRUCTURAL UNITS:

- Advanced processor circuits consist of RISC architecture.
- It executes most of the instruction in single clock cycle by using multiple register sets, windows and files and by reducing the dependency on the external memory access for data.
- A RISC has only few addressing modes for arithmetic and logic instructions.
- It contains the floating point unit.
- Pipelining allows a processor to overlap the execution of several instructions, so that more instructions can be executed in the same period of time.
- Example: DSP processors TMS320C6000, Davinci processor, OMAP processor etc.

1. **Instruction Level Parallelism Unit (ILP):**

   For instruction level parallelism, the multistage pipeline processing, multiline superscalar processing and dual, quad or multicore processing speeds up the performance from one instruction per clock cycle.
2. **Instruction Queue (IQ):**

   It is a queue of instruction so that the IR does not have to wait for the next instruction after one has been processed.
3. **Prefetch Control Unit (PFCU):**

A unit that controls the fetching of data into the I and D. caches in advance from the memory units. The instructions and data are delivered when needed by the processor's execution units. The processor does not have to fetch data just before executing the instruction. Pre-fetching unit improves performance by fetching instruction and data in advance for processing.

4. **Instruction Cache (I-Cache):**

   It sequentially stores, like an instruction queue, the instruction in FIFO mode. It lets the processor execute instructions at great speed using PFCU compare to external system memories, which are accessed at relatively much slower speeds.

5. **Branch Target Cache (BT Cache):**

   It facilitates ready availability of the next instruction – set when a branch instruction like jump, loop or call is encountered. Its fetch unit foresees a branching instruction at the I-Cache.

6. **Data Cache (D-Cache):**

   It stores the pre-fetched data from external memory. A data cache generally holds both the key (address) and value (word) together at a location. It also stores write through data when so configured. Write through data means data from the execution unit that transfer through the cache to external memory addresses.

7. **Memory Management Unit (MMU):**

   It manages the memories such that the instructions and data are readily available for processing.

8. **System Register Set (SRS):**

   It is a set of registers used while processing the instructions of the supervisory system program.

9. **Floating Point Processing Unit (FLPU):**

   A unit separate from ALU for floating point processing, which is essential in processing mathematical function fast in a microprocessor or DSP.

10. **Floating Point Register set (FRS):**

    A register set dedicated for storing floating point numbers in a standard format and used by FLPU for its data and stack.

11. **Multiply and Accumulate Unit (MAC):**

    There is also a MAC unit for multiplying coefficients of a series and accumulating these during computations.

12. Atomic Operation Unit (AOU):

It lets a user (compiler) instruction, when broken into a number of processor instruction called atomic operations, finish before an intercept of a process occurs. This prevents problems from arising out of shared data between various routines and tasks.

3. Write short note on (i) DMA (ii) Memory Management.

(i) DMA (DIRECT MEMORY ACCESS):

Definition: A direct memory access (DMA) is an operation in which data is copied from one resource to another resource in a computer system without the involvement of the CPU. The task of a DMA controller (DMAC) is to execute the copy operation of data from one resource location to another. The copy of data can be performed from:

- I/O device memory
- Memory to I/O device
- Memory to Memory
- I/O device to I/O device

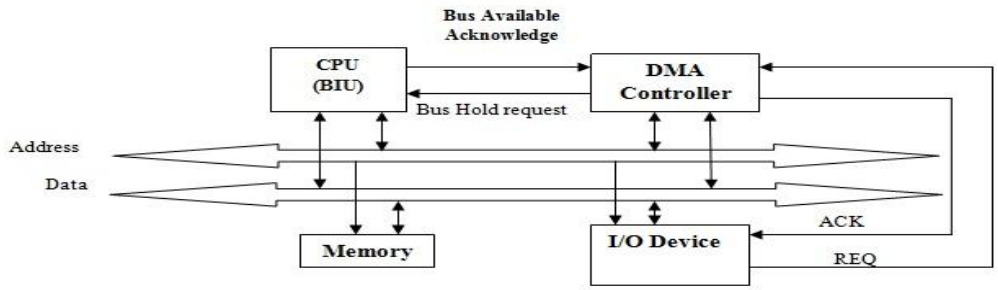


Figure: Logical Structure of System with DMA

DMAC is an independent resource of a computer added for the concurrent execution of DMA operations. The first two operation modes are 'read from' and 'write to' transfers of an I/O device to the main memory, which are the common operation of a DMA controller.

The DMAC replaces the CPU for the transfer task of data from I/O device to the main memory, otherwise it is executed by the CPU using the programmed input output mode (PIO). The '*memcopy*' is used for PIO operation. The DMAC is a master/slave resource on the system bus, because it supply the address for the resources involved in a DMA transfer. It requests the bus whenever a data is available for transport, which is signalled from the device by "REQ" signal.

DMA Operation: There are many modes of operation such as,

- Single block transfer (most commonly used)

- Chained block transfers
- Linked block transfers
- Fly- by transfers

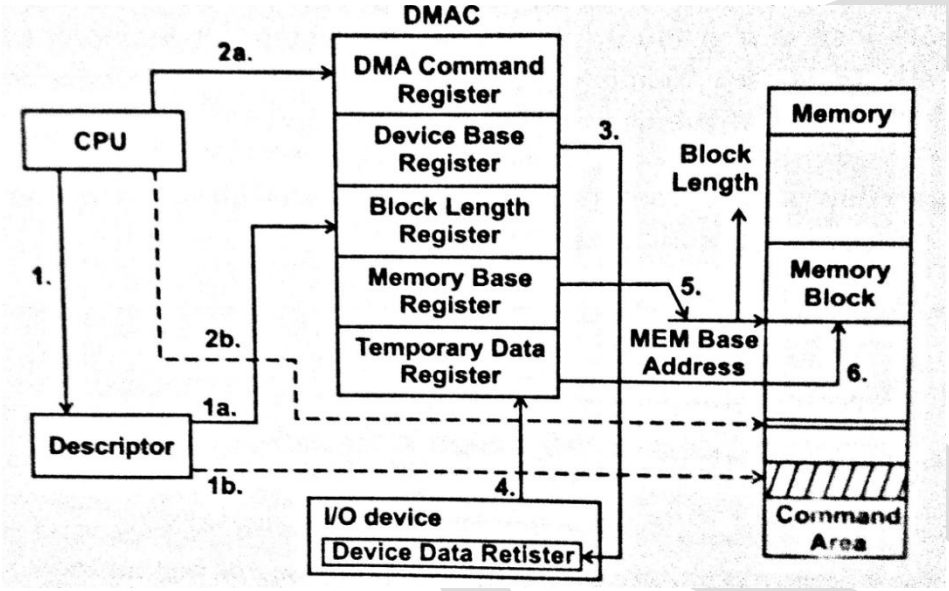


Figure: DMA Operation

Execution of a DMA – Operation (Single block Transfer)

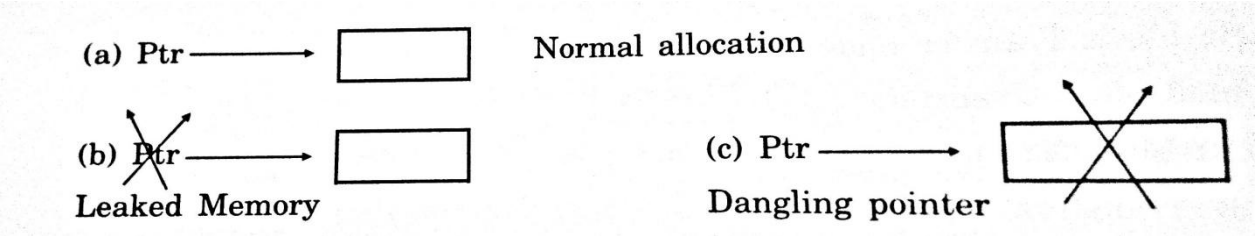
For DMA operation, the CPU prepares the construction of a descriptor (1). It contains all information for the DMAC to independently perform the DMA operation. It initializes the operation by writing a command to a register in a DMAC (2a) or to a command area, where the DMAC can poll for the command and/or the descriptor (2b). Then the DMAC addresses the device data register (3) and read the data into a temporary data register (4). In another bus transfer cycle, it addresses the memory block (5) and writes the data from the temporary data register to the memory block (6).

The DMAC increments the memory block address and continue with this loop until the block length is reached. The completion of the DMA operation is signalled to the processor by sending an IRQ signal.

(ii) MEMORY MANAGEMENT:

When a process is created, the memory manager allocates the memory addresses to it by mapping the process address space. Threads of a process share the memory space of the process.

Memory Leak: A memory allocation that does not have a corresponding de-allocation.



A memory leak is a gradual loss of available computer memory, when a program repeatedly fails to return memory that it has obtained for temporary use. Due to memory leak, run out of memory problem will occur.

Stack overflow means that the stack exceeds the allocated memory block when there is no provision for additional stack space. Memory manager controls the memory leaks and stack overflows.

Memory Management strategies are

1. *Fixed blocks allocation:* Memory address space is divided into blocks with processes of small address spaces getting a lesser number of blocks and processes of big address spaces getting a large number of blocks.
2. *Dynamic block allocation:* Memory manager allocates variable size blocks dynamically allocated from a free list of memory blocks description table at different phases of a process.
3. *Dynamic Page Allocation:* Memory has fixed sized blocks called pages and the memory management unit allocates the pages dynamically with a page descriptor table.
4. *Dynamic data memory allocation:* The manager allocates memory dynamically to different data structures like the nodes of a list, queues and stacks.
5. *Dynamic address relocation:* The manager dynamically allocates the addresses bound to the relative address. It adds the relative address to address with relocation register. This is also called run-time dynamic address binding.
6. *Multiprocessor Memory Allocation:* it adopts the tight coupling or loose coupling between two or more processors.
7. *Memory protection to OS functions:* It means that the system call and function call in user space are distinct. The memory of kernel functions is distinct and can be addressed only by the system calls. The memory space is called kernel space.
8. *Memory Protection among the tasks:* Read and write operations are not permitted to the particular memory space allocated to another task. This

protection increases the memory requirement for each task and also the execution time of the code of the task.

The Manager optimizes the memory needs and memory utilization. The memory manages the following:

1. Use of memory addresses space.
2. Mechanism to share memory space
3. Mechanism to restrict sharing of a given memory space.
4. Optimization of memory.

4. Write short note on (i) ICE (ii) Timer & Counter.

(i) IN – CIRCUIT EMULATOR (ICE):

In- circuit emulation is the use of a hardware device of in-circuit emulator used to debug the software of an embedded system. It operates by using – processor with the additional ability to support debugging operations as well as to carry out the main function of the system.

ICE is a computer chip that is used to emulate a microprocessor, so that embedded system software can be tested by developers. It allows a programmer to charge or debug the software in an embedded system. The ICEs gives the option of debugging by single stepping.

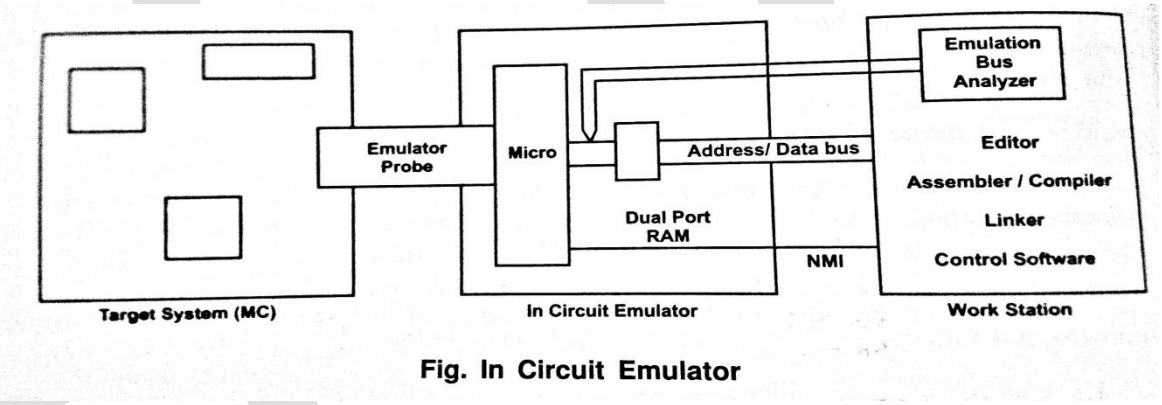


Fig. In Circuit Emulator

Working Principle:

- The programmer uses the emulator to load programs into the embedded system, run them, step through them slowly and view and change data used by the system's software. It imitates the central processing unit of the embedded system's computer.
- The house of Microchip offers in circuit emulators are of 3 types: MPLAB ICE 2000, MPLAB ICE 4000, REAL ICE.

- ICE consists of a small dual port pod. One port is a probe that plugs into the microprocessor socket of the target system. The second port is interfaced to a computer (or) workstation.

  Limitations of ICE:

  - Availability and cost
  - On chip functions
  - Transparency

  Features and capabilities of ICE are

  1. Ability to map resources between target and host.
  2. Ability to run and test code in real time without target hardware.
  3. Ability to step or run programs from/to specified states or break points.
  4. Ability to observe and modify microprocessor registers.
  5. Ability to Observe and modify memory contents.
  6. Ability to trace program execution using internal Logic Analyzers.

(ii) TIMERS AND COUNTERS:

Timer is a very common and useful peripheral. It is a device that counts the regular interval (δT) clock pulse at its input. The counts are stored and incremented on each pulse. It has output bits for the period of counts. The counts multiplied by interval δT gives the time.

Number of counting the interval x δT = Time

Timer is a programmable device, (i.e) the time period can be adjusted by writing specific bit patterns to some of the registers called timer-control registers.

A counter is more general version of the timer. It is a device that counts the input for events that may occur at irregular or regular intervals. The count gives the number of input events or pulses, since it was last read.

The simple timer has a 16 bit up counter which increments with each input clock pulse is shown in figure (a). Thus the output value 'Cnt' represents the number of pulses, since the counter was last reset to zero. An additional output 'top' indicates when the terminal count has been reached. It may go high for a predetermined time as set by the programmable control word inside the timer unit. The count can be loaded by the external program.

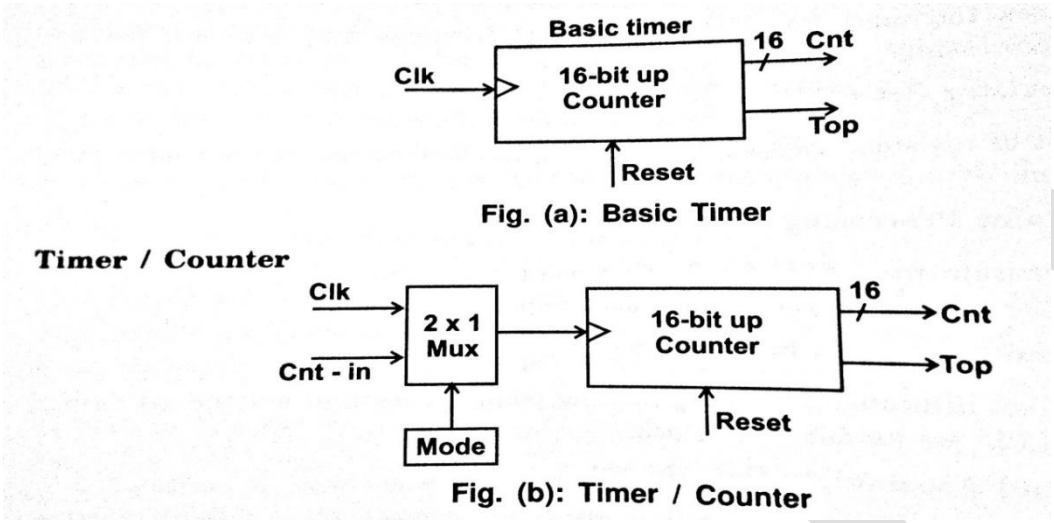


Fig. (a): Basic Timer

Fig. (b): Timer / Counter

The figure (b) provides the structure of another timer where a multiplexer is used to choose between an internal clock and external clock. The mode bit when set or reset decided the selection. For internal clock (Clk) it behaves like the timer in Figure (a). For the external coun- in (Cnt-in) it just counts the number of occurrences.

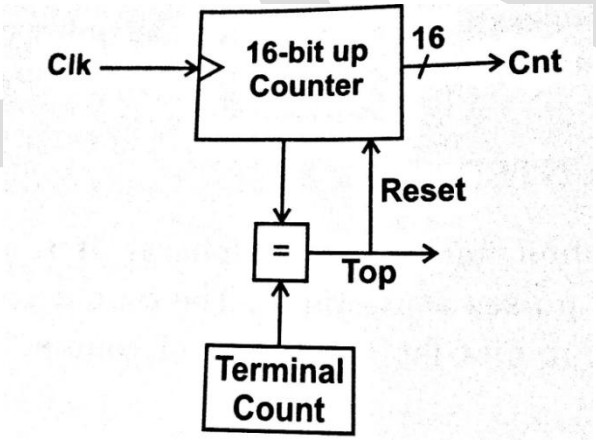


Figure (c): Timer with a terminal count

The figure (c) shows a timer with the terminal count. This can generate an event if a particular interval of time has been elapsed. The counter restarts after every terminal count. For example, in 8051 microcontroller, there is a presence of two bit timer/counter registers, Timer 0 and Timer 1.

In timer mode, register is incremented after every machine cycle.

1 machine cycle = 12 Oscillator periods

So count rate = 1/12 of Oscillator frequency.

If crystal frequency is 12 MHz, then timer clock frequency is 1 MHz.

In counter mode, Register is incremented in response to 1 to 0 transition at the corresponding external input pin T_0 and T_1.

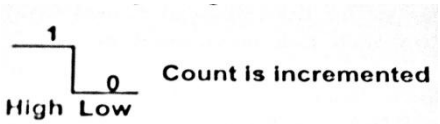


Fig. Count is Incremented

Blind Counting Synchronisation:

A blind counting free running counter with prescaling, compare and capture registers has a number of applications. It is useful for action or initiating a chain of actions and processor interrupts at the present instances as well as for noting the instances of occurrences of the events and processor intercepts for requesting the processor to use the captured counts on the events for future actions.

Watch dog Timer:

It is a timing device that resets the system after a predefined time out. It starts after a few clock cycles from power up. After restart, system works normally, if any failure occurs in past.

A watch dog timer is a piece of hardware that can be used to automatically detect software abnormalities and reset the processor if any. It is based on the counter that counts down from some initial value to zero.

The 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.

The process of restarting the watch dog timer's counter is sometimes called **"Kicking the dog"**.

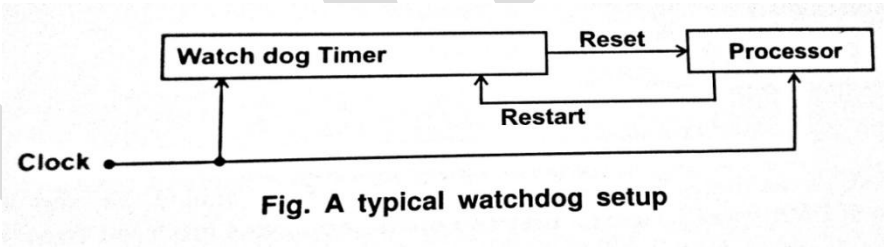


Fig. A typical watchdog setup

Real Time Clock (RTC):

It is a clock that keeps track of the time even when computer is turned off. Real time clocks (RTC) run on a special battery that is not connected to the normal power supply. In contrast, clocks that are not real time do not function when the computer is off. Do not confuse a computer's real time clock with its CPU clock. The CPU clock regulates the execution of instructions.

Real time clock provides system clock and it has a number of applications. It is a clock that generates system interrupts at preset intervals. An Interrupt Service Routine (ISR) executes on each tick or timeout or overflow of this clock. Once the device

started, it never resets or never reloaded to another value. Example: DS1307 chip is a real time clock integrated circuit.

Consider the block diagram shown below. The Arduino UNO is used for reading time from DS1307 and display it on 16X2 LCD. DS1307 sends time/data using 2 lines to Arduino. A buzzer is also used for alarm indication, which beeps when alarm is activated.

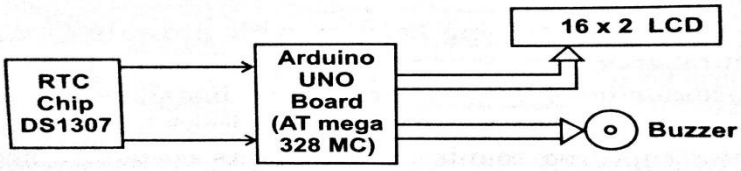


Fig. Interfacing RTC with Arduino UNO Microcontroller board

5. How to select the processor based upon its architecture and applications?

PROCESSOR:

A processor is the heart of the embedded system. For an embedded system designer, knowledge of microprocessors and microcontrollers is a prerequisite.

PROCESSOR IN A SYSTEM:

A processor has 2 essential units.

1. Program flow Control Unit (CU)
2. Execution Unit (EU)

The CU includes a fetch unit for fetching instructions from the memory. The EU has circuits that implement the instructions pertaining to data transfer operations and data conversions from one form to another. The EU includes the Arithmetic and Logic Unit (ALU) and also the circuits that execute instructions for a program control task, say halt, interrupt or jump to another set of instructions. It can also execute instructions for a call or branch to another program and for a call to a function.

A processor runs the cycles of fetch and execute. The instructions defined in the processor instruction set are executed in the sequence that they are fetched from the memory. An embedded system processor chip can be one of the following:

1. General Purpose Processor (GPP)
   - Microprocessor
   - Microcontroller
   - Embedded Processor
   - Digital Signal Processor (DSP)
   - Media Processor
2. Application Specific System Processor (ASSP) as additional processor.

3. Multiprocessor System using General Purpose Processor (GPP) and Application Specific Instruction Processor (ASIP)
4. GPP cores or ASIP cores integrated into an Application Specific Integrated circuit (ASIC), a Very Large Scale Integrated Circuit (VLSI) circuit, or an FPGA core integrated with processor units in a VLSI (ASIC) chip.

SELECTION OF PROCESSOR:

Most embedded systems need some type of processor. For a system designer, the following are important considerations when selecting a processor.

1. Instruction set
2. Maximum bits in an operand
3. Clock frequency in MHz
4. Processing speed in Million Instructions Per Second (MIPS)
5. Processing ability to solve the complex algorithms used in meeting the deadlines for their processing.
6. Register – windows provides fast context switching in a multitasking system.
7. Power efficient embedded system requires a processor that has auto-shut down feature for its units and programmability for disabling use of caches when the processing need for a function or instruction set does not have constraint of execution time. It is also required to have stop, sleep, and wait instructions. It may alsi require special cache design
8. Burst mode accesses external memories fast, reads fast and writes fast.
9. Atomic operation unit provides hardware solution to shared data problem, when designing embedded software or else special programming skills and efforts are to be made when sharing the variables among the multiple tasks.
10. A processor may also be configured at the initial program stage big-endian or Little endian storage of words. Eg: ARM processors.
11. The strong ARM family processors from Intel and Tiger SHARC from Analog devices have high power efficiency features.

SELECTION OF MICROCONTROLLER VERSION:

A version and microcontroller is selected for embedded system design as per the application as well as its cost.

1. Embedded system in an automobile requires CAN controller and CAN transceiver.

2. 8051 has only 2 timers Times 0 & Times 1, but 8052 has an additional timer – Timer 2, which is used for the development of Real Time Operating System.
3. Some Microcontrollers support master slave mechanism with the features of I2C and SPI supported in-built pins.
4. 8051 family member 83C152JA has two direct memory access (DMA) channels on-chip.
5. For interfacing more number of devices, we need more pins in microcontrollers to develop a particular application.
6. 80196KC has a PTS (Peripheral Transactions Server) that supports DMA functions.

MICROPROCESSORS:

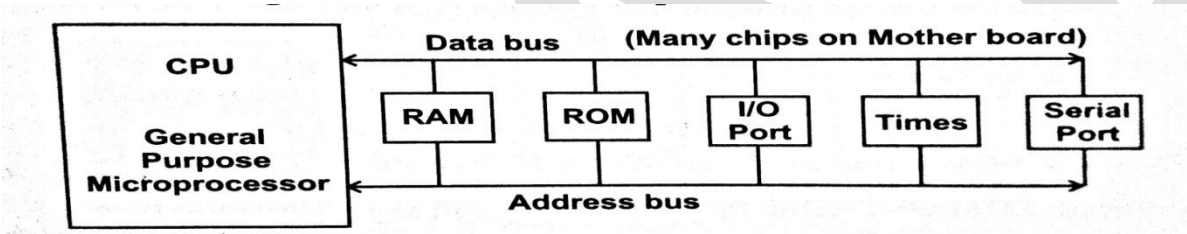
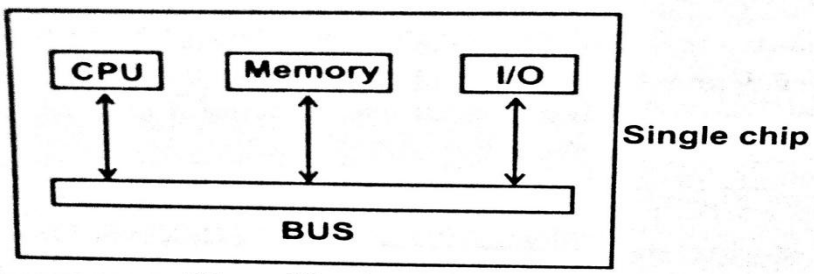


Fig. Microprocessor

Microprocessor is a single VLSI chip that has a CPU and may have some other units. Example: caches, floating point processing, arithmetic unit, pipelining etc. RAM, ROM and I/O units will not present within a CPU chip itself**.** Microprocessor accepts binary data as input & process data according to the instructions given and provides results as output. CPU has two basic functional units such as Control unit and Arithmetic Logic Unit (ALU). Example: 8085, 8086.

MICROCONTROLLER:



All on Single chip

Fig. Microcontroller

Microcontroller is a device, which integrates a number of the components of a microprocessor system onto a single microchip. It combines the CPU core, memory (ROM & RAM) and some parallel digital I/O ports onto the same microchip. Program memory is ROM and Data Memory is RAM. Example: 8051, PIC18F, PIC19F.

A microcontroller is a single chip VLSI unit, through which having limited computational capabilities, possesses enhanced input- output capabilities and a number of on-chip functional units.

MECHANISM IN MICROCONTROLLERS:

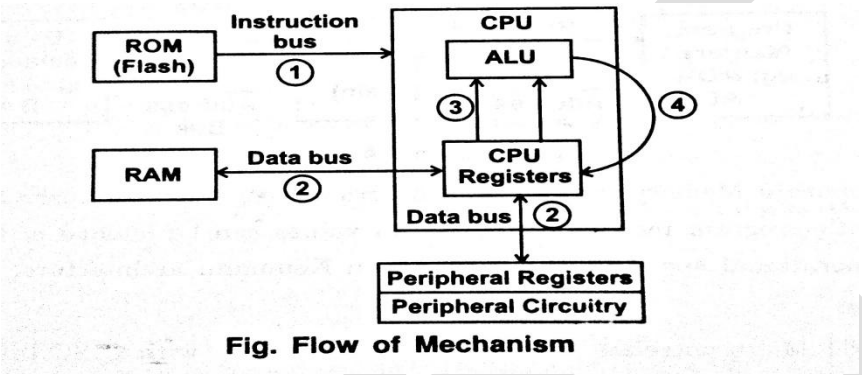


Fig. Flow of Mechanism

STEPS:

1. CPU gets the Instruction (MOV,MVI) from ROM.
2. CPU also gets the data (A=5, B=6) from RAM or from Peripheral Registers.
3. Now CPU registers are having the data and send data to the ALU unit to perform the mathematical or logical operation.
4. Finally the results are returned back to the CPU registers. In the turn CPU registers send the data to RAM or to peripheral devices.

PROCESSOR ARCHITECTURES:

A processor is the logic circuitry that responds to and processes the basic instructions that drive a system. The term processor has generally replaced the term central processing unit. The processor in a personal computer or embedded in small devices is often called a microprocessor. The presence of Microprocessor and Memory within a single chip is called as Microcontroller. The various processor architectures are

1. Von Neumann Architecture
2. Harvard Architecture
3. Super Harvard Architecture
4. Layered Architecture

VON NEUMANN ARCHITECTURE

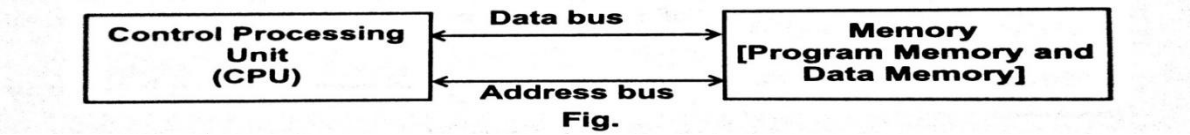


Fig.

Fig: Von Neumann Architecture

Single memory and single bus for transferring the data into and out of the CPU. Multiplication of two numbers require atleast three clock cycles.

HARVARD ARCHITECTURE

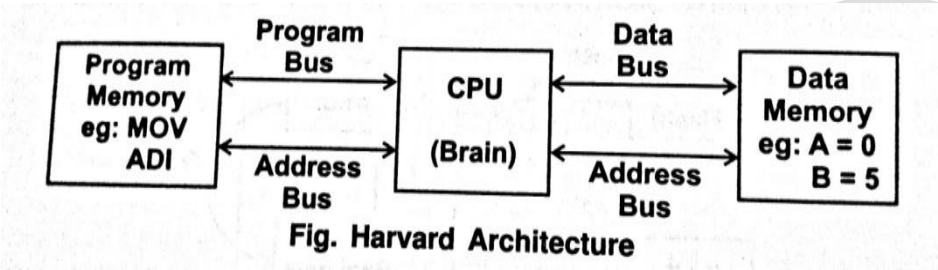


Fig. Harvard Architecture

Seperate memory for data and program with separate buses for each. Both program instructions and data values can be fetched at the same time. Operational speed is higher than Von Neumann architecture.

Examples:

8051 Microcontroller is a Harvard Architecture with CISC Instruction Set

PIC Microcontroller is a Harvard Achitecture with RISC Instruction Set

SUPER HARVARD ARCHITECTURE (DSP Processors)

DSP algorithms spend their most of the time in loops. So instruction cache is added. DSP processors are capable of processing many high frequency signals in real time. The significant feature present in the DSP processor is Multiply – Accumulate operation

Example: a <- a+(B X C)

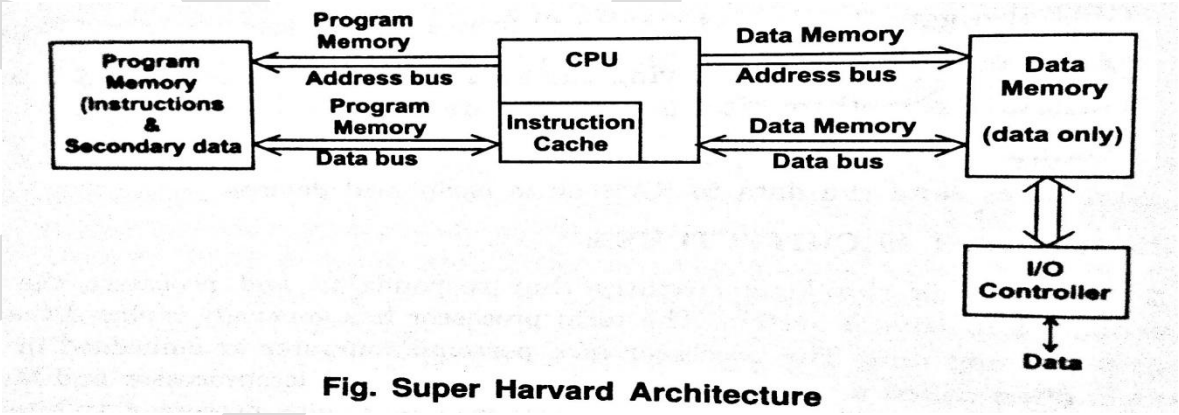


Fig. Super Harvard Architecture

By using this feature this mathematical part will performed in a single clock cycle. Applications performed in this processor are fixed point & floating point operations, matrix operations, convolution, correlation, parallelism etc.

Examples: C5000, C6000 single core processor families are manufactured by Texas Instruments.

C6000 processor fetch eight 32 bit instructions per clock cycle.

ARCHITECTURE OF EMBEDDED SYSTEMS:

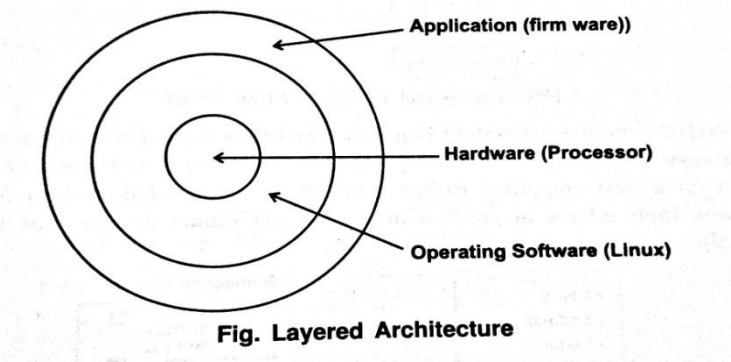


Fig. Layered Architecture

The combination of hardware, operating software and application software leads to the very effective embedded systems.

Case studies

Case-I : When the processor instruction cycle time is 1 microsecond

On chip devices and memory is sufficient. A microcontroller can be selected.

Examples : Automatic chocolate vending machine,56kbps modem, Robots, Data acquisition systems like an ECG recorder, Weather recorder, Multipoint temperature and pressure recorder and realtime Robotic controller.

Case-II : When the processor instruction cycle time is 10 to 40 nanosecond

On chip devices and memory is not sufficient. A microprocessor is required.

Examples : 52Mbps router, image processing, voice data acquisition,voice and video compression and adaptive cruise control.

Case-III : When the processor instruction cycle time is 5 to 10 nanosecond

High MIPS(Million Instruction per second) or MFLOP (Million Floating Point IPS) multi processor system is required,

Examples : Multi port 100Mbps network transceiver, fast 100 Mbps switches, routers, multichannel fast encryptions and decryptions system.

Case-IV : When the processor instruction cycle time is 1 nanosecond

Microprocessor and DSP based multiprocessor system is needed.

Examples : Voice and video processing, real time audio and video processing and mobile phone systems.

6. Discuss in detail about Target hardware Debugging.

Most of these debugging techniques can be applied to any microcontroller since they do not use any specific tools.

There are many ways to debug hardware, they are,

1. ICE (In Circuit Emulator)
2. ICD (In Circuit Debug)
3. Simulation
4. Serial RS-232
5. LCD
6. LED
7. Hardware pins
8. Logic Analyzer

ICE (In Circuit Emulator): In Circuit Emulator is the most expensive way to debug your hardware. It takes a special processor that physically takes place of the normal processor. This special processor allows software access to the internal operation of the processor.

ICD (In Circuit Debug): The next best thing to ICE is ICD (In Circuit Debug). This is for PIC microcontroller. BDM (Background Debug Mode) is for non PIC microcontroller. For ICD the processor has a small amount of built-in hardware that can halt the processor when the program reaches a specific address. The software can then read back all the registers and processor state.

Simulation: With a source code simulator you can step through the high level language code and see its effect on memory and variables without having to look at the assembler code directly. This let you focus on high level language operation and let you concentrate on the problem you are trying to solve. One great advantage of simulator is that you do not have to wait to download and program the target processor. So you can cut out the time consuming programming just by using the simulator.

Serial RS232: New microcontrollers have a built-in UART which gives virtually free debug tool that uses minimum resources and need very little software coding. For debug output you need to connect the UART output in TX to a suitable level translator circuit a MAX 232 chip. You may even get away with direct connection to the input of your PC serial port – using a translator chip will always work.

Advantages: Minimal coding, simple to use, minimal extra hardware.

Disadvantage: a) Takes long time to output a character (~1ms)

b) Takes even longer for blocks of characters (~10sf ms)

c) Need extra hardware.

Even though it takes time to output a character, it is a useful debug tool as you can output the value of variable to see what the microcontroller is really doing.

LCD: An LCD (Liquid crystal Display) gives a convenient way of displaying debugging information. It is also useful for many different applications that need a text display output. It is a module that displays text characters and a common screen size is 2 rows of 16 characters. Most LCD modules use the HD44780 controller chip which is why LCD routines built into high level language always work.

Advantage: a) Very quick update (40 µs 4 bit data bus)

b) Useful in many projects as the main display interface.

c) Simple to interface to an 8-bit port (only needs six of the 8-bits)

Disadvantage: a) Uses up an 8-bit port

b) Hardware is more expensive (Example: compared to a serial port chip)

LED: Using an LED as a microcontroller "alive" indicator. Even though it is such a simple thing to blink an LED on and off it is extremely useful as a debugging tool as you can tell at a glance whether the code you just downloaded is working sometimes you can will incorrectly set parameter on the programming software or compiler which will stop the code dead.

The LED indicator gives a quick health check for your microcontroller which is easy to see.

Pin Debugging: This is the simplest and rudest debugging method using any available port pin. Simply set and reset this pin at any point in the code that you want to monitor.

It has minimal impact on the code speed or size and can give you the following information.

- You can tell if the code is active.
- It gives you the repetition rate
- It gives you the routine time length (if you set the pin at the start and reset it at the end).

Logic Analyzer: This tool attaches to the pins you want to observe and captures the waveforms displaying multiple traces on a single display. It uses a trigger module that can be set to activate the combinations of the input signals or on their length. So you can trigger on specific patterns or on glitches or both.

For non-microcontroller based systems where the data and address bus are exposed, a logic analyzer can show the address and data organised into hex words i.e. readable. Some can disassemble the instruction showing what the processor was doing at the trigger point.

For a microcontroller based system the logic analyzer can be useful in examining peripheral operation. Example: for debugging the SPI or I^2C buses some logic analyzers also have built in support for these protocols.

Another use of the logic analyzer is to capture output over a long period of time depending on the memory capacity of the logic analyzer.