# Real Time Systems

## Unit:1 Introduction to RTS

By

**HAMSASHREE M K**
Asst. Professor
Dept of ECE
BGSIT

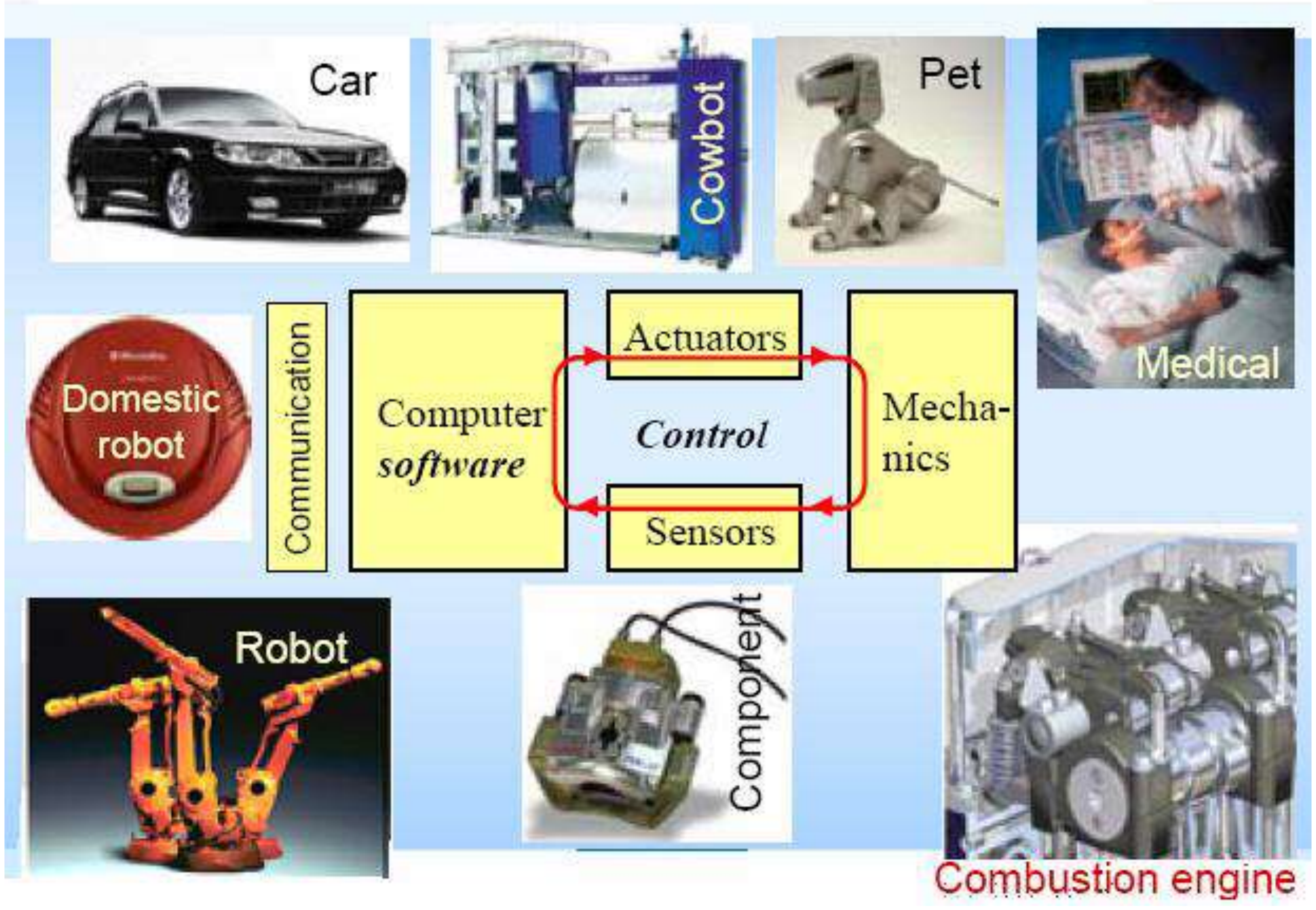
## Lecture Outline:

- Historical background.
- Elements of a real-time system.
- What is a real-time system?
- Classification of real-time systems.
- Characteristics of a real-time system.

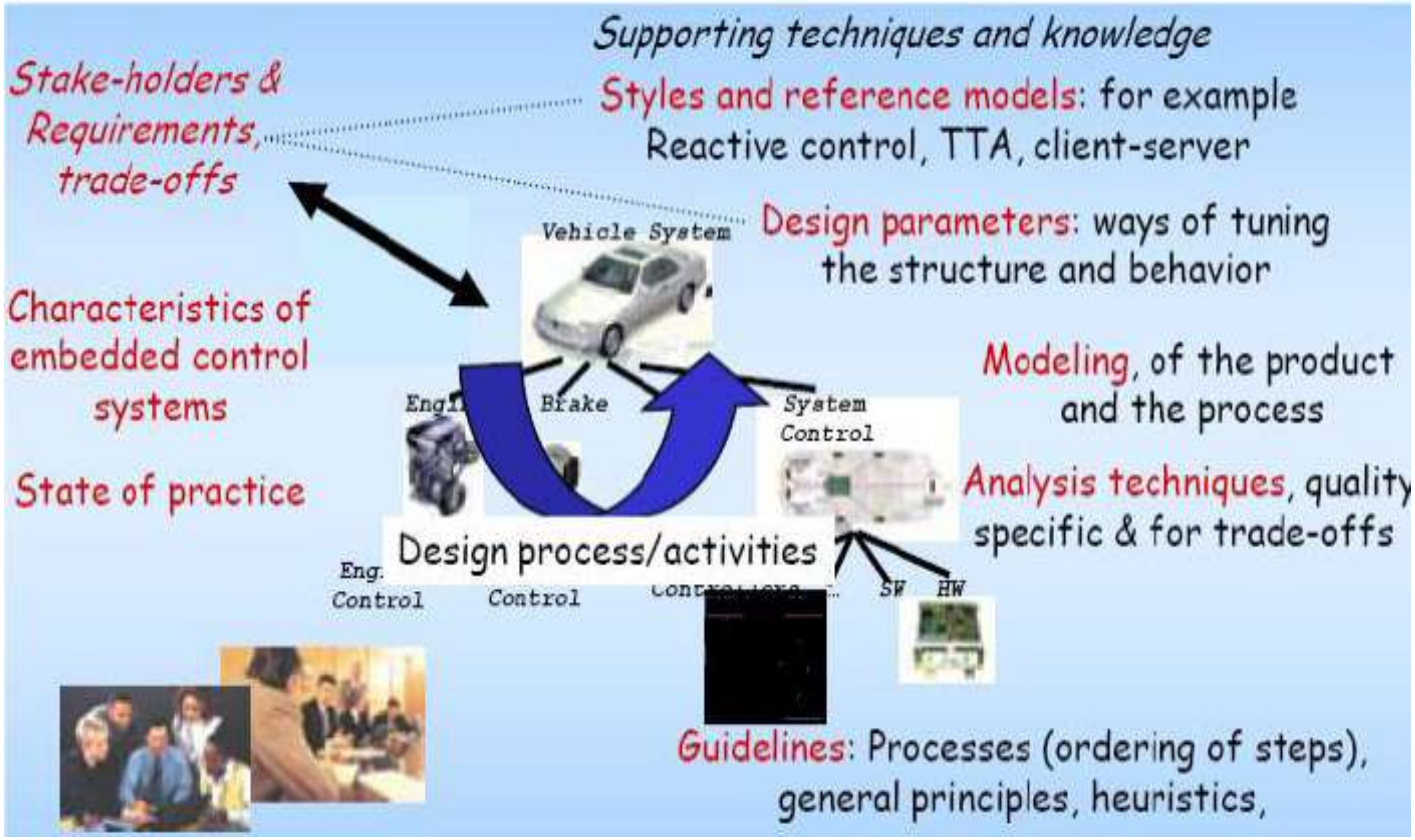
## Historical Background:

- Brown and Campbell (1950) : (in paper only)
- Using a computer operating in real-time as part of a control system . (analog computing elements)
- The 1st digital computers developed specifically for R.T.S. were for airborne operating. in 1954 a digital computer was successfully used to provide an automatic flight and weapons control system .
- The 1st industrial installation of a computer system was in September, 1958 for plant monitoring at power station in sterling , Louisiana .
- The 1st industrial computer control installation was mad by the Texaco company who installed an RW-300 system at their port Arthur refinery in Texas on 15/03/1959
- The 1st DDC computer system was the Ferranti Argus 200 system installed in November, 1962 at the ICI, Lancashire, UK. It has 120 control loops and 256 measurements .
- The advent of the Microprocessor in 1974 made economically possible the use of DDC and distributed computer control systems.

## Real-Time Systems



## Course Scope:



*Supporting techniques and knowledge*

*Stake-holders & Requirements, trade-offs*

Characteristics of embedded control systems

State of practice

Styles and reference models: for example Reactive control, TTA, client-server

Design parameters: ways of tuning the structure and behavior

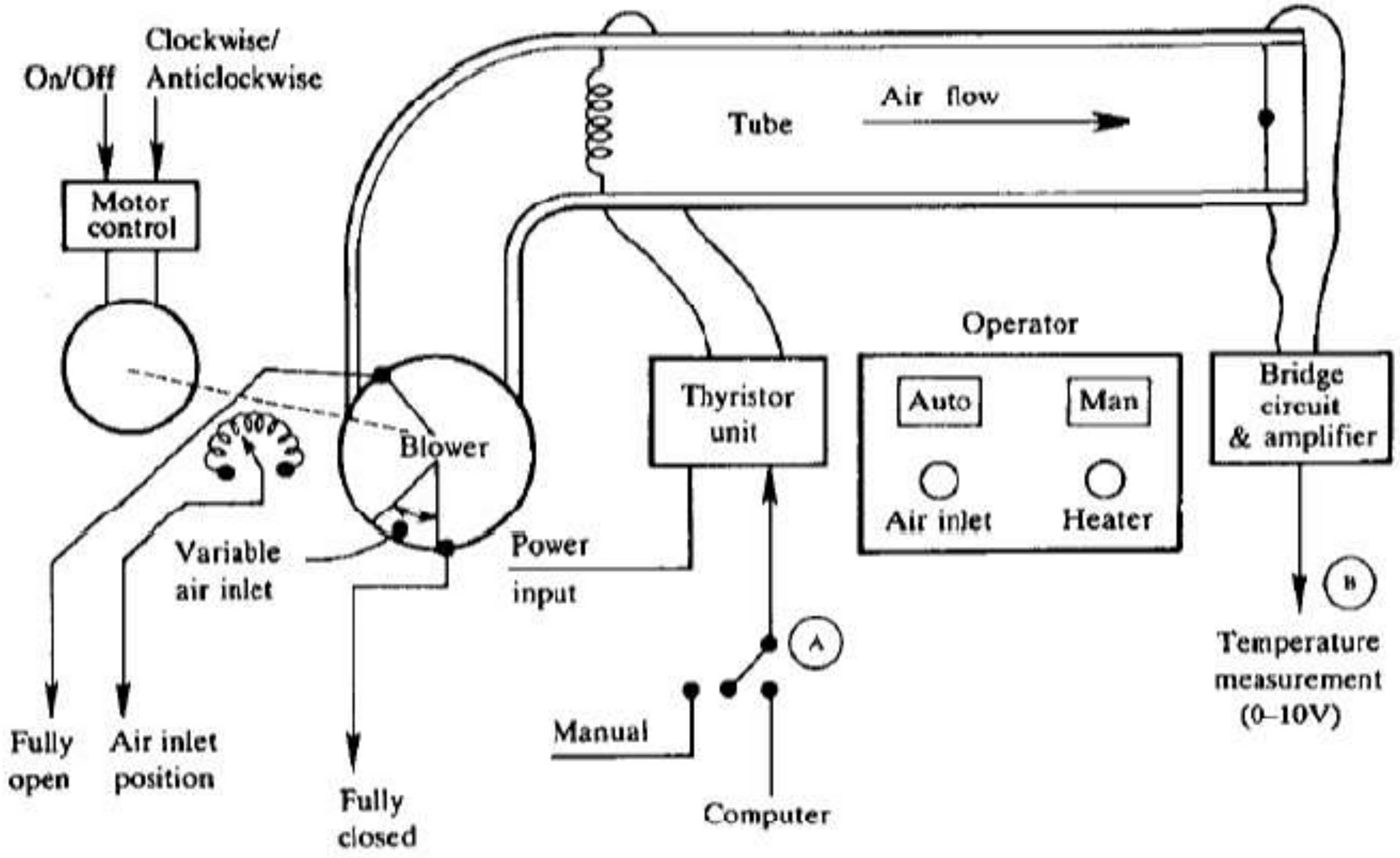
Modeling, of the product and the process

Analysis techniques, quality specific & for trade-offs

Guidelines: Processes (ordering of steps), general principles, heuristics,

## Elements of a Real-Time System:

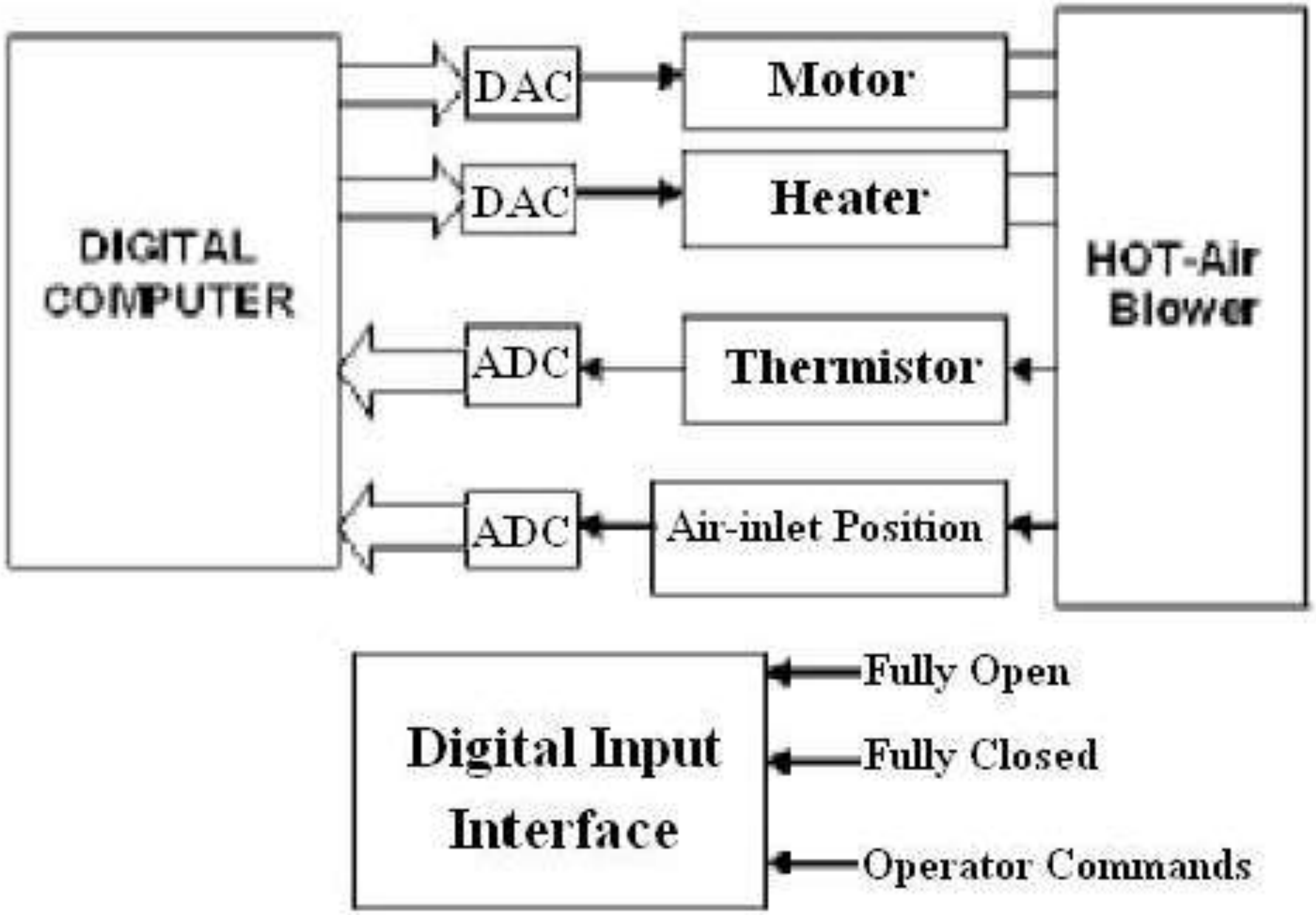
A simple plant – a hot-air blower.

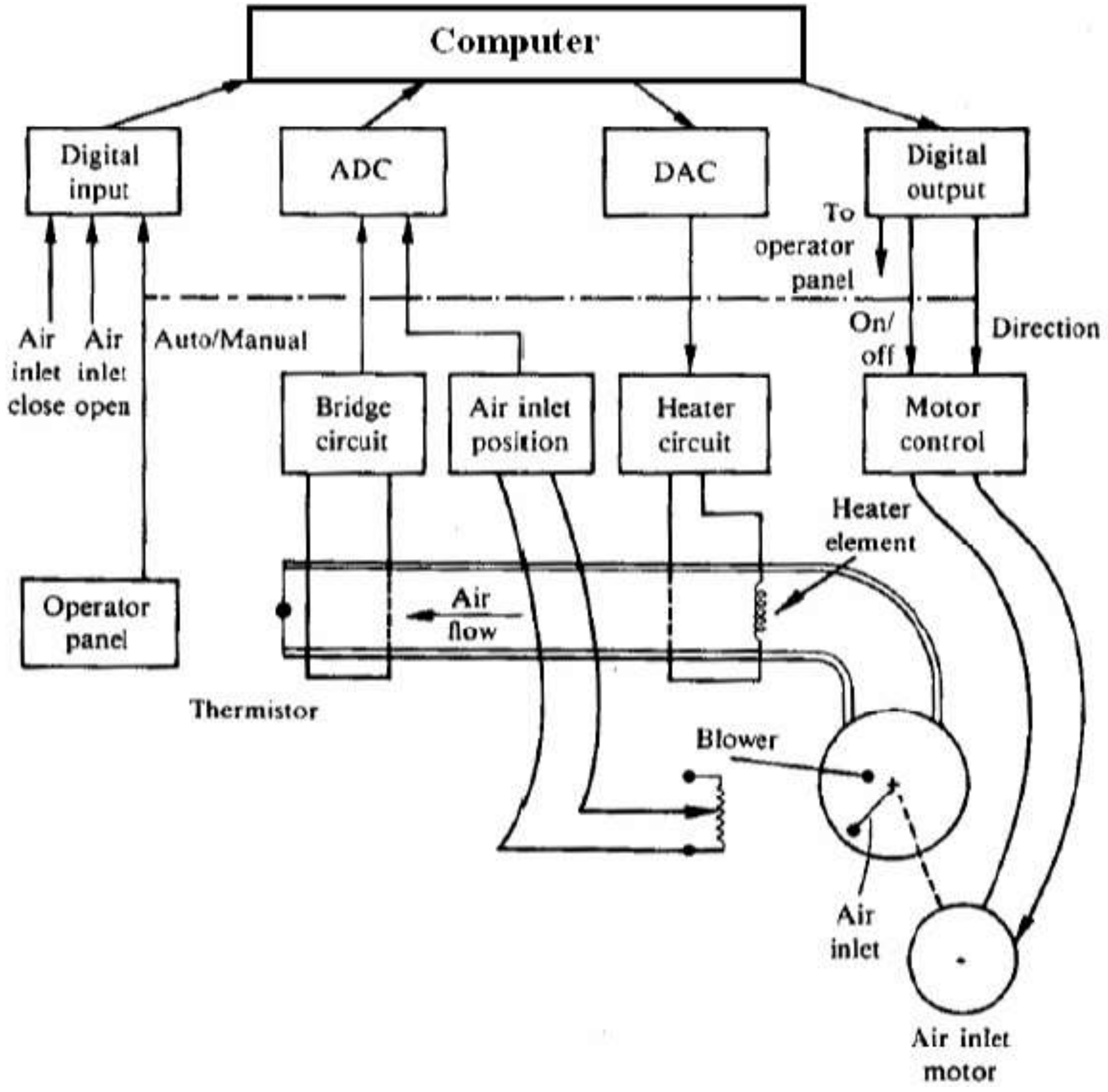


## ELEMENTS OF A COMPUTER CONTROL SYSTEM:

- A centrifugal fan blows air over a heating element and into a tube.
- A thermostat is used to detect the temp.
- The position of the air-inlet cover to the fan is adjusted by a reversible DC motor (constant speed).
- A potentiometer is attached to the air-inlet cover .
- Two 8-switches are used to detect when the cover is fully open or fully closed .
- The operator is provided with a panel from which the control system can be switched from automatic to manual control panel. Lights indicate: Fan ON, Heater ON, Cover fully-open, Cover fully-closed, Auto/Manual status.
- The information is available from the plant instrument in the following two forms:
  – Analog signals : Air Temp., Fan-inlet cover position .
  – Digital signals : Fan-inlet cover position (Fully-open, Fully-closed )
  – Status signals : Auto/Manual , Fan motor ON, Heater ON

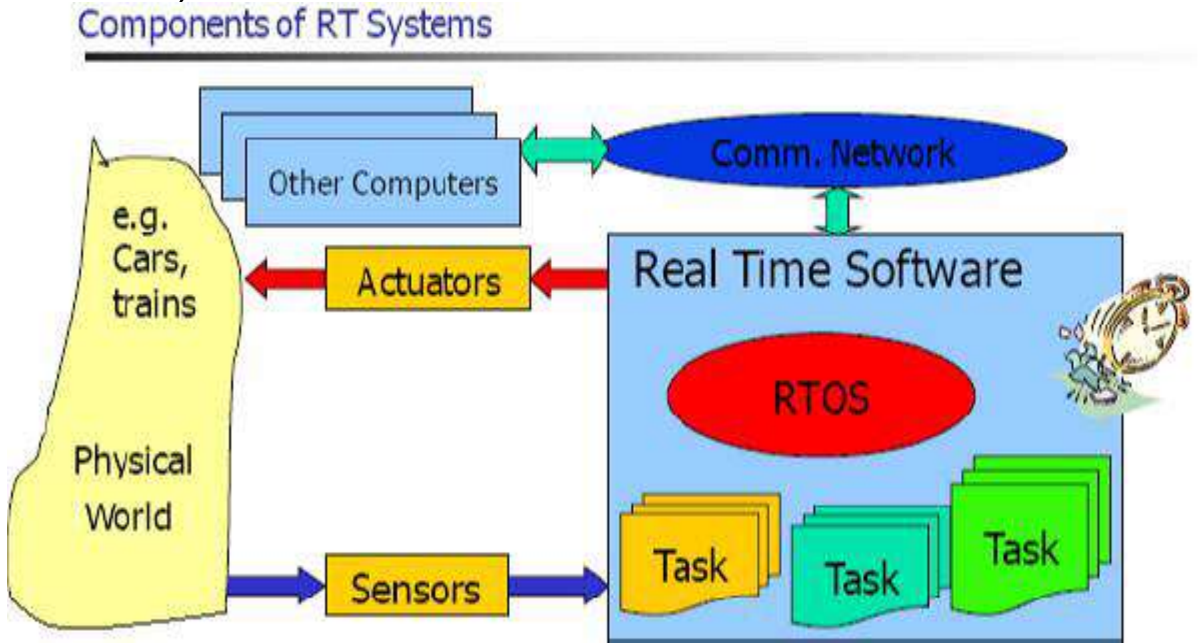
## Computer Control Of a Hot-air Blower :




Computer
Digital input
ADC
DAC
Digital output
To operator panel
Air inlet close
Air inlet open
Auto/Manual
On/off
Direction
Bridge circuit
Air inlet position
Heater circuit
Motor control
Heater element
Operator panel
Air flow
Thermistor
Blower
Air inlet
Air inlet motor

## OVERALL STRUCTURE OF RT SYSTEMS:

- Hardware (CPU, I/O devices, memory… etc)
  – Single CPU or more.
  – Clock selecion.
- A real time Operating System: function as standard OS, with predictable behavior and well-defined functionality.
- A collection of RT tasks/processes (share resources, communicate/synchronize with each other and the environment)



## WHAT IS A REAL-TIME SYSTEM?

- According to Oxford dictionary :

  "Any system in which the O/P is produced is significant."

- Alternative definitions :
  - A RTS reads I/Ps from the plant and sends control signals to the plant at times determined by plant operational C/Cs.
  - RTSs are those which must produce correct responses within a definite time limit.
  - A RTS is any information processing system that has to respond to externally generated signal within a finite and specified period.
  - A RTS is a computer system where the correct functioning of the system depends on the results produced and the time at which they are produced.

## What is a Real-Time System? (cont)

- A system, where correct timing behavior is strongly related to functionality,performance and reliability
- A computer system is a real-time one if it explicitly manages resources in order to meet timing constraints.
- A real-time system is a system where the correctness depends not only on the logical result of computation but also on the time at which the results are produced".
- A system that is synchronous with the interacting environment.

## In real-time systems:

- **Timing of actions is essential:** Compare with table tennis, air bag, engine control and music.
- **Age of data is essential:** Compare with a weather report: sample data, compute, actuate - when does the data cease to be valid?
- **Notes:**

  - Different consequences depending on context!

  - Different types of timing requirements

  - Delays need to be controlled.
- **Requirements on a real-time system:**

  1. Sufficiently fast (processing, communication, ...)

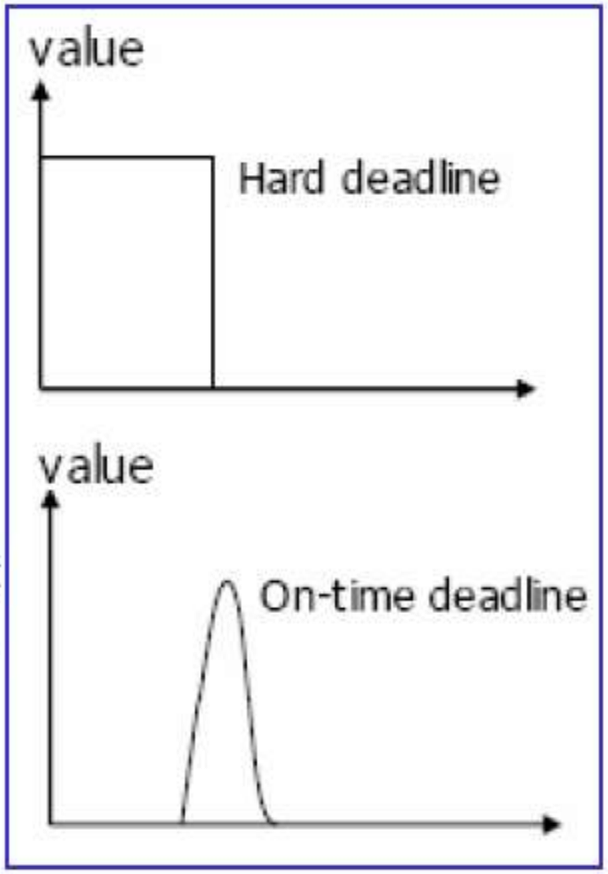
  2. Predictable resource sharing and timing!

## CLASSIFICATION OF RTSS:

Real-Time systems can be classified as:

**1. HARD REAL-TIME SYSTEM:**

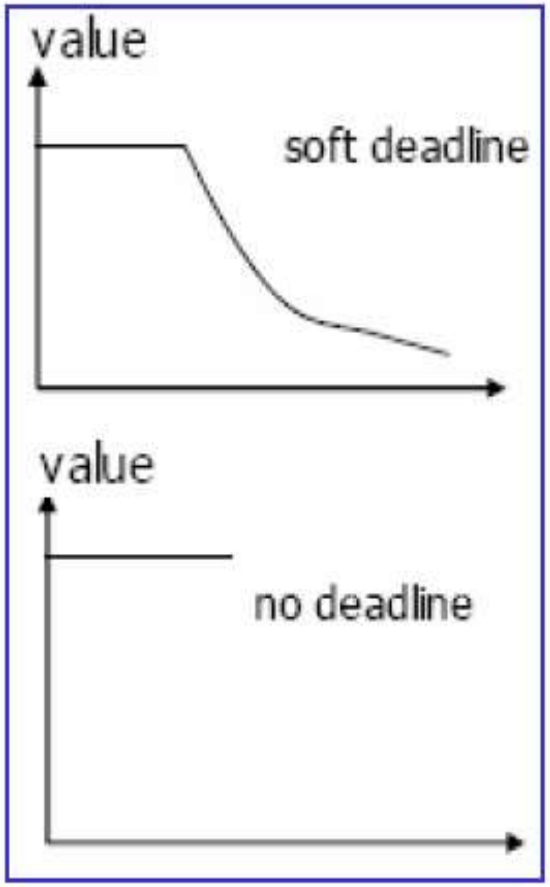
- A system whose operation is degraded if results are not proceeded according to specified timing requirements.
- System response occur within a specified deadline. Failure to meet such a timing requirement can have catastrophic consequences.
- Systems where it is absolutely imperative that responses occur within the required deadline. Example: Flight control systems, automotive systems, robotics etc.



## Classification of RTSs: (Cont)

2. **SOFT REAL-TIME SYSTEMS:**

- A system whose operation is incorrect if results are not produced according to the timing constraints. Catastrophic results will happen then.
- The response times are important but not critical to the operation of the system . Failure to meet the timing requirements would not impair the system.
- Systems where deadlines are important but which will still function correctly if deadlines are occasionally missed. Example: Banking system, multimedia etc.

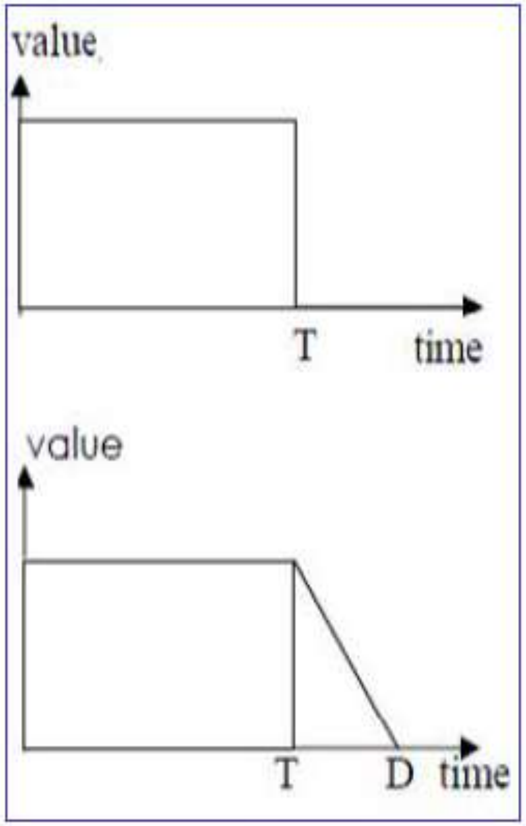


## Classification of RTSs: (cont.)

3. **FIRM REAL-TIME SYSTEMS:** There is no value for a response that occurs past a specific deadline. Failure to meet the timing requirements is undesirable .

**Notes:**

- ➢ A single system may have both hard and soft real-time Subsystems.
- ➢ In reality many systems will have a cost function associated with missing each deadline.



## Clock-Based & Event-Based Systems:

- Synchronization between the external processes and internal actions (tasks) carried out by the computer may be defined in terms of the passage of time, or the actual time of day, in which case the system is said to be **"Clock-based system"** or it may be defined in terms of events, and the system is said to be **"Event-based system".**
- If the relationship between the actions in the computer and the system is much more loosely defined, then the system is said to be **"interactive system".**

- Real-Time systems can be classified as:

**1. Clock-Based Tasks: (Cyclic and Periodic):**

– The completion of the operations within the specified time is dependent on the number of operations to be performed and the speed of the computer .

– Synchronization is usually obtained by adding a clock to the computer system , and using a signal from this clock to interrupt the operation of the computer at predetermined fixed time interval .

$$\begin{tikzcd}\text{Plant time constant} \arrow[r] & \text{Sampling time (Ts)} \arrow[r] & \text{Interrupt}\end{tikzcd}$$

## 2. Event-Based Tasks: (A periodic):

Action are to be performed not at particular times or time intervals but in response to some event . The system must respond within a given max. time

to a particular event .

– Events occur at non-deterministic intervals and event-based tasks are referred to as “aperiodic task”.

## 3. Interactive Systems:

- They represent the largest class of RTSs such as automatic bank tellers, reservation systems for hotels , airlines and car rental……etc.
- The real-time requirement is usually expressed in terms such as "the average response time must not exceed ……"
- Example: an automatic bank teller system might require an average response time not exceeding 20 sec.

## Classification of Programs:

- A real-time program is defined as a program for which the correctness of  operation depends on the logical results of the computation and the time at which the results are produced.
- In general there are three types of programming:

  1. **Sequential:** Actions are ordered as a time sequence , the program behavior depends only on the effects of the individual actions and their order .

  2. **Multi-tasking:** Actions are not necessarily disjoint in time , it may be necessary for several actions to be performed in parallel .

  3. **Real-Time:** Actions are not necessarily disjoint in time , and the sequence of some of program actions is not determined by the designer but the environment

- (by events occurring in the outside world which occur in real-time and without reference to the internal operation of the computer)
- A real-time program can be divided into a number of tasks but communication between the tasks can not necessarily wait for a synchronization signal. The environment task can not be delayed.
- In RT programs, the actual time taken by an action is an essential factor in the process of verification .

## NOTE:

- RTSs have to carry out both periodic activities .
- RTSs have to satisfy time constraints that can be either:
  - A hard constraint , or
  - A soft (average value) constraint.
- RT software is more difficult to specify, design and construct than non real-time software

## Characteristics of a RTS:

- **Large and complex:** vary from a few hundred lines of assembler or C to 20 million lines of Ada estimated for the Space Station Freedom.
- **Concurrent control of separate system components:** devices operate in parallel in the real-world; better to model this Parallelism by concurrent entities in the program.
- **to interact with special purpose hardware:** need to be able to program devices in a reliable and abstract way.
- **Mixture of Hardware/Software:** some modules implemented in hardware, even whole systems, SoC.
- **Extreme reliability and safety:** real-time systems typically control the environment in which they operate; failure to control can result in loss of life, damage to environment or economic loss.
- **Guaranteed response times:** we need to be able to predict with confidence the worst case response times for systems; efficiency is important but predictability is essential.

Thank you☺

# REAL TIME SYSTEMS

## Unit:5 Real-Time Operating Systems

By

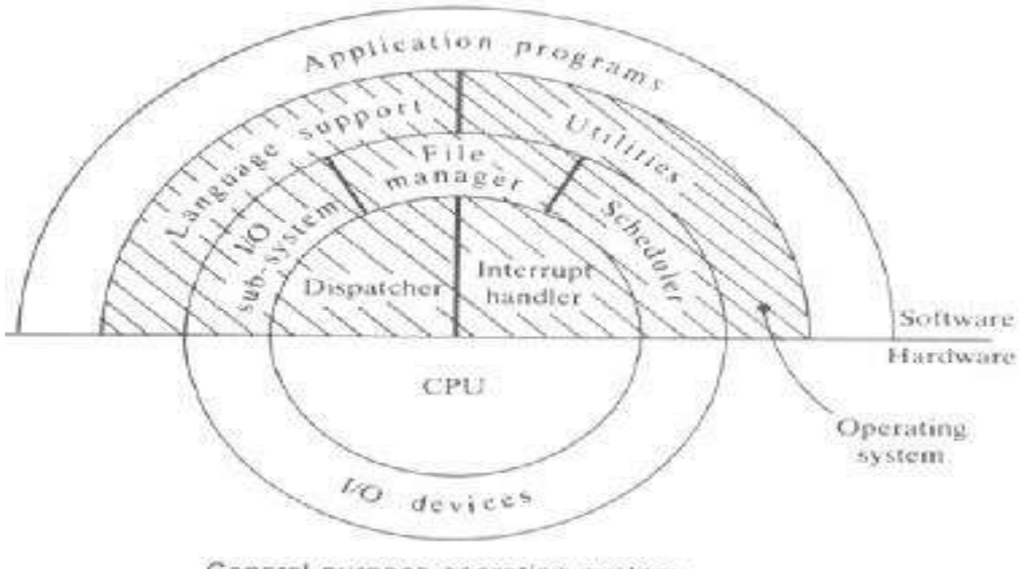
Hamsashree M K
Asst. Professor
Dept of ECE
BGSIT

## Lecture Outline:

- Explain components of a simple operating system.
- Describe types of operating systems.
- Why we use a RTOS?
- What an RTOS does? How it works?
- Benefits and drawbacks of an RTOS.
- Describe and explain by examples the basic task synchronization mechanisms.

## General Purpose Operating System:

- Access to the hardware of the system and to the I/O devices is through the operating system(OS).
- In many real-time and multiprogramming systems restriction of access is enforced by hardware and software traps.
- A general purpose operating system will provide some facilities that are not required in a particular application.
- Recently, operating systems which provide only a minimum kernel have become popular, additional features can be added.

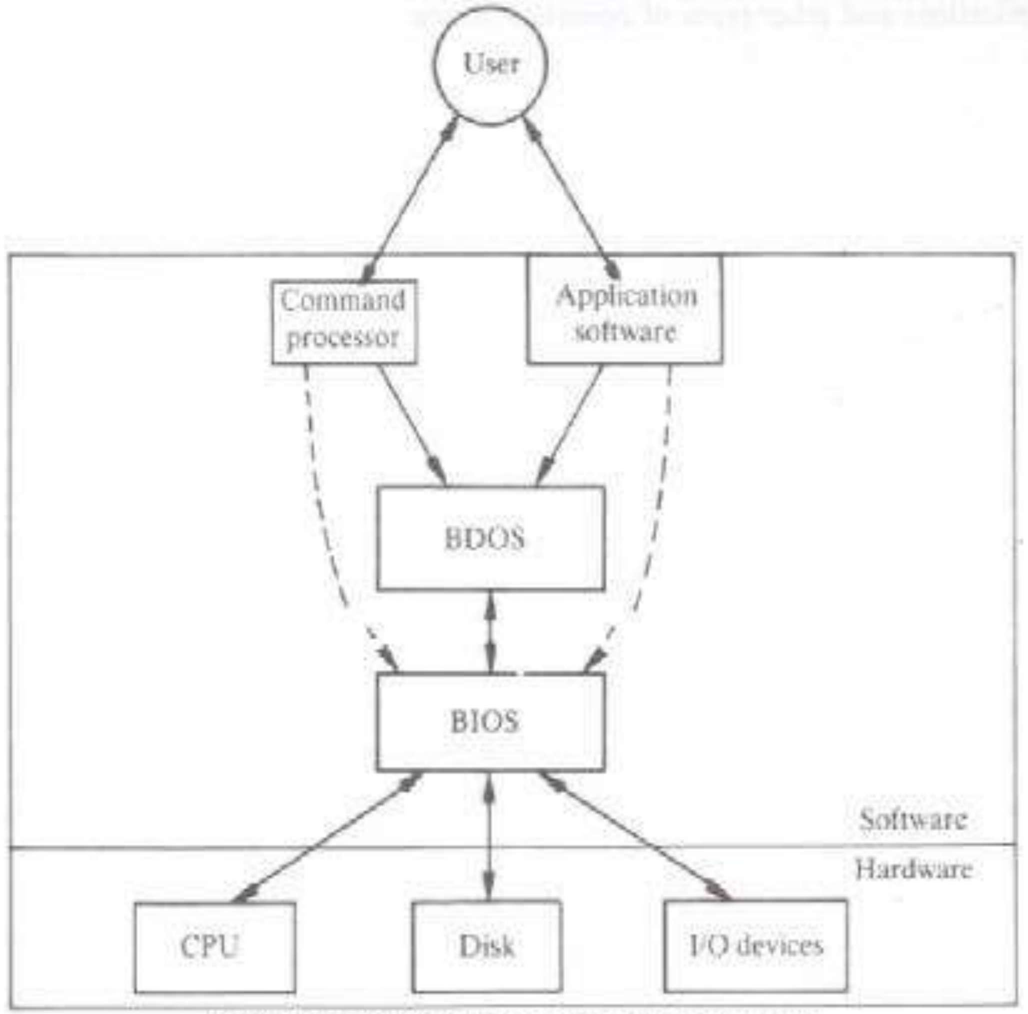


General purpose operating system

## General Structure of a Simple OS:

- The command processor provides a means by which the user can communicate with the OS.
- The actual processing of the user commands is done by BDOS, which also handles the I/O and the file operations on the disks.
- The BDOS makes the actual management of the file and I/O operations transparent to the user.
- Application programs will normally communicate with the hardware of the system through *system calls which are processed by the BDOS*.
- The BIOS contains the various device drivers which manipulate the physical devices and OS.
- Devices are treated as *logical or physical units*. Logical devices are software constructs used to simplify the user interface. User programs perform I/O to logical devices and the BDOS connects the logical devices to the physical device.

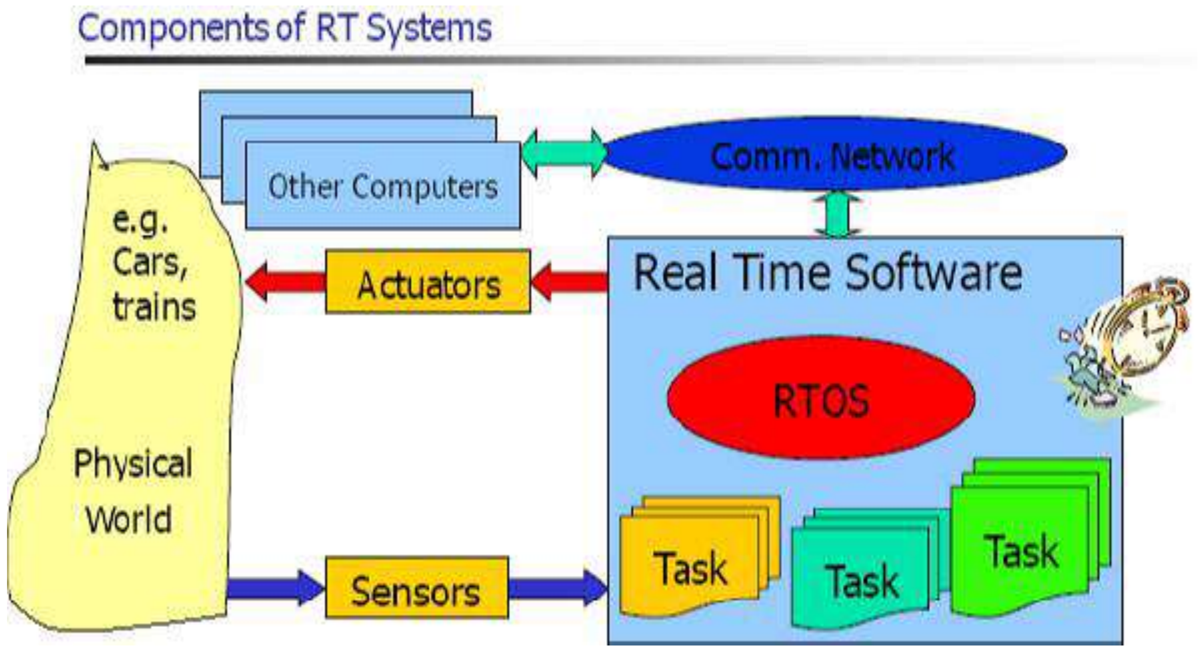
## General Structure of a Simple OS:



General structure of a simple operating system

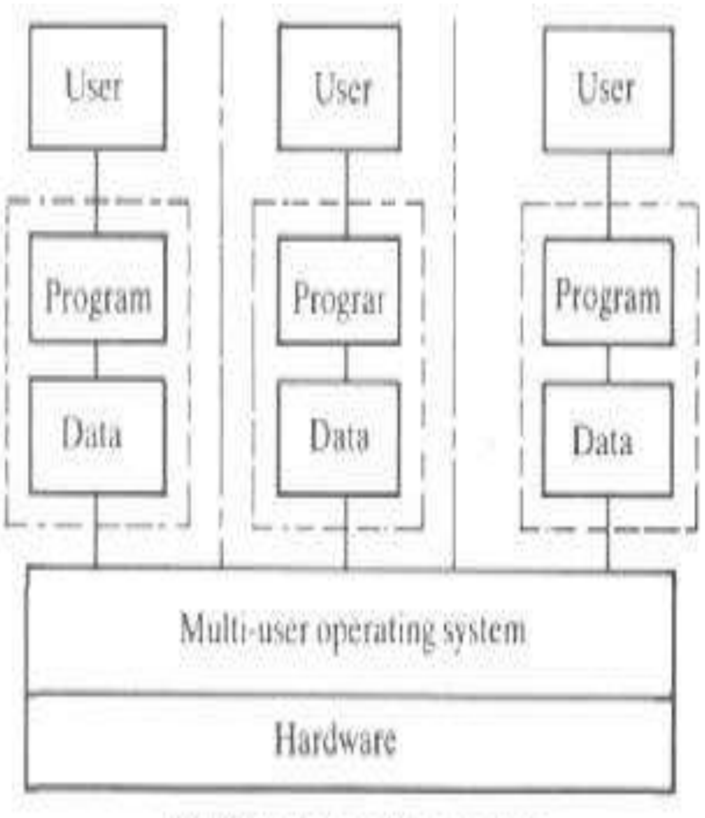
## TYPES OF OPERATING SYSTEMS:

- There are different types of OSs:
  - – Single-user and Multi-user operating systems.
  - – Single-task and Multi-tasking operating systems.
  - – Real-time operating systems.



## Multi-User Operating Systems:

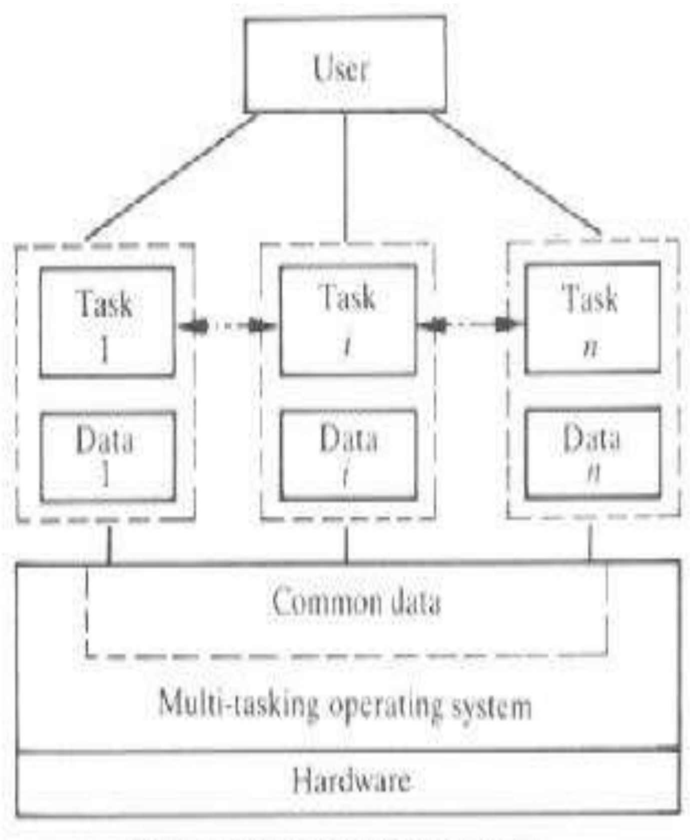
- The OS ensures that each user can run a single program as if the had the whole computer system.
- At any given instance, it is not possible to predict which user will have the use of the CPU.
- The OS ensures that one user program cannot interfere with the operation of another user program. Each user program runs in its own protected environment.



Multi-user operating system

## Multi-Tasking Operating Systems:

- In a multi-tasking operating system, it is assumed that there is a single user and that the various tasks co-operate to serve the requirements of the user.
- Co-operation requires that all tasks communicate with each other and share common data.
- Task communication and data sharing will be regulated so that the OS is able to prevent inadvertent communication or data access, and hence protect data which is private to a task.



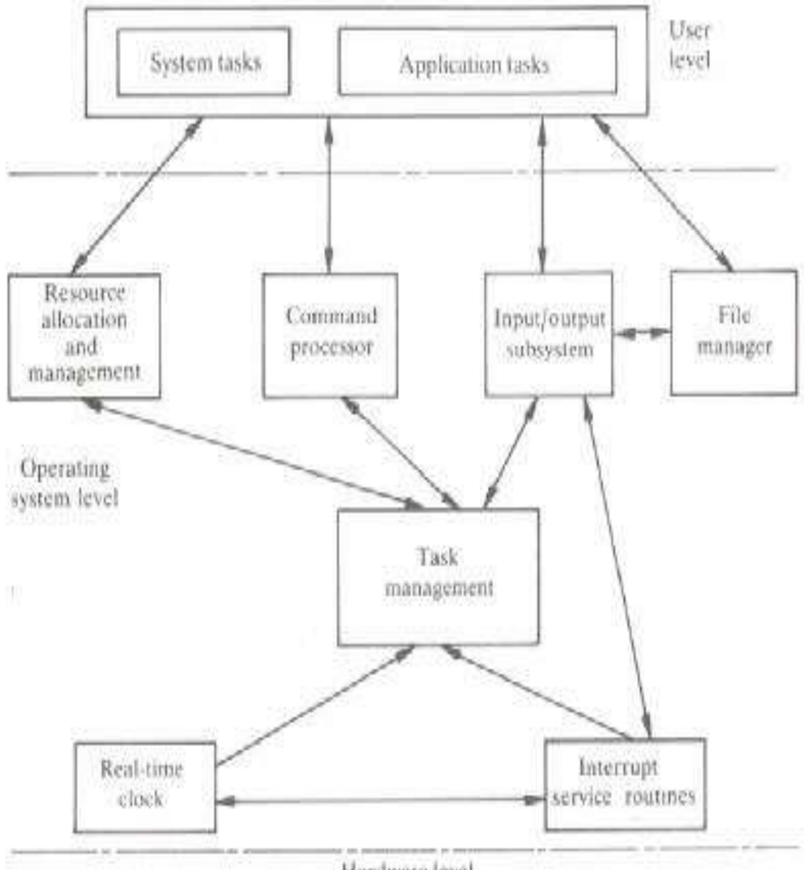
Multi-tasking operating system

## Real-Time Operating System (RTOS):

- A fundamental requirement of an operating system is to allocate the resources of the computer to the various activities which have to be performed.
- In a RTOS this allocation procedure is complicated by the fact that some of the activities are time critical and hence have a higher priority than others. Therefore, there must be some means of allocating priorities to tasks and of scheduling allocation of CPU time to the tasks according to some priority scheme.
- A task may use another task, thus tasks may need to communicate with each other. The OS must have some means of enabling tasks either to share memory for the exchange of data or to provide a mechanism by which tasks can send messages to each other.
- Tasks may need to be invoked by external events and hence the OS must support the use on interrupts.
- Tasks may need to share data and they may require access to various hardware and software components, hence there has to be a mechanism for preventing two tasks from attempting to use the same resource at the same time.

## Real-Time Operating System (RTOS):

- A real-time multi-tasking operating system has to support the resource sharing and the timing requirements of the tasks an the functions can be divided as follows:
  - – **Task Management:** the allocation of memory and processor time (scheduling) to tasks.
  - – **Memory Management:** control of memory allocation.
  - – **Intertask Communication & Synchronization:** provision of support mechanisms to provide safe communication between tasks and to enable tasks to synchronies their activities.



Typical structure of a real-time operating system

## Scheduling Strategies:

- There are two basic strategies for the scheduling of time allocation on a single CPU, these are:
- **1. Cyclic Strategy:**
  - The task uses the CPU for as long as it wishes.
  - It is a very simple strategy which is highly efficient in that it minimizes the time lost in switching between tasks.
  - It is an efficient strategy for small embedded systems for which the execution times for each task run are carefully calculated and for which the software is carefully divided into appropriate task segments.
  - This approach is too restrictive since it requires that the task units have similar execution times. It is difficult to deal with random events using this approach.

## Scheduling Strategies:

- **2. Pre-emptive Strategies:**
  - There are many pre-emptive strategies, all involve the possibility that a task will be interrupted before it has completed a particular invocation.
  - The simplest form of pre-emptive scheduling is to use a time slicing approach. Using this strategy each task is allocated a fixed amount of CPU time (number of clock ticks), and at the end of this time it is stopped and the next task in the list is run. If a task completes before the end of its time slice, the next task in the list is run immediately.

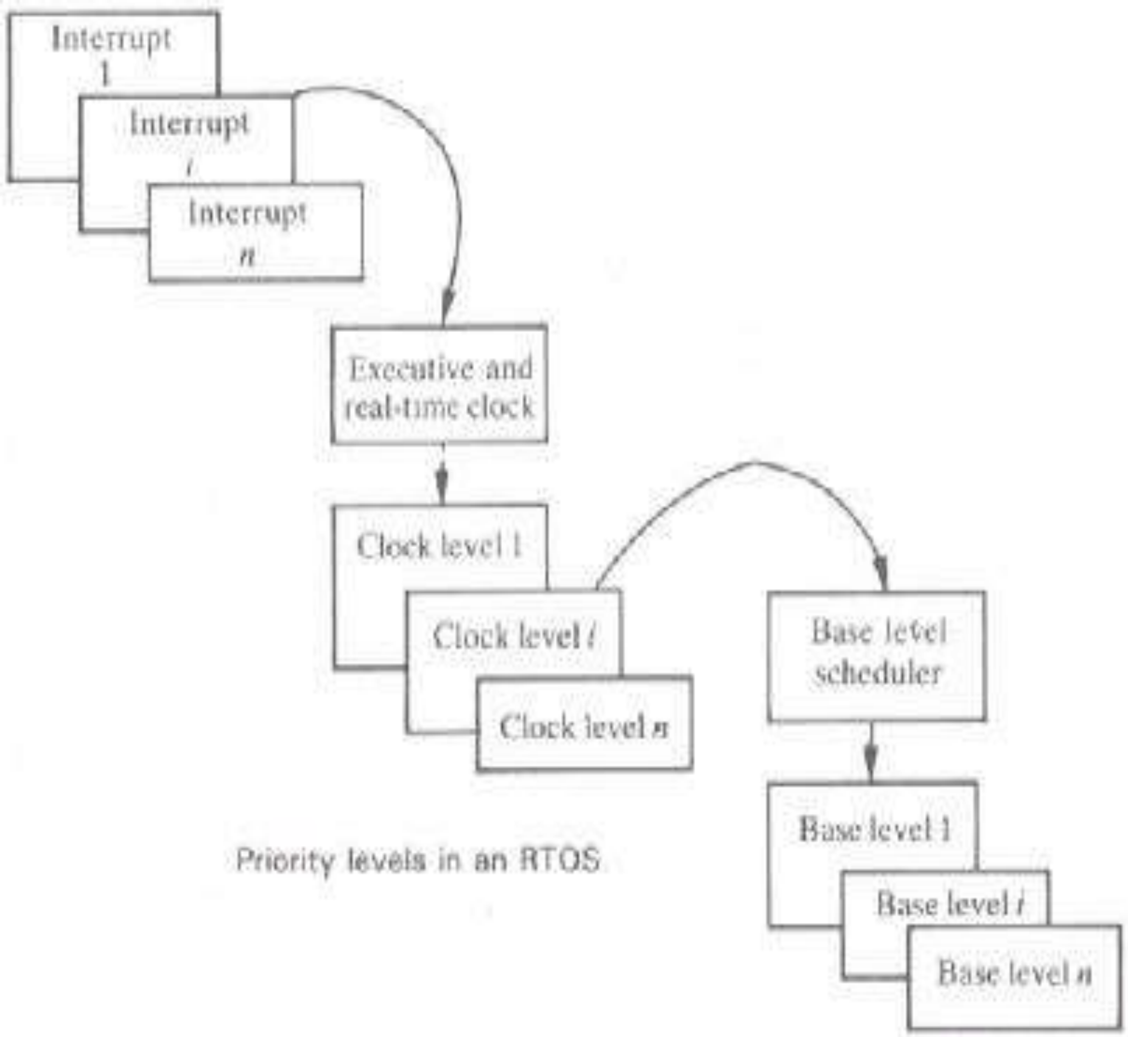
## Priority Scheduling Mechanism:

- Tasks are allocated a priority level and at the end of a predetermined time slice, the task with highest priority of those ready to run is chosen and is given control of the CPU.
- Task priorities may be fixed (static priority system) or may be changed during system execution (dynamic priority system).
- Dynamic priority schemes can increase the flexibility of the system.
- Changing priorities is risky as it makes it much harder to predict and test the behavior of the system.
- The task management system has to deal with the handling of interrupts. These may be hardware interrupts caused by external events, or software interrupts generated by a running task.

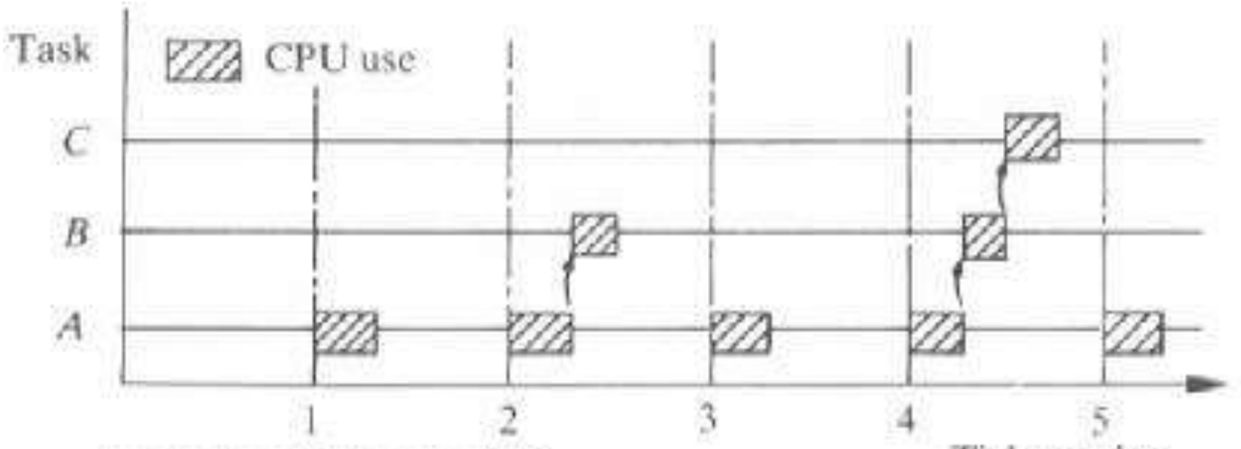
## Priority Structures:

- In a real-time system the designer has to assign priorities to the tasks in the system.
- The priority will depend on how quickly a task will have to respond to a particular event.
- Most RTOSs provide facilities such that task can be divided into three board levels:
    - **1. Interrupt Level:** at this level are the service routines for the tasks and devices which require very fast response (measured in msec.) Example: real-time clock task.
    - **2. Clock Level**: at this level are the tasks which require accurate timing and repetitive processing, such as the sampling and control tasks.
    - **3. Base Level:** tasks at this level are of low priority and either have no deadlines to meet or are allowed a wide margin of error in their timing. Tasks at this level may be allocated priorities or may all run at a single priority level.

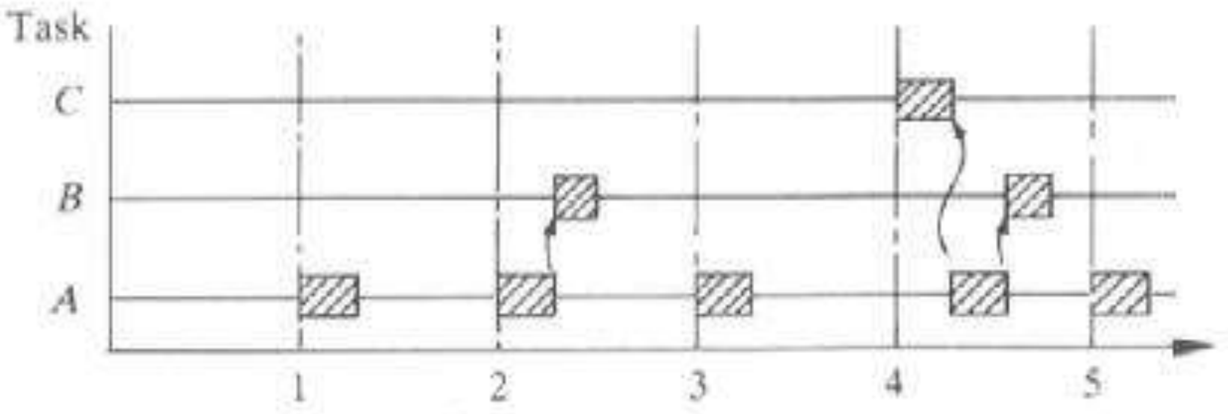
## Priority Structures:



Priority levels in an RTOS



(a) task priorities A,B,C



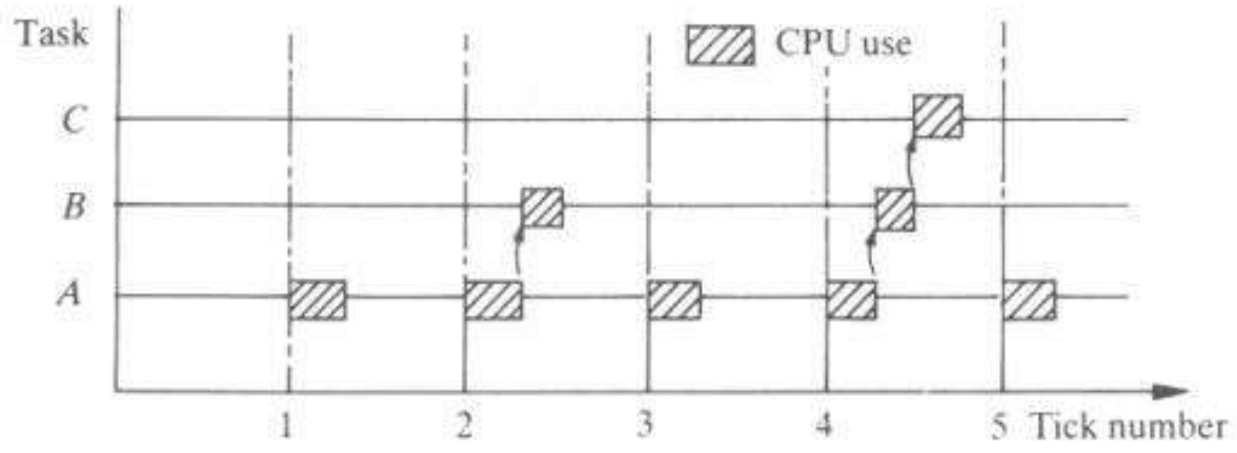
(b) task priorities C,A,B

## Clock Level:

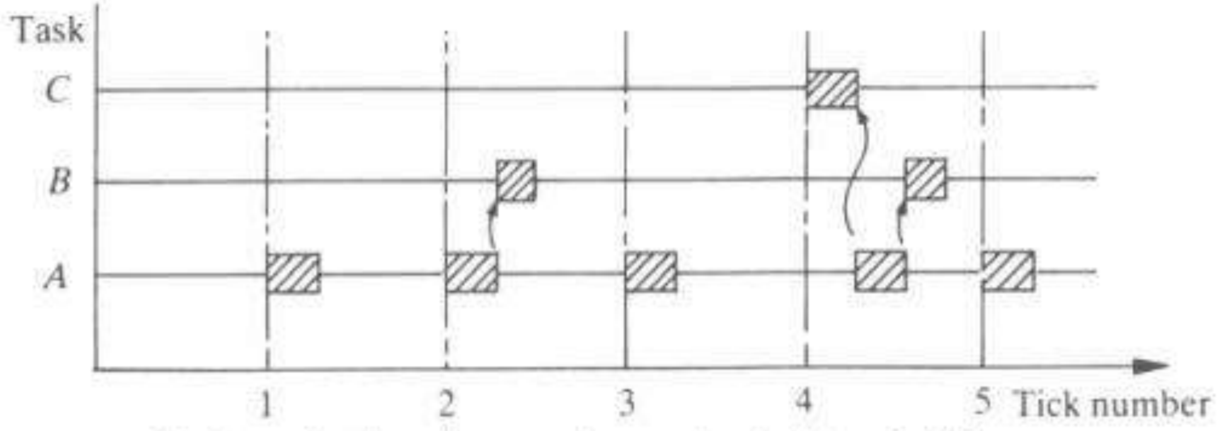
- One interrupt level task will be the real-time clock.
- Typical values 1-200 msec.
- Each clock interrupt is known as a tick and represents the smallest time interval in the system.
- The function of the clock interrupt handling routine is to update the time of day clock in the system and to transfer control to dispatcher.
- The scheduler selects which task is to run at a particular clock rate.
- Clock level tasks divided into two categories;
    - – **Cyclic:** these are tasks which require accurate synchronization with outside world**.**
    - – **Delay:** these tasks simply wish to have a fixed delay between successive repetitions or to delay their activities for a given period of time.

## Clock Level:

- Cyclic tasks are ordered in a priority which reflects the accuracy of timing required for the task, those which require high accuracy being given the highest priority
- Tasks of lower priority within clock level will have some jitter since they will have to await completion of the higher-level tasks.
- **Example:**
  - Three tasks A, B, and C are required to run at 20 msec, 40 msec and 80 msec intervals. If the clock interrupt rate is set at 20 msec. if the task priority order is set as A,B,and C with A as the highest priority.
  - The following slid shows task activation diagram for this example in two cases;
  - Case (a): Task priorities are: A, B, then C.
  - Case (b): Task priorities are: C, A, then B.



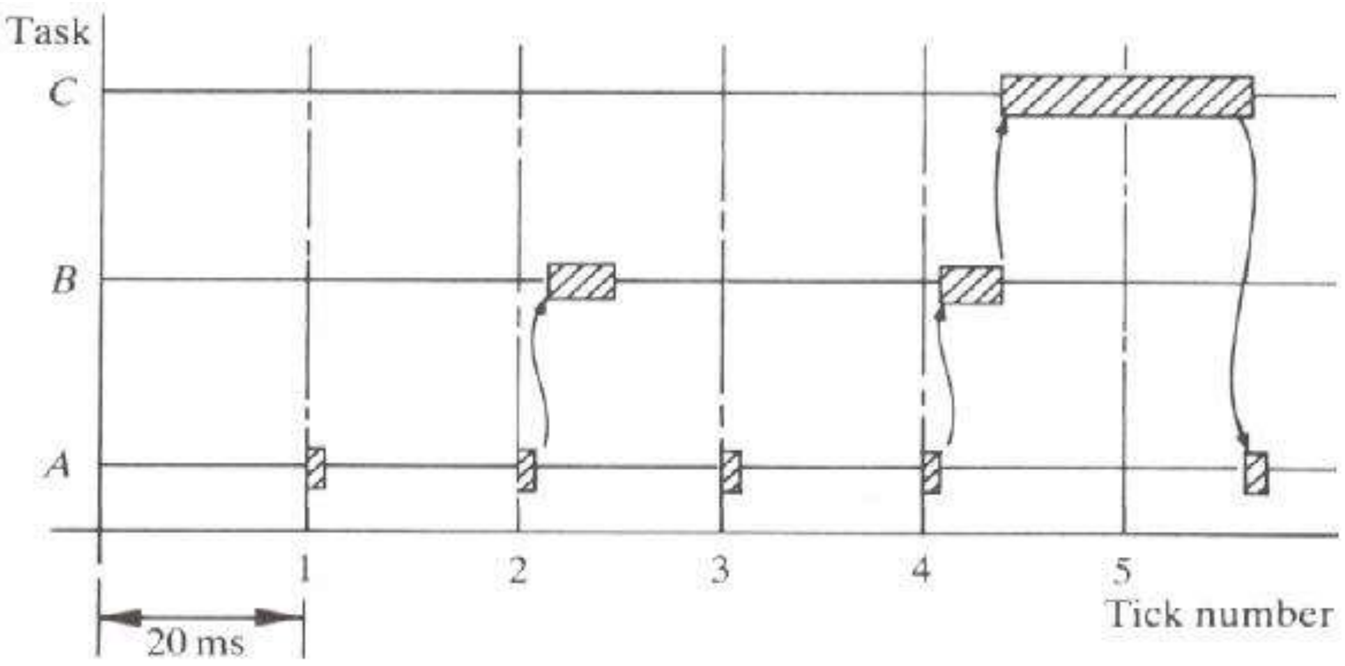
Task activation diagram for task priorities A,B,C·



Task activation diagram for task priorities C,A,B.

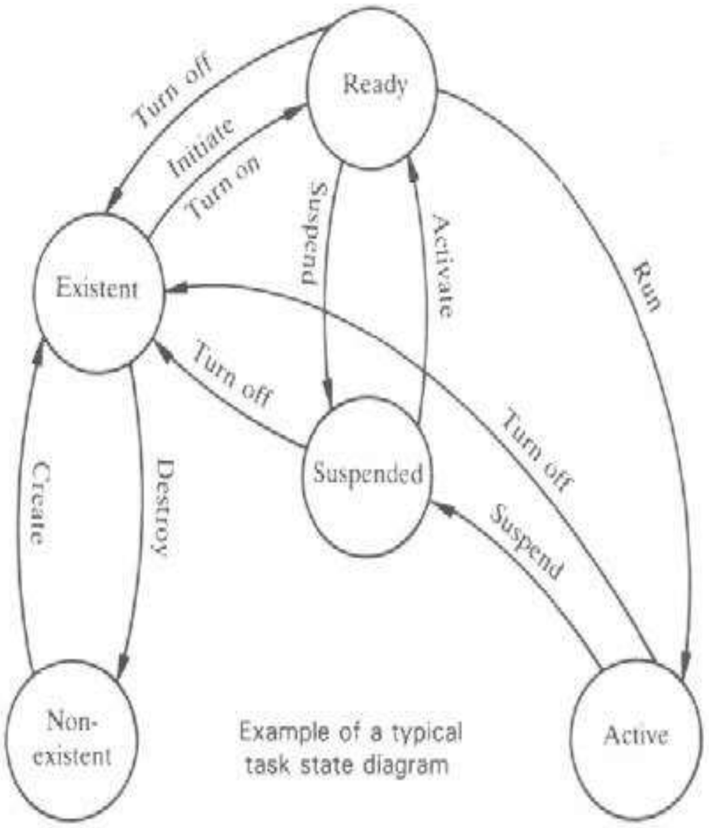
## EXAMPLE:

- Now assume that task C takes 25 msec to complete, task A takes 1 msec and task B takes 6 msec. if task C is allowed to run until completion then the activity diagram is given bellow.
- Task A will be delayed by 11 msec at every fourth invocation..



## TASK STATES:

- Tasks are in one of four states:
  - 1. Running
  - 2. Ready to Run (but not running)
  - 3. Waiting (for something other than the CPU.)
  - 4. Inactive
- Only one task can be Running at a time (unless we are using a "multi-core" CPU).
- A task which is waiting for the CPU is Ready. When a task has requested I/O or put itself to sleep, it is Waiting.
- An Inactive task is waiting to be allowed into the schedule. It is like Microsoft Word when you are NOT running it.



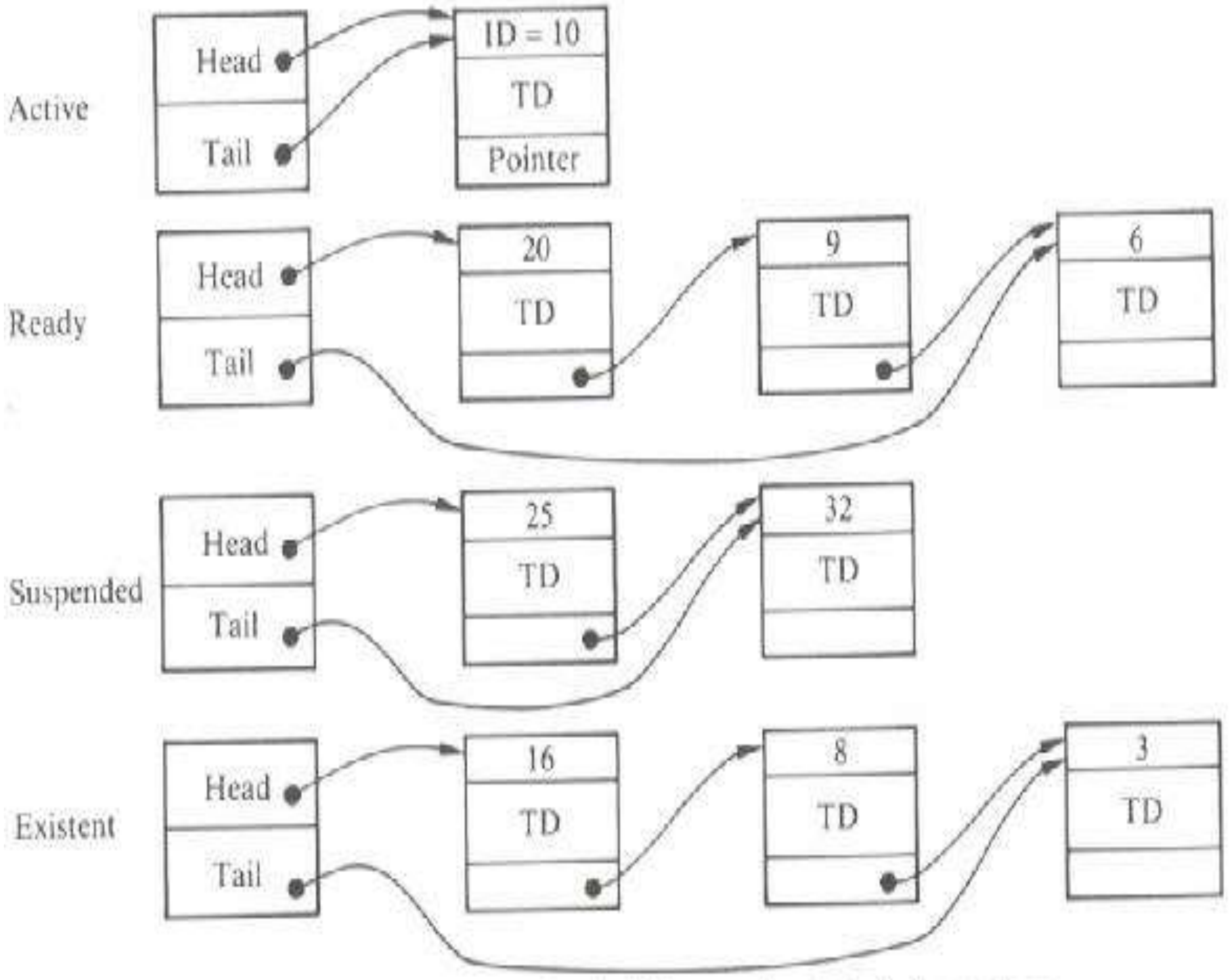
Example of a typical task state diagram

## TASK DESCRIPTOR:

- Information about the status of each task is held in a block of memory by the RTOS. This block is called Task Descriptor (TD), or Task Control Block (TCB) or Task Data Control (TDC).
- The information is held in the TD will vary from system to system, but will typically consist of the following:
  - – Task Identification.
  - – Task Priority.
  - – Current state of task.
  - – Area to store volatile environment (or a pointer to an area for storing the volatile environment).
  - – Pointer to next task in list.

## Example:

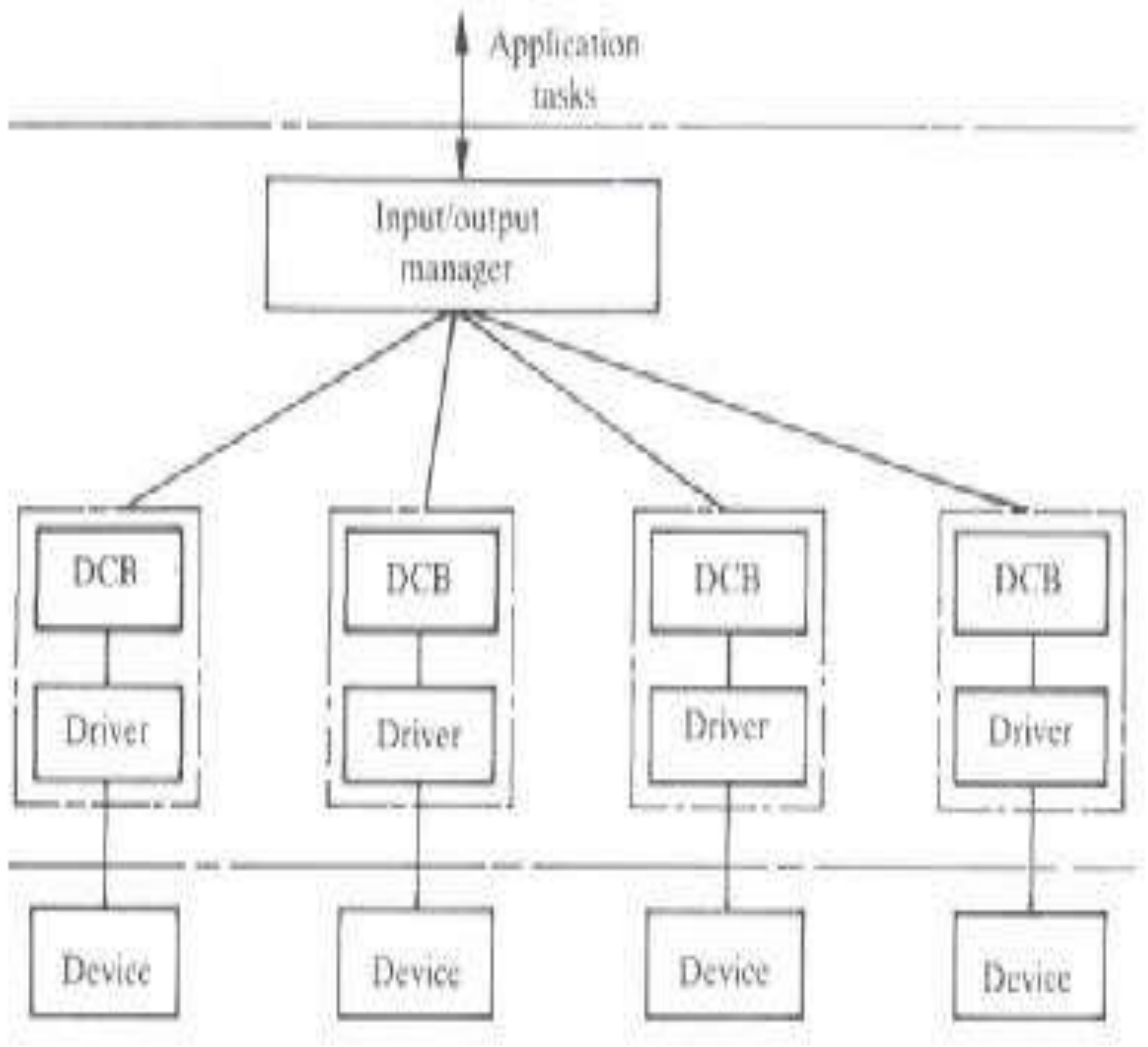
- The next slide shows list structure for holding task state information:
  - There is one active task (task ID=10).
  - There are three tasks ready to run (ID=20, ID=9 and ID=6). The entry held in the executive for the ready queue head points to task 20, which in tern points to task 9 and so on.
  - The advantage of the list structure is that the actual TD can be located anywhere in the memory and hence the OS is not restricted to a fixed number of tasks as the case in older OSs which used fixed length tables to hold task state information.



List structure for holding task state information

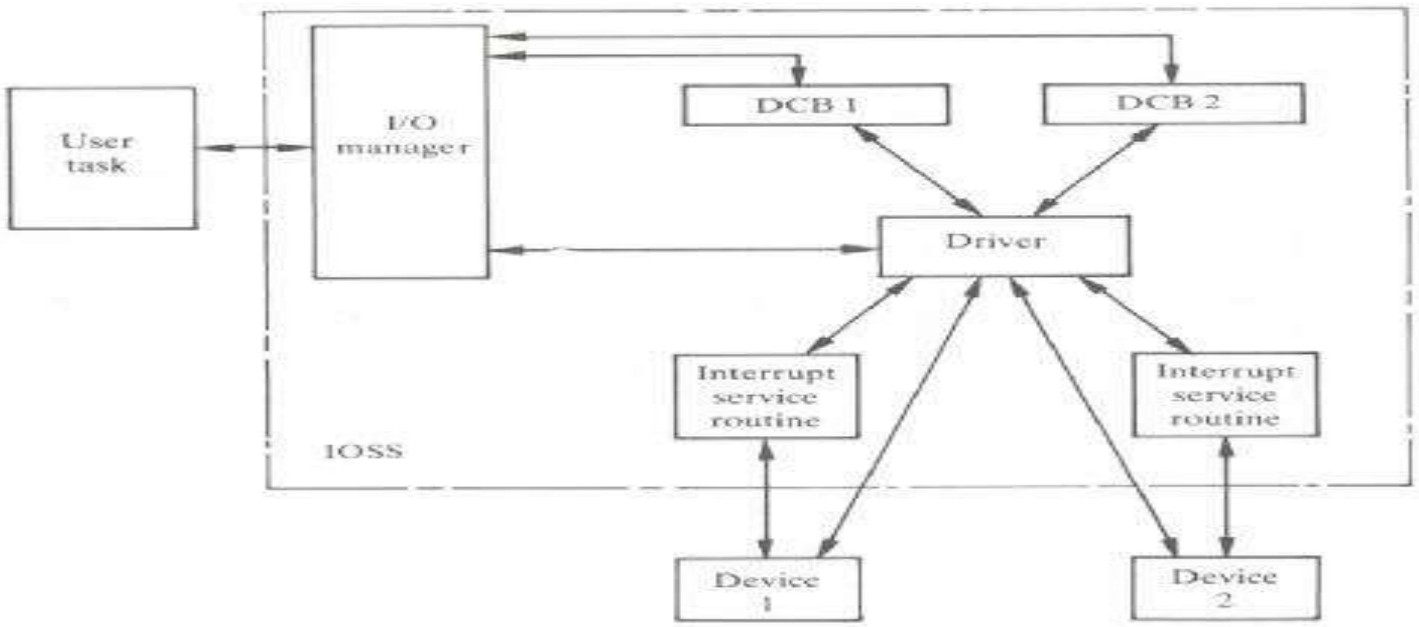
## Resource Control:

- One of the most difficult areas of programming is the transfer of information to and from external devices. The availability of a well-designed and implemented I/O subsystem (IOSS) in an OS is essential for efficient programming. This enables programmer to perform input output by means of system calls either from a HLL or from the assembler. The IOSS handles all the details of the devices.
- A typical IOSS will be divided into two levels.
- The **I/O manager accepts the system calls from** the user tasks and transfers the information contained in the calls to the **device control block (DCB) for the particular device.**
- The information supplied in the call by the user task will be;
  - – the location of a buffer area in which the data to be transferred is stored (o/p) or is to be stored (i/p),
  - – the amount of data to be transferred,
  - – type of data,
  - – direction of transfer, and
  - – the device to be used.


Application tasks
Input/output manager
DCB
Driver
DCB
Driver
DCB
Driver
DCB
Driver
Device
Device
Device
Device

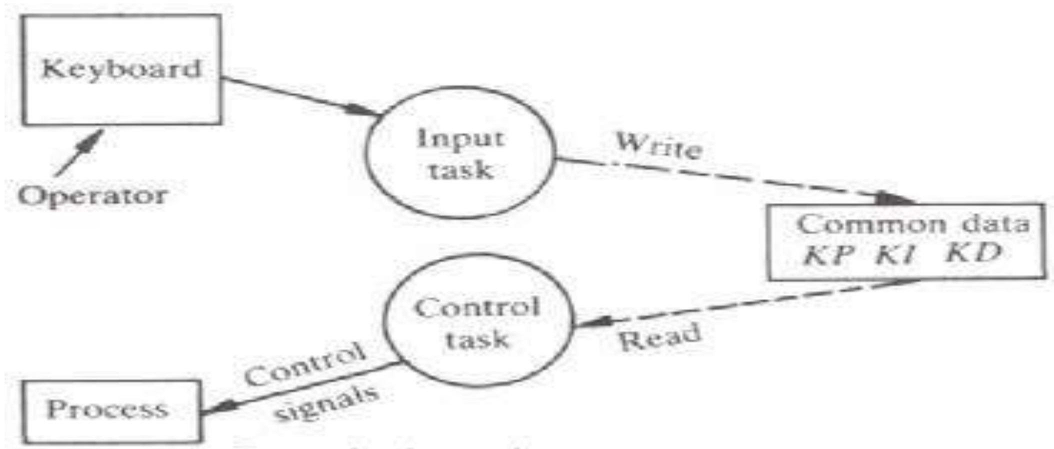
## DETAILED ARRANGEMENT OF IOSS:

- The actual transfer of the data between the user task and the device will be carried out by the device driver and this segment of code will make use of other information stored in the DCB.
- A separate device driver may be provided for each device.
- A single driver may be shared between several devices, however, each device will require its own DCB.
- The OS will normally be supplied with DCBs for the more common devices.



## Mutual Exclusion:

- Consider the transfer of information from i/p task to a control task. The i/p task gets the values for the controller i/p parameters (gain, Ti and Td). From these it computes the controller parameters (KP, KI, and KD) and these are transferred to the CONTROL task.
- A simple method is to hold the parameters values in an area of memory (common data area) and hence is accessible to both tasks.



Data sharing using common memory

Thank you☺

# Real Time Systems

## Unit:2 Concepts of Real-Time Computer Control Systems

By

Hamsashree M K
Asst. Professor
Dept of ECE
BGSIT

## LECTURE OUTLINE:

- Concepts of computer control systems.
- Analog and digital control.
- Data acquisition system.
- Sequence control.
- Direct digital control.
- Adaptive control.
- Supervisory control.
- Centralized and distributed computer control.
- Human computer interface..

## Activities of Computer Control

The activities being carried out by a computer , in a RTS, will include the following:

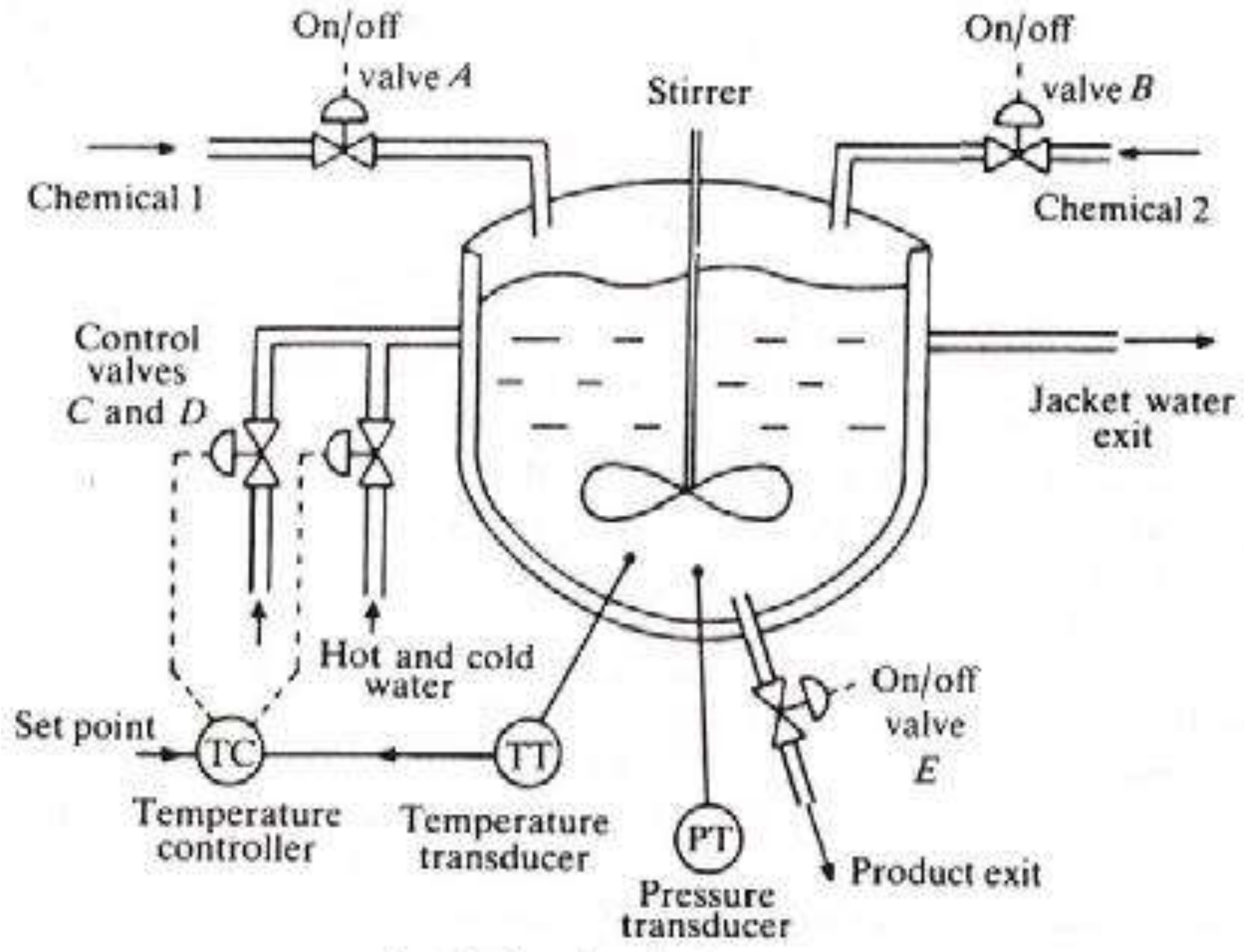
- - Data acquisition .
- - Sequence control .
- - Direct digital control (DDC).
- - Supervisory control (SC).
- - Data analysis .
- - Data storage .
- - Human – computer interface (HCI).

## Objectives of Computer Control

The objectives of using a computer in a RTS will include the following :

- - Efficiency of operation
- - Ease of operation .
- - Safety.
- - Improved products
- - Reduction in waste .
- - Reduced environmental impact .
- - Reduction in direct labor .

## Sequence Control:

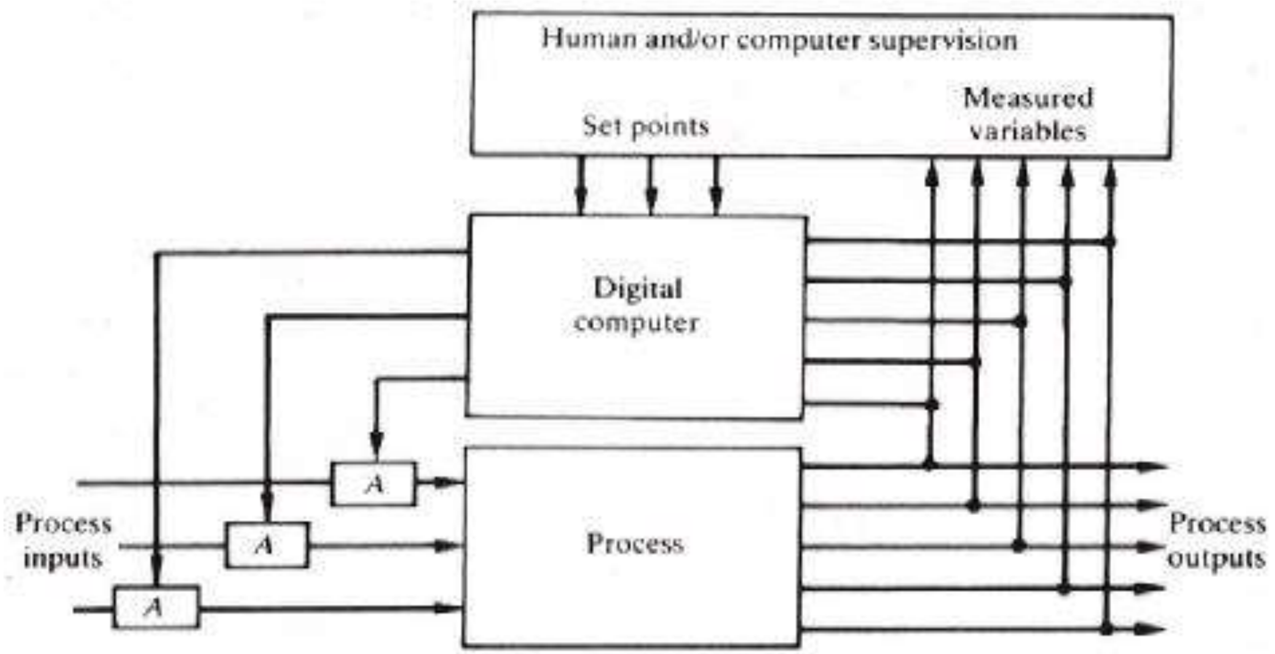


A simple chemical reactor vessel

- Sequence control systems are widely used in the food processing and chemical industries.
- The procedure for this simple reactor are:
  - 1. Open valve A.
  - 2. Check the level of chemical 1.
  - 3. Start the stirrer to mix the chemical reactor.
  - 4. Repeat steps 1 and 2 with valve B.
  - 5. Switch ON the PID controller.
  - 6. Monitor the reaction temp, when it reaches the set-point, start a timer.
  - 7. When the timer indicates that the reaction is complete, switch OFF the controller and open valve C to cool down the reactor contents. Switch OFF the stirrer.
  - 8. Monitor the temp, when the contents have cooled, open valve E to remove the product from the reactor.

## Direct Digital Control (DDC):

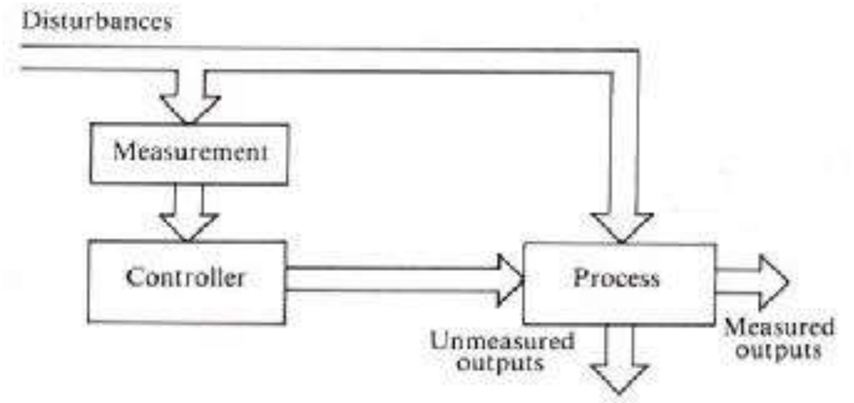
- The computer is in the feedback loop of the system. It is a critical component in terms of the reliability of the system.
- In the event of a failure of the computer, the system remains in a safe condition.
- The advantages for DDC over analog control are:
  - 1. Cost. 2. Performance. 3. Safety.



Direct digital control.

## DDC TECHNIQUES

- Feedback control
- Inferential control
- Feed-forward control



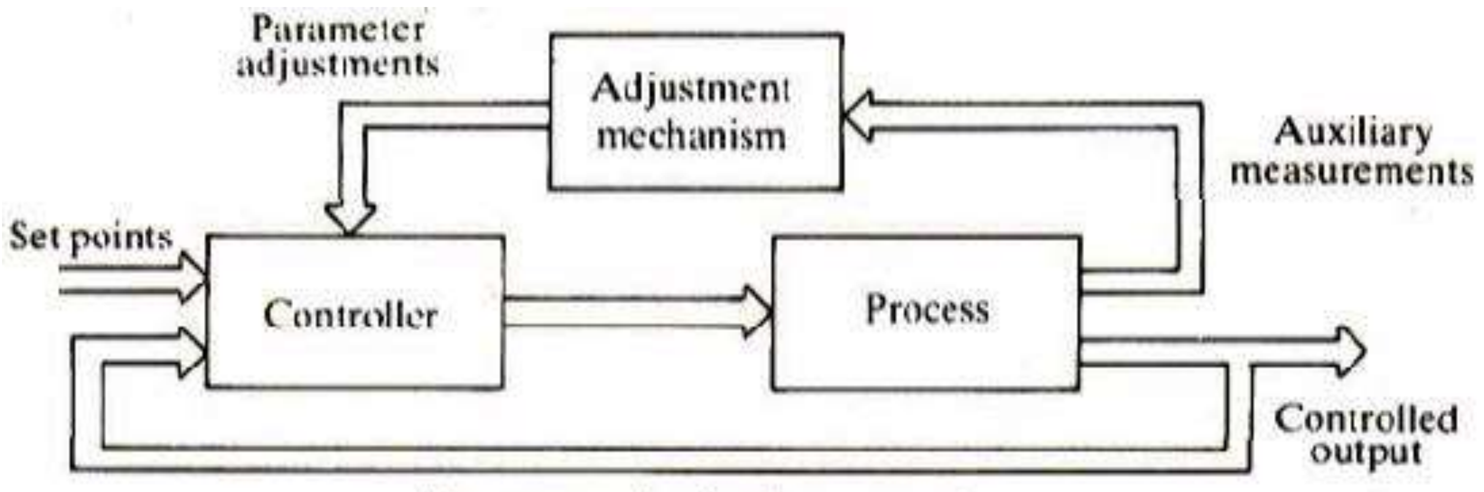
General structure of a feedforward control configuration.

- Adaptive control

## ADAPTIVE CONTROL:

Adaptive control can take several forms. Three of the most common are:

- 1. Preprogrammed adaptive control.
- 2. Self-tuning control.
- 3. Model-reference adaptive control.

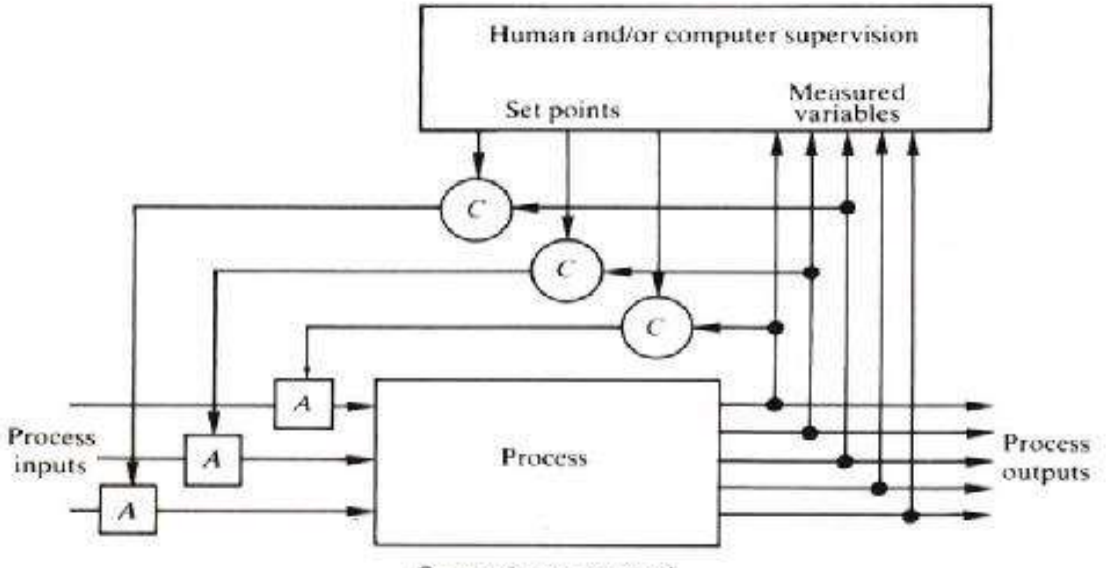


Programmed adaptive control

## SUPERVISORY CONTROL:

Many of early computer control schemes used the computer in a supervisory role and not for DDC. The main reason for this were;

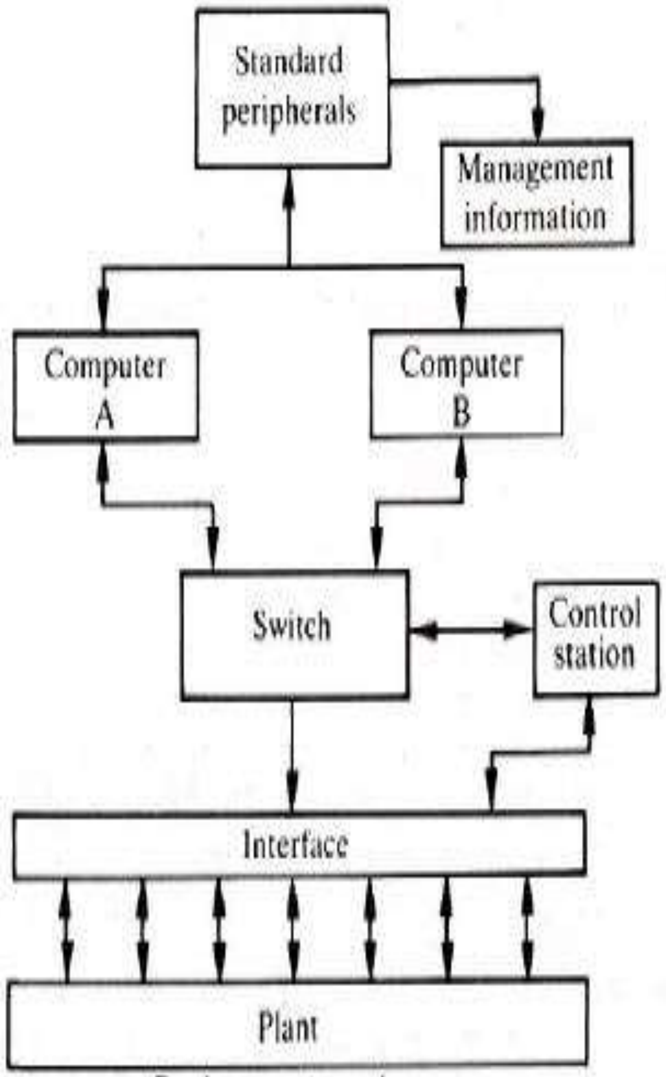
- 1. Computers were not always very reliable and caution dictated that the plant should still be able to run in the event of a computer failure.
- 2. computers were very expensive and it was not economically viable to replace the analog control equipment in current use.



Supervisory control.

## Centralized Computer System:

- Most of the 1960s computer control systems implied the use of one central computer for the control of the whole plant. The reason for this was largely financial (computers were expensive).
- By 1970 the cost of computer hardware had reduced to such an extent that it became feasible to consider the use of dual computer systems.
- Automatic failure and change-over equipment when used becomes a critical component.
- The continued reduction of the cost of hardware and the development of the microprocessor has made multi-computer systems feasible. These fall into two types:
    - 1. Hierarchical systems : tasks are divided according to function, e.g.: one computer performing DDC.
    - 2. Distributed systems : many computers perform essentially similar tasks in parallel.

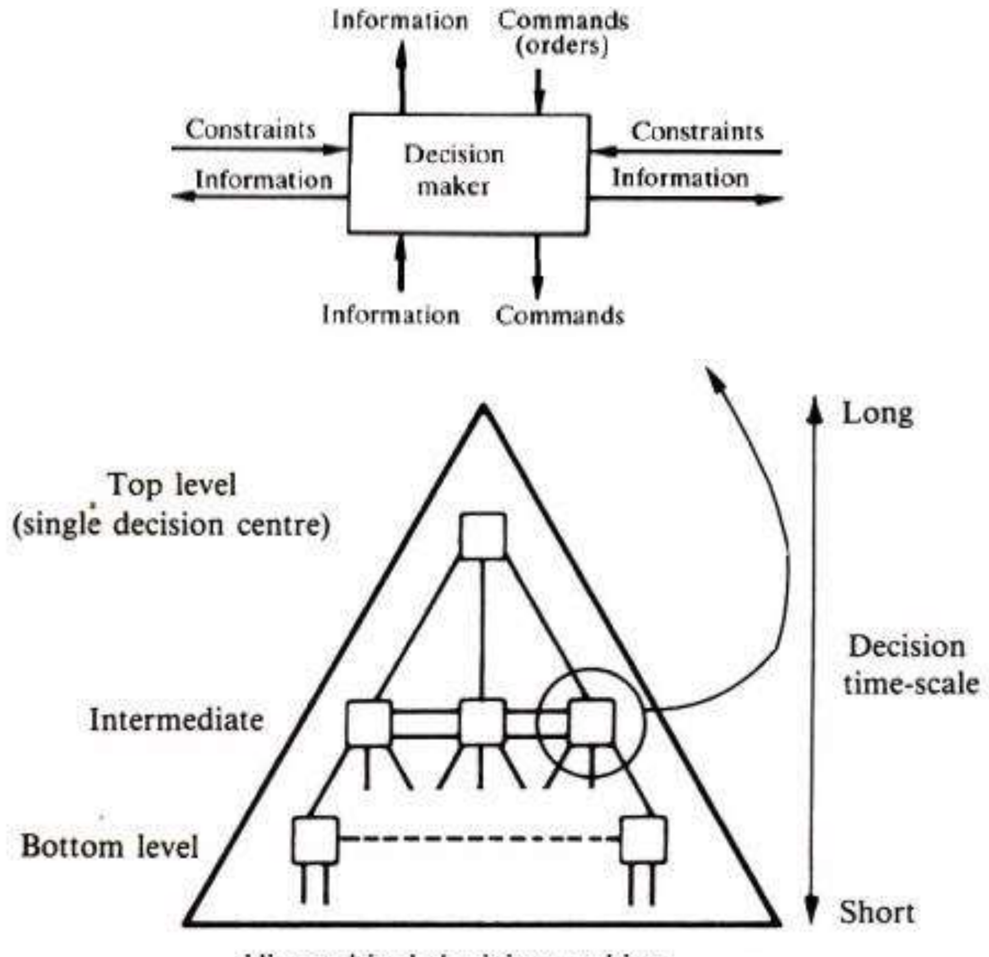


Dual computer scheme

## Multi-Computer Systems:

- Several computers can be configured for real-time computer control applications.
- These include dual computer systems to increase reliability, and distributed and hierarchical configurations.
  - **1. Hierarchical Systems: tasks are divided according to function, for example;** one computer performing DDC, other performing sequence control, other performing supervisory control..
  - **2. Distributed Systems: many computers perform essentially similar tasks in** parallel.

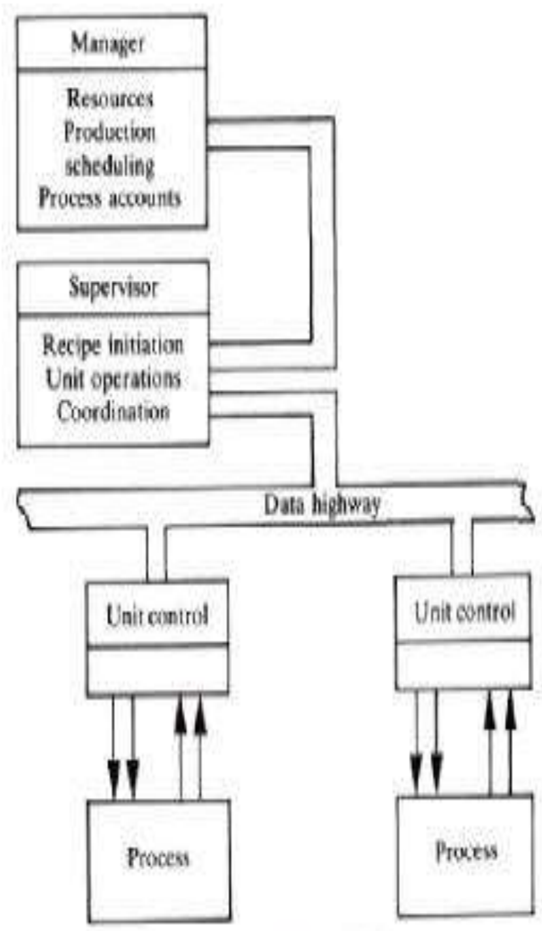
## Hierarchical Decision Making:



Hierarchical decision-making

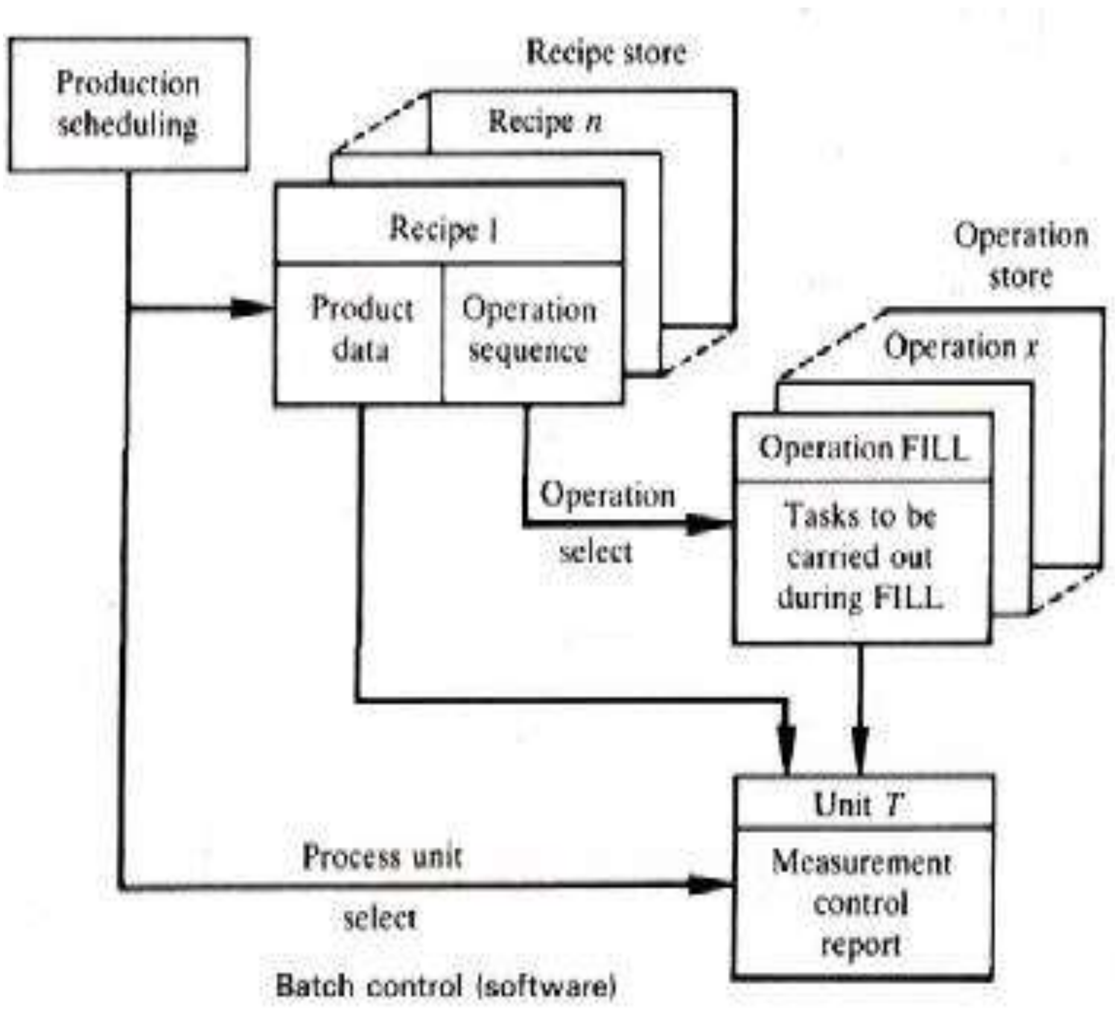
## Hierarchical System: An Example

- A typical example of a hierarchical system is the batch system given below.
- It has three levels; Manager, Supervisor, and unit Control.
- It is assumed that single computers are used for manager and supervisor functions, and that for each processing unit a single unit control computer is used.



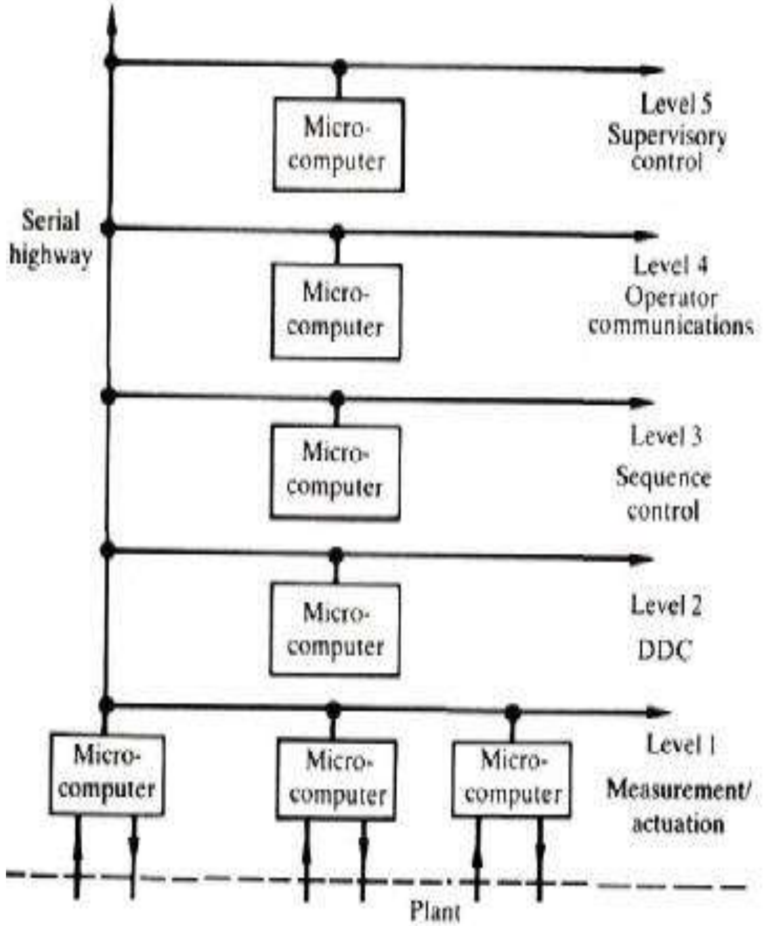
Batch control using a hierarchical system.

## Hierarchical System: An Example



## DISTRIBUTED SYSTEMS:

- In real-time systems , consider:
  - Each unit is carrying out essentially similar tasks to all the other units.
  - In the event of failure or overloading of a particular unit all or some of the work can be transferred to other units.



A distributed and hierarchical system

## ADVANTAGES:

- 1. Sharing of tasks between µCs.
- 2. More flexible than using one µC.
- 3. Failure of a unit will cause less disruption.
- 4. It is easier to make changes .
- 5. Linking by serial highway means that the computer units can be widely dispersed .

Thank you☺

# Real Time Systems

## Unit:3 Computer Hardware Requirements for Real-Time Applications

By

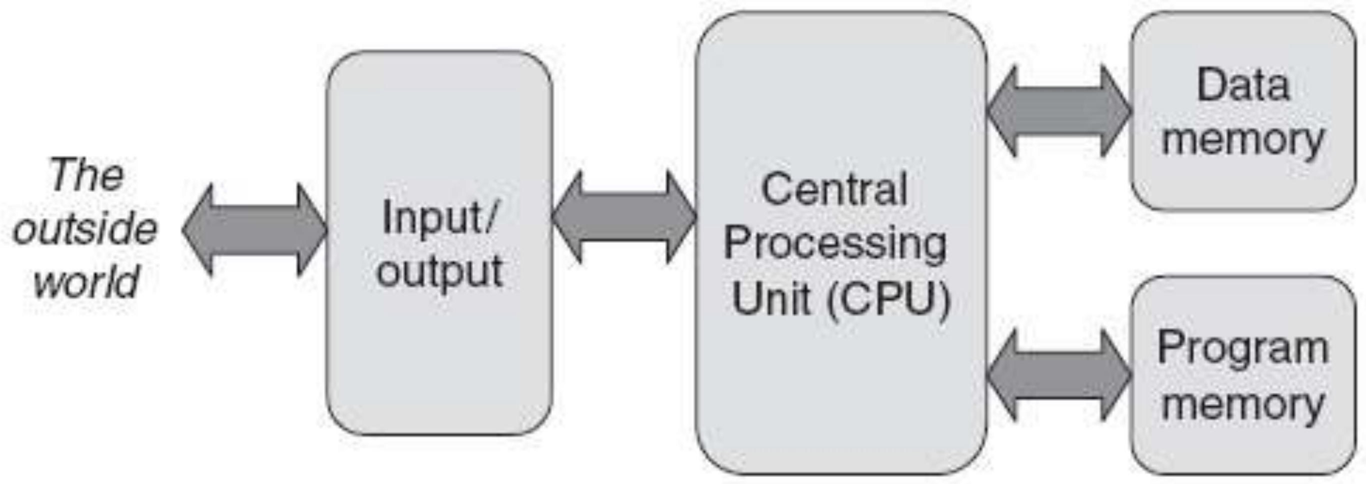
Hamsashree M K
Asst. Professor
Dept of ECE
BGSIT

## Lecture Outline:

- Features of microcomputers and microcontrollers.
- Standard interfacing techniques.
- Digital input/output interface.
- Analog input/output interface.
- Pulse input/output interface.
- Data acquisition system design.
- Management of data acquisition system.

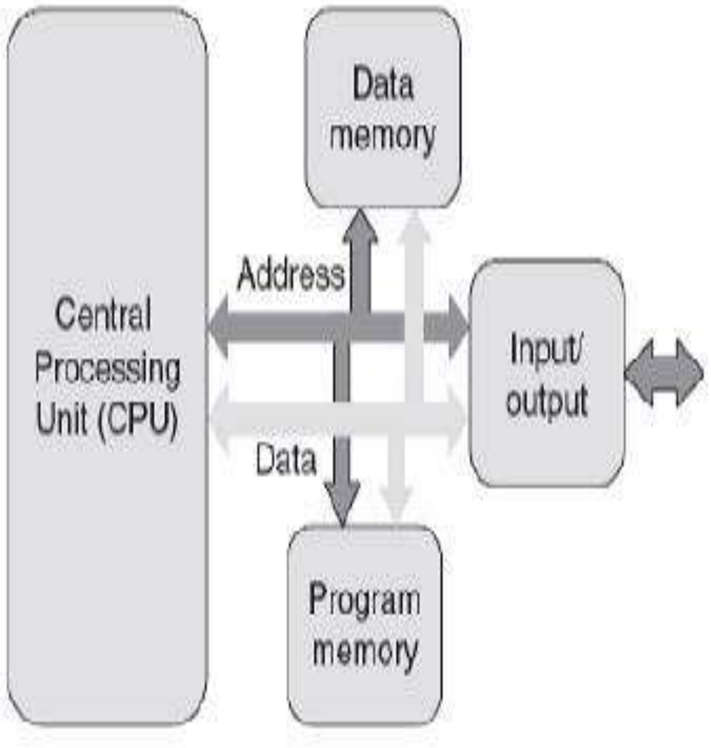
## Microcomputers & Microcontrollers:

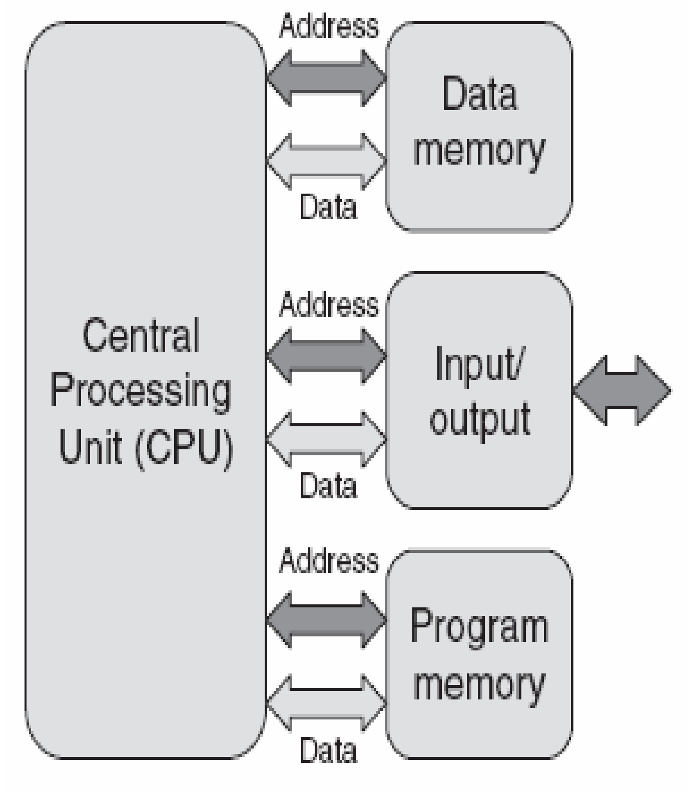
- General purpose microprocessors include the Intel xx86 series, Motorola 680xx series, National 32xxx series, and the Zilog Z8000 series.
- The ALU together with control unit and the general purpose registers make up the CPU.
- The CPU, memory and input/output units represent a microcomputer. The CPU in a single chip microcomputer or a microcomputer board is called microprocessor.



## COMPUTER ARCHITECTURE:

- 1. The Von Neumann System.
- 2. The Harvard System.





## General-Purpose Computer :

**1. CPU : Features :**

– Word length. - Instruction set .
– Addressing methods. - No. of registers.
– Information transfer rates. - Interrupt structure.

**2. Storage:**

– RAM, ROM, EPROM and auxiliary storage unit .
– DMA for fast I/O information transfer.

**3. Input and Output:**

– Process I/O
– Operator I/O
– Computer I/O

**4. Bus structure:**

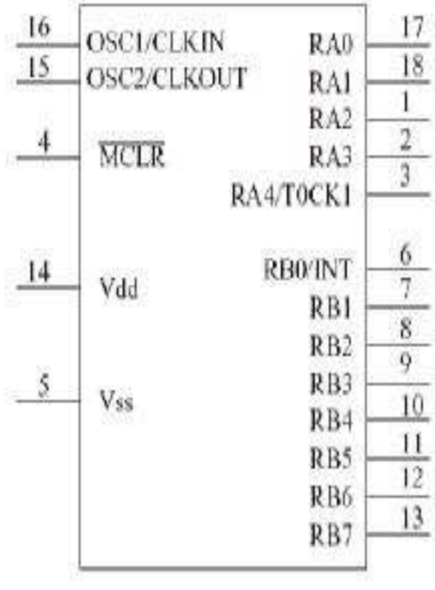
– Mechanical (physical) structure
– Electrical
– Functional

## Specialized Computers:

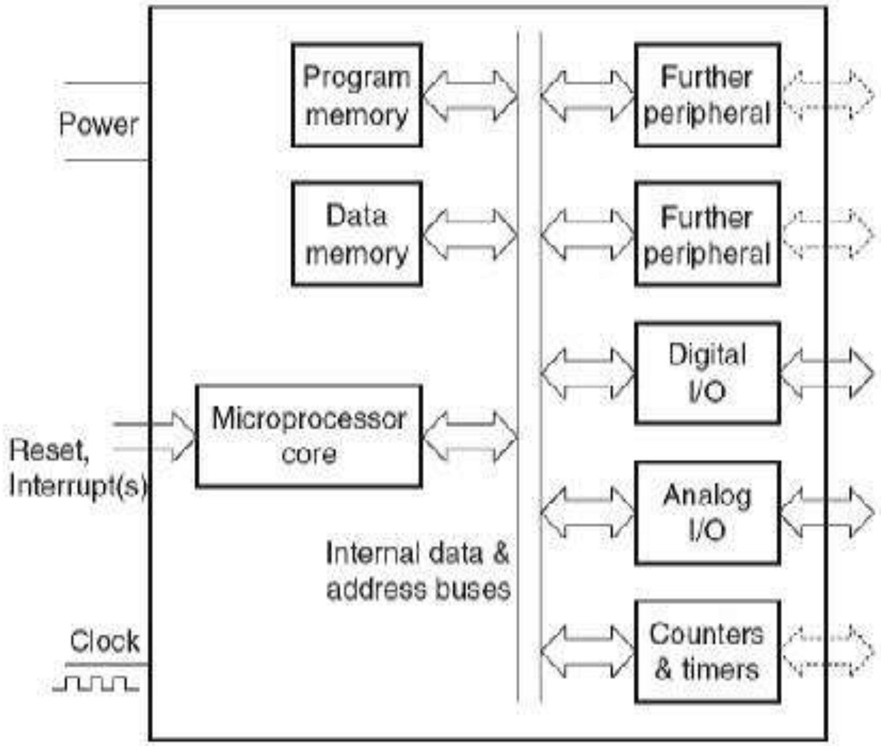
- Specialized processors have been developed for two main purposes:
  - – Safety-critical applications.
  - – Increased computation speed .
- For safety-critical applications , use RISC computers.
- The advantage of simplifying the instruction set is:
  - 1. The possibility of formal verification (using math. proofs) that the logic of the processor is correct.
  - 2. It is easier to write assemblers and compilers for simple instruction set.
- Many different forms of parallel computer architecture have been used SIMD, MISD, and MIMD .
- Digital signal processors .

## Single Chip Microcontrollers:

- Small amount of RAM and EPROM , it can be extended.
- • Instruction set .
- • DAC and ADC
- • Interrupt structure
- • I/O lines
- • Timers.



PIC16F84 pin configuration



## Microcontroller Selection:

Comparison of PIC families

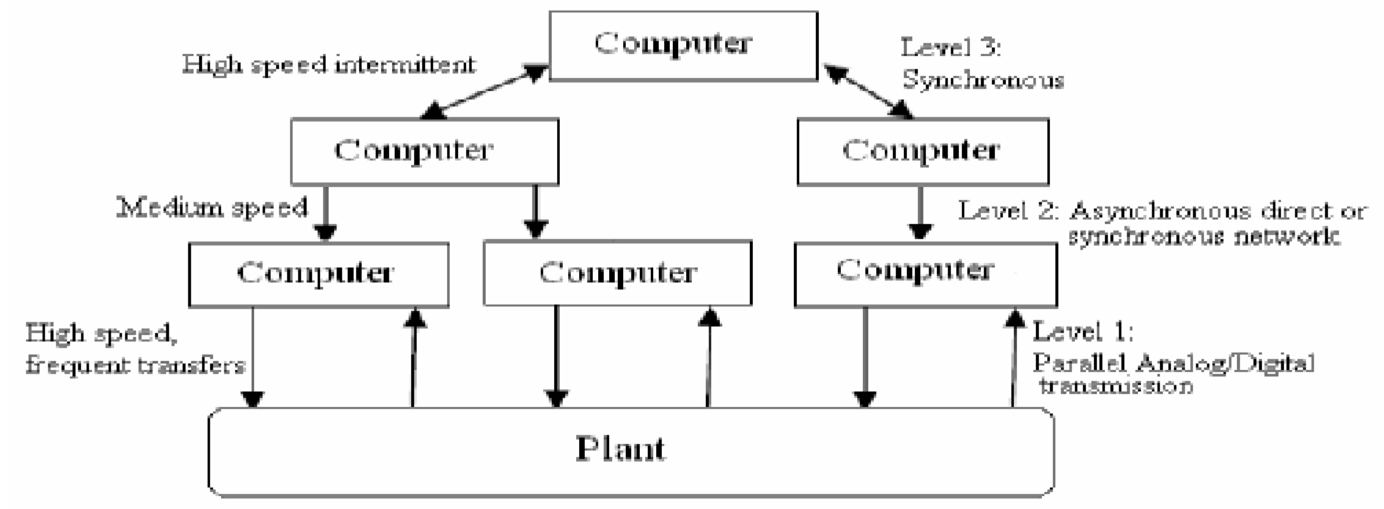
| PIC family | Stack size (words) | Instruction word size | Number of instructions | Interrupt vectors |
|---|---|---|---|---|
| 12CXXX/12FXXX | 2 | 12- or 14-bit | 33 | None |
| 16C5XX/16F5XX | 2 | 12-bit | 33 | None |
| 16CXXX/16FXXX | 8 | 14-bit | 35 | 1 |
| 17CXXX | 16 | 16-bit | 58, including hardware multiply | 4 |
| 18CXXX/18FXXX | 32 | 16-bit | 75, including hardware multiply | 2 (prioritised) |

| Device number | No. of pins* | Clock speed | Memory (K = Kbytes, i.e. 1024 bytes) | Peripherals/special features |
|---|---|---|---|---|
| 16F84A | 18 | DC to 20 MHz | 1K program memory, 68 bytes RAM, 64 bytes EEPROM | 1 8-bit timer 1 5-bit parallel port 1 8-bit parallel port |
| 16F873A | 28 | DC to 20 MHz | 4K program memory 192 bytes RAM, 128 bytes EEPROM | 3 parallel ports, 3 counter/timers, 2 capture/compare/PWM modules, 2 serial communication modules, 5 10-bit ADC channels, 2 analog comparators |
| 16F874A | 40 | DC to 20 MHz | 4K program memory 192 bytes RAM, 128 bytes EEPROM | 5 parallel ports, 3 counter/timers, 2 capture/compare/PWM modules, 2 serial communication modules, 8 10-bit ADC channels, 2 analog comparators |

## THE PIC16F84 MICROCONTROLLER:

- The 16F84A architecture is representative of all 16 Series microcontrollers, with Harvard structure, pipelining and a RISC instruction set.
- The PIC 16F84A has a limited set of peripherals, chosen for small and low-cost applications. It is thus a smaller member of the family, with features that are a subset of any of the larger ones.
- A particular type of memory location is the Special Function Register, which acts as the link between the CPU and the peripherals.
- Reset mechanisms ensure that the CPU starts running when the appropriate operating conditions have been met, and can be used to restart the CPU in case of program failure.
- The parallel port allows ready exchange of digital data between the outside world and the controller CPU.
- It is important to understand the electrical characteristics of the parallel port and how they interact with external elements.
- A microcontroller needs a clock signal in order to operate. The characteristics of the clock oscillator determine speed of operation and timing stability, and strongly influence power consumption.
- Interrupts and counter/timers are important hardware features of almost all microcontrollers. They both carry a number of important hardware and software concepts, which must be understood.

## COMMUNICATIONS:

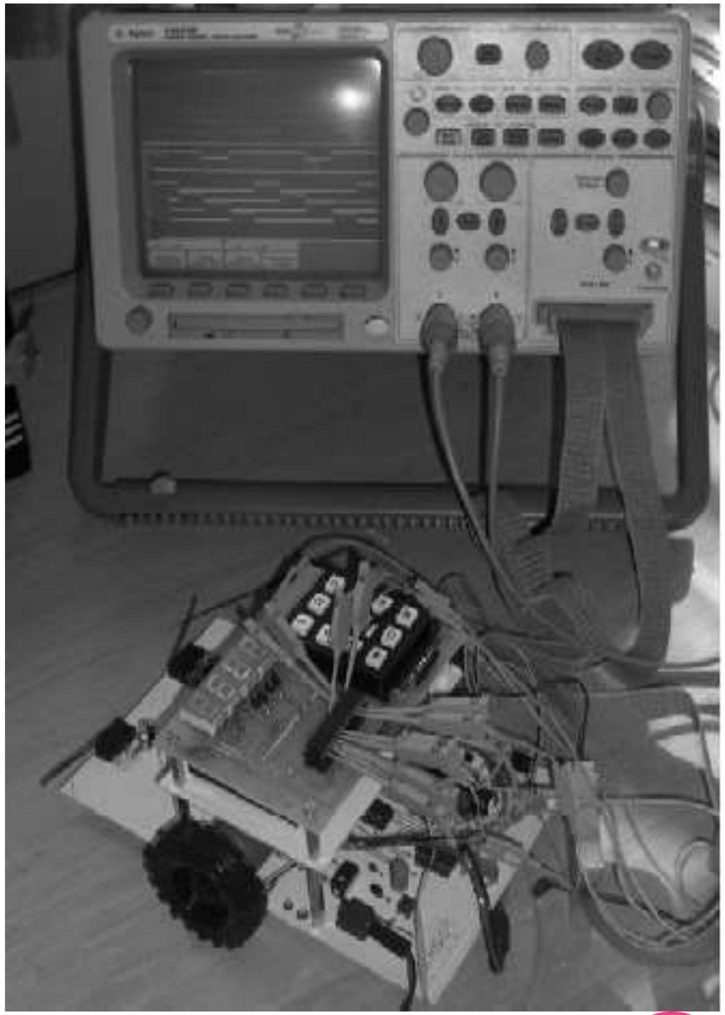

Computer
High speed intermittent
Level 3: Synchronous
Computer
Computer
Medium speed
Level 2: Asynchronous direct or synchronous network
Computer
Computer
Computer
High speed, frequent transfers
Level 1: Parallel Analog/Digital transmission
Plant

- Level (1): parallel analog/digital transmission (High speed, frequent transfer)
- Level(2): Asynchronous direct or synchronous network (Medium speed)
- Level (3): Synchronous (High speed , intermittent)

- At high levels, it is more usual to use serial communication methods due to the distances between computers (few hundred meters).
- At plant level , parallel analog and digital signal transmission techniques are involved, since the distances are small.
- Serial communication techniques can be characterized in several ways:
  - 1. Mode: Synchronous and Asynchronous .
  - 2. Quantity: Character by character and block .
  - 3. Distance: Local and remote (Wide area).
  - 4. Code: ASCII and others..

## Process Related Interface:

- Instruments and actuators connected to the plant can take a wide variety of forms; they may be used for  measuring a variable, they could be used to control an actuator.
- There is a need to convert a digital quantity to a physical quantity, or an analog signal generated from a sensor into a digital quantity.
- Most devices can be allocated to one of the following categories;
  - 1. Digital quantities.
  - 2. Analog quantities.
  - 3. Pulses and pulse rates.
  - 4. Telemetry. DDC, other performing sequence control, other performing supervisory control..



## SENSORS USED IN RT SYSTEMS:

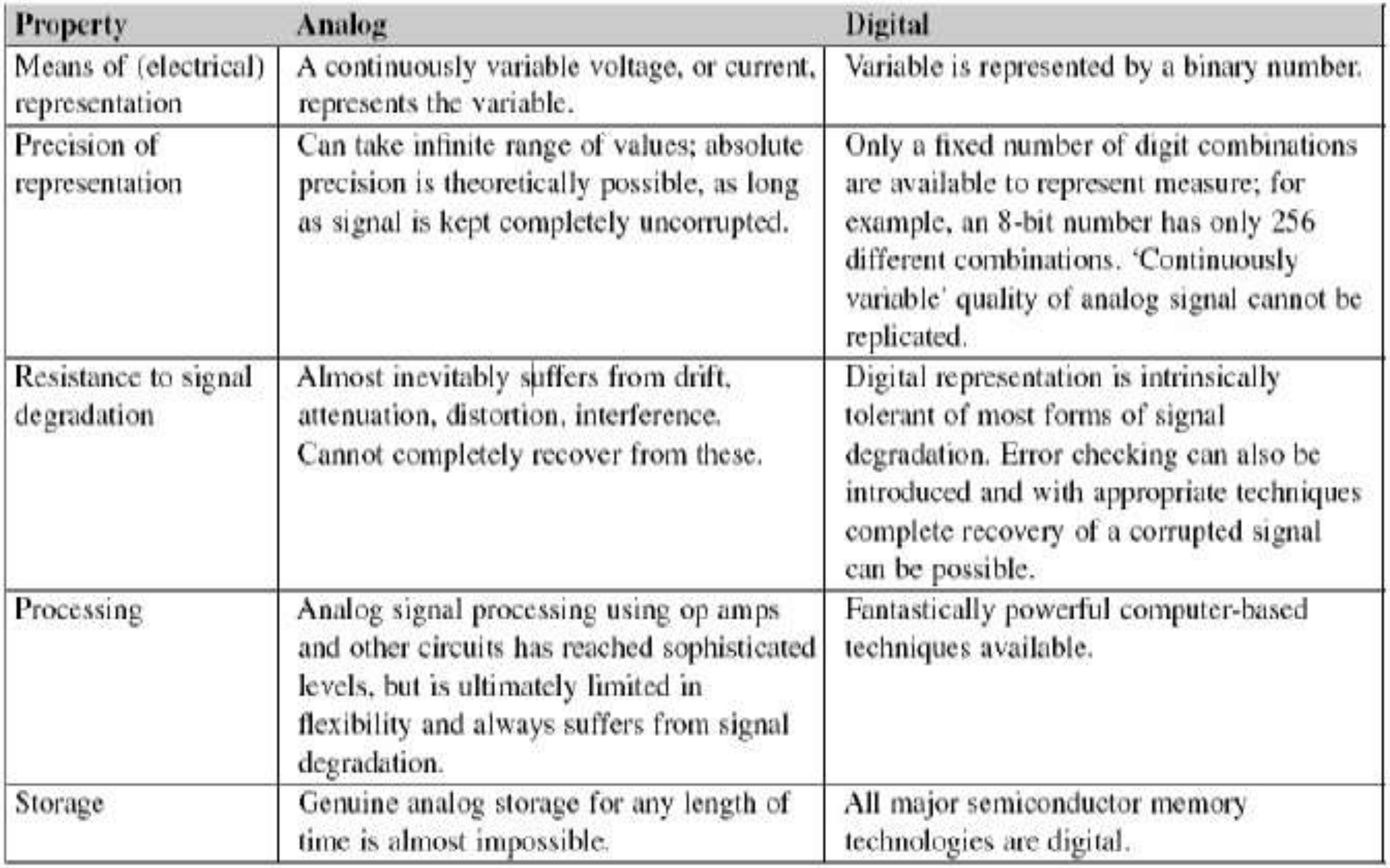
- A sensor is a device that outputs a signal which is related to the measurement of a physical quantity such as temperature, speed, force, pressure, displacement, acceleration, torque, flow, light or sound.
- Sensors are used in RT systems in the feedback loops, and they provide information about the actual output of a plant. For example, a speed sensor gives a signal proportional to the speed of a motor.
- Sensors can be classified as analog or digital;
  - – **Analog sensors** are more widely available, and their outputs are analog voltages. For example, the output of an analog temperature sensor may be a voltage proportional to the measured temperature. Analog sensors can only be connected to a computer by using an A/D converter.
  - – **Digital sensors** are not very common and they have logic level outputs which can directly be connected to a computer input port.
- The choice of a sensor for a particular application depends on many factors such as the cost, reliability, required accuracy, resolution, range and linearity of the sensor.

## THE CHOICE OF A SENSOR:

- **Range:** The range of a sensor specifies the upper and lower limits of the measured variable for which a measurement can be made. For example, if the range of a temperature sensor is specified as 10–60 LC then the sensor should only be used to measure temperatures within that range.
- **Resolution:** The resolution of a sensor is specified as the largest change in measured value that will not result in a change in the sensor's output, i.e. the measured value can change by the amount quoted by the resolution before this change can be detected by the sensor. In general, the smaller this amount the better the sensor is, and sensors with a wide range have less resolution. For example, a temperature sensor with a resolution of 0.001K is better than a sensor with a resolution of 0.1 K.
- **Repeatability:** The repeatability of a sensor is the variation of output values that can be expected when the sensor measures the same physical quantity several times. For example, if the voltage across a resistor is measured at the same time several times we may get slightly different results.
- **Linearity:** An ideal sensor is expected to have a linear transfer function, i.e. the sensor output is expected to be exactly proportional to the measured value. However, in practice all sensors exhibit some amount of nonlinearity depending upon the manufacturing tolerances and the measurement conditions.
- **Dynamic response:** The dynamic response of a sensor specifies the limits of the sensor characteristics when the sensor is subject to a sinusoidal frequency change. For example, the dynamic response of a microphone may be expressed in terms of the 3-dB bandwidth of its frequency response.

# ANALOG INPUT/OUTPUT INTERFACING:

Some properties of analog and digital quantities



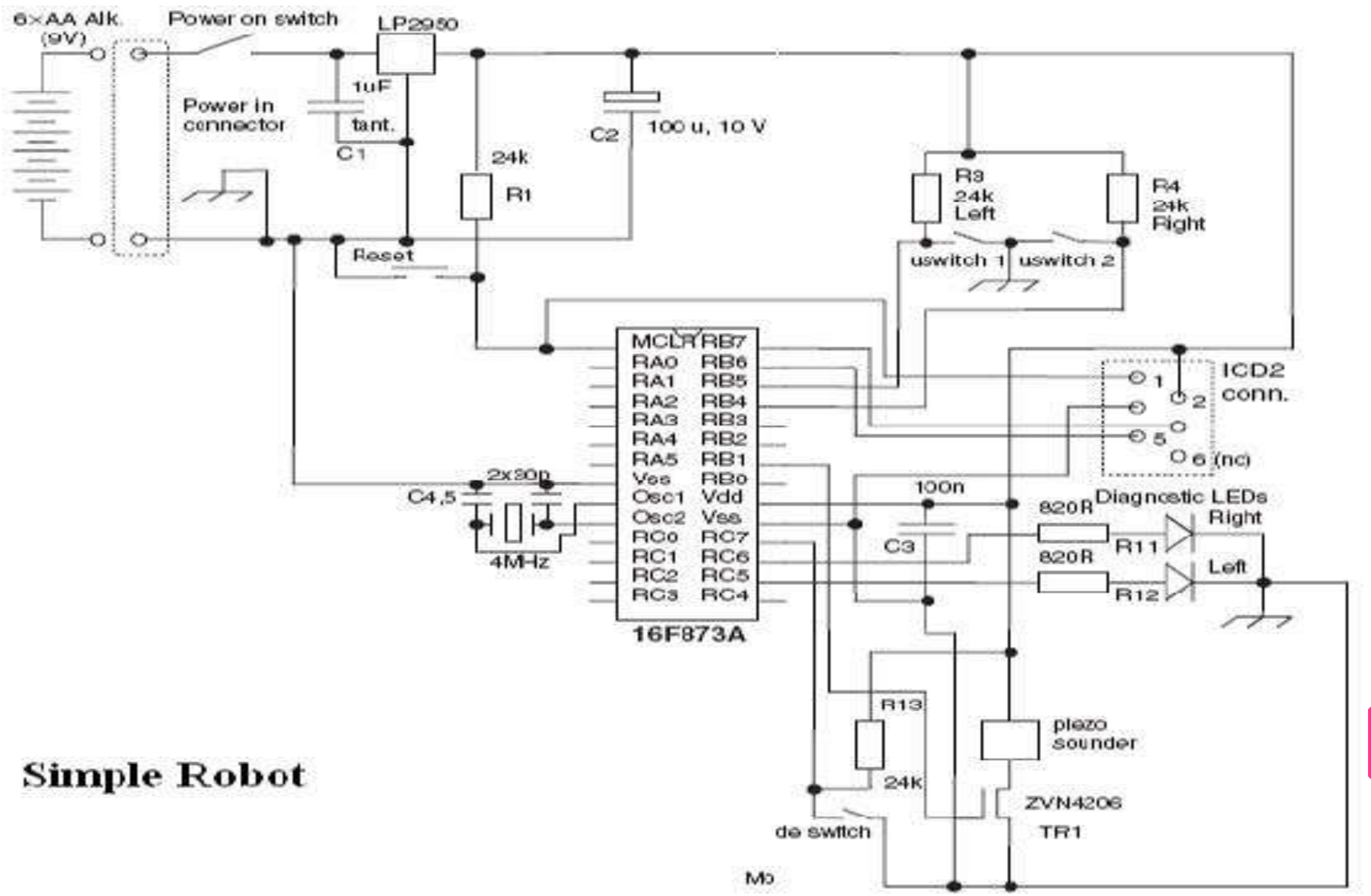
| Property | Analog | Digital |
|---|---|---|
| Means of (electrical) representation | A continuously variable voltage, or current, represents the variable. | Variable is represented by a binary number. |
| Precision of representation | Can take infinite range of values; absolute precision is theoretically possible, as long as signal is kept completely uncorrupted. | Only a fixed number of digit combinations are available to represent measure; for example, an 8-bit number has only 256 different combinations. 'Continuously variable' quality of analog signal cannot be replicated. |
| Resistance to signal degradation | Almost inevitably suffers from drift, attenuation, distortion, interference. Cannot completely recover from these. | Digital representation is intrinsically tolerant of most forms of signal degradation. Error checking can also be introduced and with appropriate techniques complete recovery of a corrupted signal can be possible. |
| Processing | Analog signal processing using op amps and other circuits has reached sophisticated levels, but is ultimately limited in flexibility and always suffers from signal degradation. | Fantastically powerful computer-based techniques available. |
| Storage | Genuine analog storage for any length of time is almost impossible. | All major semiconductor memory technologies are digital. |

## Pulse Input/Output Interfacing:

- - Reading sequence of pulses generated from a sensor.
- - Reading the width of a pulse width modulated signal.
- - Generating number of pulses with fixed frequency.
- - Generating a controllable pulse width modulated signal.

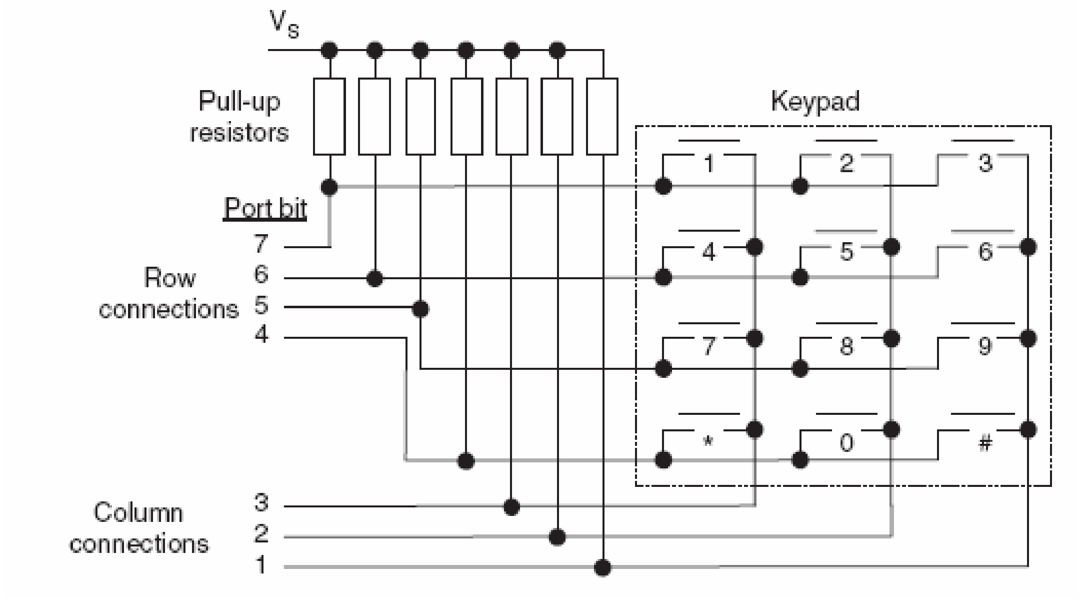
## EXAMPLE:

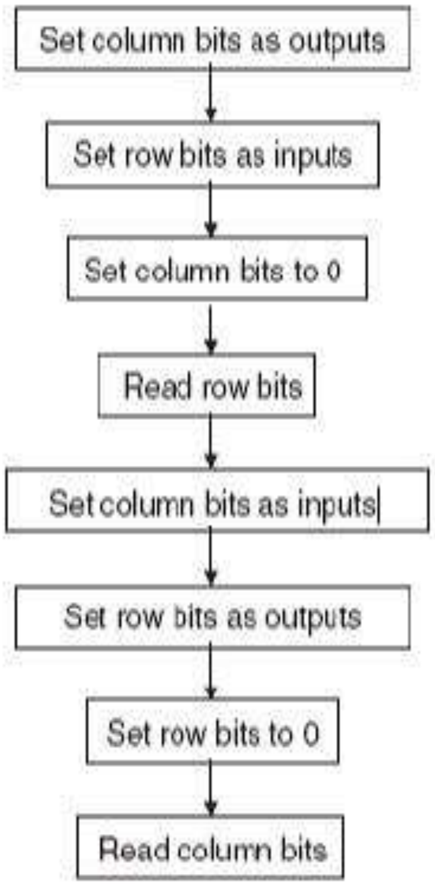
- Simple Robot System: Hardware interfacing with an 8-bit microcontroller.



Simple Robot

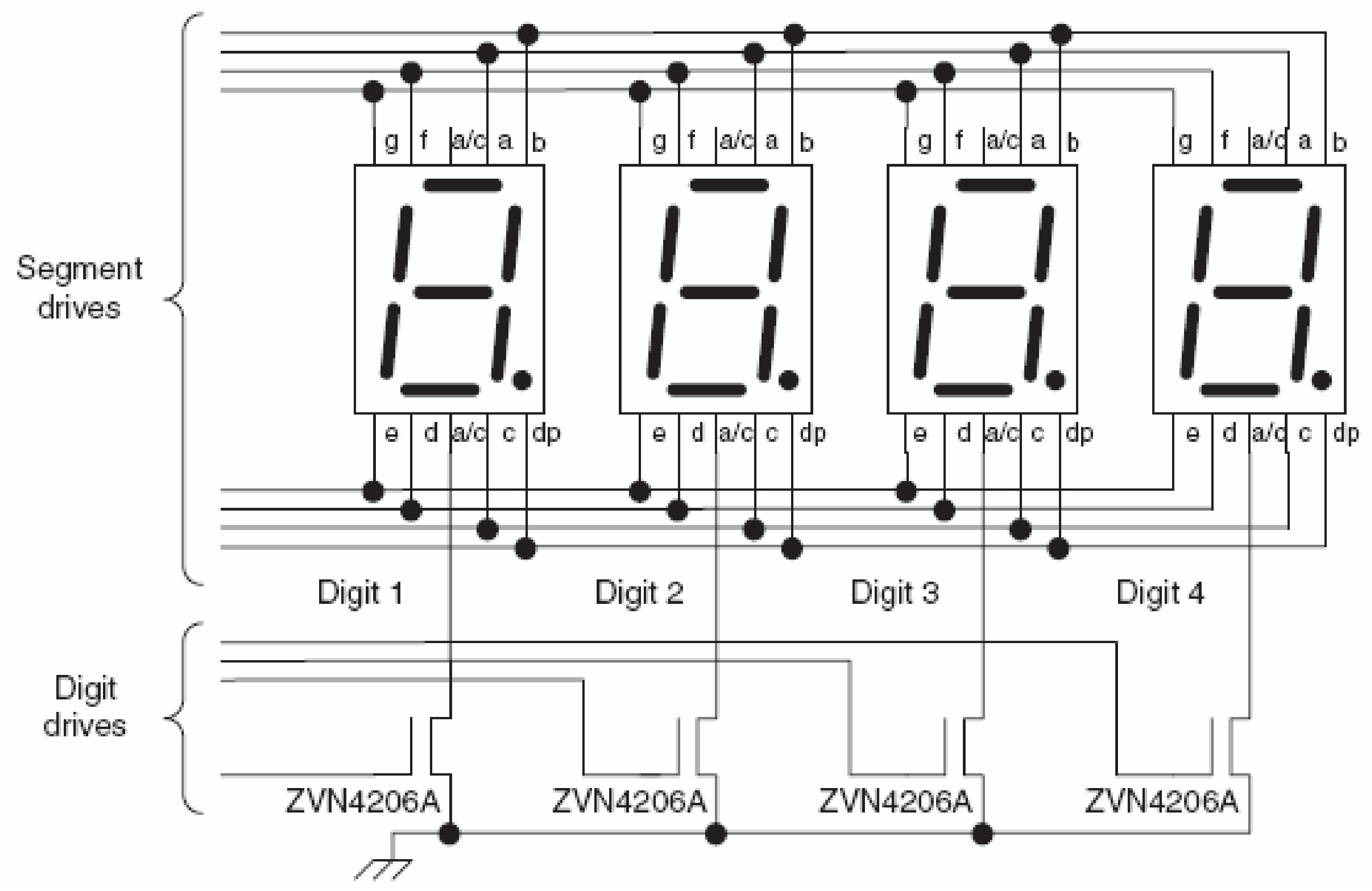
## Keypad circuit diagram with pull-up resistor:





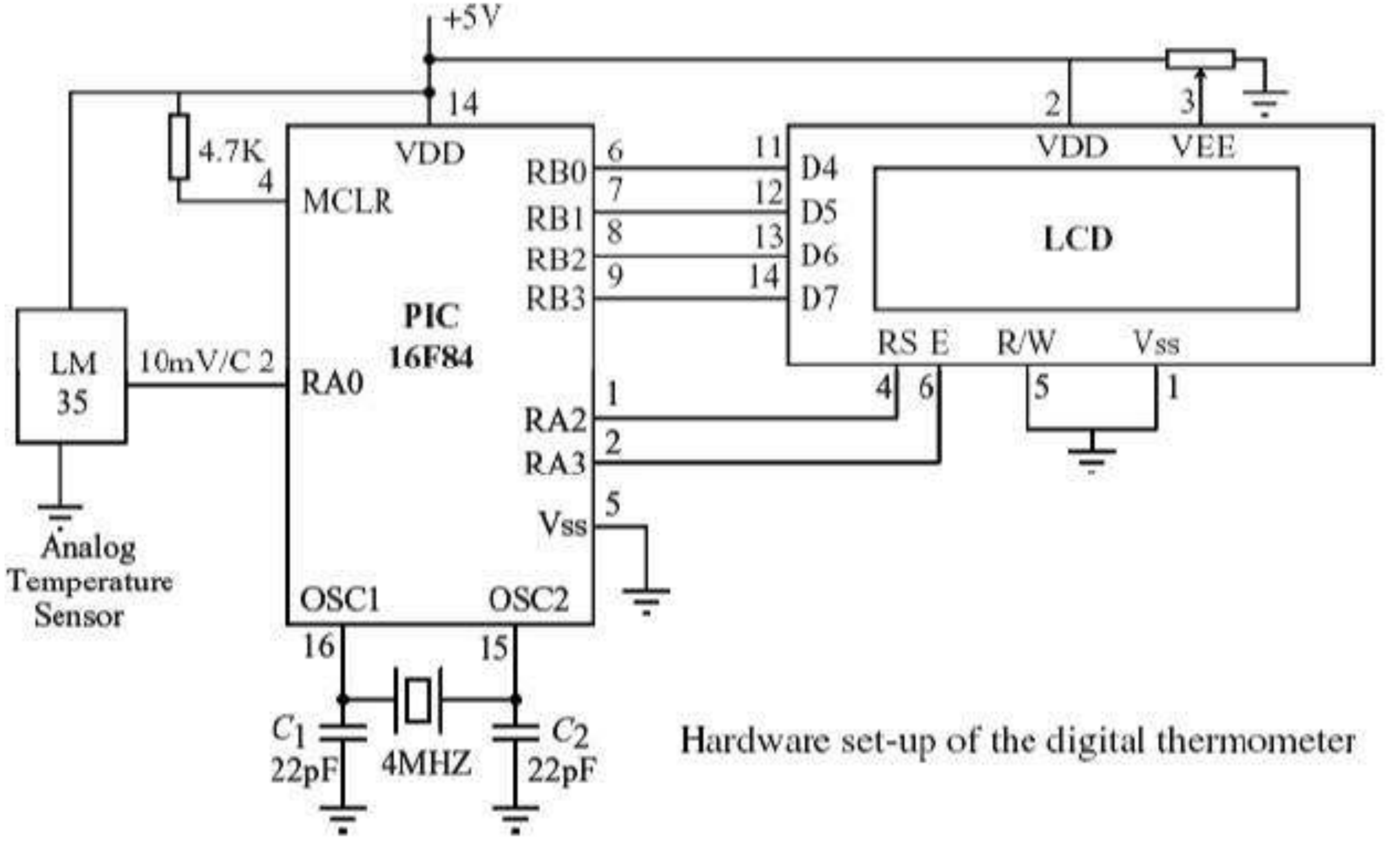
## EXAMPLE:

- Four-digit display unit design.



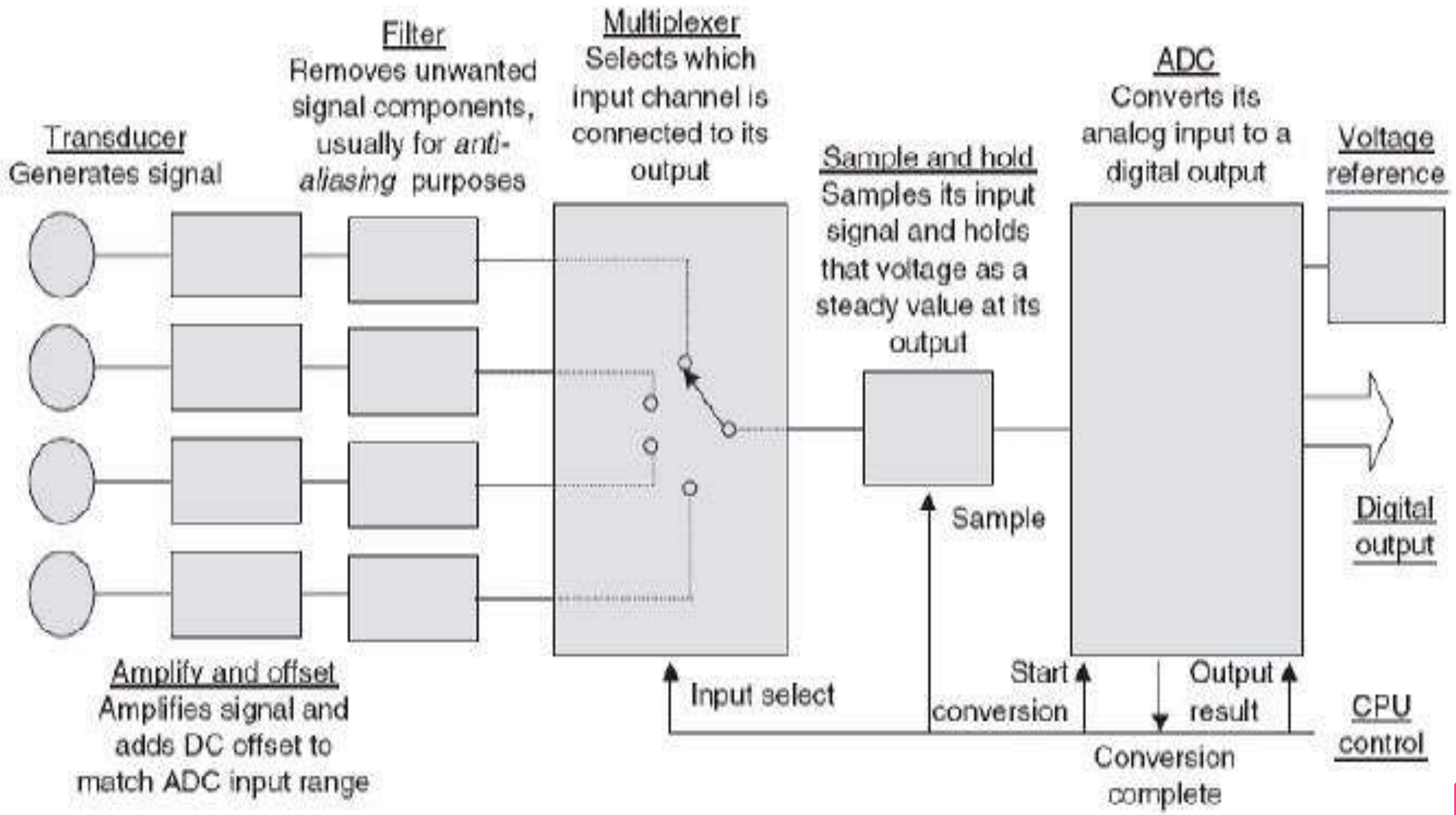
## EXAMPLE:

- LCD interfacing with an 8-bit microcontroller.



Hardware set-up of the digital thermometer

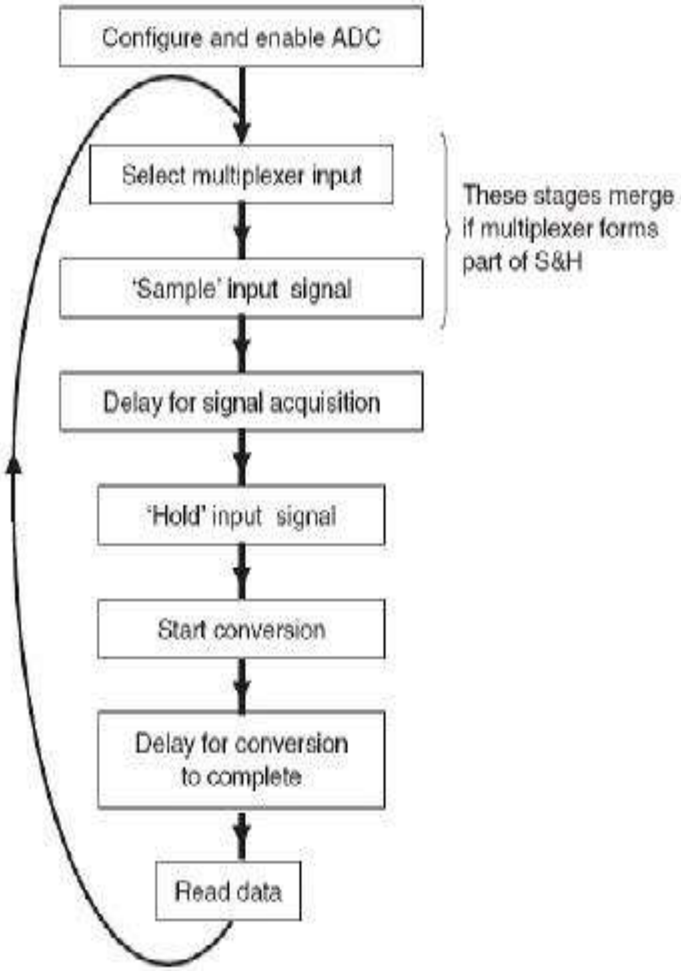
## DATA ACQUISITION SYSTEM DESIGN:


Filter
Removes unwanted signal components, usually for *anti-aliasing* purposes
Multiplexer
Selects which input channel is connected to its output
ADC
Converts its analog input to a digital output
Transducer
Generates signal
Sample and hold
Samples its input signal and holds that voltage as a steady value at its output
Voltage reference
Sample
Digital output
Amplify and offset
Amplifies signal and adds DC offset to match ADC input range
Input select
Start conversion
Output result
CPU control
Conversion complete

Elements of a (four-channel) data acquisition system

## DAS: Software Design:

- - Using flowchart.
- - Writing an assembly program.
- - Different sampling rates.
- - Selecting the suitable sampling frequency.
- - Task execution time.
- - Dealing with interrupts.
- - Memory management.



Typical timing requirement of one A-to-D conversion

Thank you☺