Sub: Embedded System (KOE-062)

Notes: Unit-1

Topic

1.1. Embedded Systems – Introduction to Embedded System.

1.2. The Build Process for Embedded System.

1.3. Structural units in Embedded Processor.

1.4. Selection of processor & Memory devices (ROM, RAM).

1.5. DMA.

1.6. Memory Management Methods-Timer and Counting devices.

1.7. Watchdog Timer.

1.8. Real time Clock.

1.9. In circuit emulator.

1.10. Hardware Debugging.

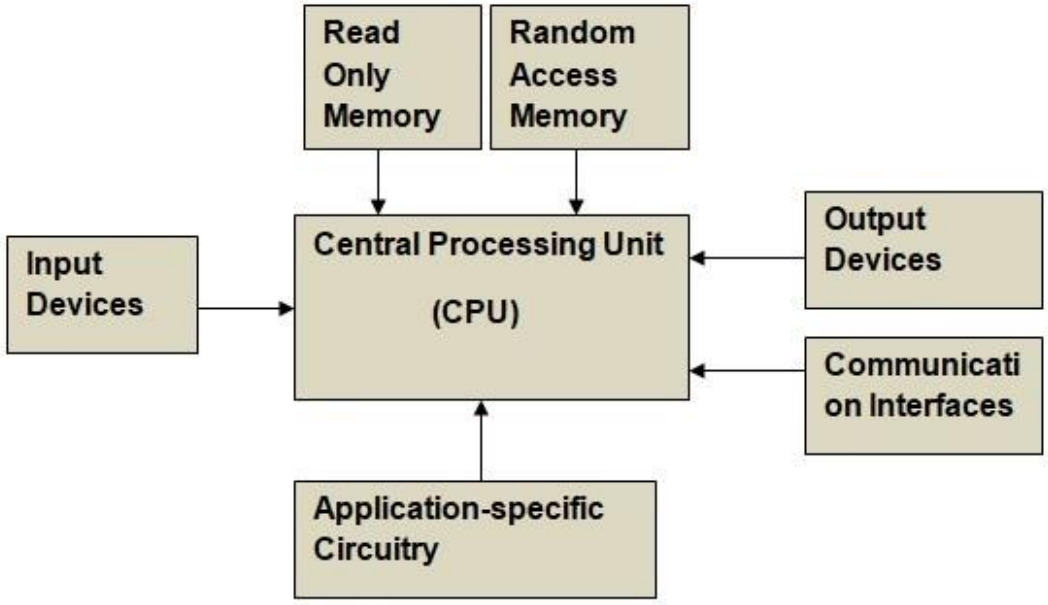
1.1. Embedded Systems – Introduction to Embedded System.

- **System** is a way of working, organizing or performing one or many tasks according to a fixed set of rules, program or plan.

  It is an arrangement in which all the unit combined to perform a work together by following certain set of rules in real time computation. It can also be defined as a way of working, organizing or doing one or many tasks according to a fixed plan.

Definition:

- An embedded system is an electronic/electro-mechanical system designed to perform a specific function and is combination of both hardware and firmware (software). The program instructions written for embedded systems are referred to as firmware, and are stored in Read-Only-Memory or Flash memory.
- Every ES is unique, and the hardware as well as software is highly specialized to the application domain. Embedded systems are designed to do some specific task, rather than be a general purpose computer for multiple tasks.



- **An embedded system is a combination of three major components:**
- **Hardware:** Hardware is physically used component that is physically connected with an embedded system. It comprises of microcontroller based integrated circuit, power supply, LCD display etc.
- **Application software:** Application software allows the user to perform varieties of application to be run on an embedded system by changing the code installed in an embedded system.

- **Real Time Operating system (RTOS):** RTOS supervises the way an embedded system work. It act as an interface between hardware and application software which supervises the application software and provide mechanism to let the processor run on the basis of scheduling for controlling the effect of latencies.

Characteristics of Embedded System:

1. Application and Domain Specific

- Embedded Systems do a very specific task; they can't be used for any other purpose. **Both hardware & software in an embedded system are optimized for the specific task.**
- For example, we can't replace the embedded control unit of microwave oven with air-conditioners embedded control unit.

2. Reliability

- Embedded Systems have to work for thousands of hours without break. This calls for very reliable hardware and software.
- For ex., if the embedded system comes to a halt state due to hardware error, the system should reset itself without any human intervention.
- Reliability is a measure of how much percentage we can rely upon the proper functioning of the system without failures.

3. Reactive

- Embedded systems are in constant interaction with the real world through sensors and user-defined input devices.
- Any changes happening in the Real world (called as an Event) are captured by the sensors/input devices and the control algorithm reacts in a designed manner to bring the controlled output variables to the desired level.
- Hence the Embedded systems are referred **as Reactive systems**, as they produce changes in output in **response** to the changes in the input.

4. Real-Time Performance

- Real Time System operation means the timing behavior of the system should be deterministic, meaning that the system should respond to requests in a known amount of time.
- In real-time embedded systems, a specific job has to be completed within a specific time. They have time constrains and they have to work against some deadlines. *(Examples of Real Time Systems – Missiles, flight control systems, nuclear plane safety system)*
- A Real Time System should not miss any deadlines for tasks (or) operations. For example, in a nuclear plant safety system, missing a deadline may cause loss of life / damage to property. Hence in hard real-time embedded systems which are subject to very strict deadlines for performing specific tasks, the timing analysis is of great importance.

5. Low Power consumption

- Most of the Embedded Systems operate through a battery. To reduce the battery drain and to avoid frequent recharging of the battery, the power consumption has to be very low.
- Hardware designers must address this issue – for example, by reducing the no. of components, by designing the processor to revert to low-power mode when there is no operation to perform.

6. Operates in harsh environments

- Some of the embedded systems are to be operated in harsh environment conditions like high temperature zone, areas subject to vibrations and shock etc. Systems placed in such areas should be capable to withstand all these adverse operating conditions.

General Purpose Computing system	Embedded system
It is combination of generic hardware and a general purpose OS for executing a variety of applications.	It is combination of special purpose hardware and software for executing specific set of applications
It contains general purpose operating system	It may (or) may not contain an operating system for functioning.
Applications are alterable (programmable) by the user.	Applications are non-alterable by the user.
Performance is the key deciding factor in the selection of the system. Always 'faster is better'.	Application specific Requirements (performance, power requirements, memory usage, size, design and manufacturing cost, etc) are the key deciding factors.
Less tailored towards reduced operating power requirements.	Highly tailored to take the advantage of the power saving modes supported by the hardware.
Response requirements are not time-critical.	For certain category of embedded systems like mission critical systems, the response time requirement is highly critical.
Need not be deterministic in execution behavior.	Execution behavior is deterministic for certain types of Embedded systems like 'Hard Real Time' systems.

Classification of Embedded System:

The classification of embedded system is based on following criteria:

(a) Based on generation
(b) Based on complexity
(c) Based on functionality & performance requirements

(a) Classification based on generation

1. First generation (1G):

- Built around 8-bit microprocessors like 8085, Z80.
- Simple in hardware circuit with firmware developed in Assembly code.
- *Examples: Digital telephone keypads, stepper motor control units*

2. Second generation (2G):

- Built around 16-bit µp and 8-bit µc.
- They are more complex & powerful than 1G µp & µc.

Examples: Data Acquisition system (DAS),

Supervisory Control and Data Acquisition systems (SCADA)

3. Third generation (3G):

- Built around 32-bit µp & 16-bit µc.
- A new concept of application and domain specific processors / controllers like Digital Signal Processors(DSPs), Application Specific Integrated Circuits(ASICs).

Examples: Robotics, Media, Industrial and process control, networking, etc.

4. Fourth generation:

- ✓ Built around 64-bit µp & 32-bit µc.
- ✓ The concept of System on Chips (SoC), reconfigurable processors, multi-core processors
- ✓ High performance, tight integration, miniaturization, and very powerful.

Examples: Smart Phones, Mobile Internet Devices

(b) Classification based on Complexity

1. Small-scale:

- Simple applications where the performance requirements are **not time-critical**.
- Built around low performance and low cost 8 or 16 bit µp/µc.

Example: an electronic toy

2. Medium-scale:

- Slightly complex in hardware and firmware requirement.
- Built around medium performance and low cost 16 or 32 bit µp/µc.
- Usually contain embedded operating system (RTOS) for functioning
  Examples: Industrial applications, Monitoring & Control of Manufacturing Equipment

3. Large-scale / complex:

- Highly complex hardware & firmware.
- Built around 32 or 64 bit RISC µp/µc (or) PLDs (or) SoC (or) multi-core processors.
- Response is time-critical.

Examples: Mission critical applications like Aircraft operating and control Systems, Electric power systems.

(c) Classification based on functionality and performance requirements

1. Stand alone embedded systems

- As the name implies, stand-alone systems work in stand-alone mode.
- They take inputs, process them and produce outputs.
- The inputs can be electrical signals from transducers / commands from human being such as pressing of a button. The outputs can be electrical signals to drive another system, an LED display / LCD display/DC motors.
- Many ES used in consumer electronics, automobiles, process control in manufacturing units fall into this category.

Examples: Digital Camera, Air Conditioner, Refrigerator, Microwave oven, DVD player

2. Real time systems

- Embedded systems in which some specific task has to be done in a specified period are called Real-Time systems.
- They have time constrains and they have to work against some deadlines. Meeting the deadlines is the most important requirement of Real-Time system.
- A Real Time System should not miss any deadlines for tasks (or) operations. For example, in a nuclear plant safety system, missing a deadline may cause loss of life /damage to property. Hence in hard real-time embedded systems which are subject to very strict deadlines for performing specific tasks, the timing analysis is of great importance.

  Ex: Missile embedded with a tracking system, flight control systems, nuclear plant safety system, and navigation system

3. Networked Information appliances

- Embedded systems that are provided with network interfaces and accessed by networks such as Local Area Network (LAN) or Internet are called as Networked Information appliances.
- Such embedded systems are connected to a network, typically a network running TCP/IP (Transmission Control Protocol/ Internet Protocol) protocol suite.
- They run the complete TCP/IP protocol stack and can communicate with other nodes on the network.

  Examples:

  ✓ ***A networked process control system*** *consists of a no. of embedded systems connected as a LAN. Each embedded system can send real-time data to a central location from where the entire process control system can be monitored.*

  ✓ ***A web camera*** *which is connected to internet camera can send pictures in real-time to any computer connected to internet. In such case, the web camera has to run the HTTP server software in addition to the TCP/IP protocol stack.*

  ✓ ***IoT applications*** *– the door lock of your home can be controlled from your desktop over the internet.*

4. Mobile devices

- Mobile devices such as mobile phones, Personal Digital Assistants (PDA), smart phones etc. are a special category of embedded systems.
- Mobile devices are capable of supporting high data rate services in addition to the voice services. Accessing internet services such as e-mail, World Wide Web and so on can be done while a person is on the move. They are capable of handling multimedia applications.
- The limitation of mobile devices – small size, lack of good user interfaces such as full-fledged keyboard and display, memory constrains, battery life time etc.

APPLICATIONS OF EMBEDDED SYSTEMS

The different applications/Examples of Embedded systems are given below

1. *Consumer Electronics*: Camera, Camcorders(Video capture and recording)
2. *Household appliances*: Digital TVs, DVD players, Set top boxes, Washing machine, Refrigerator.
3. *Automotive industry*: Anti-lock breaking system (ABS), engine control, automatic navigation system, engine control
4. *Home automation & security systems*: Air conditioners, sprinklers, fire alarms, CC cameras, home security system
5. *Telecom*: Cellular phones, telephone switches, handset multimedia applications
6. *Computer peripherals*: Printers, scanners, fax machines
7. *Computer networking systems*: Network routers, switches, hubs, firewalls
8. *Healthcare*: EEG, ECG machines.
9. *Banking & Retail*: Automatic teller machines (ATM), Currency counters
10. *Card Readers*: Barcode, smart card readers.

11. *Measurements & Instrumentation* : Logic Analyzers, Spectrum analyzers, PLC systems,

12. Missiles and Satellites : Defense, Aerospace, Communication, tracking system
13. Robotics : stepper motor controllers for a robotic system
14. Motor control systems : accurate control of speed and position of dc motor
15. Entertainment systems : video games, music system

- **Advantages** of Embedded System:
  Small size.
  Enhanced real-time performance.
  Easily customizable for a specific application.

- **Disadvantages** of Embedded System:

  High development cost.
  Time-consuming design process.
  As it is application-specific less market available.

1.2. BUILD PROCESS IN EMBEDDED SYSTEM

Definition: The process which converts source code to executable code is called as the build process.

The build process for embedded systems is different. This is because the code to be run on an embedded system is written one platform i.e. general purpose computer and executed on another platform i.e. the target hardware.

An Embedded system would also use tools such as a Compiler, Linker, Locater and Debugger to perform the entire build process. These tools would be a part of a larger IDE.

A compiler which produces the executable code to be run on a different platform is called a cross-compiler; else it is called a native compiler.

Ex. Turbo C++ is a native compiler. The compiler in case of embedded systems development is a cross compiler.

The build process involves three steps:

Compiling

Linking

Locating

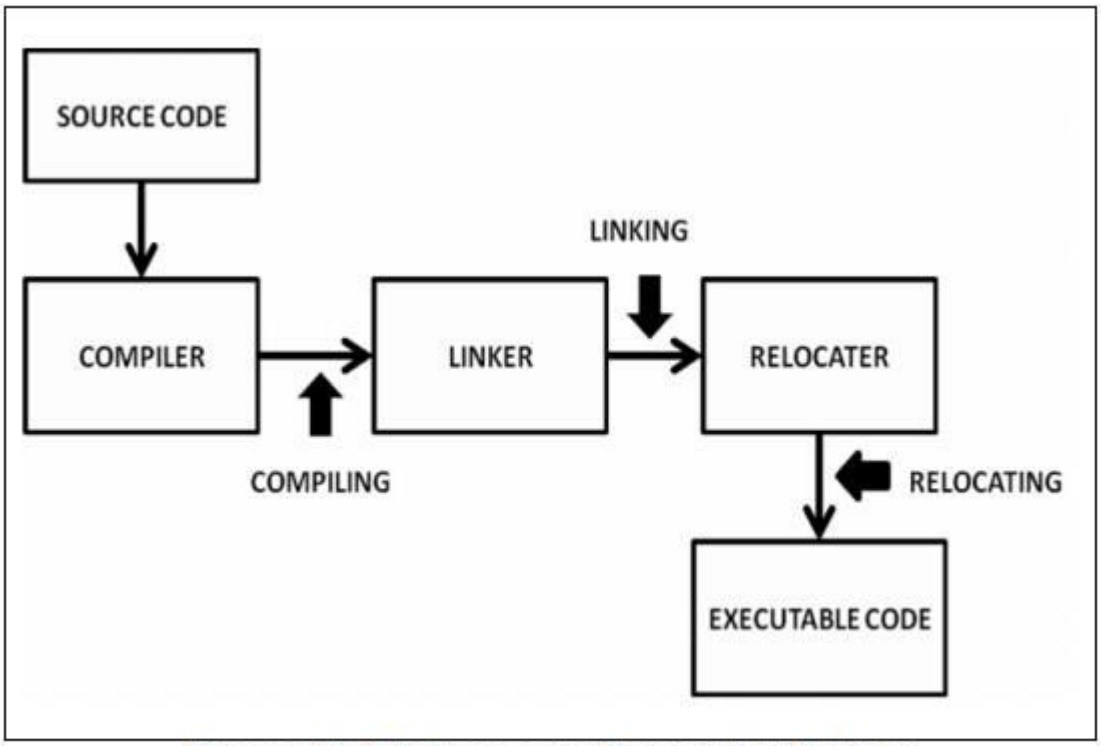


Figure: Build Process in Embedded System

Compiling

The process of compiling is done by the compiler.

The compiler takes input as source code files and gives output as multiple object files.

Compilers for embedded systems are essentially cross-compilers. For example while compiling the programmer has to select the target processor for which the code has to be generated.

The contents of the object files depend on its format. Two commonly used formats are:

1. Common Object file format (COFF)

2. Extended file format (ELF) o Object files generally have the following structure:

Linking

o The process of linking is carried out by the linker

o The linker takes input as multiple object files and gives output as a single object file which is also called as the relocatable code.

The output of compiler is multiple object files. These files are incomplete in the sense that they may contain reference to variables and functions across multiple object files which need to be resolved.

The job of the linker is to combine these multiple object files and resolve the unresolved symbols.

The Linker does this by merging the various sections like text, data, and bss of the individual object files. The output of the linker will be a single file which contains all of the machine language code from all of the input object files that will be in the text section of this new file, and all of the initialized and uninitialized variables will reside in the new data section and bss section respectively.

Locating

The process of relocating is carried out by the relocater.

The relocater takes input as the relocatable code produced by the linker and gives output as the final executable code.

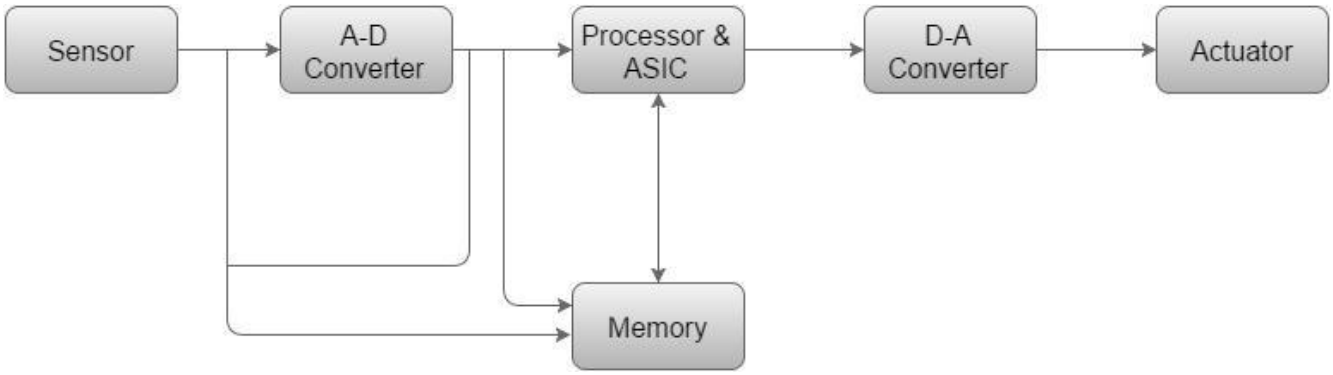
This output is a binary executable file which is called hex code.

The locator needs to be given information about the memory available on the target processor.

The locator will use this information to assign physical memory addresses to each of the code and data sections within the relocatable program code. Finally it produces an output file that contains a binary memory image that can be loaded into the target processors ROM.

1.3. Basic Structure in Embedded System

Thefollowingillustrationshowsthebasicstructureofanembeddedsystem−



- **Sensor**− It measures the physical quantity and converts it to an electrical signal which can be read by an observer or by any electronic instrument like an A2D converter. A sensor stores the measured quantity to the memory.

- **A-D Converter**− An analog-to-digital converter converts the

analog signal sent by the sensor into a digital signal.

- **Processor & ASICs**– Processors process the data to measure the output and store it to the memory.

- **D-A Converter**– A digital-to-analog converter converts the digital data fed by the processor to analog data

- **Actuator**– An actuator compares the output given by the D-A Converter to the actual (expected) output stored in it and stores the approved output.

1.4. Selection of processor & Memory devices (ROM, RAM).

Selection of processor:

Processor • should operate at higher clock speed for processing more instructions per second. • High computing performance of computing when there exist (a) Pipeline(s) and superscalar architectures, (b) pre-fetch cache unit, caches, and register-files and MMU and (c) RISC architecture. • Register-windows provides fast context switching in a multitasking system.

Power-efficient embedded system requires a processor that has auto-shut down feature for its units and programmability for the disabling these use of these 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 also require special cache design. Burst mode accesses external memories fast, reads fast and writes fast. • Atomic operation unit provides hardware solution to shared data problem when designing embedded software, else special programming skill and efforts are to be made when sharing the variables among the multiple tasks

Processor- specific features -

Big endian or little endian

Energy efficient

The selection of a processor is not an easy task for an embedded system. It is not possible to simply select a processor from your favourite list. Several factors are to be considered while selecting a processor for an embedded system. Because embedded systems are designed to operate in different climatic conditions, critical environments, etc.

Depending on the type of applications, the necessary performance, power constraints, special processing, modern technology, etc, the processor is selected. There are different types of embedded processors available in the market.

The selection of embedded processor should enhance the user experience by reducing the chance of poor speed, system overheating, insufficient memory, etc. The embedded system in the customer's hand should not make them, curse the designer. Instead, the system should make them proud of the technology.

Hence utmost care should be taken in the selection of embedded processors. In the below section, we have discussed some of the factors to be considered while selecting a processor for an embedded system.

TABLE OF CONTENTS

Speed and Performance

The most important factor to consider when choosing a processor for an embedded system is its performance. A processor's speed is primarily determined by its architecture and silicon design.

The number of instructions executed per second and the number of operations per clock cycle must be evaluated for assessing the performance. At the same time, the efficiency of the computation units is also important while talking about the performance.

The advancement of fabrication techniques enabled the packing of more transistors in the same area, reduces the propagation delay. Furthermore, the presence of a cache reduces the time required to fetch instructions/data.

Processor architectures that support additional instruction can aid in improving performance for specific applications. Pipelining and super-scalar architectures boost processor performance even further.

Other techniques for increasing execution rate include branch prediction, speculative execution, and so on. Multi-core processors are the new trend in performance enhancement. So the size of the cache, processor architecture, instruction set, etc has to be taken into account when comparing the performance.

Optimal Power usage

Increasing the logic density and clock speed have an adverse impact on the power requirement of the processor. Faster charging and discharging cycles in the capacitor, leakage currents may lead to more power consumption.

More logic leads to higher power density thereby making the heat dissipation difficult. With more emphasis on greener technologies and since many systems are becoming battery operated, it is important to design the system for optimal power usage.

Preemptions and context switching between different process will also cause more power consumption. Techniques like frequency scaling and voltage scaling can help in achieving lower power usage. Silicon-on-Chip (SoC) comes with advanced power gating techniques that can shut down clocks and power to unused modules.

Peripheral support

Apart from the processor, the embedded system has many other peripherals to perform input and output operations. It is important to have the right peripherals to assist the processor in optimized performance.

In recent days, almost all the processors used are SoCs. So it is better if the necessary peripherals are available in the chip itself and are called on-chip peripherals. It offers various benefits such as optimal power consumption and effective data communication compared to external peripherals

The Peripheral set located outside the SoC but on the same PCB is called off-chip peripherals or external peripherals. Timers, A/D and D/A converters, PWM controllers, serial communication controllers are some of the external peripherals used for processors.

Advanced Processing

Along with the core processor, the presence of various co-processors and specialized processing units can add more value to the processing performance. The instructions fetched by the core processor are executed by the co-processors in parallel, thereby reducing the processing load.

Some of the popular co-processors include Floating Point Co-processor, Graphics Processing Unit(GPU), Digital Signal Processors(DSP), etc. Floating Point co-processor can be very

helpful in applications involving complex mathematical operations including multimedia, imaging, codecs, etc.

GPU is responsible for rendering the images on a digital display. It is a parallel processing technology, which processes images faster than the normal unit. It is sometimes called a Visual processing unit.

GPUs are used to accelerate the 3D graphics in gaming applications, video and content creations, Artificial Intelligence and machine learning applications, etc. For example, Intel Iris Xe MAX is a GPU along with 11th Gen Intel Core processors, it provides advanced graphics performance and immersive game play.

DSP processors are designed specifically for signal processing applications. Its architecture supports the processing of multiple signals at a faster rate. It can manipulate real-time signals and is used in different applications like telecommunications, image processing, signal processing, navigation systems, etc.

Cost

For all the required functionalities to be built up in a system, certainly the price will be the determining factor during the selection of processor for an embedded system.

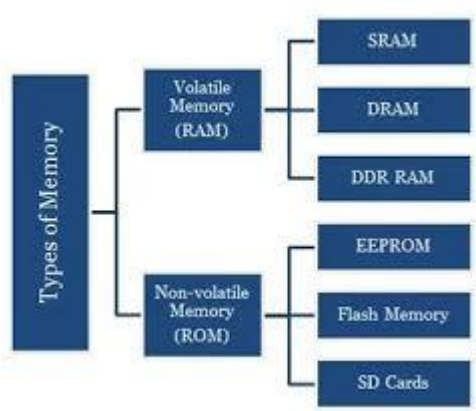
Memory devices (ROM, RAM)

Memory is the physical storage element or device used for storing two kinds of information.

1. Data or information
2. Program instructions or opcodes

The embedded system fetches the data from the memory, processes and produces output data, which is again stored in the memory. The stored data can be intermediate data that is produced during the execution.

Program information is nothing but the instructions or opcode that executes the function performed by the processor. When the program is executed, the CPU fetches the instruction from the memory and executes it.



Volatile Memory

Volatile memories will retain the data as long as the power is applied to the memory device. When the power is turned off, the contents will be erased from the memory.

It is used to store the data as well as the program instructions. Random Access Memory (RAM) is a type of volatile memory, which is divided into two types: SRAM and DRAM.

Random Access Memory (RAM)

RAM is the short-term memory that is accessed by the processor to execute all the applications. Hence, accessing the RAM is very fast. The embedded system cannot operate without the RAM.

The operation of RAM can be explained with a simple analogy –

Imagine your working table. To do simple mathematics, you will scribble on the paper nearby you without any delay. Once you have finished the work, you will throw the paper in the dustbin.

In the same way, RAM is used in the system.

In earlier days, there are three types of RAM: Static Random Access Memory(SRAM), Dynamic Random Access Memory(DRAM) and Synchronous Dynamic Random Access Memory(RAM).

Static Random Access Memory (SRAM)

It is a type of RAM that uses latching circuitry to store each bit, which is built with flip-flops. Hence the memories need not be refreshed. Thus the data stored will be retained till the duration of power being applied.

The Static Ram has high-speed operation as compared to DRAM and consumes less power. It is also used as cache memory. They are easier to use with low-end microcontrollers. The capacity of SRAM ranges from 1 bit to 256 Megabit. The major disadvantage is that the cost is comparatively higher for SRAM.

SRAMs are manufactured by top manufacturers such as Toshiba, IBM, Intel, Texas Instruments, Hitachi, etc.

Dynamic Random Access Memory (DRAM)

DRAM uses capacitors and transistors to store each bit. The capacitors have a charging and recharging cycle. In order to retain its content, periodic refreshing is to be done and hence called “Dynamic RAM”.

It consumes more power, which generates more heat. DRAM can be packed much denser than SRAM. DRAMs are available at an affordable price.

There are different types of DRAM such as SDRAM, DDR RAM, Graphics DRAM, Video DRAM, etc.,

Synchronous DRAM or SDRAM is a type of DRAM with improved performance. In this, the data transfer is established by synchronizing between the main memory and the microprocessor.

Double Data Rate RAM (DDR RAM) is the most common type available in recent days. It allows multiple data transfers at the same time. There are different versions of DDR RAM such as DDR1, DDR2, DDR3, DDR4 and DDR5. The latest DDR4 RAM has the highest speed of 25 GB per second. DDR5 RAM has a further highest speed of 50 GB per second.

When you buy a laptop or desktop computer system, the type and size of the RAM are the most important specifications. Nowadays, most laptops are equipped with DDR4 RAM. If it is a lower-end model, it is equipped with a size of 4 GB. For higher-end devices, 8 GB or 16 GB memory sizes are preferred.

Non-volatile memory

*Non-volatile memor*y will retain the content when the power applied to the memory device is turned off. Using this memory, the stored data can be retrieved after restarting the system.

The bootup configurations are typically stored in the non-volatile memory. They are slower than volatile memory but more information can be stored for a longer time.

There are different types of non-volatile memory available in the market. All the non-volatile memories can be classified under two categories: Electrically addressed and mechanically addressed.

EPROM, EEPROM, Flash memory belongs to Electrically addressed categories. Disk drive, magnetic tape comes under mechanically addressed categories.

Let us discuss some of the non-volatile memories.

EEPROM

PROM is a programmable Read Only memory made of logic gates. Data can be stored in the PROM by the programmer. It cannot be programmed once the information is written in the device.

EPROM is the Erasable PROM, consisting of floating gate transistors. The information can be stored, erased and also reprogrammed by the end-user. EPROM must be removed from the computer to erase its contents. The stored data can be erased by exposing it to the ultraviolet light source.

EEPROM or E^2PROM is an Electrically Erasable PROM. The data can be stored and erased by applying electrical signals. It is not necessary to remove the EEPROM from the computer

to erase its content. Large blocks of data can be erased at a time. However, it has a definite lifetime for erasing and reprogramming the data.

Flash memory

Flash memory is a non-volatile electronic memory used in embedded systems. It was invented by Toshiba in 1980, based on EEPROM technology. The stored data in the flash memory can be read much faster than the write operation.

It is an advanced memory technology used in various applications such as mobile phones, computers, PDA, cameras, etc. There are two types of flash memories: NAND and NOR flash memories. They are built from NAND and NOR logic gates.

In NAND type of flash memory, write and read operations are performed page-wise. Whereas the erasing operation is performed blockwise. The NAND memory is generally smaller in size and is used in USB flash drives. Error-correcting code like Hamming code is used in NAND type of flash memory to ensure normal operation.

The read operation in NOR flash memory is similar to that of random access memory. Hence, a faster read operation is possible with NOR than NAND flash memory. It has low storage capacity compared to NAND flash memory.

SD card

SD card or Secure Digital Card is a portable flash type non-volatile memory. It has high capacity and high access speed. Hence used for lot of consumer electronics like mobile phones, digital cmeras, etc.,

These cards have its own processor which handles the interface requirements and operations like error correction, etc., They are available in different sizes for diffeent applications. Different variations are available in SD cards such as

- Secure digital High Capacity(SDHC) card
- SD eXtended Capacity (SDXC) card
- SD Ultra Capacity (SDUC) card

So far, we have discussed only a few types of memory. But lot more types of memory are available in real practice such as Masked ROM, NVRAM, VRAM, GRAM, etc., It is always a challenging task to select a perfect memory for an embedded system.

Let us discuss the selection criteria for selecting memory for embedded systems.

Selection of Memory

The selection of suitable memory is a very essential step in designing an embedded system. The designers have to choose the best memory for their system. The selected memory device should reflect the goal of your embedded application.

The performance of the entire embedded system depends on the selection of memory. Improper selection of memory may lead to insufficient memory, slow speed operation, more power consumption, etc. Hence such challenges must be considered while designing an embedded system.

Here we have discussed some of the factors that should be considered while selecting the memory for an embedded system.

List of Factors

- **Speed**: The time to read or write the data should be greatest consideration while selecting the memory. In general, speed will not be a greater issue for small embedded applications. But when you go for medium or high range applications, read/write access time time should be faster.
- **Latency:** It is the time between initiating the request of data until it is received. When executing the processor instructions, it request the data stored in the memory. The requested data must be retrieved by the processor for quick operation. Less latency, then more speed operation.
- **Memory Capacity**: If your application need below 60 MB, it is advisable to choose the memory size as 64 MB. At the same time, running out of storage is the worst feeling we face with the digital camera. So it is important to choose the capacity as needed for the application.
- **Size:** The size of the memory device should be compatible withe the embedded system. For hand held devices, the size of the memory should be compact in nature. If it is a desktop computer system, the size can be of medium sized. Proper size selection is also an essential criteria while selecting a memory.
- **Power consumption**: Memory needs power to read or write data. For high access speed, the power consumptions will be more, results in more power dissipation. More heat will reduce the lifetime of the embedded system. Hence the designers should go with optimum power consumption memory devices.
- **Cost:** Cost plays a significant role in deciding any product. While planning to design an embedded system, similar importance should be given to memory as that of processor selection. Money should be allocated, considering the type of memory to be used in the embedded system.

Differences:

RAM	ROM
It is a Random-Access Memory.	It is a Read Only Memory.
Read and write operations can be performed.	Only Read operation can be performed.
Data can be lost in volatile memory when the power supply is turned off.	Data cannot be lost in non-volatile memory when the power supply is turned off.

It is a faster and expensive memory.	It is a slower and less expensive memory.
Storage data requires to be refreshed in RAM.	Storage data does not need to be refreshed in ROM.
The size of the chip is bigger than the ROM chip to store the data.	The size of the chip is smaller than the RAM chip to store the same amount of data.
Types of RAM: DRAM and SRAM	Types of ROM: MROM, PROM, EPROM, EEPROM

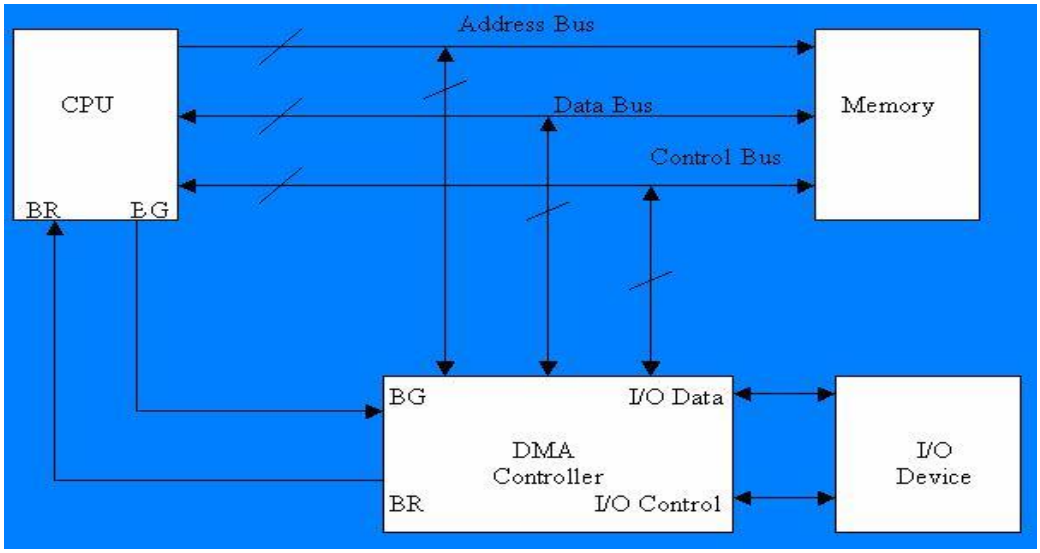
SRAM	**DRAM**
It is a Static Random-Access Memory.	It is a Dynamic Random Access Memory.
The access time of SRAM is slow.	The access time of DRAM is high.
It uses flip-flops to store each bit of information.	It uses a capacitor to store each bit of information.
It does not require periodic refreshing to preserve the information.	It requires periodically refreshing to preserve the information.
It uses in cache memory.	It is used in the main memory.
The cost of SRAM is expensive.	The cost of DRAM is less expensive.
It has a complex structure.	Its structure is simple.

Direct Memory Access (DMA)

Definition: A direct memory access (DMA) is an operation in which data is copied (transported) from one resource to another resource in a computer system without the involvement of the CPU.

Direct Memory Access is a capability provided by some computer bus architectures that allows data to be sent directly from an attached device (such as a disk drive) to the memory on the computer's motherboard. The microprocessor is freed from involvement with the data transfer, thus speeding up overall computer operation.

An alternative to DMA is the Programmed Input/output (PIO) interface in which all data transmitted between devices goes through the processor.



Fig(1) Computer system with DMA

Note//BG: bus grant signal and BR: bus ready signal

Advantages of DMA

-Fast memory transfer of data

-CPU and DMA run concurrently under cache mode.

-DMA can trigger an interrupt, which frees the CPU from polling the channel

Use of this mechanism can greatly increase throughput to and from a device.

DMA Controller

The DMA controller can issue commands to the memory that behave exactly like the commands issued by the CPU. The DMA controller in a sense is a second processor in the system but is dedicated to an I/O function. The DMA controller as shown below connects one or more I/O ports directly to memory, where the I/O data stream passes through the DMA controller faster and more efficiently than through the processor as the DMA channel is specialized to the data transfer task

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

- A DMAC is an independent (from CPU) resource of a computer system added for the concurrent execution of DMA-operations. The first two operation modes are ’read from’ and ’write to’ tran sfers of an I/O-device to the main memory, which are the common operation of a DMA-controller. The other two operations are slightly more difficult to implement and most DMA-controllers do not implement device to device transfers The DMAC replaces the CPU for the transfer task of data from the I/O-device to the main memory (or vice versa) which otherwise would have been executed by the CPU using the programmed input output (PIO) mode. PIO is realized by a small instruction sequence executedby the processor to copy data. The ’memcpy’ functio n supplied by the system is such a PIO operation.

The DMAC is a master/slave resource on the system bus, because it must supply the addresses for the resources being involved in a DMA transfer. It requests the bus whenever a data value is available for transport, which is signaled from the device by the REQ signal. The functional unit DMAC may be integrated into other functional units in a computer system, e.g. the memory controller, the south bridge, or directly into an I/O-device.

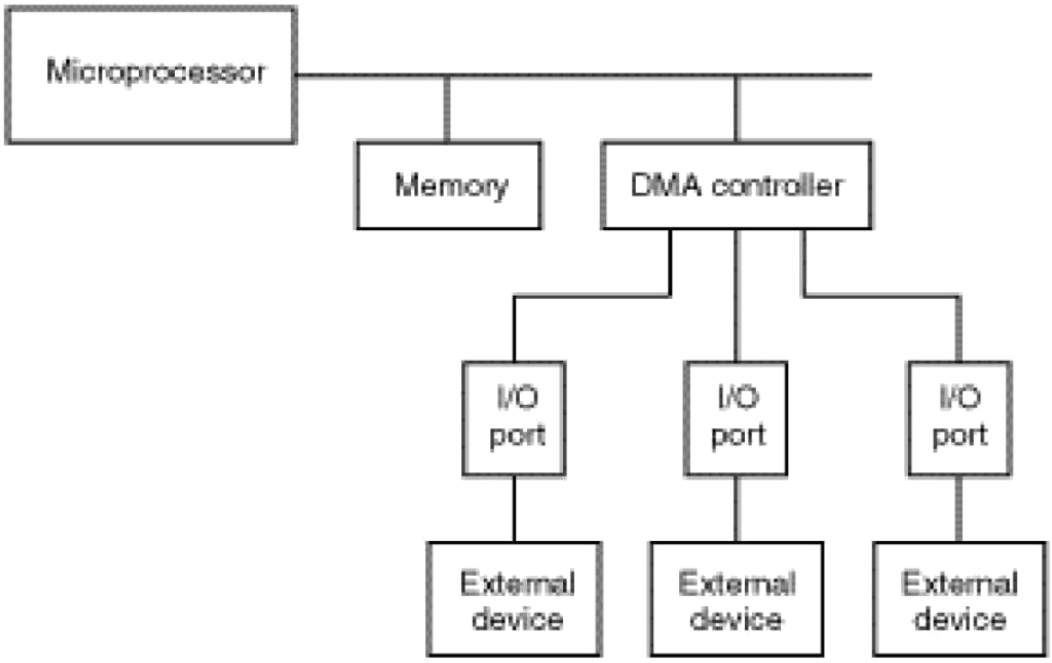


Fig (2): A microcomputer with a direct memory-access controller

The DMA controller includes several registers :-

- The DMA Address Register contains the memory address to be used in the data transfer. The CPU treats this signal as one or more output ports.

- The DMA Count Register, also called Word Count Register, contains the no. of bytes of data to be transferred. Like the DMA address register, it too is treated as an O/P port (with a diff. Address) by the CPU.

- The DMA Control Register accepts commands from the CPU. It is also treated as an O/P port by the CPU.

Although not shown in this fig., most DMA controllers also have a Status Register. This register supplies information to the CPU, which accesses it as an I/P port.

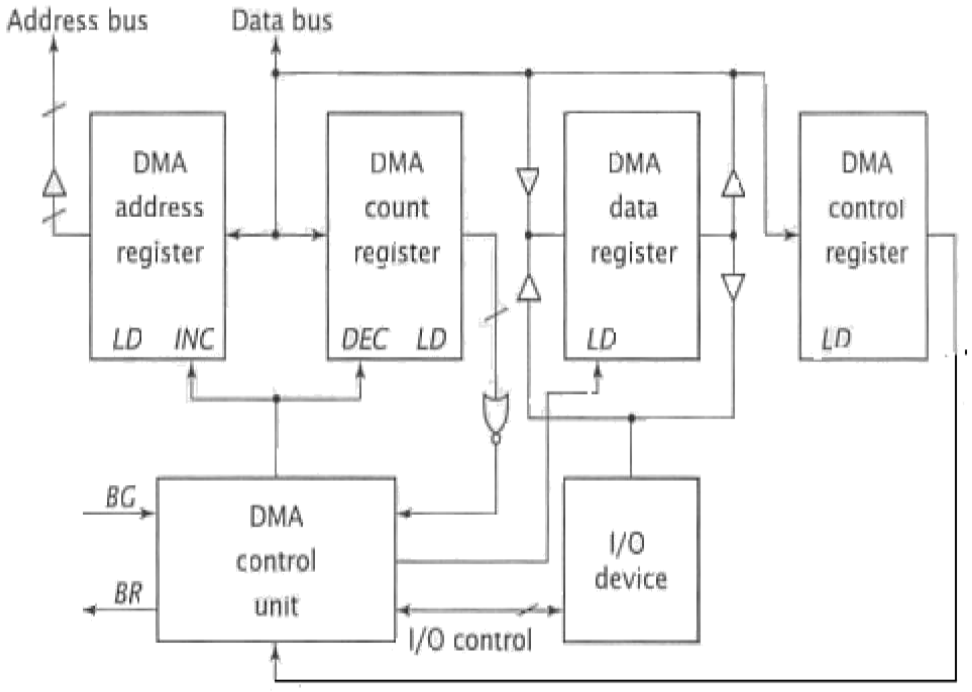


fig (3)Internal Configuration of DMA Controller

How DMA and CPU use the Buses

• DMA sends a signal called HOLD to the CPU

• The CPU will respond by sending back the signal HLDA (hold acknowledge) to indicate to the DMA that it can go ahead and use the buses.

• While the DMA is using the buses to transfer data , the CPU is sitting idle,

• And when the CPU is using the bus, the DMA is sitting idle.

After DMA finishes its job it will make HOLD go low and then the CPU will regain control over the buses

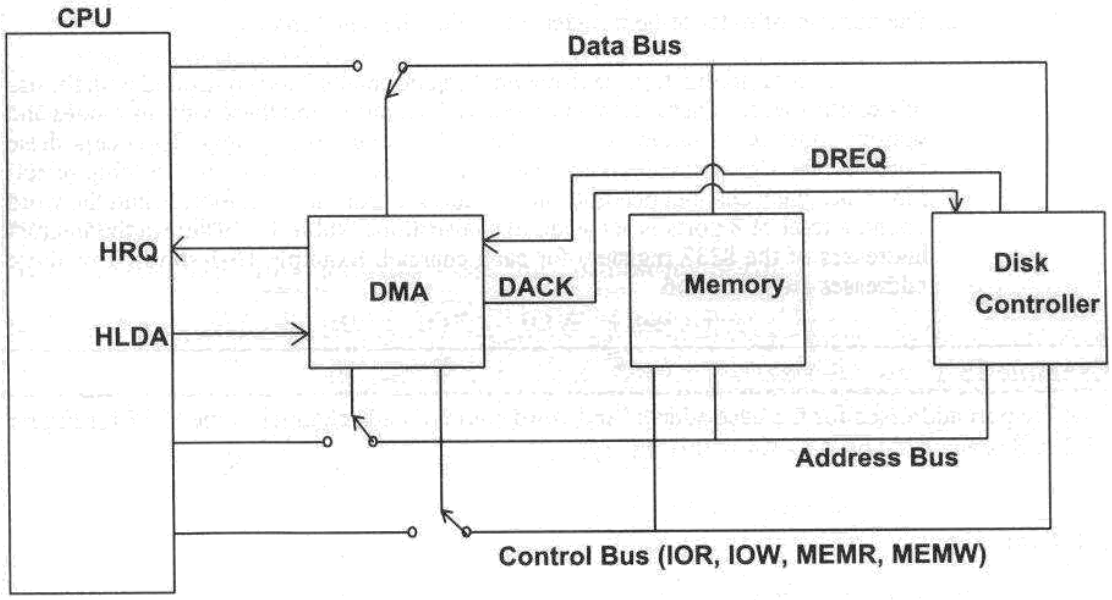


Fig (4) DMA Usage of System Bus

DMA will go through the following steps:

1. The peripheral device (such as the disk controller) will request the service of DMA by pulling DREQ (DMA request) high.

2. The DMA will put a high on its HRQ (hold request), signaling the CPU through its HOLD pin that it needs to use the buses.

3. The CPU will finish the present bus cycle and respond to the DMA request by putting high on its HLDA (hold acknowledge), thus telling the 8237 DMA that it can go ahead and use the buses to perform its task.

4. HOLD must remain active high as long as DMA is performing its task.

5. DMA will activate DACK (DMA acknowledge), which tells the peripheral device that it will start to transfer the data.

6. DMA starts to transfer the data from memory to peripheral as follows

• DMA puts the address of the first byte of the blo ck on the address bus

• Activates MEMR,

• Reads the byte from memory into the data bus

• Then activates IOW to write it to the peripheral.

Memory Management Methods- Timer and counting devices.

What is Memory Management?

In a multiprogramming computer, the Operating System resides in a part of memory, and the rest is used by multiple processes. The task of subdividing the memory among different processes is called Memory Management. Memory management is a method in the operating system to manage operations between main memory and disk during process execution. The main aim of memory management is to achieve efficient utilization of memory.

Why Memory Management is required?

- Allocate and de-allocate memory before and after process execution.
- To keep track of used memory space by processes.
- To minimize fragmentation issues.
- To proper utilization of main memory.
- To maintain data integrity while executing of process.

Now we are discussing the concept of Logical Address Space and Physical Address Space

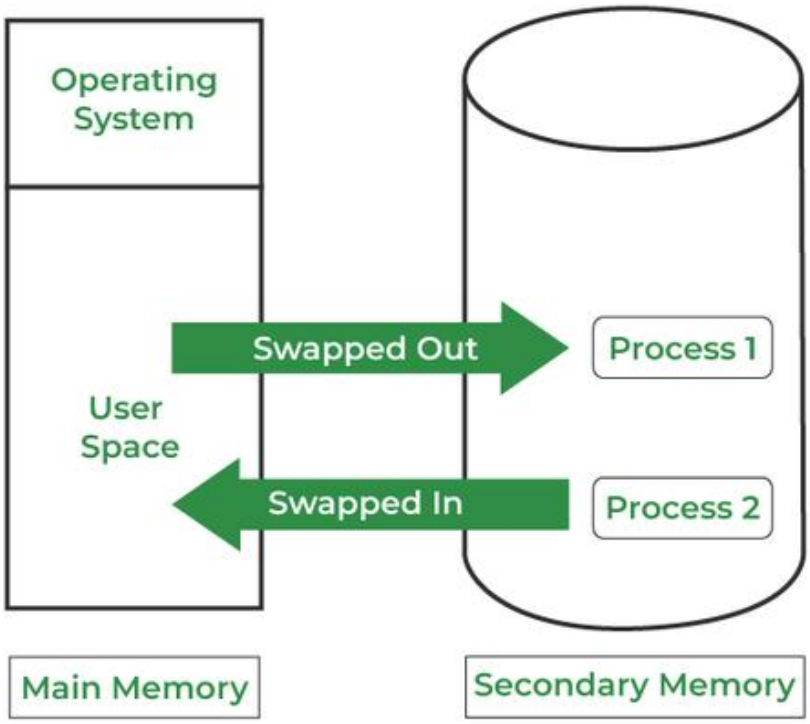
Logical and Physical Address Space

- **Logical Address Space:** An address generated by the CPU is known as a "Logical Address". It is also known as a Virtual address. Logical address space can be defined as the size of the process. A logical address can be changed.
- **Physical Address Space:** An address seen by the memory unit (i.e the one loaded into the memory address register of the memory) is commonly known as a "Physical Address". A Physical address is also known as a Real address. The set of all physical addresses corresponding to these logical addresses is known as Physical address space. A physical address is computed by MMU. The run-time mapping from virtual to physical addresses is done by a hardware device Memory Management Unit(MMU). The physical address always remains constant.

Swapping

When a process is executed it must have resided in memory. Swapping is a process of swapping a process temporarily into a secondary memory from the main memory, which is fast compared to secondary memory. A swapping allows more processes to be run and can be fit into memory at one time. The main part of swapping is transferred time and the total time is directly proportional to the amount of memory swapped. Swapping is also known as

roll-out, or roll because if a higher priority process arrives and wants service, the memory manager can swap out the lower priority process and then load and execute the higher priority process. After finishing higher priority work, the lower priority process swapped back in memory and continued to the execution process.



Swapping in memory management

A **timer** is a specialized type of clock which is used to measure time intervals. A timer that counts from zero upwards for

Measuring time elapsed is often called a **stopwatch**. It is a device

That counts down from a specified time interval and used to generate a time delay, for example, an hourglass is a timer.

A **counter** is a device that stores (and sometimes displays) the number of times a particular event or process occurred, with respect to a clock signal. It is used to count the events happening outside the microcontroller. In electronics, counters can be implemented quite easily using register-type circuits such as a flip-flop.

Difference between a Timer and a Counter

The points that differentiate a timer from a counter are as follows −

Timer	Counter
The register incremented for every machine cycle.	The register is incremented considering 1 to 0 transitions at its corresponding to an external input pin (T0, T1).
Maximum count rate is 1/12 of the oscillator frequency.	Maximum count rate is 1/24 of the oscillator frequency.
A timer uses the frequency of the internal clock, and generates delay.	A counter uses an external signal to count pulses.

Watchdog Timer;

Watchdog timer is a piece of hardware in micro-controller. **Watchdog timer is used to generates system reset if system gets stuck somewhere** i.e. if system goes into endless loop of execution watchdog timer will reset the system to come out of endless loop. Watchdog is safety mechanism in embedded system which makes your system reliable, but **it depends on how you make use of watchdog timer.**

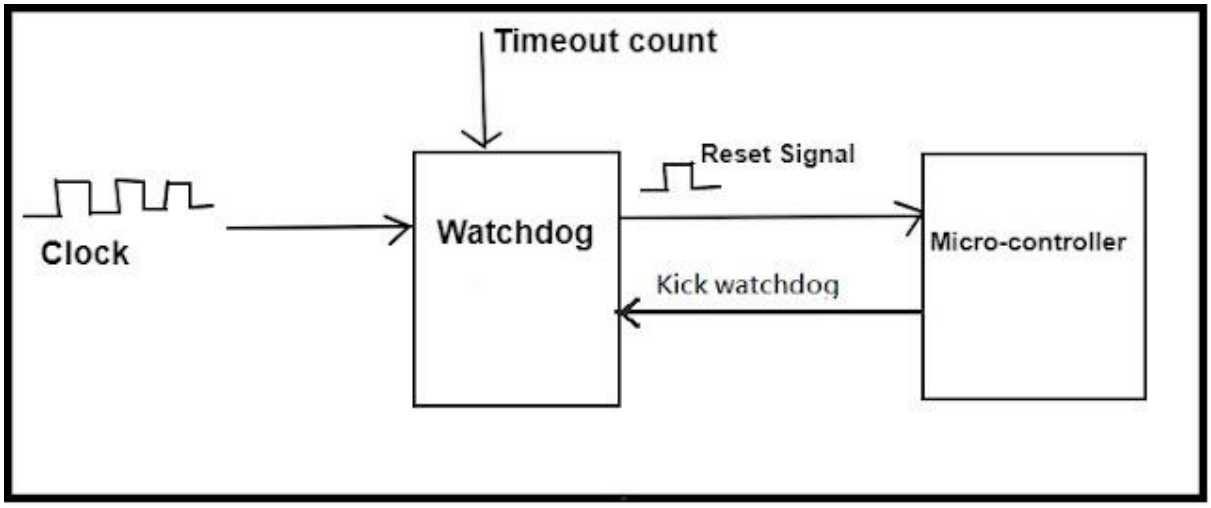


Fig. 1 Watchdog timer

Here's a brief overview of how a watchdog timer works:

Initialization: The watchdog timer is initialized and started when the embedded system is powered on or reset.

Timer Reset: The software periodically resets or "feeds" the watchdog timer to prevent it from timing out. This is typically done by writing a specific value or sequence to a specific register in the microcontroller.
Timer Expired: If the software fails to reset the watchdog timer within a certain predefined time interval, the timer will expire or "timeout."
System Reset: When the watchdog timer times out, it triggers a system reset, which restarts the microcontroller or the entire embedded system. This reset helps recover from potential system lock-ups, software crashes, or other faults.

How does watchdog works:

Watchdog is basically **a counter, which starts from counting zero** and reaches to a certain value. **If counter reaches to certain value then watchdog hardware will generates a watchdog reset. To avoid system reset, software needs to kick the watchdog** i.e. need to reset the counter to zero. In case software stuck into endless loop it system will not able to kick the watchdog hence counter reaches to certain value and resets the system.

Watchdog is initially loaded with certain value. This value is calculated based on timeout time of watchdog (**Further section it is been shown how to calculate counter value based on timeout value**). Before timeout time, system should reset the counter.

e.g. If your system if performing 3 tasks periodically and to perform 3 tasks is takes 500 ms. Then timeout time is considered as 600 ms (considering worst case scenario), counter value is calculated with respect to 600 ms and loaded into watchdog. Following figures shows watchdog hardware. Input to watchdog hardware is clock.
Based on every clock tick watchdog internal counter increments. Then there is comparator which compares count value with loaded count value (timeout value) and if count matches watchdog hardware generates and reset signal.

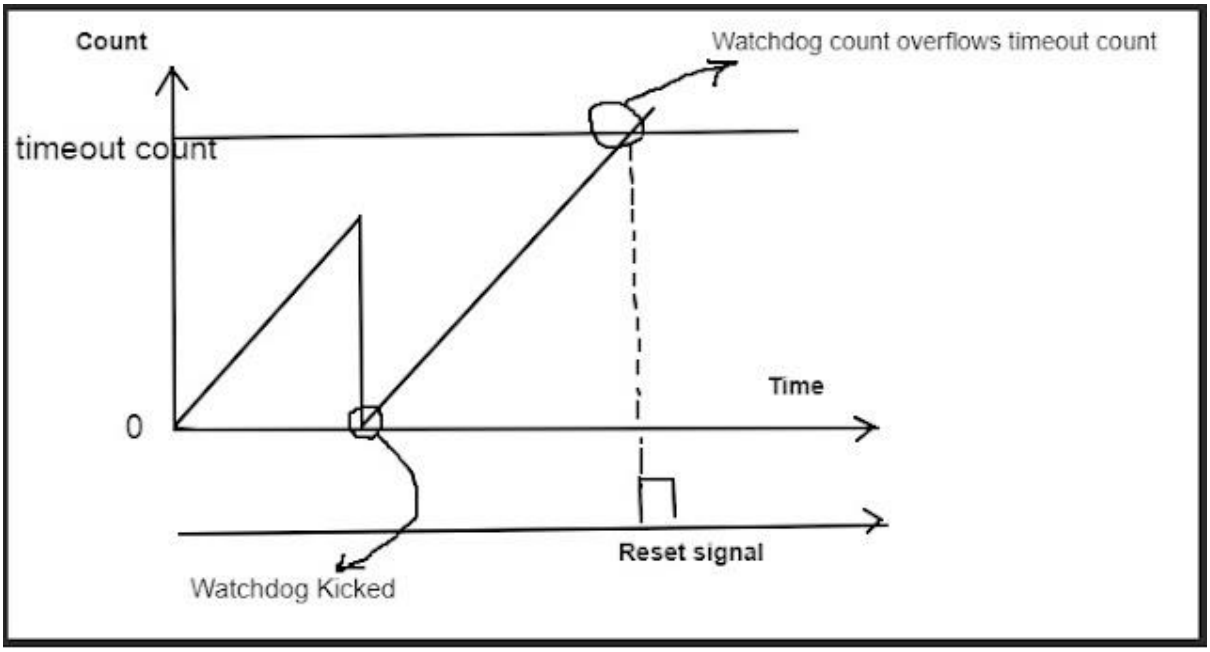


Fig. 2 Watchdog Working

Watchdog Calculations:

Consider a system in watchdog is working on **4 kHz clock**. **System finishes its work in 450 ms** and **worst case time to finish work is 500 ms**. Let us consider **500 ms as timeout time.**

1/4 kHz= 0.25 ms.
1 tick of clock = 0.25 ms.
500 ms =2000 ticks.

When clock ticks 2000 times 500 ms is completed. Counter value related to timeout is 2000.
If watchdog counter reaches to 2000 it will generates a reset signal as per Fig 2. Before reaching to 2000 system need to reset counter to 0.

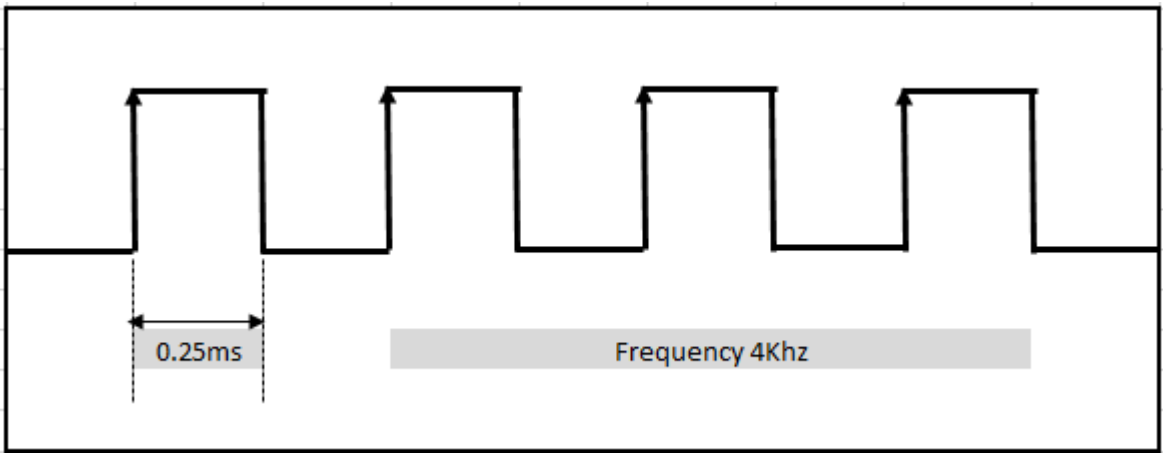


Fig 3. Clock to Watchdog

Watchdog Implementation:

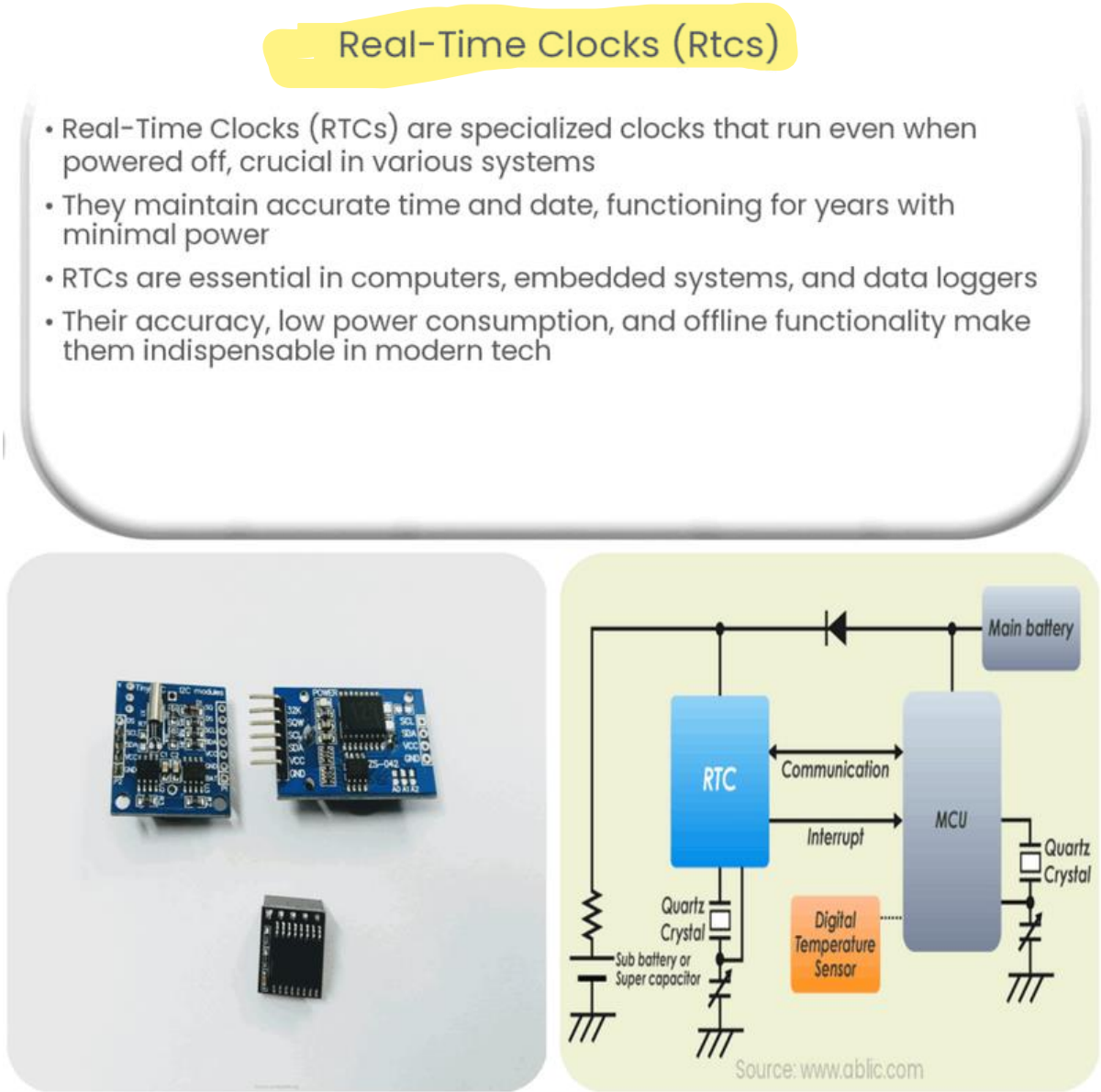
Consider system having 3 periodic tasks as shown in below code task1, task2 and task3. To finish 3 tasks it takes 450 ms as per above discussion. So after finishing all 3 tasks software should reset (kick) watchdog counter. So that system will never get reset.

If system stuck in any one of task, it will never kick watchdog timer. Then watchdog counter will increment and once counter reaches to 2000 count (as per above calculation), **watchdog hardware will generates an interrupt i.e. reset signal and system will get reset (Fig. 2).**

```
void main()
  {
     while(1)
      {
         task1();
         task2();
         task3();
         kick_wdg(); // called before 500ms
```

```
    } //end of while


} // end of main
```



RTC is an IC (integrated circuit) device and also a computer clock that keep track of the current time. It assists to keep and maintain accurate time in devices.

It is powered by an internal lithium battery. As a result of which even if the power of the system is turned off, the **RTC** clock keeps running. Below picture is an anatomy of rtc IC and it has a tiny lithium battery.

Introduction to Real-Time Clocks (RTCs)

A **Real-Time Clock** (RTC) is a computer clock that keeps track of the current time. It's not just any ordinary clock, but a specialized one that runs even when the system is powered off. This ability makes it an indispensable component in a variety of systems.

Modern RTCs are usually present on the motherboard of computers, embedded systems, and servers. They are highly precise and designed to consume minimal power, allowing them to function for several years with a small battery. The primary function of an RTC is to maintain accurate time, date, and often, other information like day of the week.

Working Principle of RTCs

Inside an RTC, a **crystal oscillator** serves as the heart of the timekeeping function. It uses the mechanical resonance of vibrating crystal to create an electrical signal with a precise frequency. This allows the RTC to accurately keep time, often down to the second

Applications of RTCs

1. **Computers and Laptops:** RTCs in these devices keep track of time and date, even when they're turned off. When you boot your system, the operating system reads the RTC to get the current time and date.

2. **Embedded Systems:** Many embedded systems like digital cameras, GPS devices, and home automation systems use RTCs to timestamp events or to perform tasks at certain times.

3. **Data Loggers:** RTCs play a crucial role in data logging devices, which need to record the exact time when data is collected. This is particularly useful in scientific experiments, environmental monitoring, and industrial processes.

Types of Real-Time Clocks (RTCs)

There are various types of RTCs, each designed to cater to different requirements. Here are a few common ones:

- **Battery-backed RTCs:** These RTCs have a separate lithium battery which powers the clock even when the main power supply is turned off. This design allows the RTC to maintain accurate timekeeping for several years.

- **Capacitor-backed RTCs:** Instead of a battery, these RTCs use a capacitor for power backup. While they tend to have shorter backup duration compared to battery-backed RTCs, they are often chosen for their longer lifespan and eco-friendliness.

- **Crystal RTCs:** These RTCs use a crystal oscillator for timekeeping. They offer high accuracy and stability and are commonly used in computers and embedded systems

In-circuit emulator (ICE)

- An in-circuit emulator (ICE) is a hardware device used to debug the software of an embedded system. It was historically in the form of bond-out processor which has many internal signals brought out for the purpose of debugging. These signals provided information about the state of the processor.
- • An in-circuit emulator provides a window into the embedded system. 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.
- • More recently the term also covers JTAG based hardware debuggers which provide equivalent access using on-chip debugging hardware with standard production chips.
- • ICE's attach a terminal or PC to the embedded system. The terminal or PC provides an interactive user interface for the programmer to investigate and control the embedded system.
- • In usage, an ICE provides the programmer with execution breakpoints, memory display and monitoring, and input/output control.

  Benefits- In-circuit emulator offers additional benefits when integrated with compilers. When in-circuit emulators are integrated with compilers, the build/make process is streamline so when bugs are found the code can be quickly changed in-circuit emulator environment and retested. You can get in-circuit emulators that can support multiple microprocessors built by different manufacturers.

  Features- One of the greatest feature of in-circuit emulator is the ability to set trace condition. Complex trace condition and filters allow developers to trap bug and then look back the events that led up to the error. The trace offers the engineers a view of what he would normally not be able to see. The trace happens in real time which help of events that may need to happen without delays of stopping or even slowing down the microprocessors.

What is a Debugger?

A debugger is a software tool that facilitates the identification and correction of errors, commonly known as bugs, in a program. In the context of embedded systems, a debugger is specifically designed to work with the constraints and intricacies of embedded hardware. It allows developers to observe and manipulate the program's execution, inspect variable values, set breakpoints, and trace the flow of code, providing valuable insights into the system's behaviour.

In-Circuit Debugging

In-circuit debugging involves connecting a debugger directly to a running embedded system, allowing developers to monitor and control its execution. Benefits include real-time debugging capabilities, improved system visibility, and the ability to debug hardware-related issues.

Some popular in-circuit debugging tools for embedded systems include:

- JTAG: A widely used hardware debugging interface, as mentioned in the Dynamic Analysis section
- Segger J-Link: A popular JTAG/SWD debugger for ARM-based systems
- P&E Micro: A provider of in-circuit debugging solutions for various microcontroller platforms
- Atmel-ICE: An in-circuit debugger and programmer for Atmel microcontrollers

Hardware Debugging

Hardware debugging involves diagnosing and fixing issues related to the physical components of an embedded system, such as circuitry, sensors, and actuators. Benefits include improved system reliability, reduced development time, and the ability to identify and resolve hardware-specific issues.

Some popular hardware debugging tools for embedded systems include:

Oscilloscopes: Essential tools for analyzing and troubleshooting electrical signals

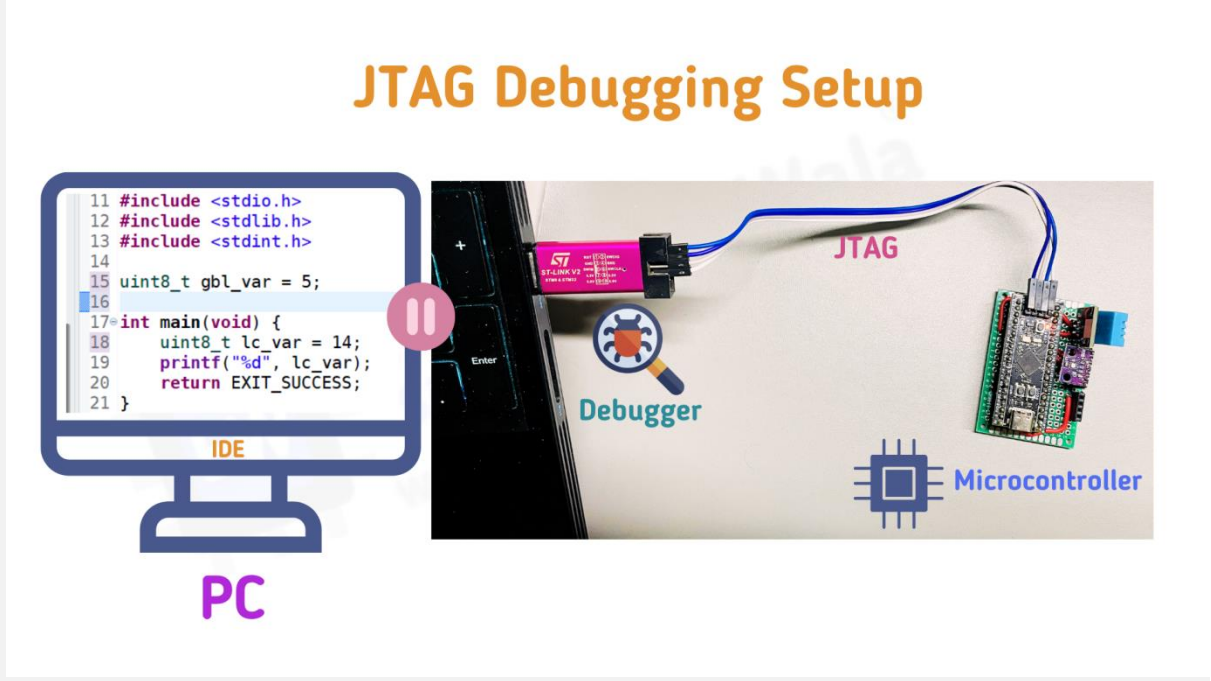
- Logic Analyzers: Devices used for monitoring and analyzing digital signals
- Protocol Analyzers: Tools for capturing and analyzing communication data between system components
- Power Analyzers: Instruments for measuring and analyzing power consumption in embedded systems

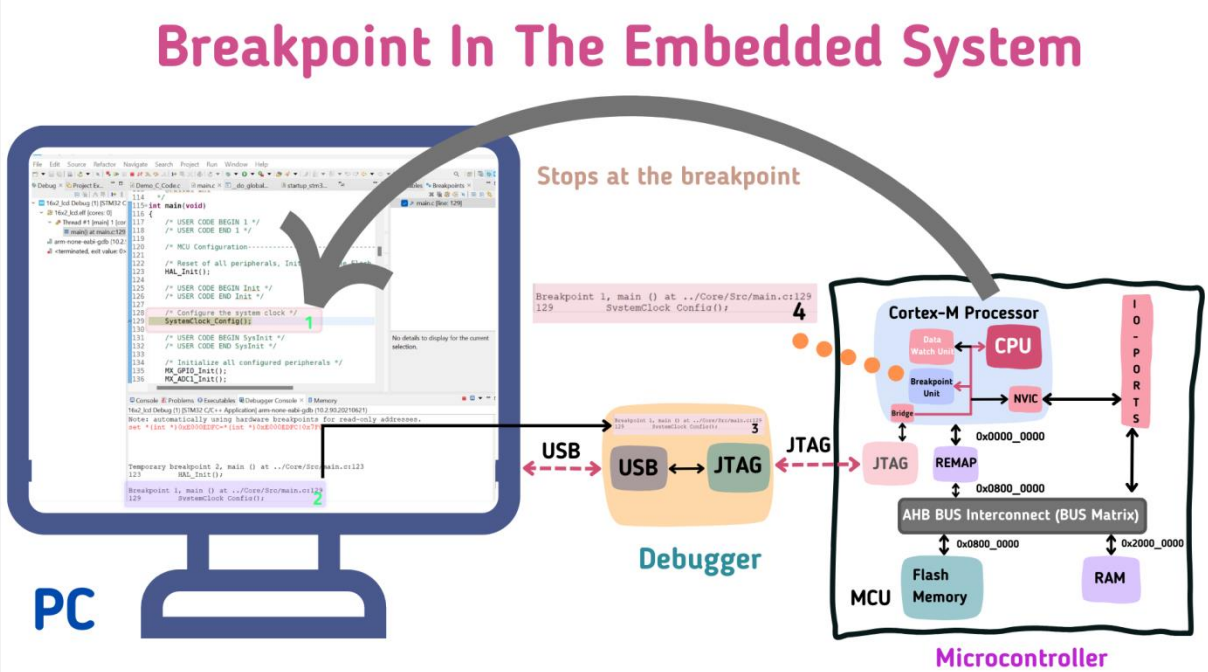
Firmware Debugging

Firmware debugging involves diagnosing and fixing issues related to the low-level software that controls an embedded system's hardware. Benefits include improved system stability, increased reliability, and reduced development time.

Some popular firmware debugging tools for embedded systems include:

- GDB: The GNU Debugger, as mentioned in the Dynamic Analysis section
- OpenOCD: An open-source on-chip debugger for embedded systems
- LLDB: A high-performance debugger for C, C++, and other languages
- Intel System Debugger: A powerful debugger for Intel-based systems





In-Direct Debugging (e.g., UART, SPI):

In-direct debugging is a cost-effective solution that utilizes less dedicated hardware, often everaging common digital interfaces such as **UART (Universal Asynchronous Receiver-Transmitter)** and **SPI (Serial Peripheral Interface)** etc.