

REAL TIME SYSTEM

Course Code	: 18EC731	CIE Marks	: 40
Lecture Hours/Week	: 3	SEE Marks	: 60
Total Number of Lecture Hours	: 40 (08 Hrs / Module)	Exam Hours	: 03
CREDITS – 03			

Course Learning Objectives: This Course will enable students to:

- Understand the fundamentals of Real-time systems and its classifications.
- Describe the concepts of computer control and hardware components for Real-Time Application.
- Discuss the languages to develop software for Real-Time Applications.
- Explain the concepts of operating system and RTS development methodologies.

Module-1

Introduction to Real-Time Systems: Historical background, Elements of a Computer Control System, RTS- Definition, Classification of Real-time Systems, Time Constraints, Classification of Programs.

Concepts of Computer Control: Introduction, Sequence Control, Loop Control, Supervisory Control, Centralized Computer Control, Hierarchical Systems.

(Text: 1.1 to 1.6 and 2.1 to 2.6),

L1, L2

Module-2

Computer Hardware Requirements for Real-Time Applications: Introduction, General Purpose Computer, Single Chip Microcomputers and Microcontrollers, Specialized Processors, Process-Related Interfaces, Data Transfer Techniques, Communications, Standard Interface.

(Text: 3.1 to 3.8).

L1, L2

Module-3

Languages for Real-Time Applications: Introduction, Syntax Layout and Readability, Declaration and Initialization of Variables and Constants, Cuts, Modularity and Variables, Compilation of Modular Programs, Data types, Control Structures, Exception Handling, Low-level facilities, Co-routines, Interrupts and Device Handling, Concurrency, Real-Time Support, Overview of Real-Time Languages.

(Text: 5.1 to 5.14),

L1, L2, L3

Module-4

Operating Systems: Introduction, Real-Time Multi-Tasking OS, Scheduling Strategies, Priority Structures, Task Management, Scheduler and Real-Time

Clock Interrupt Handler, Memory Management, Code Sharing, Resource Control, Task Co-Operation and Communication, Mutual Exclusion.

(Text: 6.1 to 6.11).

L1, L2

Module-5

Design of RTS – General Introduction: Introduction, Specification Document, Preliminary Design, Single-Program Approach, Foreground/Background System.

RTS Development Methodologies: Introduction, Yourdon Methodology, Ward and Mellor Method, Hately and Pirbhai Method.

(Text: 7.1 to 7.5 and 8.1, 8.2, 8.4, 8.5).

L1, L2, L3

Course Outcomes: At the end of the course, students should be able to:

1. Explain the fundamentals of Real time systems and its classifications.
2. Understand the concepts of computer control and the suitable computer hardware requirements for real-time applications.
3. Describe the operating system concepts and techniques required for real time systems.
4. Develop the software algorithms using suitable languages to meet Real time applications.
5. Apply suitable methodologies to design and develop Real-time Systems.

Text Book:

- Real-Time Computer Control, Stuart Bennet, 2nd Edn. Pearson Education. 2008.

Reference Books:

1. "Real –Time Systems", C.M. Krishna, Kang G. Shin, McGraw –Hill International Editions, 1997.
2. Real-Time Systems Design and Analysis, Phillip. A. Laplante, second edition, PHI, 2005.
3. Embedded Systems, Raj Kamal, Tata McGraw Hill, India, third edition, 2005.

REAL TIME SYSTEMS

UNIT-1

INTRODUCTION TO REAL TIME SYSTEMS: Historical Background ,
RTS Definition , Classification of Real Time Systems, Time Constraints,
Classification of Programs.

Historical Background :-

- ↳ In 1950, → Brown and Campbell proposed that Computers used in real time, Assumption was that, only analog computers were used. It consisted of feedforward and feedback loops . Digital Computers were not there
- ↳ In 1954, digital Computers were developed for real time Control . and used for automatic flight and weapon Control System.
- ↳ In late 1950's . Digital Computers were used by Industries for Plant Control
- ↳ In 1958, Louisiana power and light Company Installed Computer systems for plant Monitoring.
- ↳ First Industrial Computer Installation was done by Texas Company at Texas , ie; Ramo Woolbridge (RW 300) System .
- ↳ In Oct, 1958 - it planned to have computer Control for ammonia plant at Louisiana.
- ↳ In 1960 , Closed loop Control Systems were used , but there were lot of noise problems were faced.

- ↳ In 1960's, Early computers were used with Combined magnetic core memories and drum stores.
- ↳ Cost of Earlier computer increased in attempts to resolve some problems and to use only one computer for both Supervisory and DDC.
- ↳ Later, In Late 1960's, micro computers became less cost - and made them suitable, to load large number of task onto one machine.
- ↳ In 1970, 2 computers were used, one computer acting as simply standby to function in event of failure of working computer.
- ↳ In 1974, Microprocessors advent - made it economical to use distributed Computer control System.

Elements of a Computer Control System:

A Simple Example illustrate the various operations of a Computer Control System.

Let us consider an example of hot-air blower.

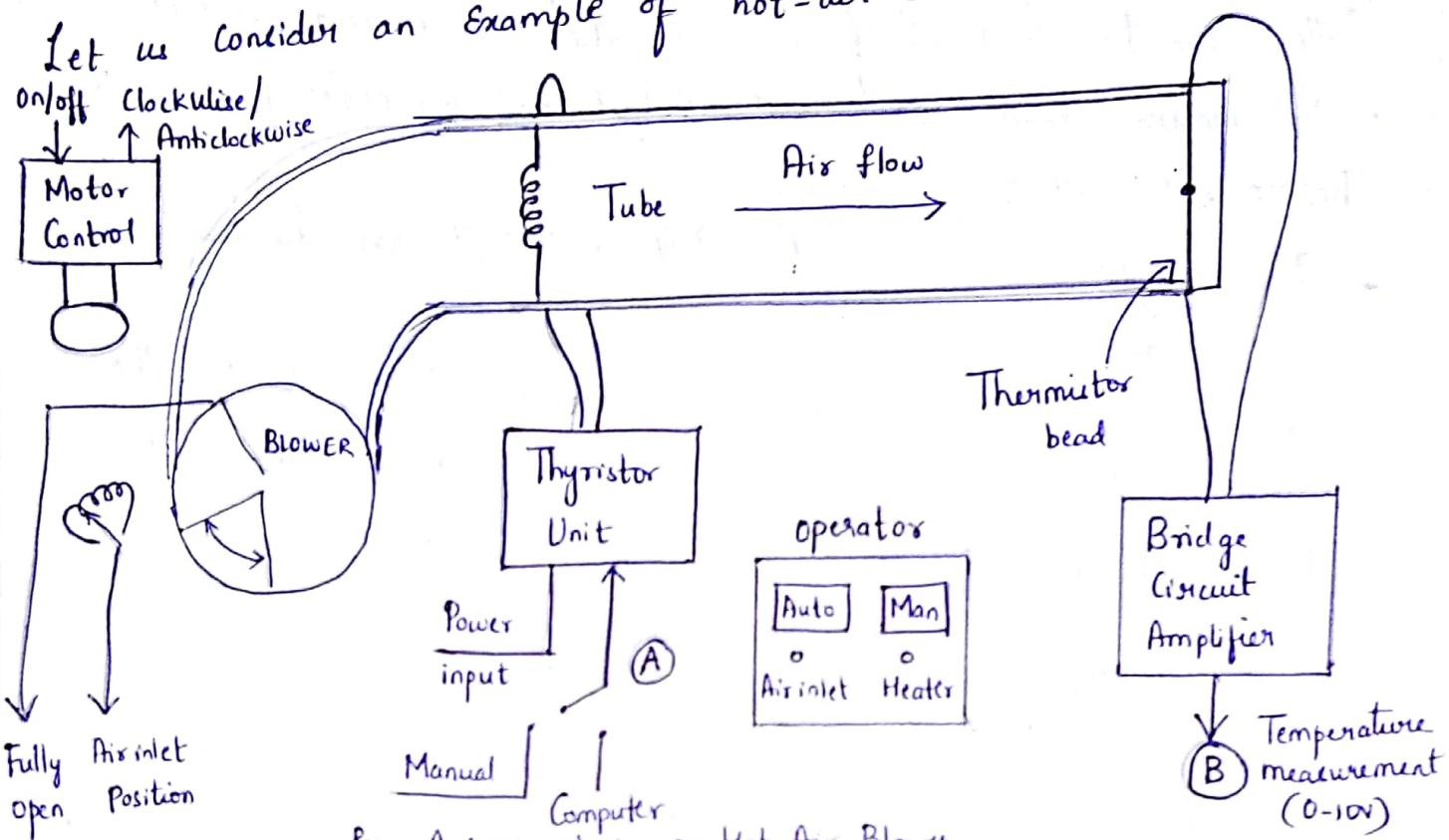
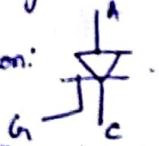


Fig. A Simple plant - an Hot Air Blower

- A Centrifugal fan blows air over a heating Element and into a tube. A thermistor bead is placed at the outlet end of the tube and forms one arm of a bridge circuit.
- The amplified output of the bridge circuit is available at B and provides a voltage in the range 0 to 10 Volts, proportional to temperature.
- The current supplied to heating Element can be varied by supplying a dc voltage in the range 0 to 10 Volts to point A.
- The position of air-inlet cover to the fan is adjusted by means of a reversible motor. The motor operates at constant speed and is turned ON/OFF by a logic signal applied to its Controller. [Note: Thyristor basic operation: Whenever small gate voltage is applied current



- A Second logic signal determines the direction of rotation. Potentiometer is used to air Inlet cover and Output Voltage is proportional to position of the cover.
- The operator is provided with a panel from which the control system can be switched from automatic to manual control.
- In manual mode the heat output and fan cover position can be adjusted using potentiometers.
- Panel lights are used to indicate heater ON, fan ON and auto/manual status.
- The desired output temperature - called as Set point for the control system, is set by operator using slider potentiometer, the setting of which can be read by the computer.

Computer Control of an Hot Air Blower

The operation of this system, using a computer requires software to support monitoring, control and actuation of the plant.

- Monitoring involves obtaining information about the current state of the plant
- Information about the plant instruments in the following forms:-

Analog Signals

- Air temperature, Fan-inlet Cover position

Digital (logic) Signals -

Fan-inlet Cover position,

Status Signals - Auto/manual, fan motor ON, heater ON.

- Control involves the digital equivalent of continuous feedback control for the control of temperature and for position control of fan-inlet Cover.

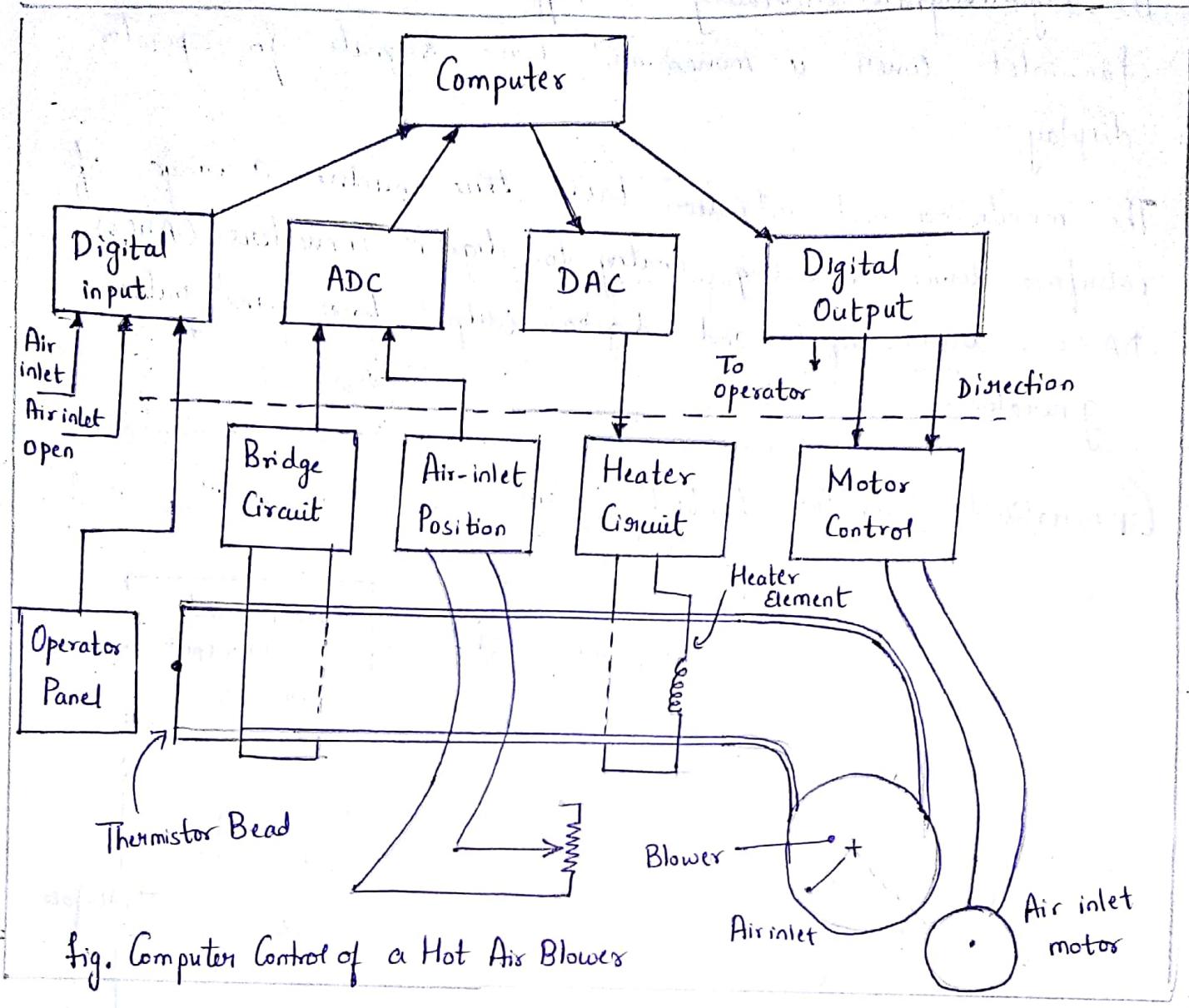


fig. Computer Control of a Hot Air Blower

- The Sequence and Interlock control operations are also required, since, for example, the heater should not be ON, if fan is not running.

The Computer has to handle automatic Change-over from Simply tracking the manual Control operations to Control the System when the operator requests a change from manual to automatic Control.

- Activation requires the provision of Voltage proportional to the demanded heat output to drive the heater Control.

The logic signals indicating ON/off and direction in which fan-inlet cover is moved and logic signals for operator display.

The monitoring and actuation tasks thus involve a range of interface devices including analog-to-digital Converters (ADCs), DACs, digital input and digital output lines and pulse generators.

Generalised Computer Control System

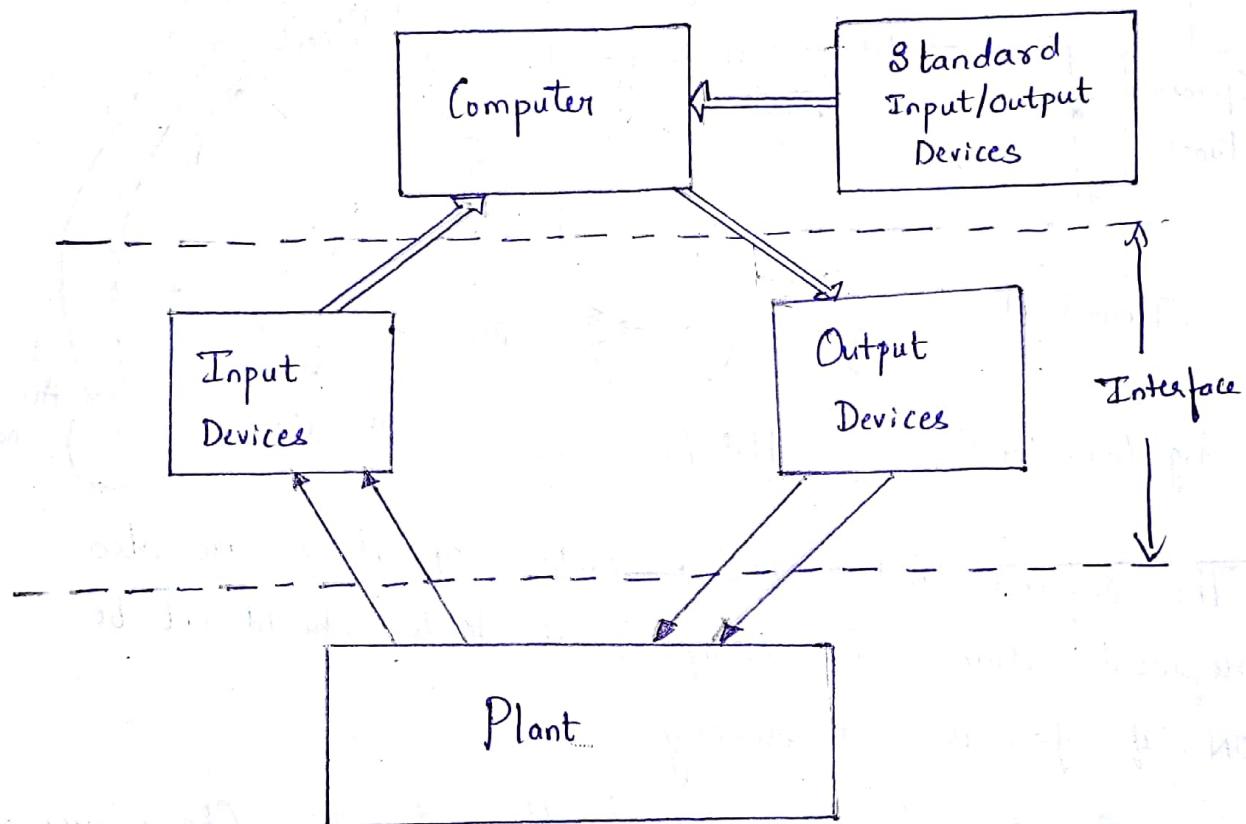


fig. Generalised Computer System.

- The Input devices are devices which transmit information to the Computer. These devices require Software to operate it and this Software is represented as Input and Output tasks.

Generalized Computer Control System Showing Hardware and Software Interfaces.

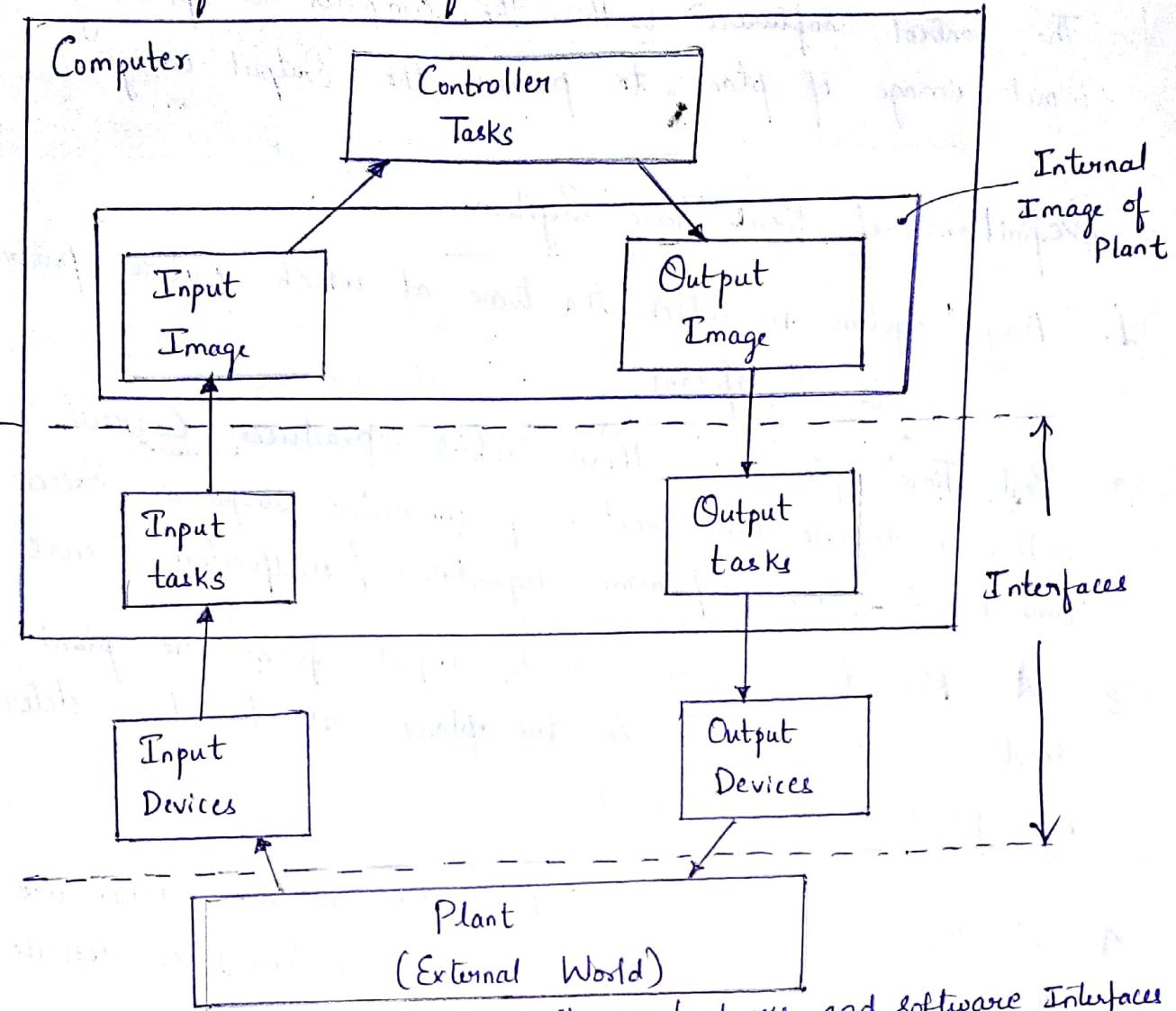


fig. Generalized Computer Control showing hardware and software Interfaces

- The figure shows the simplified block diagram of feedback control part of the system. This diagram represents a Continuous Control System.
- The Input devices plus the Input software gather the information needed to create an Input Image of the plant.
- The Input is External Information is in analog form, the process of obtaining Snapshot will involve digitization as well as Sampling.
- The Output Image represents the current set of Outputs

generated by Control Calculations.

- The Output image will be updated periodically by control Tasks.
- The Control software within the computer is operating on an input image of plant to produce the Output image.

Definitions of Real Time Systems

1. "Any system in which the time at which output produced is significant."
2. Real Time System are those which produce correct response within a definite time limit. If Computer responses exceed these time bounds, then performance degradation / malfunction occurs.
3. A Real time system reads input from the plant and sends control signals to the plant at the time determined by plant operating conditions
4. A System is called a Real Time System, when we need a quantitative expression of time (or real time) to describe the behavior of the System.

Classifications of Real Time Systems.

1. Clock based Systems (cyclic, periodic)
2. Event Based Tasks
3. Interactive Systems

1. Clock Based Tasks:-

- Synchronisation between the External processes and Internal actions are carried out by Computer is defined in terms of Passage of Time , said to be clock based.
- Plant operates in real Time.
- Time Constant is a measure of time taken by a plant to respond to a change in input
- Time constant may be measured in hours for Chemical process or milliseconds for an aircraft System.
- The computer which is used to control plant must be able to carry out all operations within - Each Sampling Interval.
- Completion of operation within Specified time is dependent on Number of operations performed and speed of Computer.
- Synchronisation is obtained by adding a Real Time Clock (RTC) which is used to interrupt operations of Computer at some predetermined fixed time Interval.
- Clock based tasks are typically referred to as cyclic/ Periodic tasks given by cycle Time T .

2. Event Based Tasks (aperiodic)

- There are operations where actions have to be performed not at particular Time , but in response to Some Event
- Event based Systems are used to indicate alarm conditions and initiate alarm actions.
- Spec. ... in FBS , is that System must respond within a maximum

- In Event based Systems, polling may be used, where computer periodically asks various sensors to see if action is required.
- Event occurs at Non-deterministic Intervals, hence EBS are referred to as aperiodic tasks.
- These are Expressed in terms of having Start or finish Times.

Interactive Systems:-

These are probably represent the largest class of Real Time Systems.

Ex: Automatic Bank Tellers (ATM's), Reservation Systems etc.

Real Time requirement is Expressed in such a way that

"Average response time must not Exceed".

These Systems can be Event based and Clock based System.

ATM systems might require an avg response time must not exceed 2sec, if avg period calculated for 24 hours.

Time Constraints :-

There are two categories of RTS :-

1. Hard Real Time Systems - These are systems that must satisfy the deadlines on Each and Every occasion.

2. Soft Real Time :- These are systems in which occasional failure to meet deadline, does not affect the system.

Typical Example of hard real time control system is temperature control loop of hot air blower system.

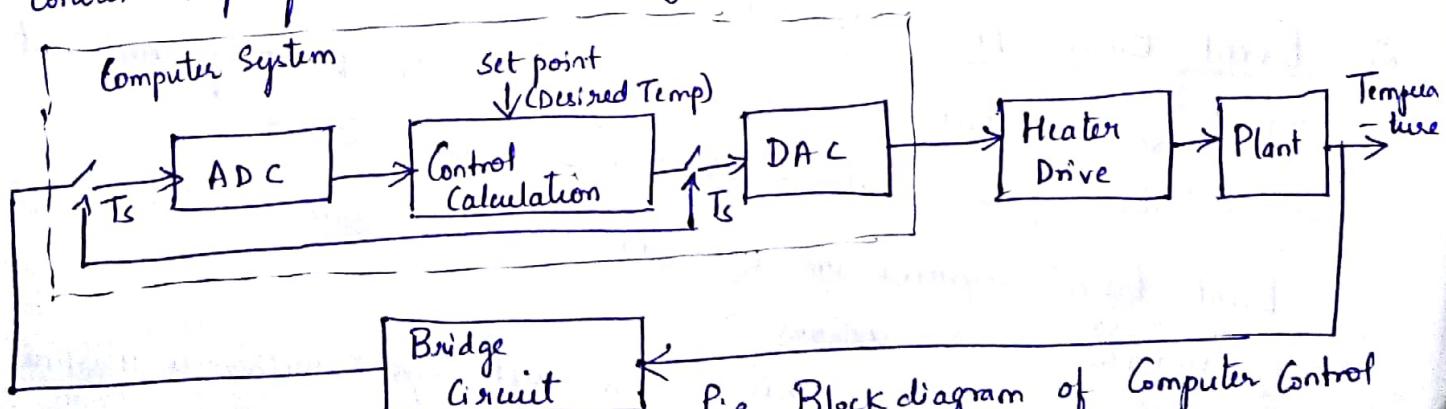


Figure shows the temperature control loop of an hot air blower system. The Sampling Interval T_s , is determined by a suitable Control Algorithm for this system. The Input value must be read, Control Calculations are performed and Output value is calculated within Specified Sampling Interval.

- heater
- This value is sent to Output drive and then to the plant.
 - The same system can be made Event based System, i.e., if Overtemperature is detected, the heater should be OFF within T_{sec} if Sampling Interval is 10 sec.
 - i.e., the Computer must respond to the Event within some Specified Maximum time.

Example for soft Real Time Systems, is an ATM.

- The Automatic Teller Machine (ATM) is a typical System, which is event initiated in that it is started by the customer placing their card in the machine.
- The time constraint on machine responding will be specified in terms of an average response time say 20 sec, with average being measured over a 24 hour period.

Formal definitions of Real Time Systems:-

<u>Hard</u>	<u>Soft</u>
Periodic (cyclic) $t_c(i) = t_s \pm a$	Aperiodic (Event) $t_e(i) \leq T_e$ $\frac{1}{n} \sum_{i=1}^n t_c(i) = t_s \pm a$ $n = \frac{I}{t_s}$

where, $t_c(i)$ \rightarrow Interval between i and $i-1$ cycles

$t_e(i)$ = Response time to i^{th} occurrence of Event e,

t_s \Rightarrow desired periodic (cyclic) Interval.

T_e \Rightarrow Maximum permitted response time to event e measured over T.

$n \rightarrow$ Number of occurrence of Event e within Time Interval T.

$\alpha \rightarrow$ Small Timing Tolerance.

Classifications of Programs.

Three types of programming are Identified.

1. Sequential:-

- The sequential programming are strictly ordered as a time sequence.

- Behavior of the program depends only on individual actions and their order.

Verification requires two kinds of argument.

① Particular statement in a program defines a stated action.

② Various program structures (like for, while etc.) produce state sequence of Events.

2. Multitasking

- Method in which multiple tasks are performed at the same time

- They are Executed Concurrently.

- They share common resources such as CPU and Main Memory.

- Verification requires application for arguments for sequential programs with some additions.

The task can be verified separately only if the constituent variables of each task are distinct.

3. Real Time programming :- It involves programming for events occurring in the outside world in real time and without reference to internal operations of the computer

- It has a specific deadline to produce output.

4. Non-Real Time programming

- It has no deadline, even if fast response or high

These are called sporadic timer.

REAL TIME SYSTEMS

①

UNIT-2

CONCEPTS OF COMPUTER CONTROL: Introduction, Sequence Control, Loop Control, Supervisory Control, Centralised Computer Control, Distributed System, Human-Computer Interface [HCI], Benefits of Computer Control Systems.

INTRODUCTION:

The Basic features of Computer Control Systems using Examples drawn from Industrial process control.

In this field application, they are typically classified as:

1. Batch
2. Continuous
3. Laboratory

Batch:-

- * The term Batch is used to describe processes in which sequence of operations are carried out to produce a quantity of product.
- * The batch is one in which the sequence is repeated to produce further batches.
- * Specification of product or Exact composition may be changed between different runs.
- * An important measurement in batch production is Set up time. i.e., Time taken to produce the equipment for next production batch.
- * Batch Size =
$$\frac{\text{Operation time}}{\text{Set up time}}$$
 where, Operation time is Time taken during which product is being produced.

Continuous:

- * The term continuous is used for systems in which production is maintained for long periods of time without interruption typically over several months or years.

- * The ratio of different fractions is changed but this is done without halting the process.
- * Continuous Systems may be considered as batches, in that the product combination is changed from time to time.
- * But they are still classified as continuous since the change in composition is made without halting the production process.

Laboratory Systems:

- * Laboratory based systems are frequently an Operator initiated type in which the computer is used to control some complex experimental test.
- Ex: The audiometer has to produce sound levels at different frequencies. Each audiometer has to be tested against a sound level meter connected to a computer and using the output from computer to drive the audiometer through its frequency range. The result is printed out from test computer which provide the test certificate.

Objectives of using a computer to control the process will include:

- Efficiency of Operation
- Ease of Operation
- Safety
- Reduction in Waste
- Improved products
- a reduction in direct labour.

SEQUENCE CONTROL:-

(3)

Sequence control occurs in Systems which often predominates batch Systems and hence batch Systems is used to illustrate it.

- * Batch Systems are widely used for food processing and chemical Industries where Operations are carried out which involve mixing raw materials, carrying out some process and discharging the product. Chemical

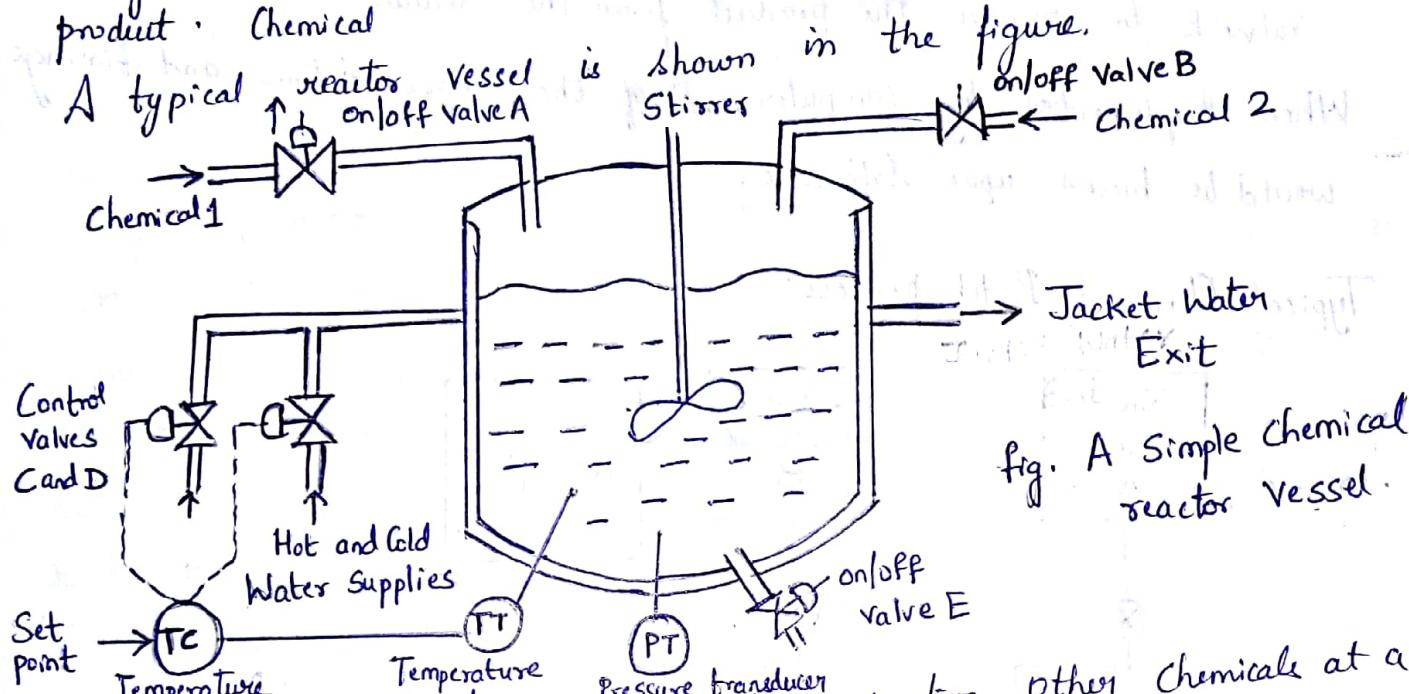


fig. A Simple chemical reactor vessel.

- * A Chemical is produced by the reaction of two other chemicals at a Specified temperature.

The chemicals are mixed together in a sealed vessel and the temperature of the reaction is controlled by feeding hot or cold water through the water jacket which surrounds the Vessel.

The procedure for the Operation of the System may be as follows:-

The procedure for the Operation of the System may be as follows:-

1. Open Valve A to charge the vessel with Chemical 1.
2. Check the level of the Chemical in the Vessel ; when the correct amount of Chemical has been admitted , close valve A.
3. Start the stirrer to mix the Chemicals together.
4. Repeat Stages 1 and 2 with valve B in order to admit the Second Chemical.
5. Switch on the three-term controller and supply a set point so that the Chemical mix is heated up to the required temperature.

6. Monitor the reaction temperature & when it reaches the set point, Start the timer to time the duration of the reaction.
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 temperature, when the contents have cooled, Open valve E to remove the product from the reactor.

When implemented by computer, all of the above actions and timings would be based upon software.

Typical Chemical Batch process:

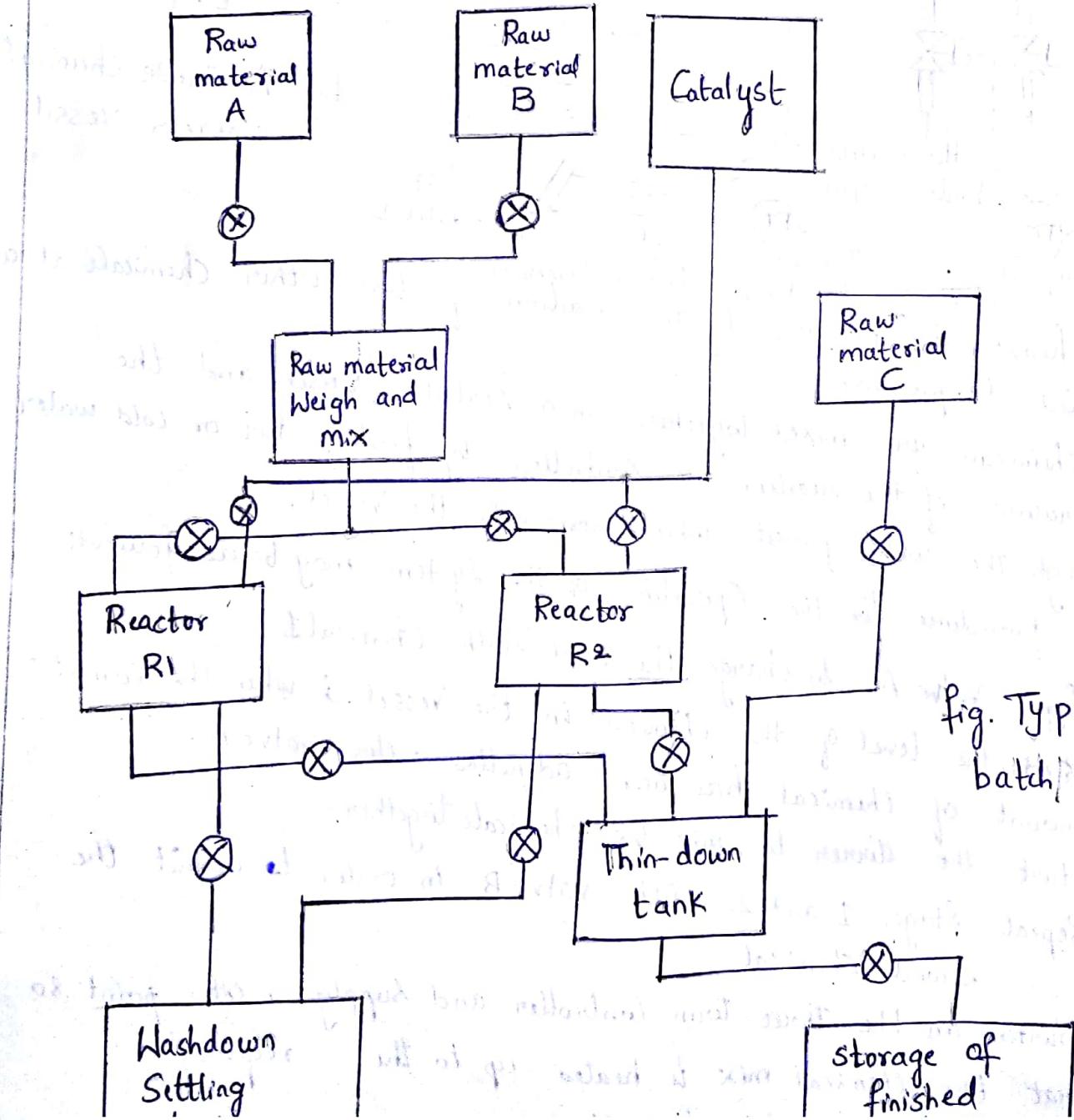


Fig. Typical Chemical batch process

- In this plant two reactor vessel (R_1 and R_2) are used alternatively so that the processes of preparing for the next batch and cleaning up after a batch can be carried out in parallel with the actual production.
- Assuming that R_1 has been filled with the mixture and the catalyst, and reaction is in progress, there will be for R_1 : loop control of the temperature and pressure; Operation of the stirrers; and timing of the reaction.
- In parallel with this, vessel R_2 will be cleaned - the washdown sequence and the next batch of raw material will be measured and mixed in the mixing tank.
- The previous batch will be thinned down and transferred to the appropriate storage tank and, if there is to be a change of product or a change in product quality - the thin-down tank will be cleaned.
- Once this is done the next batch can be loaded into R_2 and then, assuming that the reaction in R_1 is complete, the contents of R_1 will be transferred to the thin-down tank and the Washdown procedure for R_1 initiated.

LOOP CONTROL: (DIRECT DIGITAL CONTROL)

The direct digital control (DDC) the computer is in the feedback loop as is shown in figure.

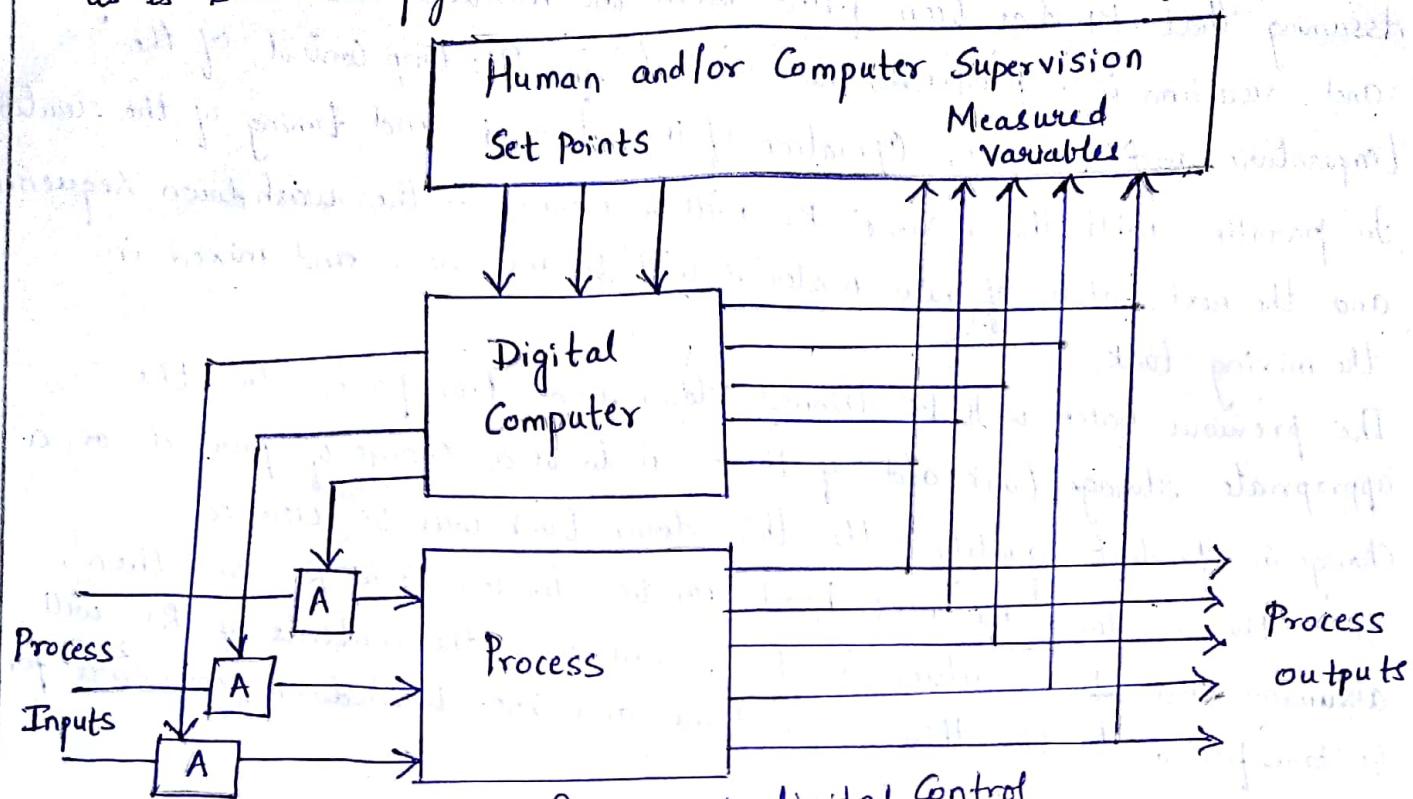


fig. Direct digital Control

Disadvantage:-

The consequence of the computer being in the feedback loop is that it forms a critical component in terms of the reliability of system, hence great care is needed to ensure that, in the event of failure or malfunctioning of computer, the plant remains in a safe condition.

The usual means of ensuring safety are to limit the DDC unit to making incremental changes to the actuators on the plant.

Advantages of DDC over Analog Control:

1. Cost:- a single digital computer can control a large number of loops. The introduction of microprocessors a single loop DDC unit can be cheaper than an analog unit.
2. Performance:- Digital Control offers simpler implementation of a wide range of control algorithms, improved controller accuracy and reduced drift.
3. Safety - Modern digital hardware is highly reliable with long

mean time - of between failures and hence can improve the safety of Systems.

PID Control

The PID Control algorithm has the General form.

$$m(t) = K_p [e(t) + 1/T_i \int e(t) dt + T_d de(t)/dt]$$

where $e(t) = r(t) - l(t)$

$l(t) \rightarrow$ measured variable

$r(t) \rightarrow$ reference value or set point

$e(t) \Rightarrow$ error

$K_p \rightarrow$ Overall controller gain

$T_i \rightarrow$ Integral action time

$T_d \rightarrow$ derivative action time.

It is difficult to Improve on the Control performance that can be obtained by using Either PI or PID Control.

- * Using a Control signal that is made proportional to the Error between the desired value of an Output and the actual Value of Output is an obvious and a reasonable strategy
- * The ratio b/w the Control signal and Error signal can be adjusted using the proportionality constant K_p . Choosing the value of K_p involves a compromise.
 - a high value of K_p gives a small steady-state Error and a fast response. but response will be Oscillatory and may be un acceptable in many applications.
 - a low value of K_p gives a slow response and a large steady state Error.
- * By adding the Integral action term the Steady State Error can be reduced to zero since the Integral term Integrates the Error signal with respect to time.
- * For a given Error value the rate at which the Integral term Increases is determined by the Integral action time T_i .

A purely proportional controller Operates correctly only under one particular set of process conditions

- * Changes in load on the process or some Environmental condition will result in a Steady state Error.

* ... thus ... and reduces the Error

DDC Application

→ DDC may be applied to a single loop system implemented on a small microprocessor. The loops may be cascaded, that is with the output of one loop acting as the set point for another loops, signals may be added together and conditional switches may be used to alter signal connections.

Ex 1: Typical Industrial System - A boiler Control Scheme.

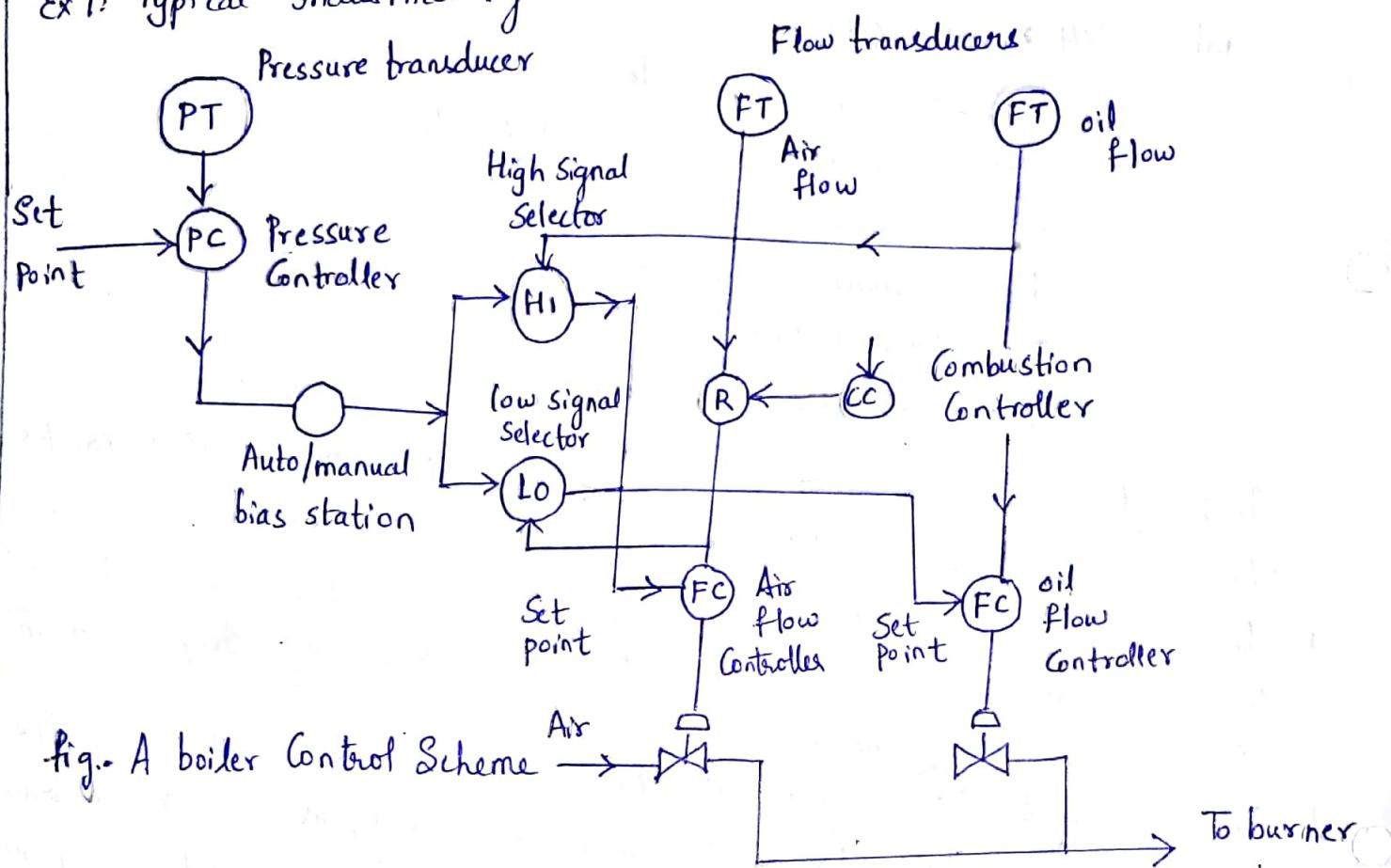


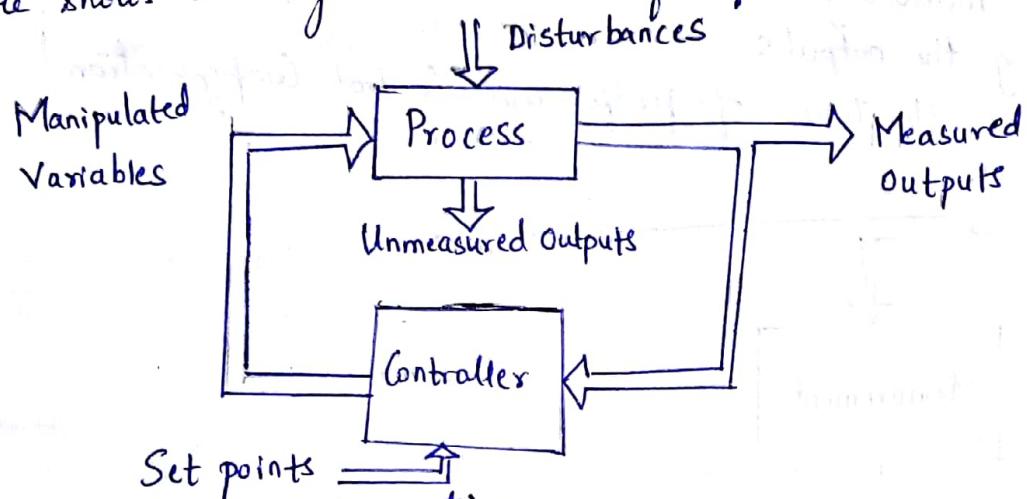
fig. A boiler Control Scheme

- The steam boiler control system is shown. The steam pressure is controlled by regulating the supply of fuel oil to the burner. In order to comply with pollution regulations a particular mix of air and fuel is required.
- The steam pressure control system generates an actuation signal which is fed to an auto/manual bias station. If the station is switched to auto then the actuation signal is transmitted. If it is in manual mode a signal which has been entered manually is transmitted [(Ex) Keyboard]
- The signal from Bias has 2 units low signal selector & Inputs & one o/p.
- The high selector transmits the higher of 2 Input signals, low selector transmits the lower of 2 inputs. Signal from the low selector provides the set points for, DDC loop controlling the Oil flow, the signal from high selector provides set points for air flow controller. A ratio unit is controlled in air flow.

DDC Techniques:-

- Inferential
- feed forward
- Adaptive or Self tuning Control.

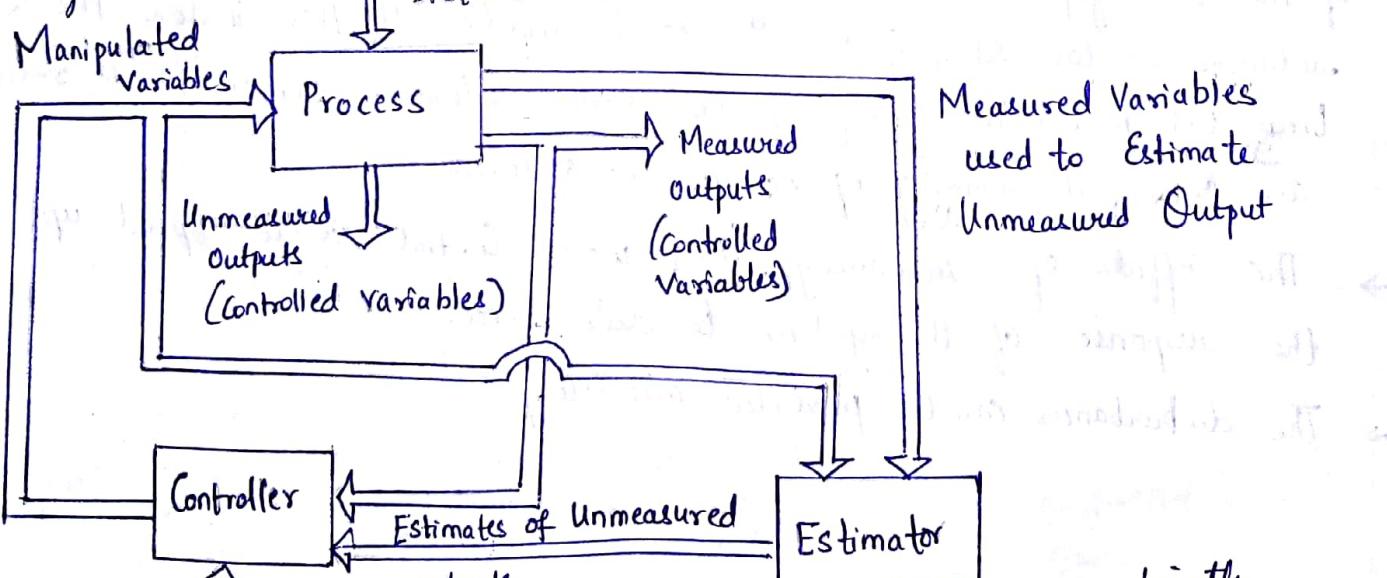
DDC is not necessarily limited to simple feedback control. figure shows the general structure of a feedback control configuration.



Inferential Control Configuration.

Inferential Control is the term applied to control where the variables on which feedback control is to be based cannot be measured directly, but have to be 'inferred' from measurements of some other quantity.

Figure Shows General Structure of Inferential Control Configuration.



Measured Variables used to Estimate Unmeasured Output

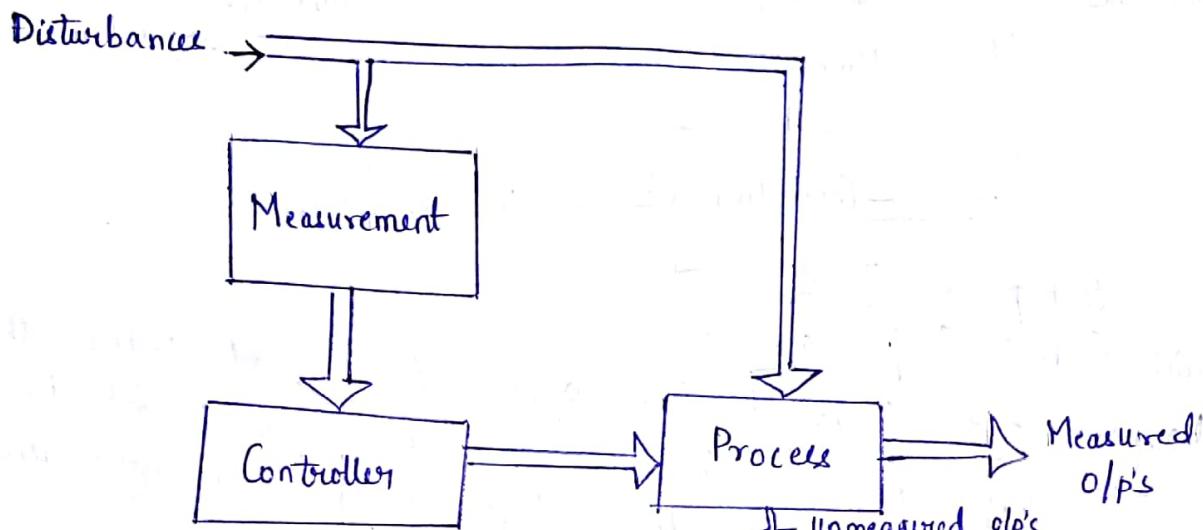
Some of the outputs can be measurements and used directly in the feedback control.

Other outputs required by the controller cannot be measured directly. Learn this the

value of the controlled variable is inferred.
Inferential measurements are frequently used in distillation column control. [Refer Schematic of binary distillation column in page 49]

Feed Forward Control Configuration.

This control is frequently used in the process industries. It involves measuring the disturbances on the system rather than measuring the outputs figure. General structure of feedforward control configuration.



for Ex: in hot rolling of sheet steel, if the temperature of the billet is known as it approaches the first stage mill, the initial setting of the roll gap can be calculated accurately and estimate of the reduction at each stage of the mill can be made. If this is done the time taken to get the gauge of steel within tolerance can be much reduced and hence the quantity of scrap steel reduced.

- The effect of introducing feedforward control is to speed up the response of the system to disturbances.
- The disturbances can be predicted accurately.

Adaptive Control

(11)

3 of most Common adaptive Control are:-

- Preprogrammed adaptive Control (gain Scheduled Control)
- Self tuning
- Model reference adaptive Control.

Preprogrammed adaptive Control:-

This is illustrated as shown in figure.

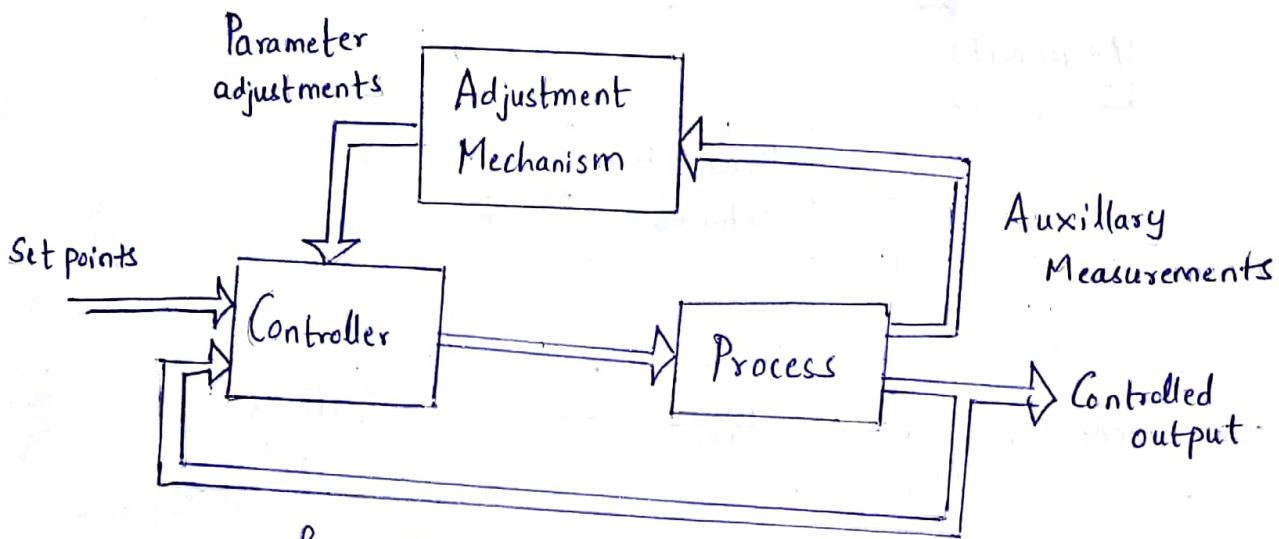


figure (a) Auxillary process Measurements

The adaptive or adjustment Mechanism makes preset changes on the basis of Changes in auxillary process Measurements. Ex: in many aircraft Controls the measured air Speed is used to select Controller parameters according to a preset Schedule.

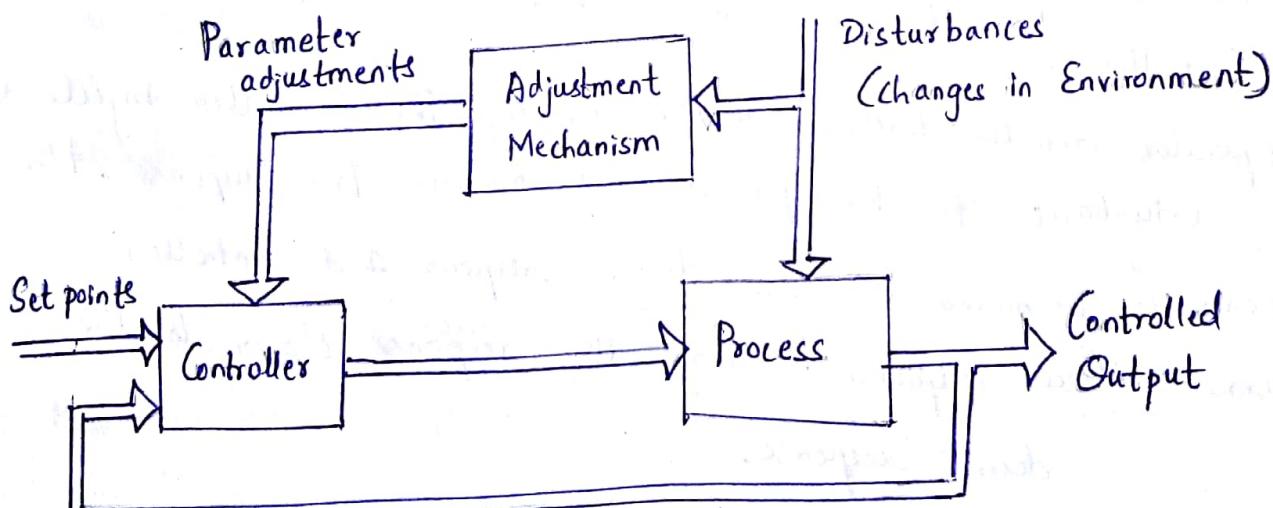


Fig (b) External Environment

Changes in the External Environment are used to select the gain or other controller parameters.
for Example, in an aircraft auto stabilizer, control parameters maybe changed according to the External air pressure.

Self-tuning Adaptive Control.

This is illustrated as shown below.

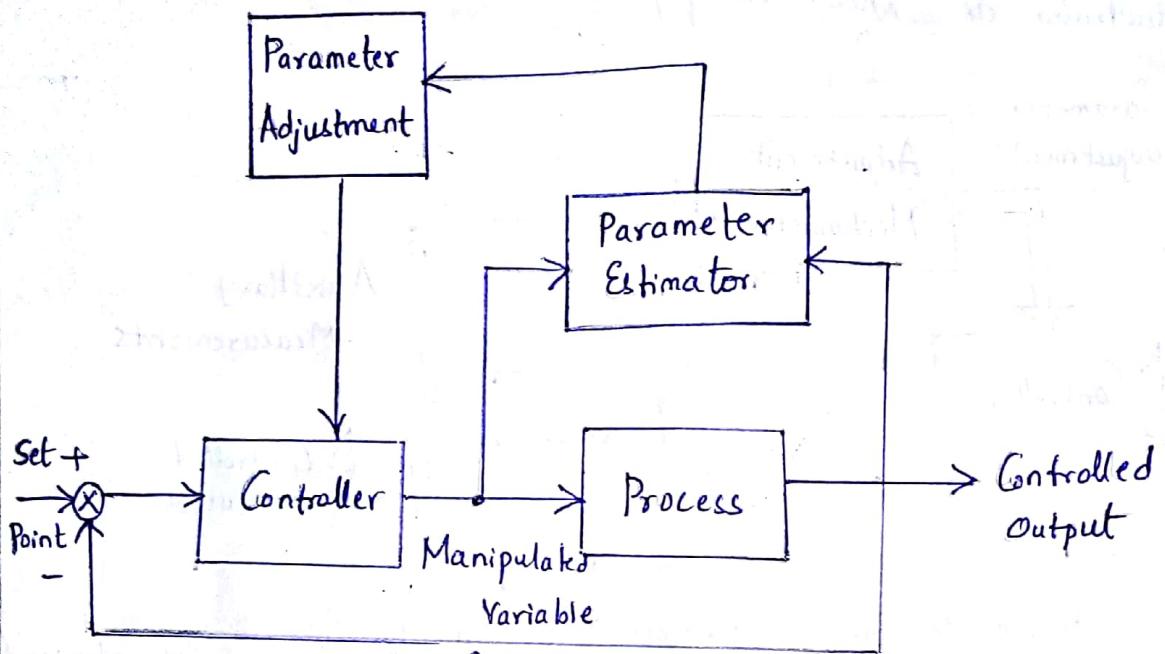


fig. Self tuning Adaptive Control

If uses Identification techniques to achieve continual determination of the parameters of the process being controlled ; changes in the process parameters are then used to adjust the actual controller.

An alternative form of Self tuning is frequently found in Commercial PID Controllers -

At operator initiated Intervals or periodically the Controller injects a small disturbance to the process and measures the response - the response is compared to some desired response and controller parameters are adjusted to bring the response closer to the desired response.

Model reference Adaptive Control.

(13)

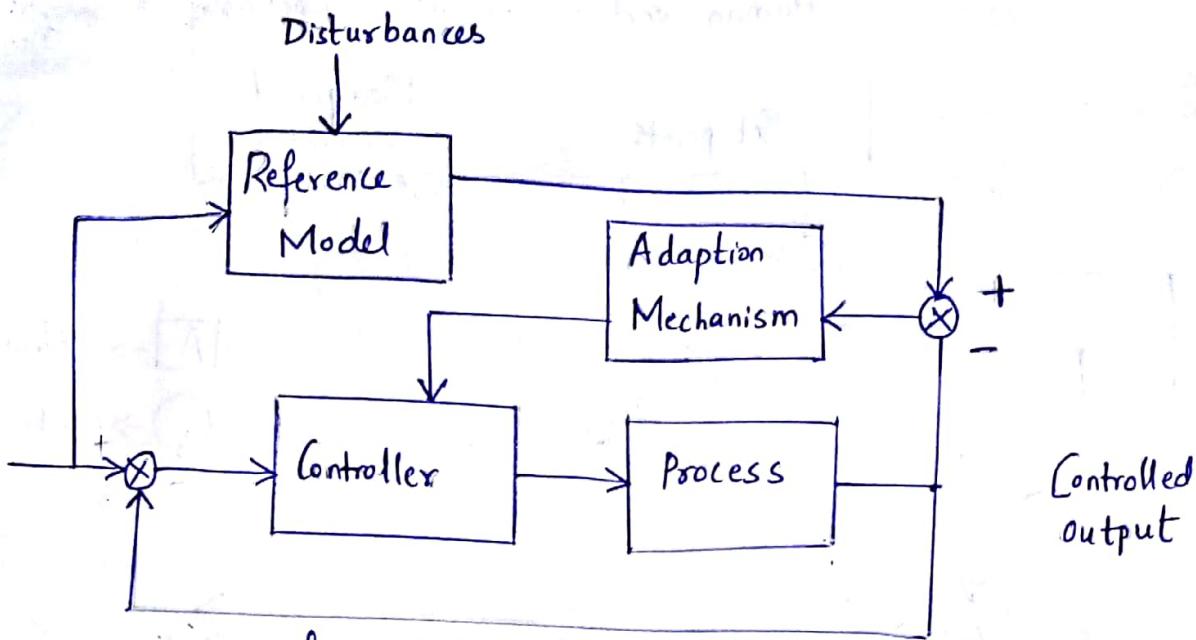


fig. Model reference Adaptive Control

It relies on ability to construct an accurate model of the process and measures the disturbances which affect the process.

SUPERVISORY CONTROL

Direct digital control (DDC) is often simply the computer implementation of the techniques used for the traditional analog controllers.

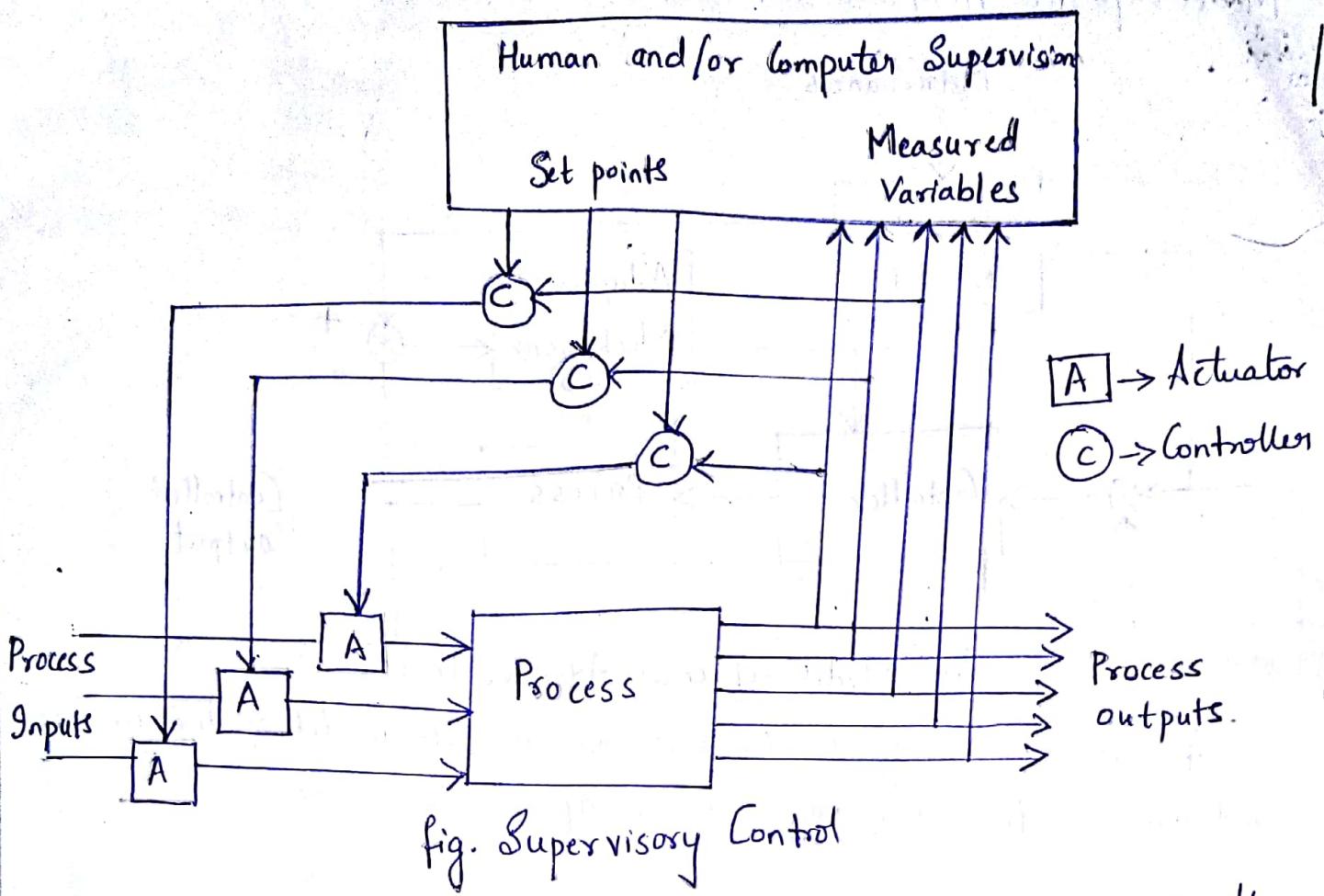
Many of early computer control schemes used the computer in a supervisory role and not for DDC.

The main reason for this were.

- ① Computers in the early days 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

The basic idea of a "Supervisory Control" is illustrated as shown below.

Applications of Supervisory Control are very simple and are based upon knowledge of steady state characteristics of the plant.

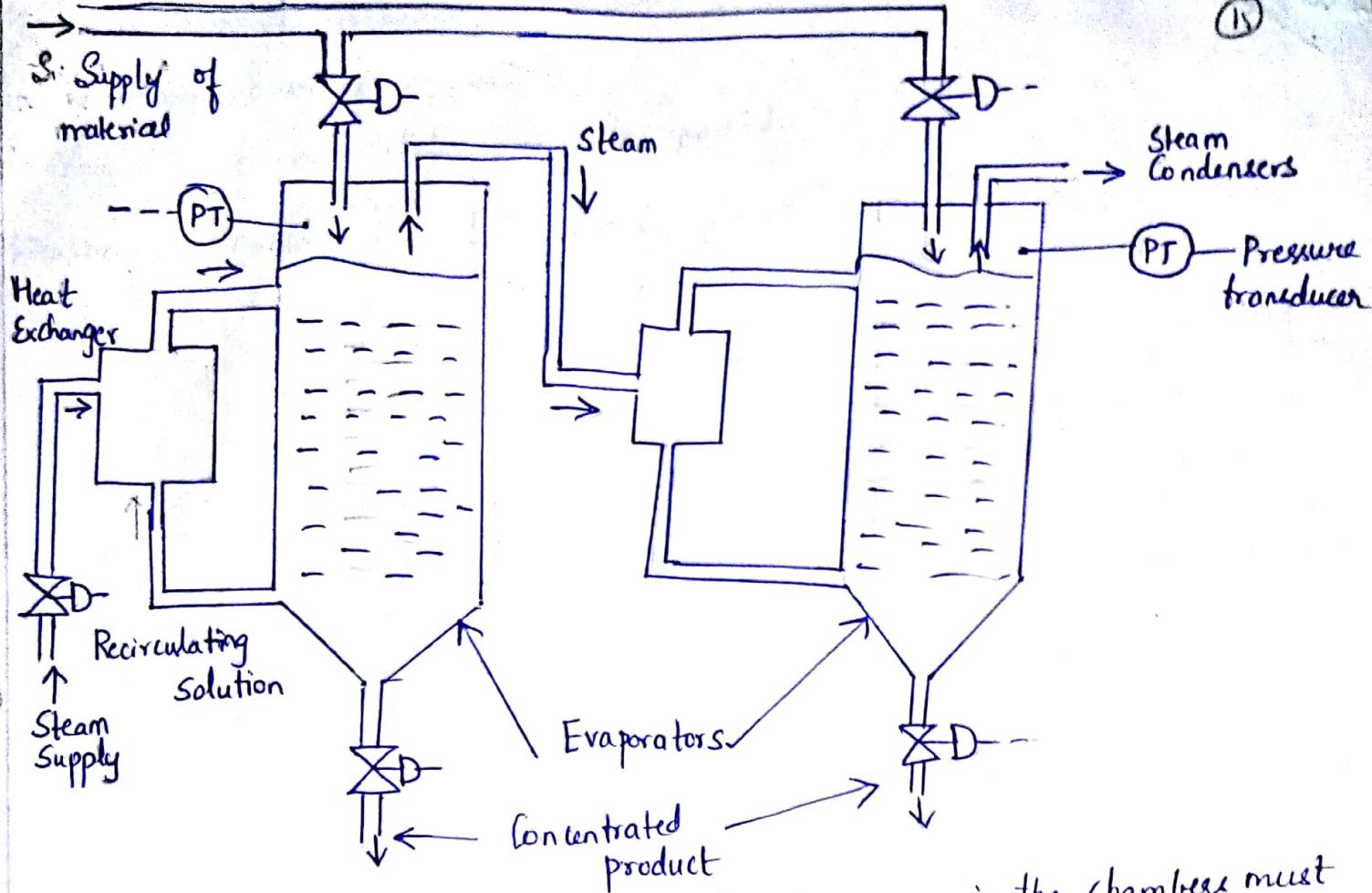


The circles labelled C represent Individual Controllers in the feedback loop; these can be themselves digital computers but their operation is supervised by a digital computer. actuators are used to control the mechanism.

Example of Supervisory Control.

An Example of Supervisory Control i.e., an Evaporation plant is shown in figure.

- * Two Evaporators are connected in parallel and material in solution is fed to each unit.
- * The purpose of plant is to evaporate as much water as possible from the solution.
- * Steam is supplied to a heat Exchanger linked to the first Evaporator and steam for second Evaporator is supplied from the vapours boiled off from the first stage.



- * To achieve maximum Evaporation the pressure in the chamber must be as high as Safety permits.
- * It is necessary to achieve a balance between the two Evaporators, if first one is driven at its maximum rate it may generate so much steam that the Safety thresholds for second Evaporator are exceeded.
- * A Supervisory Control Scheme can be designed to balance the operation of two Evaporators to obtain the best overall Evaporation rate.

Centralised Computer Control.

In 1960 - One single computer were used to control the whole plant.
because the computer were expensive.

A consequence of centralised control was the considerable resistance to the use of DDC schemes, with one central computer in the feedback loop, failure of computer results in loss of control of the whole plant.

- Computer were not reliable hence they were failures in the system.

In Mid 1960s:- Companies began to produce digital controllers with analog back-up.

* These units were based on the standard analog controllers but allowed a digital control signal from the computer to be passed through the controller to the actuators; the analog system tracked the signal and if the computer did not update the controller within a specified interval, the unit dropped over to local analog control.

This scheme enabled DDC to be used with confidence, that is even if computer is failed, a plant can still be operated.

In 1970 - the cost of computer hardware had reduced to such an extent that it became feasible to consider the use of dual computer system.

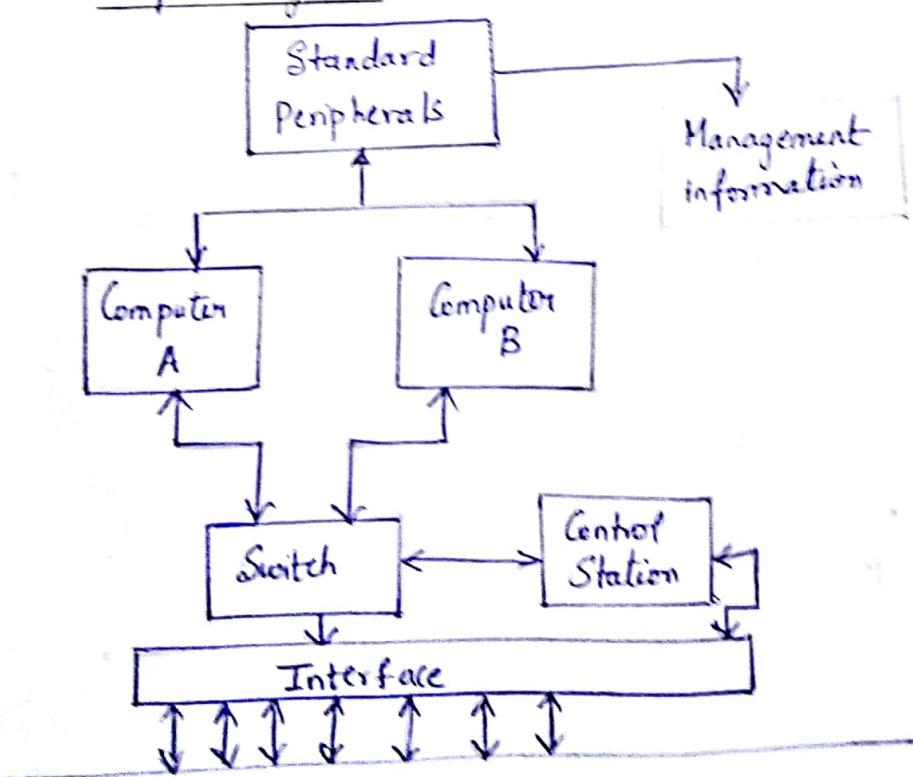


fig. Dual Computer Scheme

In Dual Control Computer Scheme,

→ the change over is manual.

→ Automatic failure detection is incorporated.

Furthermore, the problems of designing, programming, testing and maintaining the software are not reduced. If anything they are further complicated in that provision for monitoring ready for change-over has to be provided. Cost of

The reduction of hardware has made multicomputer systems feasible.

2 types:

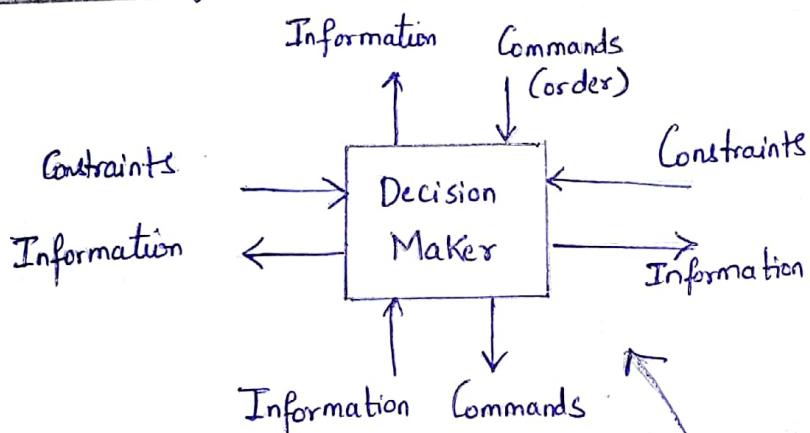
1. Hierarchical - Tasks are divided according to function.

Ex: One computer performs DDC calculations and another which performs Supervisory Control.

2. Distributed - Many computers perform essentially similar tasks in parallel.

Hierarchical Systems

(a)



(b)

Top level
(single decision centre)

Intermediate

Bottom level
(many decision centres)

long

Decision time-scale

short

④ Decision-making function

- Each division Element receives commands from the level above and sends Information back to that level and, On basis of Information received from the Element and from constraints Imposed by Elements at the same level , sends commands to the Elements below and Information to Elements at the same level.
- This structure follows a natural division of production process in terms of the time-response requirements of different levels.
- At the bottom , a fast response to simple problems is required. as one progresses up the hierarchy the complexity of Calculations Increases as does the time allowed for the response.

A typical Example of a hierarchical system is the batch system is shown in figure.

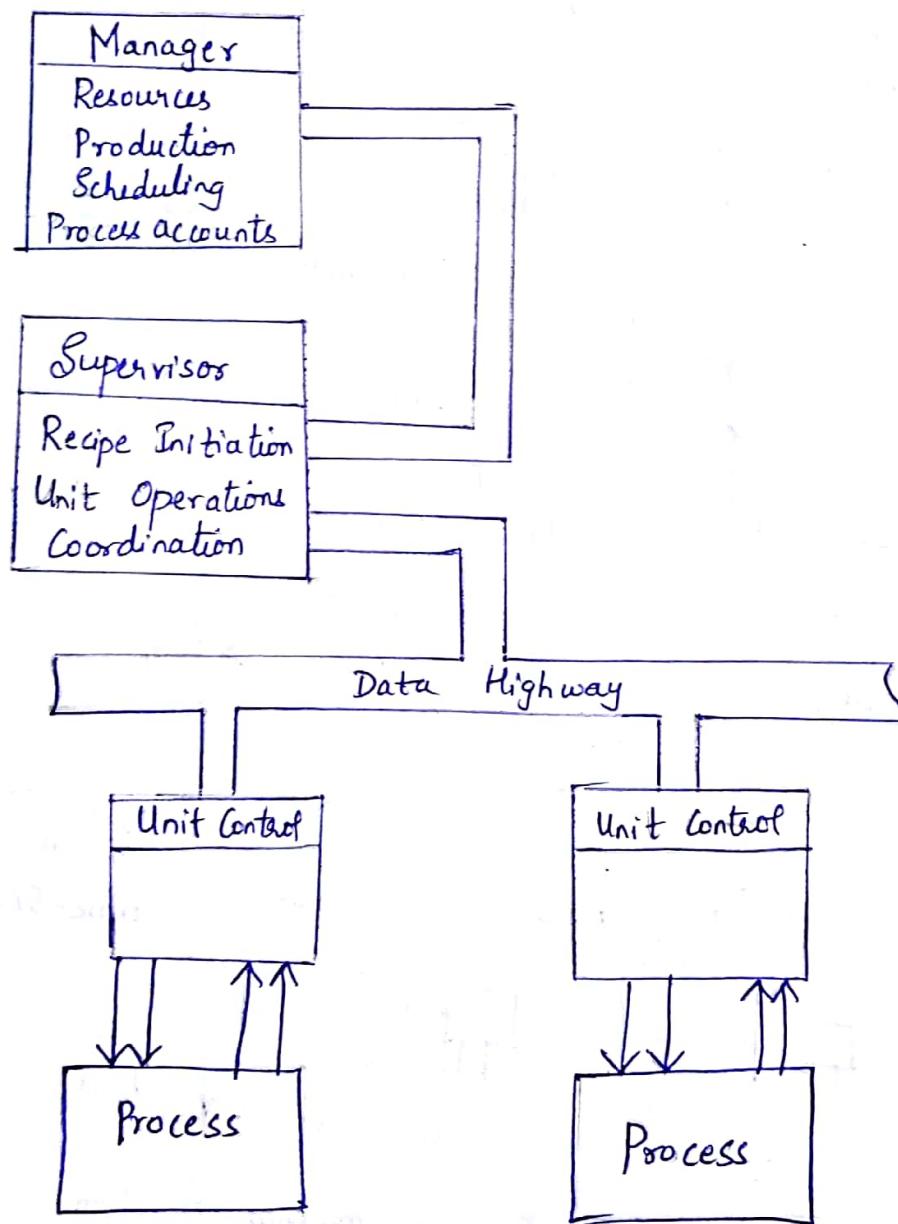


fig. Batch Control
using a
Hierarchical
System

This System has 3 levels -

→ Manager

→ Supervisor

→ Unit Control

- At the manager level, the functions such as resource allocation, production Scheduling and production accounting are carried out. On the basis of this information the production Schedule, that is the list of products to be produced and quantities and process units to be used will be calculated.
- The Information regarding the production Schedule is transferred to the Supervisor.
 - ↳ The supervisor has a store containing the product recipe and store of operation sequences for making the product.
 - ↳ when the appropriate unit, as selected by the production plan, ready the information on the product - set points, alarm conditions, tolerance etc. is loaded down into the unit controller.
- At the lowest level, the unit controllers, are responsible for operating the plant; opening and closing valves and switches, controlling temperatures, pressures, speeds & flows, monitoring alarms and reporting plant conditions.

Distributed Systems.

Assumptions of distributed approach are that:

1. Each unit is carrying out Essentially similar tasks to all the other units.
2. In the Event of failure or overloading of a particular unit all or some of the work can be transferred to other units.

In most modern schemes a mixture of distributed and hierarchical approaches is used as shown in figure.

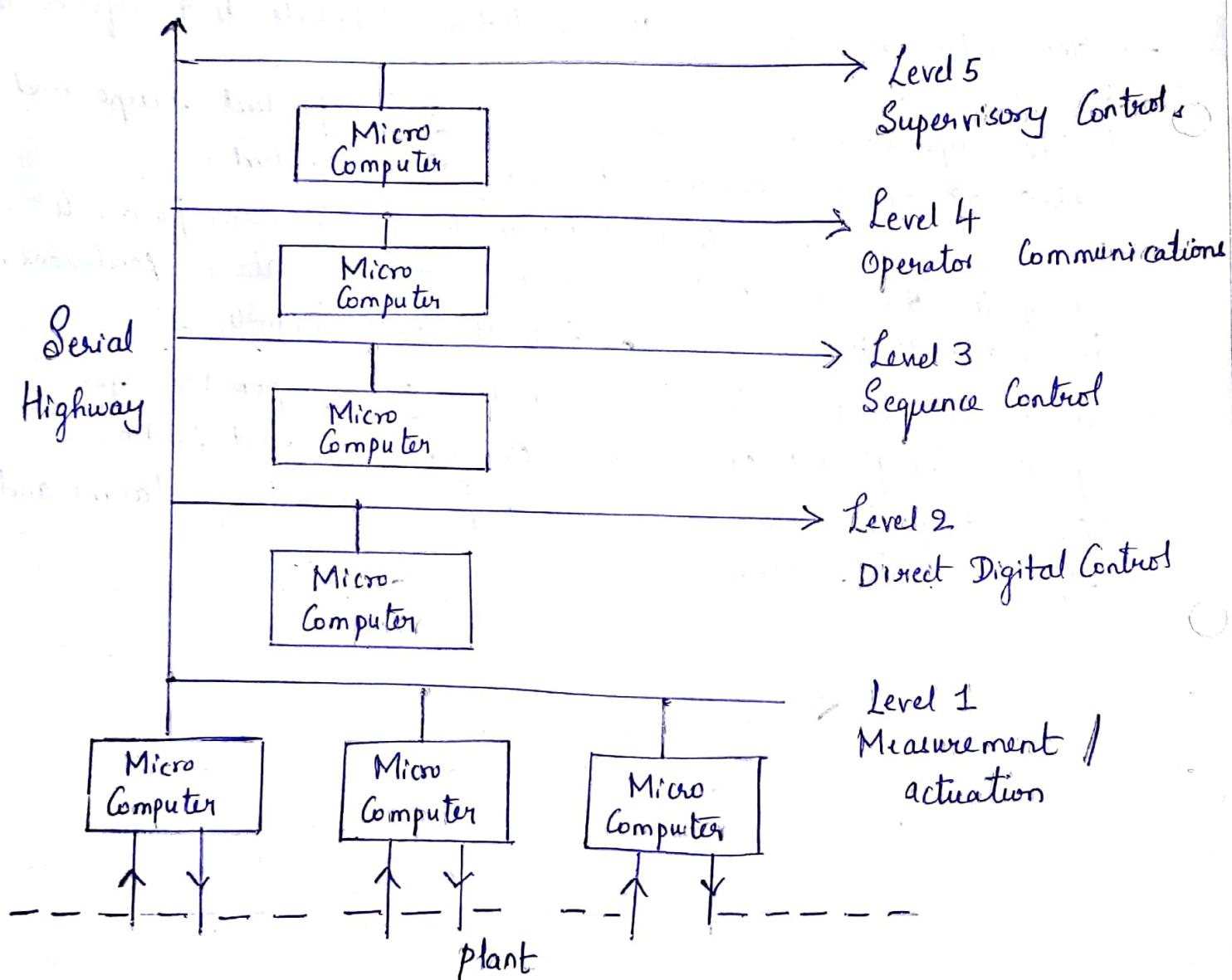


Fig. A distributed and hierarchical System

→ The tasks of measurement, DDC, operator Communication etc are distributed among a number of Computers which are linked together via a common Serial communications highway and are configured in a

Five broad divisions of functions are shown:

Level 1: All computations and plant interfacing associated with measurement and actuation. This level provides a measurement and actuation database for the whole System.

Level 2: All DDC Calculations.

Level 3: All Sequence Calculations

Level 4: Operator Communications

Level 5: Supervisory Control.

Level 6: Communications with other Computer Systems.

Major advantages of this approach are:-

1. The System Capabilities are greatly Enhanced by sharing of tasks between processors - the burden of Computation for a single processor becomes very great if all of the described control features are included.
 2. The System is much more flexible than the use of a single processor: If more loops are required, all that is necessary is to add more boxes to communication link.
 3. It allows Standardisation, since it is much easier to develop standard units for well-defined single tasks than for overall control schemes.
 4. Failure of a unit will cause much less disruption in that only a small portion of Overall system will not be working.
 5. It is much easier to make changes to the system, in the form of either hardware replacements.
- Changing large programs is hazardous because of possibility of side-effects: with the use of small modules such effects are less likely to occur and are more easily detected and corrected.
6. Linking by serial highway means that the computer units can be widely dispersed: hence it is unnecessary to bring cables carrying transducer signals to a central control room.

Human Computer Interface (HCI)

- * The key to successful adoption of a computer control scheme is often the facilities provided for the plant Operator of the system.
- * A day-to-day operation of the plant must be provided.
- * The information relevant to the current state of its operation should be available and facilities to enable interaction with the plant - to change set points, adjust actuators by hand, to acknowledge alarm conditions etc., should be provided.
- * A typical operator station has designed keyboards and general display and printer units; video units are frequently provided to enable the operator to see parts of the plant.
- * Standard software packages provide a range of display types:-
 - an alarm overview presenting information on alarm status of large areas of plant.
 - a no of area displays presenting information on control scheme associated with each area.
- * The plant manager requires access to different information:-
hardcopy printouts - that summarise day-to-day operation of the plant also provide a permanent plant operating history.
- * The Manager will be interested in assessing the economic performance of the plant and in determining possible improvements in plant operation.

Role of the Control Engineer.

(2)

The Control Engineers responsibility is as follows:-

1. To define the appropriate control strategy to meet the system requirements
2. To define the measurements and actuators and to set up alarm conditions, Sampling Intervals Etc:
3. To define DDC Controllers and connections with any other elements in the Control Scheme.
4. To tune the Control Scheme, that is to select appropriate gains so that it performs according to some desired specification
5. To define and program sequence control procedure necessary for automation of plant operation.
6. To determine and implement satisfactory supervisory control schemes.

Economics and Benefits of Computer Control Systems.

- ↳ Computer Control was expensive and very strong case was needed to justify the use of Computer control rather than conventional instrumentation.
- ↳ The Computer user permits repeatability.
- ↳ In order to provide sequence procedures to manufacture different products, Computer systems are highly flexible compared to conventional systems.
- ↳ Computer systems are used to maintain a database containing the product recipes and change to a new recipe quickly and reliably.
- ↳ The main area of benefit has been in control of the starting and stopping of batch operations in that Computer-based systems have significantly reduced the dead time associated with batch operations.

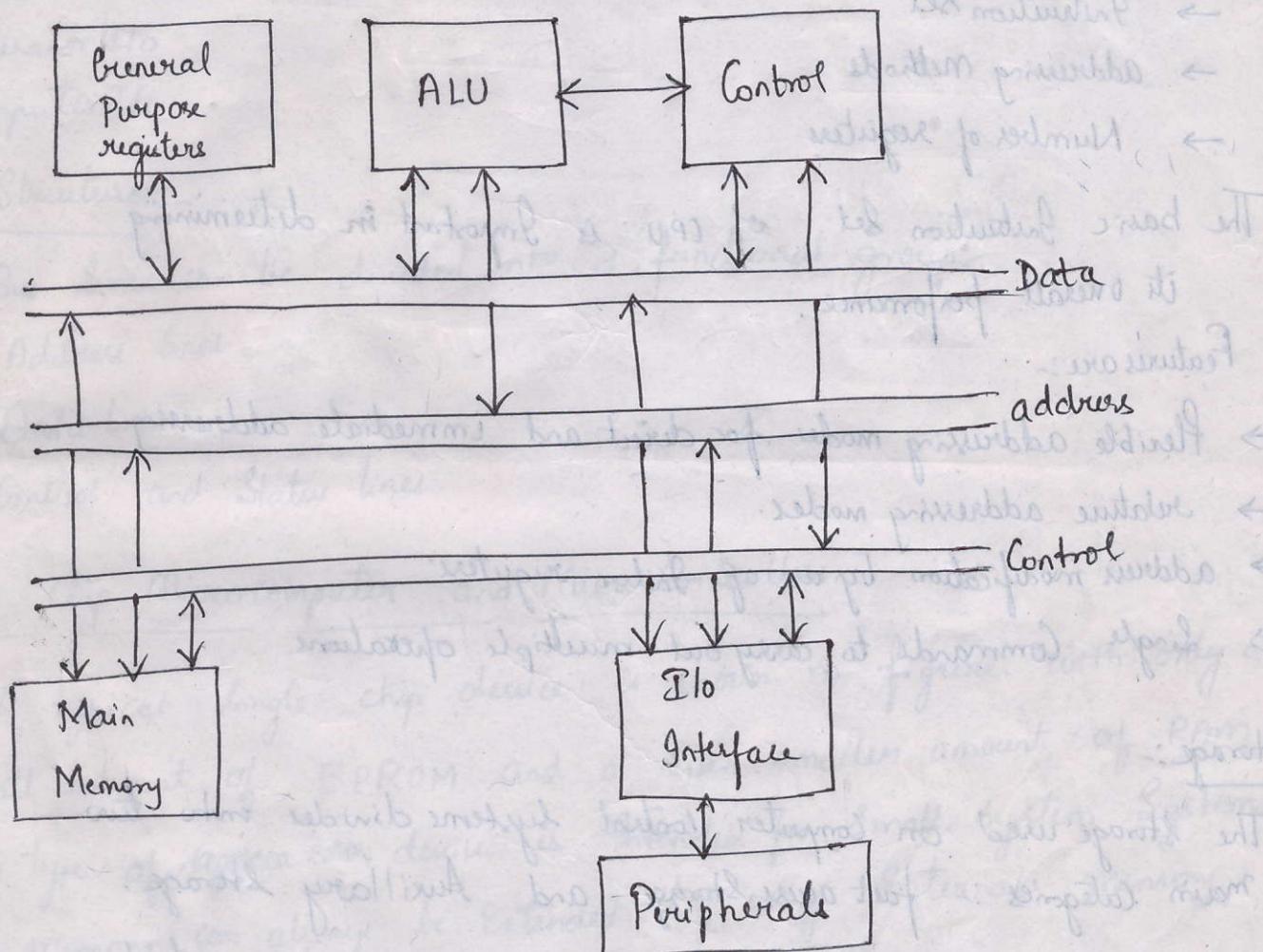
- The Economics of Computer Control have been changed drastically by the Microprocessor in that the reduction in cost and improves reliability.
- The availability of powerful, cheap and highly reliable computer hardware and communication system makes it possible to conceive and construct large, complex.

Computer Hardware Requirements for Real-time Applications

CBM

General purpose Computer :-

A Simplified block diagram of basic unit is shown in figure. the arithmetic and logic, control, register, memory and Input/Output Unit represent a general purpose digital computer.



Central Processing unit:

- The arithmetic and logic unit (ALU) together with the control unit and general purpose registers makes up Central processing unit (CPU).
- The ALU contains the circuits necessary to carry out arithmetic and logic operations, for example to add numbers, subtract numbers and compare two numbers.

→ The Control unit continually supervises the operations within the CPU; it fetches program Instructions from main Memory, decodes the Instructions and sets up the necessary data paths.

The features of CPU determine the processing power available and influence choice of Computer.

- Wordlength
- Instruction Set
- Addressing Methods
- Number of registers

The basic Instruction Set of CPU is important in determining its overall performance.

Features are:-

- flexible addressing modes for direct and immediate addressing
- relative addressing modes.
- address modification by use of Index registers.
- single Commands to carry out multiple operations.

Storage:-

The storage used on computer control systems divides into two main categories :- fast access storage and auxiliary storage.

- The fast access memory is that part of the system which contains data, programs and results which are currently being operated on.
- The auxiliary storage medium is typically disk or magnetic tape. These devices provide bulk storage for programs or data which are required infrequently at a much lower cost than fast access memory.

Input and Output:-

It is most of complex areas of a computer system, part of compilation arises because of variety of devices which have to be connected.

I/O System of most computer control can be divided into 3 sections:-

- Process I/O
- Operator I/O
- Computer I/O

Bus Structure:-

The Bus lines can be divided into 3 functional groups:

- Address lines
- Data lines
- Control and Status lines.

Single-Chip Microcomputer and Microcontrollers.

A typical single-chip device is shown in figure with only a small amount of EPROM and a even smaller amount of RAM. This type of ~~system~~ device is intended for small system systems. The memory can always be extended by using external memory chips.

The microcontroller is similarly a single-chip device that is specifically intended for embedded computer control applications. The chip may also contain a real-time clock generator and a watch-dog timer.

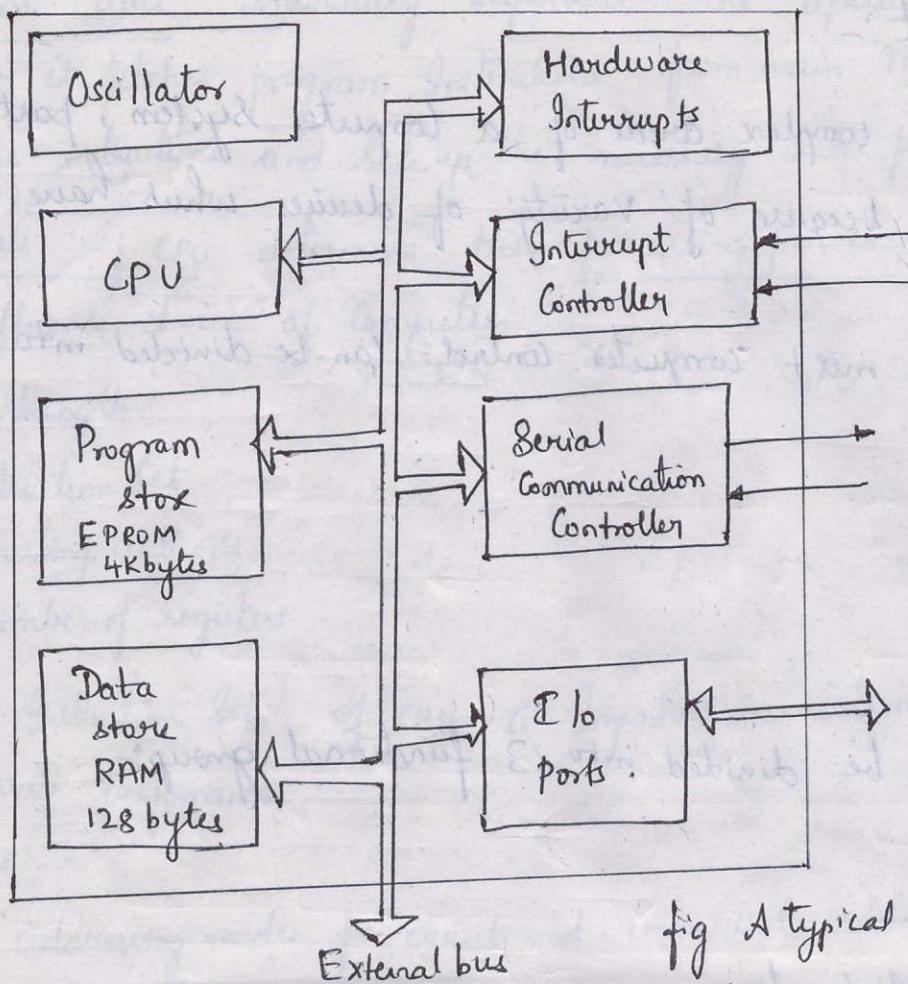


fig A typical single-chip computer

Specialised Processor:-

Specialised processors have been developed for two main purposes:-

- Safety Critical applications
- Increased Computation Speed.

For safety critical applications the approach has been to Simplify the Instruction set - so called Reduced Instruction Set Computer (RISC).

The advantage of simplifying the Instruction set is the possibility of formal Verification that the logic of the processor is correct.

1. Parallel Computers

The various Computer system architectures are:-

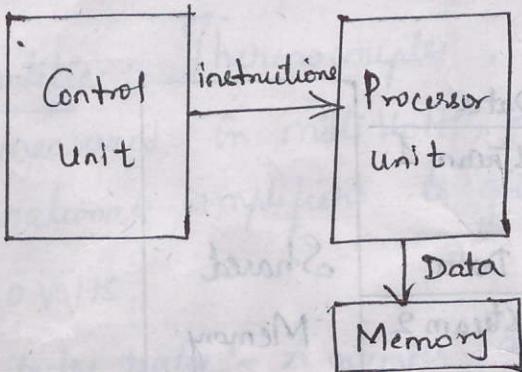
→ SISD → Single Instruction Stream, Single Data Stream

→ SIMD → Single Instruction Stream, Multiple Data Stream

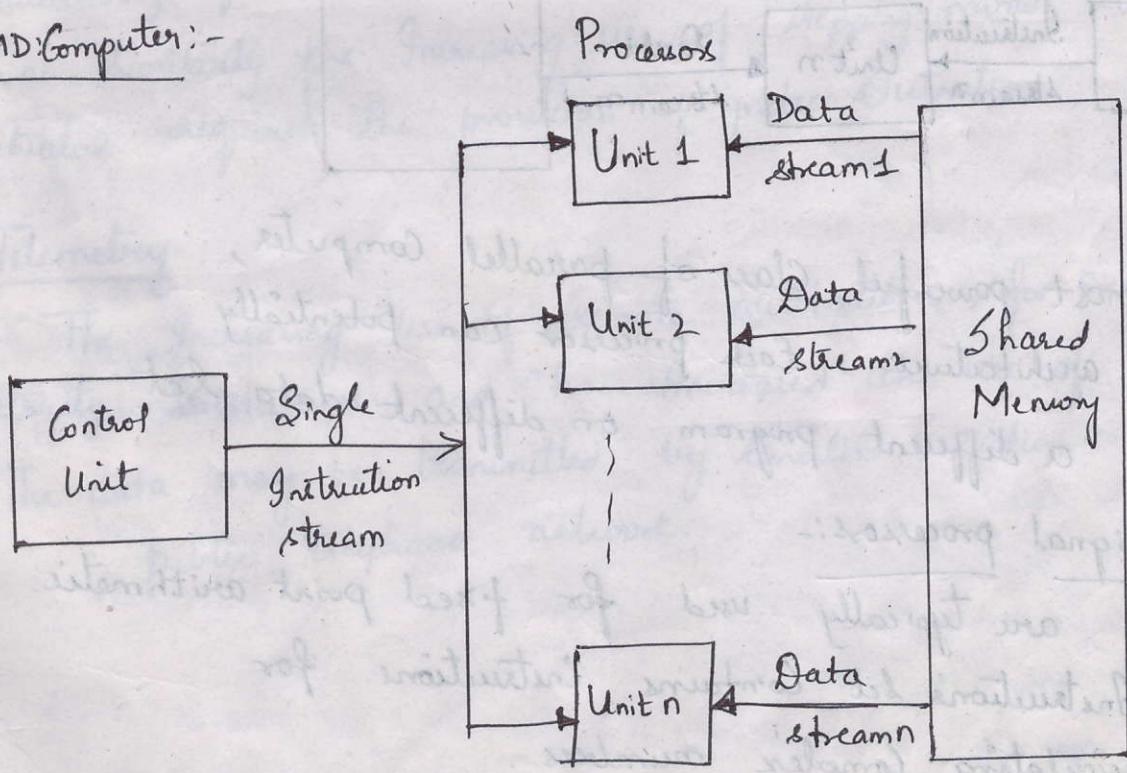
→ MISD → Multiple Instruction Stream, Single data Stream

→ MIMD → Multiple Instruction streams, multiple Data stream.

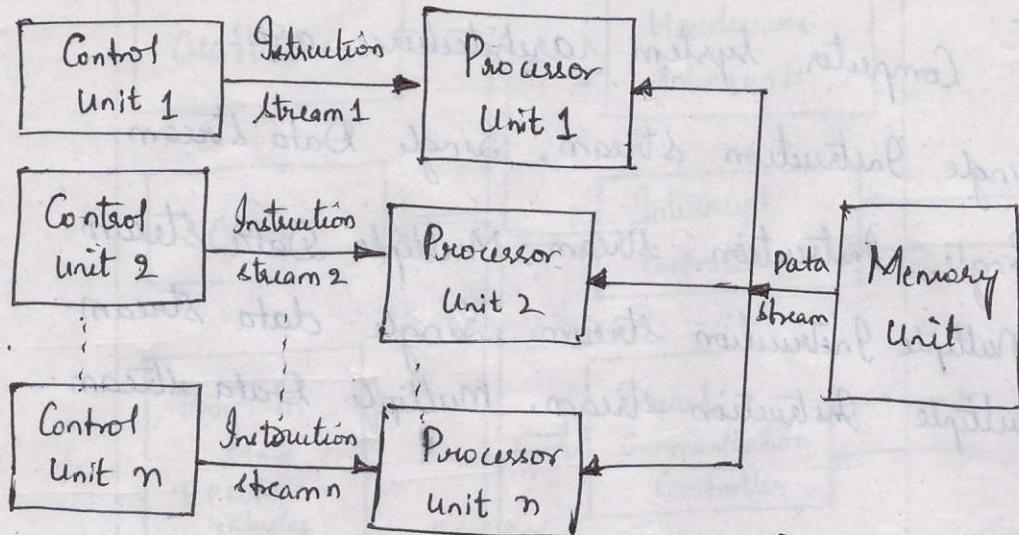
SISD Computer:-



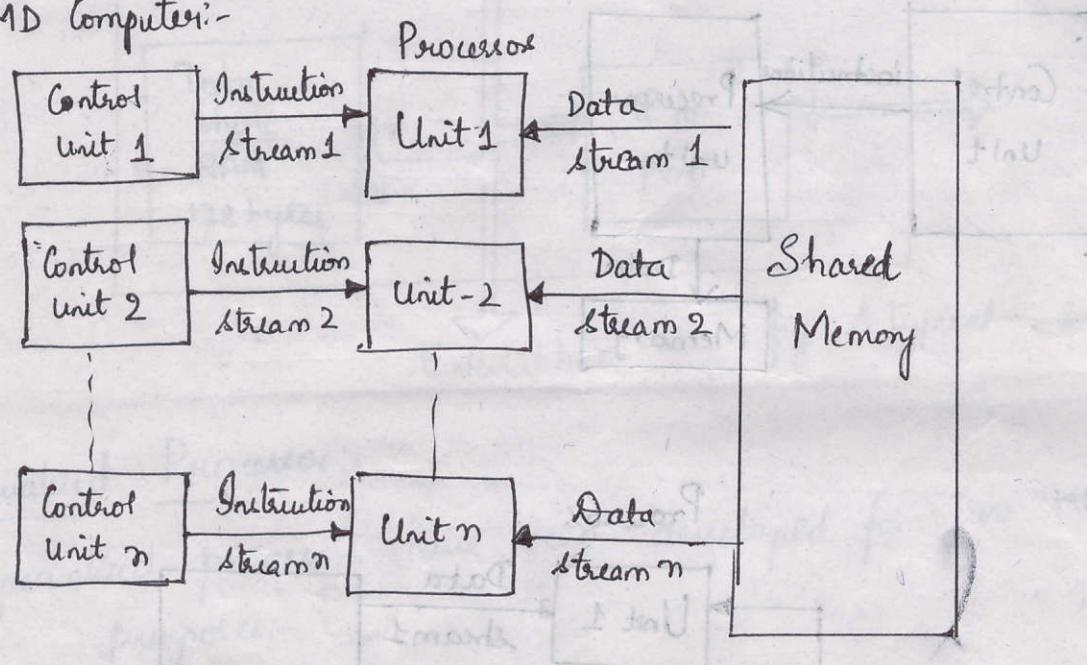
SIMD Computer:-



MISD Computer :-



MIMD Computer:-



MIMD - most powerful class of parallel computer,
 In this architecture, Each processor can potentially
 Execute a different program on different data set.

2. Digital Signal Processor:-

↳ DSP's are typically used for fixed point arithmetic
 and Instructions set Contains instructions for
 manipulating complex numbers.

Process - Related Interfaces

Instruments and actuators connected to process or plant can take a wide variety of forms: they may be used for measuring temperatures and hence we have thermocouples, resistance thermometers etc.

Most ^{RT} devices can be allocated to one of the following four categories :-

1. Digital quantities:- These can be either binary, that is a value is open or closed, a switch is on or off, a relay should be opened or closed.
2. Analog quantities:- Thermocouples, strain gauges, etc. give outputs which are measured in millivolts; these can be amplified using operational amplifiers to give voltage in the range -10 to +10 Volts.
3. Pulse and pulse rates:- A number of measuring instruments, particularly flow meters, provide output in the form of pulses. Similarly the increasing use of stepping motors as actuators requires the provision of pulse outputs.
4. Telemetry:- The increasing use of remote outstations, for example Electricity Substations and has increased the use of telemetry. The data may be transmitted by landline, radio or the public telephone network.

Digital Signal Interfaces :-

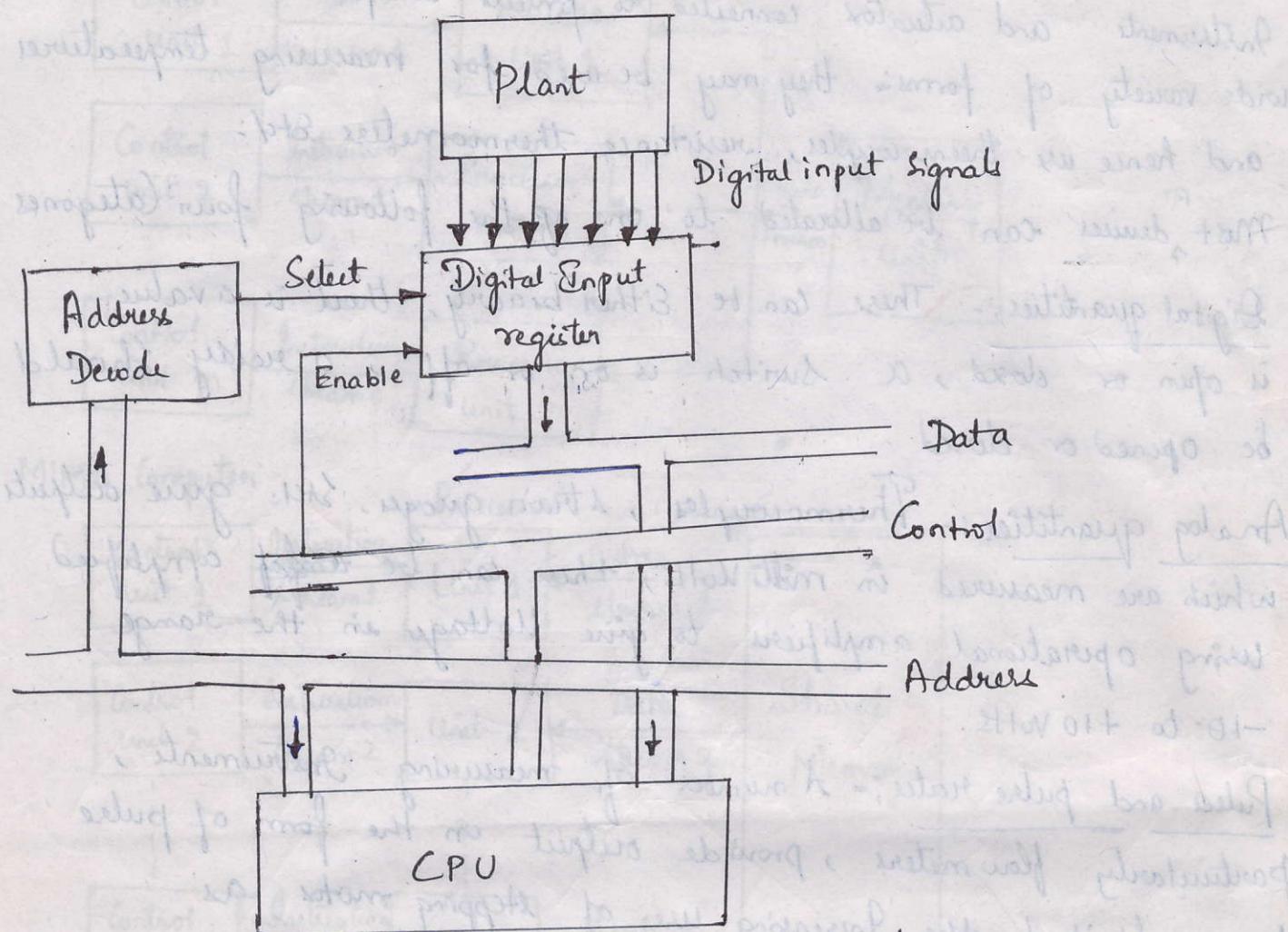


fig. Simple digital Input Interface

- The plant outputs are logic signals which appear on lines connected to digital input register. It is usual to transfer one word at a time to the computer.
- The logic levels on the input lines will typically be 0 and +5V.
- To read the lines connected to digital input register the computer has to place the address of register on address bus and decoder circuitry is required to select digital input register.
- In response to both the 'Select' and 'Enable' signals and digital input register enables its output gates and puts data onto the computer data bus.
- For proper operation of data bus the digital input register must connect its output gates to the data bus only when it is selected and enabled; if it connects at any other time it will corrupt data intended for other devices.

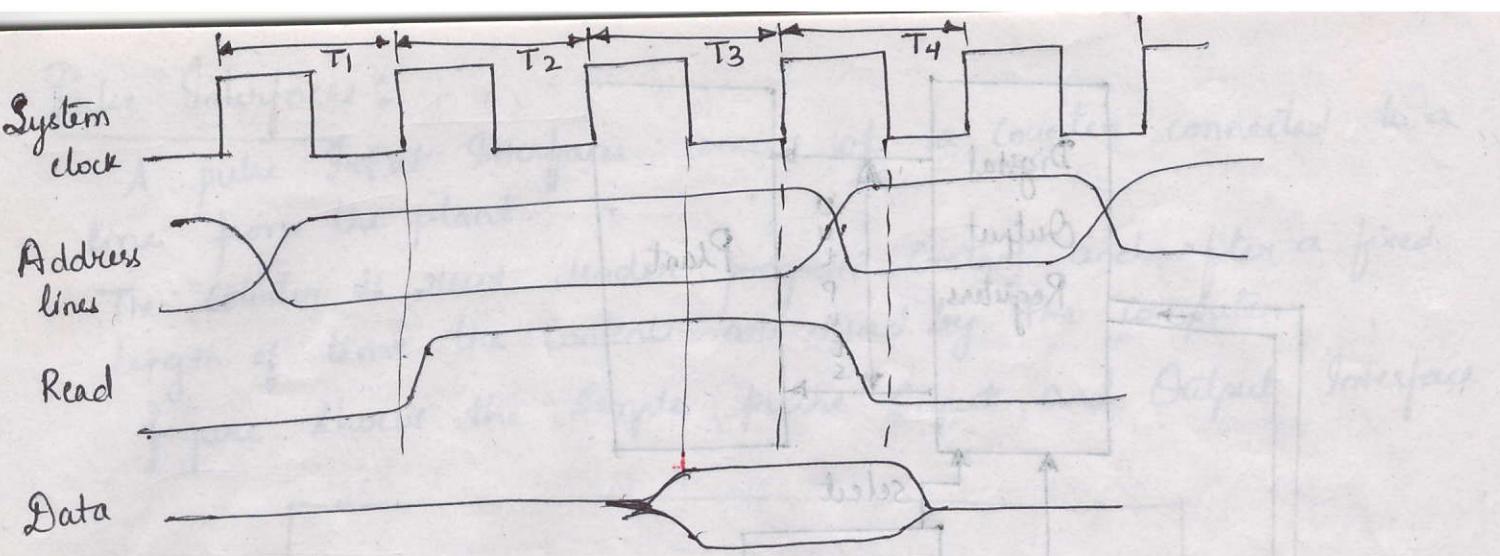


fig. Simplified READ (INPUT) timing diagram.

- The three cycles of system clock, labelled T₁, T₂ and T₃.
- The address lines begin to change at beginning of cycle T₁, and they are guaranteed to be valid by start of cycle T₂, and the READ line becomes active.
- For the correct ^{read} operation the digital input register has to provide stable ↑ data at negative-going Edge of clock during the T₃ cycle.
- The data must remain on data lines until negative going edge of clock cycle.
- Note that the actual time taken to transfer the data from the data bus to CPU may be much shorter than time for which the data is valid.

A simple Digital Output Interface is shown below.

- Digital Output is the simplest form of Output:
- all that is required is a register or latch which can hold data output from the computer.
 - The 'Enable' signal is used to indicate to device that the data is stable on data bus and can be read.
 - The latch must be capable of accepting the data in a very short length of time, typically less than 1 microsecond.

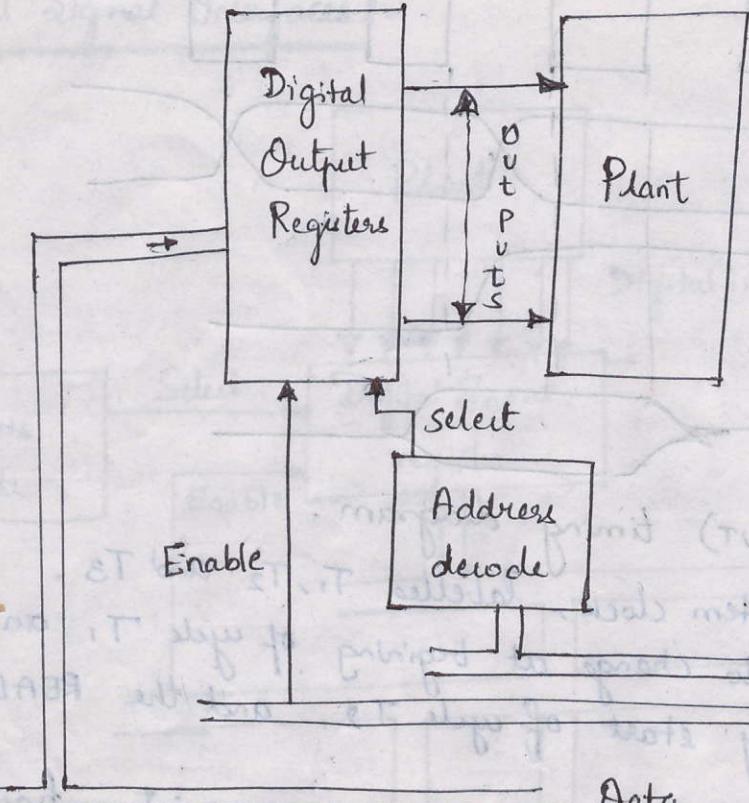


fig. Simple digital Output Interface

- The Output from the latch is a set of logic levels typically 0 to +5V.
- If these levels are not adequate to operate the actuators on the plant, some signal conversion is necessary.

Pulse Interfaces:

A pulse Input Interfaces consists of a Counter connected to a line from the plant.

The Counter is reset under program Control and after a fixed length of time the Contents are read by the computer.

figure shows the simple pulse Input and Output Interface

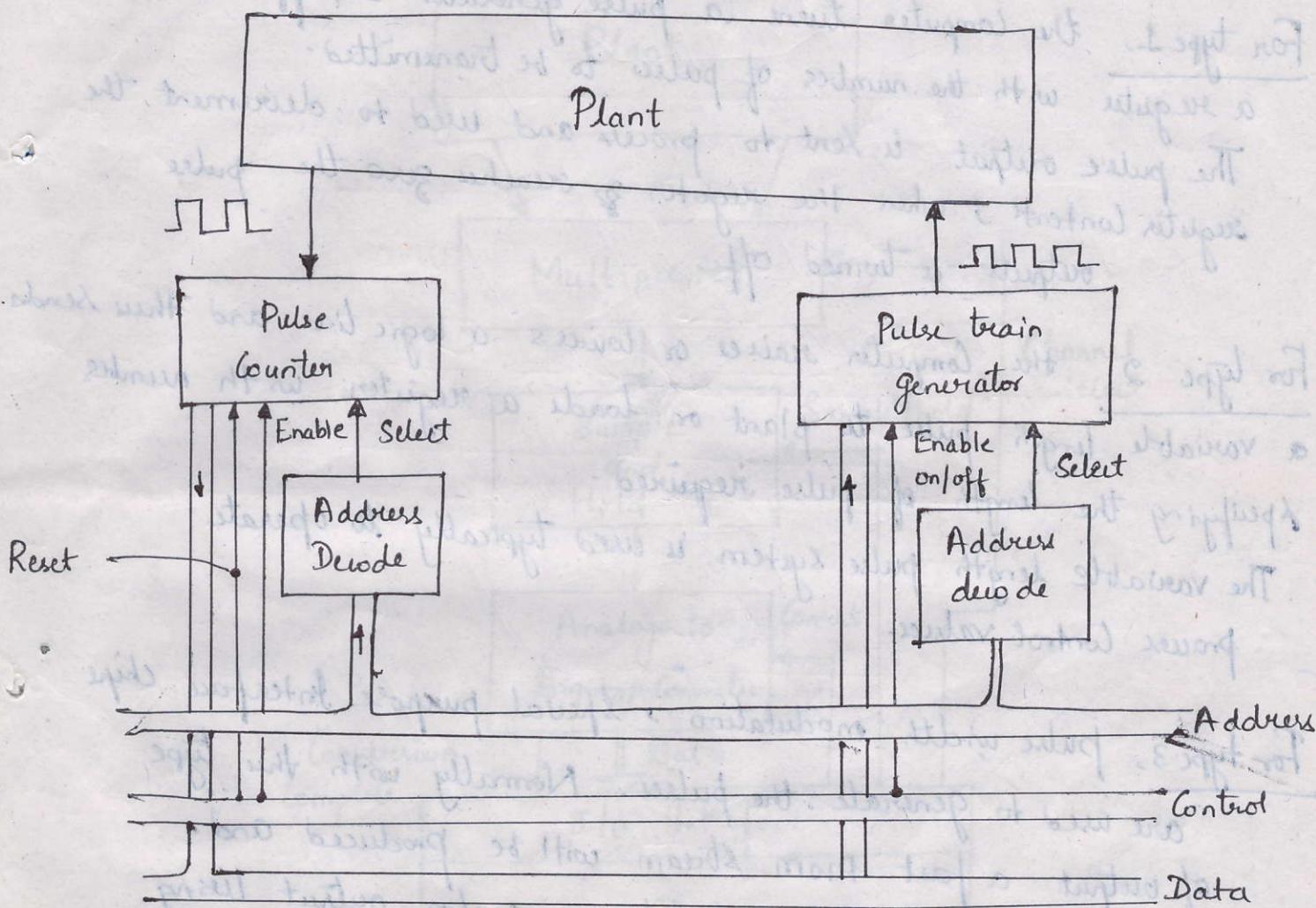


fig. Pulse Input and Output Interface

- The measurement of length of time for which the count proceeds can be carried out by logic circuit in Counter Interface
- If the timing is done by Computer then the 'Enable' signal must inhibit further Counting of pulses.

Pulse Outputs can take a variety of forms:

1. a series of pulses of fixed duration
2. a single pulse of variable length
3. Pulse width modulation - a series of pulses of different widths sent at a fixed frequency.

For type 1, the computer turns a pulse generator on/off, or loads

a register with the number of pulses to be transmitted.

The pulse output is sent to process and used to decrement the register contents; when the register reaches zero the pulse output is turned off.

For type 2, the computer raises or lowers a logic line and then sends a variable length pulse to plant or loads a register with numbers specifying the length of pulse required.

The variable length pulse system is used typically to operate process control valves.

For type 3, pulse width modulation, special purpose Interface chips are used to generate the pulses. Normally with this type of output a fast PWM stream will be produced and this can be converted into linear analog output using a low pass filter.

Analog Interfaces :-

The conversion of analog measurement to digital measurement

involves 2 operations : Sampling and quantisation.

→ Many ADCs include a 'Sample - hold' circuit on Input to device

→ This 'sample and hold' unit is used to prevent a change in the quantity being measured while it is being converted to a discrete quantity

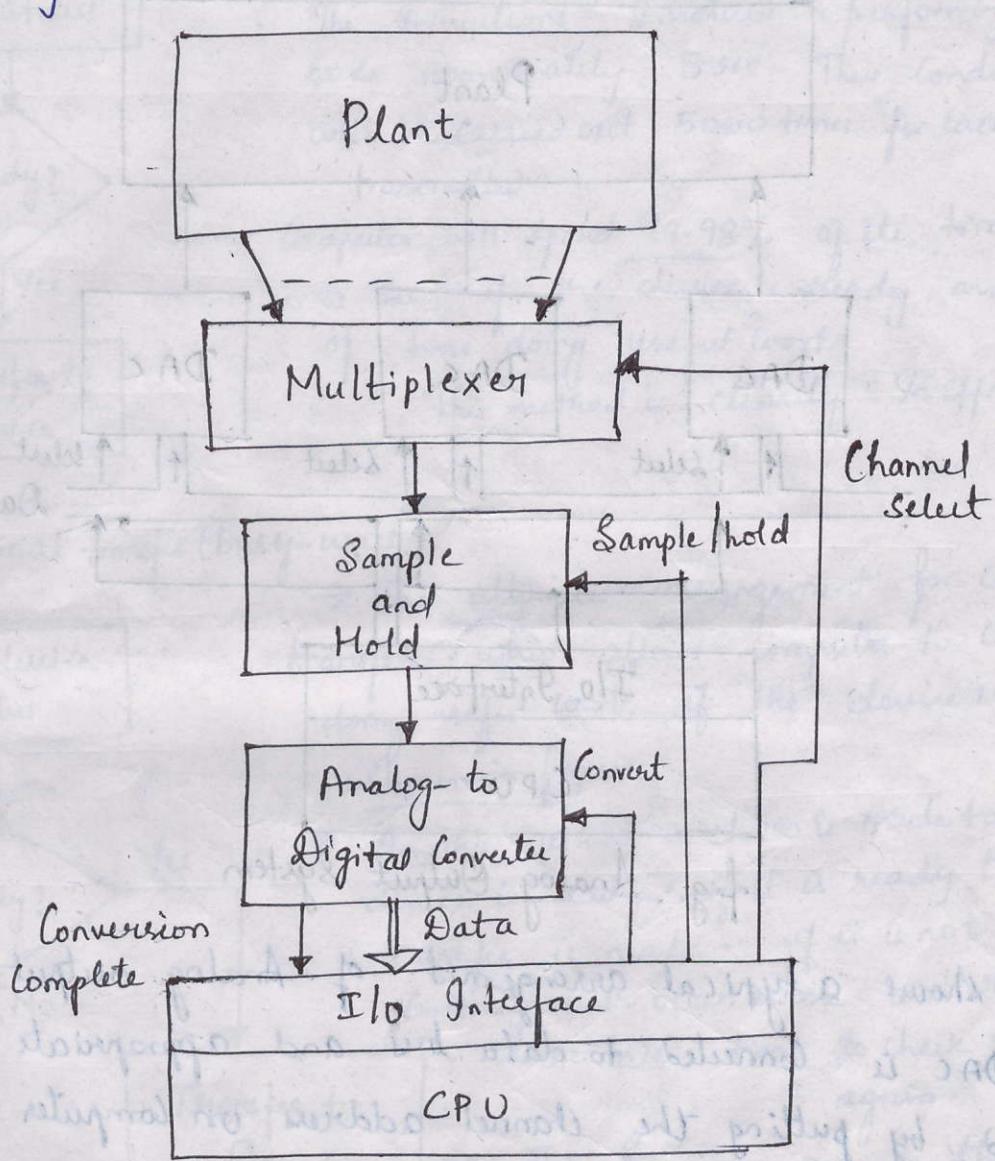


fig. Analog Input System

- To operate the analog input interface the computer uses a 'start' or 'sample' signal.
- Quantisation may take from a few microseconds to several milliseconds.
- On completion of conversion the ADC raises a 'ready' or 'complete' line which is either polled by the computer.

→ A multiplexer is used to switch the Inputs from several Input lines to a single ADC.

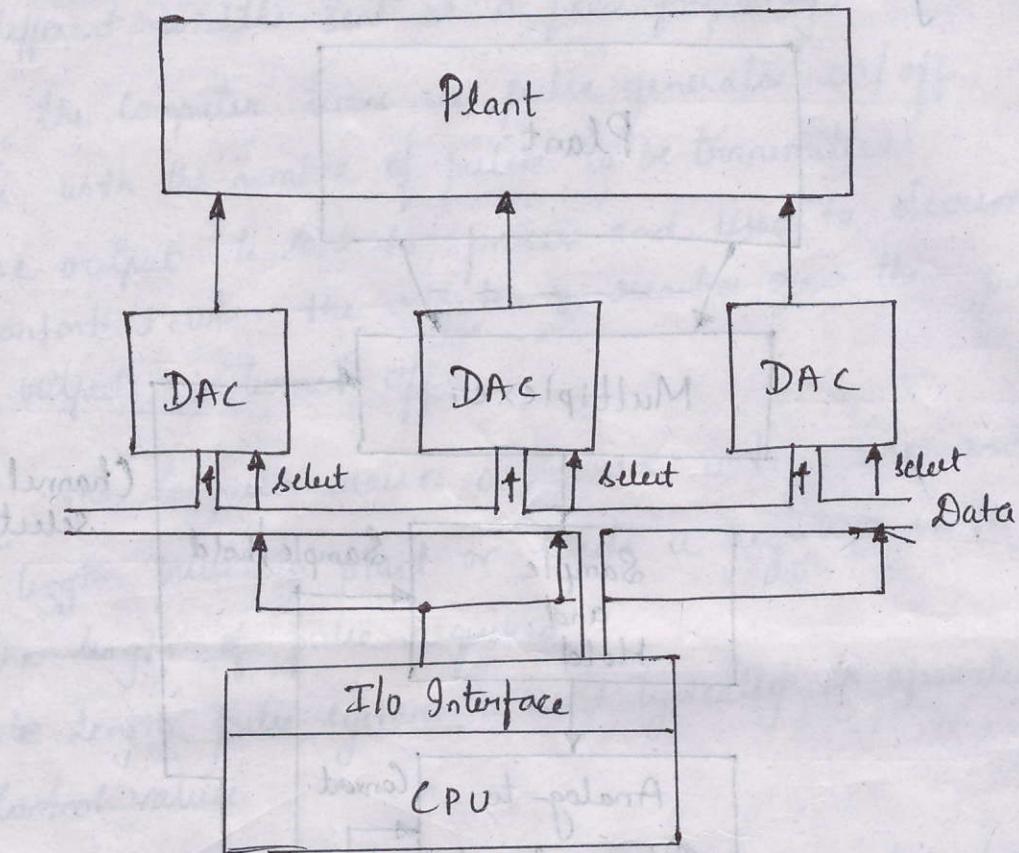


fig. Analog Output System

figure shows a typical arrangement of Analog output system. Each DAC is connected to data bus and appropriate channel is selected by putting the channel address on computer address bus.

- The DAC acts as a latch and holds the previous value sent to it until the next value is sent.
- The conversion time is typically from 5 to 20ms. and typical analog outputs are -5V to +5V, -10V to +10V or current output of 0 to 20mA.

Data Transfer Techniques

Polling:-

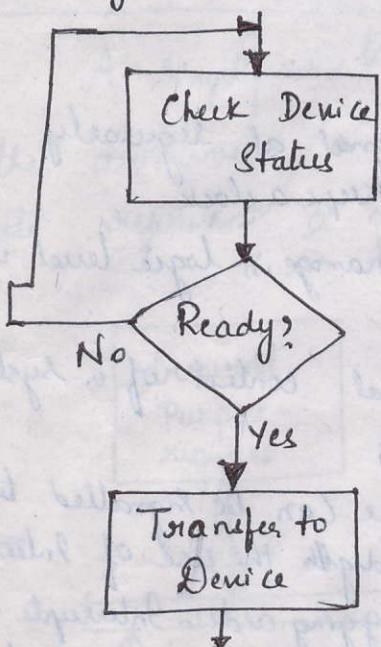


fig. Conditional transfer (busy-wait)

A simple Example of Conditional Transfer is shown in fig.

- Assuming data is being transferred to the printer which operates at 40 characters/second, the computer will find device is ready Every 25ms.
- The Instructions Involved in performing test will take approximately 5 ms. Thus Conditional test will be carried out 5000 times for each character transmitted.
- Computer will spend 99.98% of its time in checking to see if the device is ready and only 0.02% of time doing useful work.
This method is clearly In efficient.

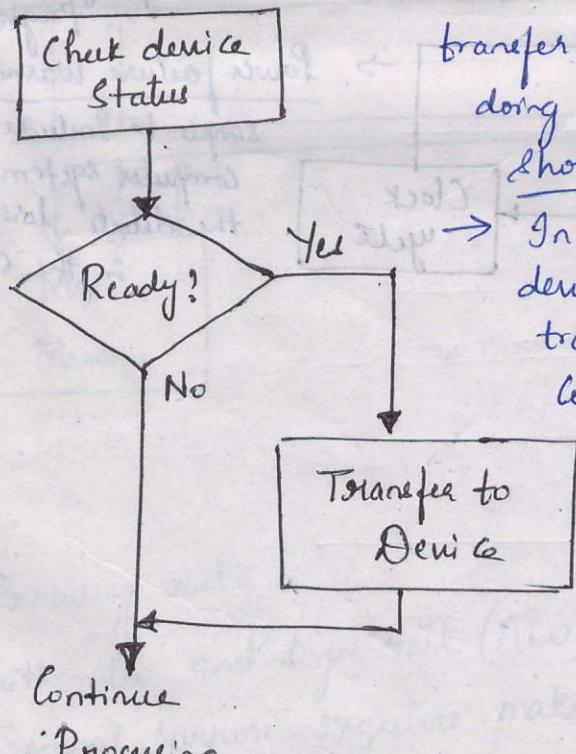


fig. Conditional Transfer.

→ The alternate arrangement for conditional transfer, which allows Computer to continue doing useful work if the device is busy is shown in figure

→ In this method a check is made to see if the device is ready & if it is ready then the transfer is made & if it is not Computer continue with other work and returns at some later time to check the device again.

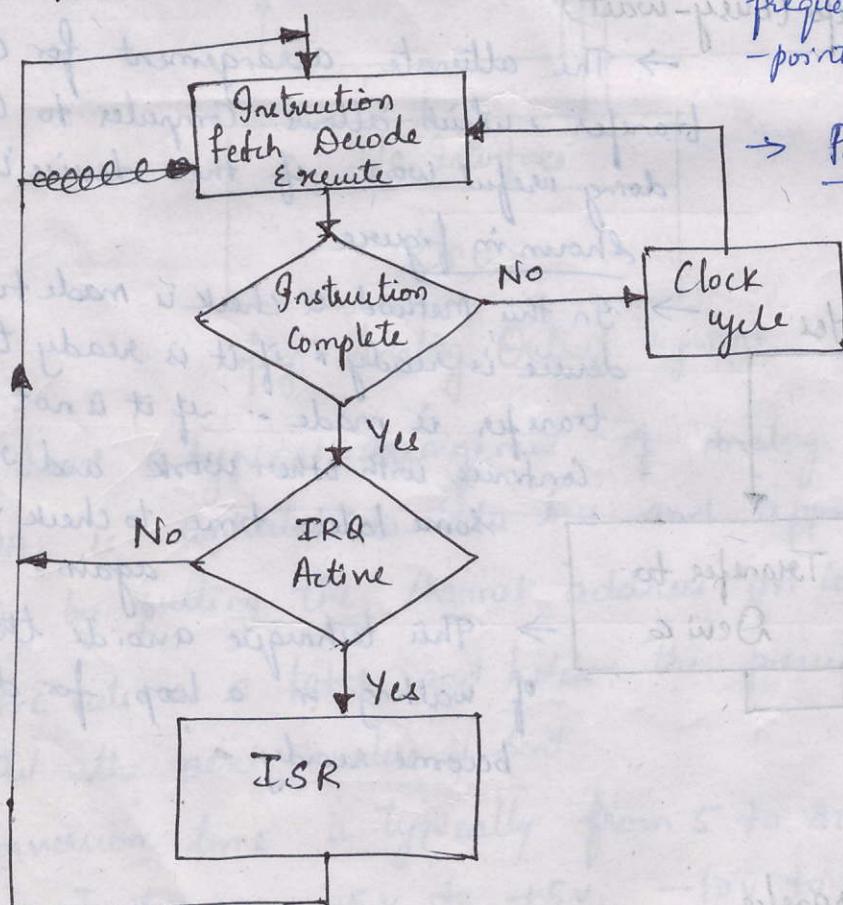
→ This technique avoids the inefficiency of waiting in a loop for device to become ready.

→ Conditional transfer technique involve polling, which is using the Computer to check whether a device is ready for a data transfer.

Interrupts :-

- An Interrupt is a mechanism by which the flow of program can be temporarily stopped to allow a special piece of software called Interrupt Service Routine (ISR).
- ✓ Use of Interrupts :-
 - Real time clock :- This External hardware provides a signal at regularly spaced Intervals - the ISR counts the signals and keeps a clock.
 - Alarm Inputs :- Various sensors can be used to provide a change in logic level in the Event of an alarm.
 - Manual override :- Use of an Interrupt can allow external control of a system to allow for maintenance and repair.
 - Hardware failure Indication :- Failure of External hardware can be signalled to the processor through the use of Interrupt

Flowchart of basic Interrupt Mechanism.



→ Debugging aids:- Interrupts are frequently used to insert break points in the program for program testing

→ Power failure Warning :- It is simple to include in the computer system a circuit that detects loss of power in the system.

Interrupt Input Mechanism:-

- A common arrangement is shown in the figure.
- The IRQ line can be enabled and disabled using software and hence computer can run in a mode in which External Events cannot disturb the processing.
- A second interrupt line is provided, thus Interrupt cannot be turned off by software and hence said to be a Non-maskable Interrupt (NMI). A typical use would be to provide the power failure detect interrupt.

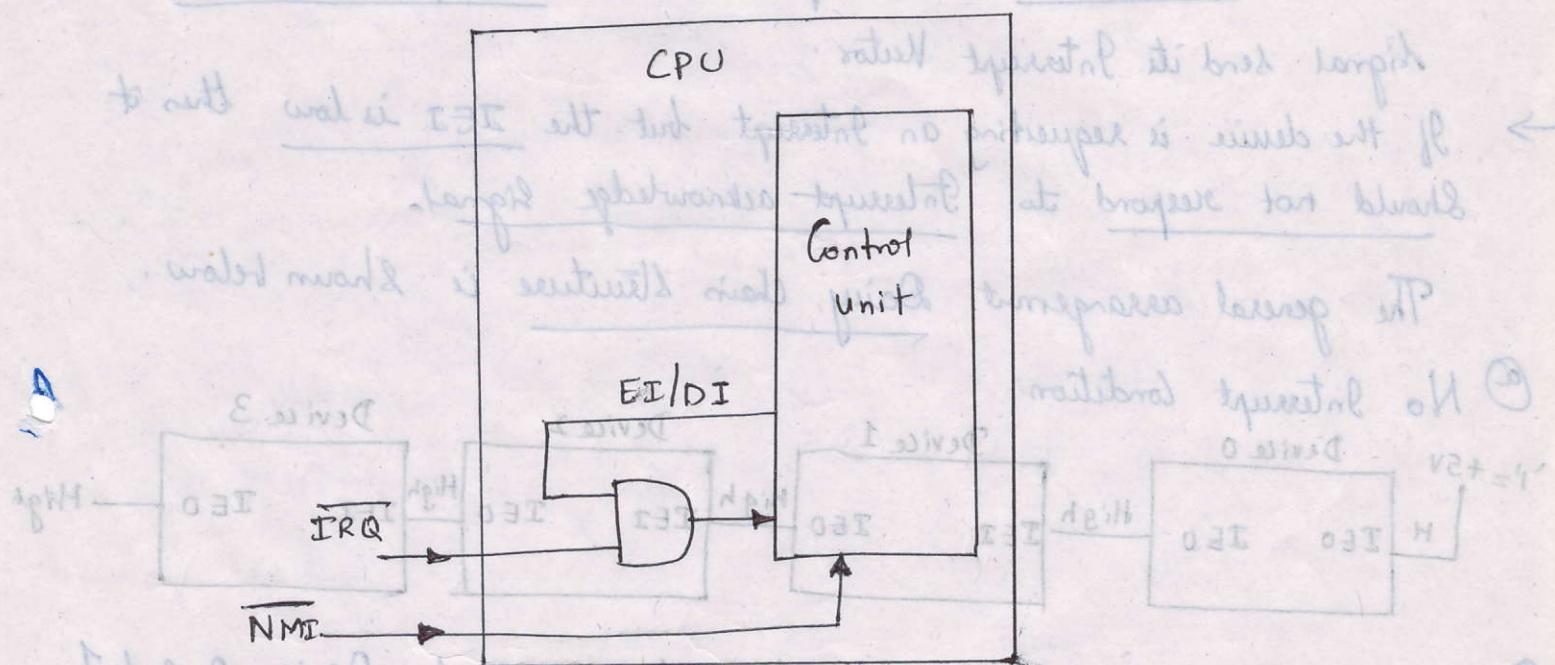
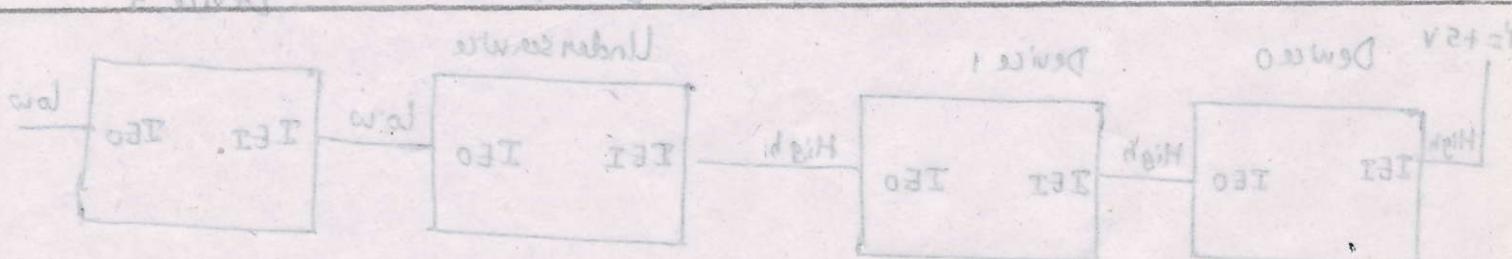


fig. Typical Basic Interrupt System.



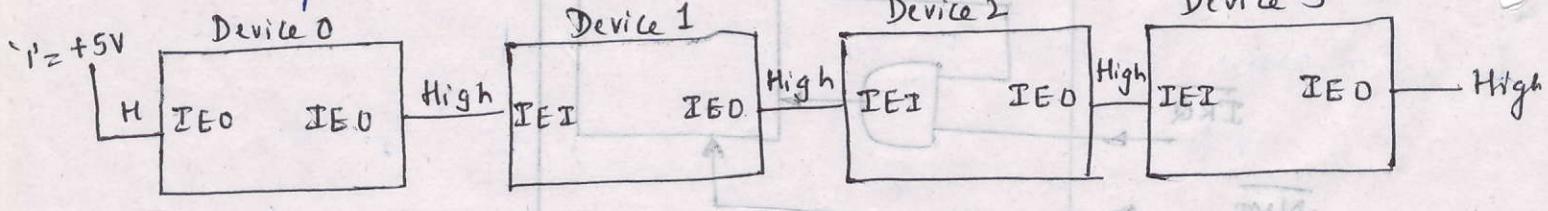
* Daisy chain Interrupt Structure:-

In Daisy chain an "acknowledge" signal is propagated through the devices until it is blocked by the interrupting device.

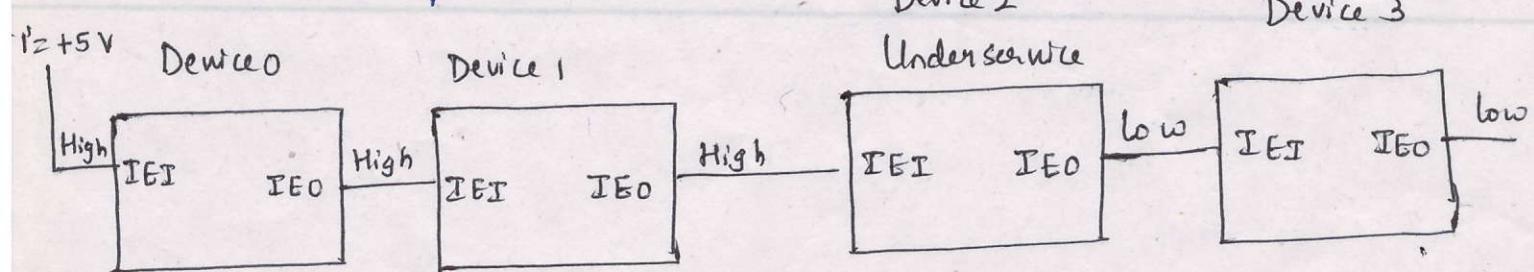
- Each unit has IEI [Interrupt Enable in] and IEO [Interrupt Enable output] pins. Both pins active signal is HIGH.
- The first IEI in chain is set permanently "HIGH" for any given unit the output pin IEO is high iff Input IEI is high and unit is not requesting an interrupt
- If a device is requesting an interrupt and IEI is high that device should set IEO low and respond to an "Interrupt Acknowledge" signal send its interrupt vector.
- If the device is requesting an interrupt but the IEI is low then it should not respond to Interrupt acknowledge signal.

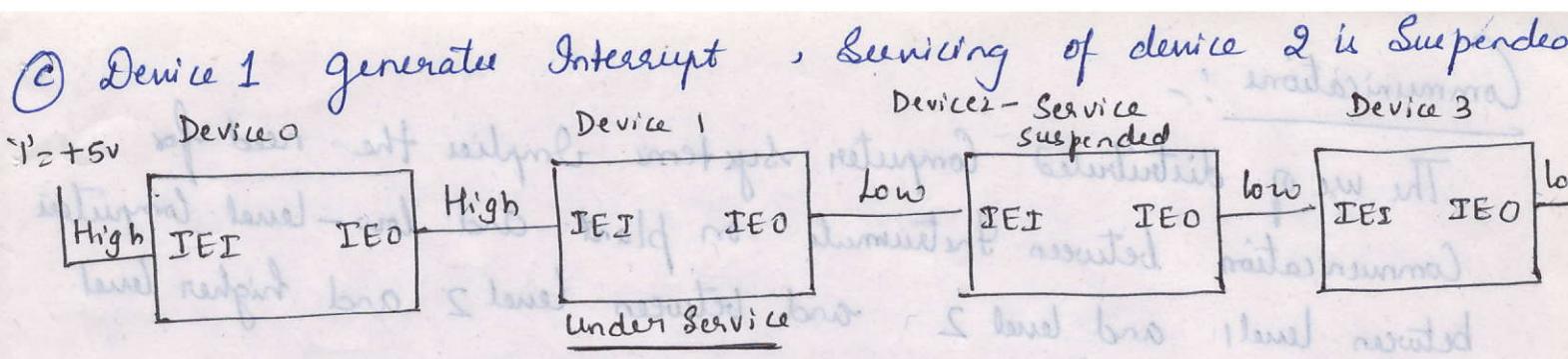
The general arrangement of Daisy chain structure is shown below.

(a) No Interrupt Condition

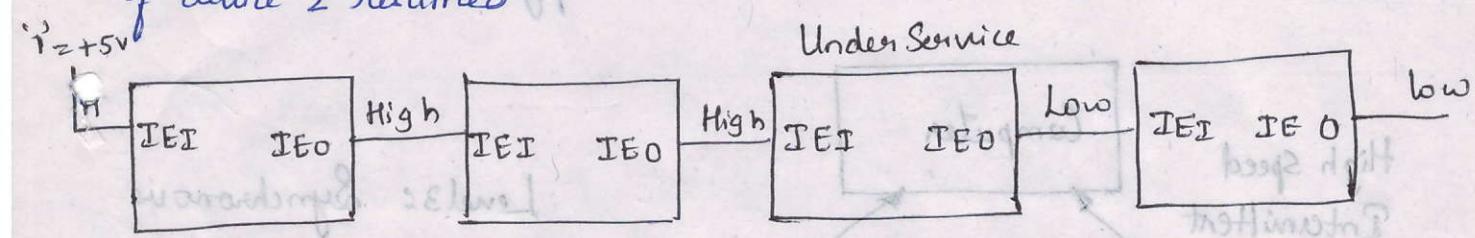


(b) Device 2 is Serviced, Device 3 is locked out and Device 0 and 1 can still interrupt

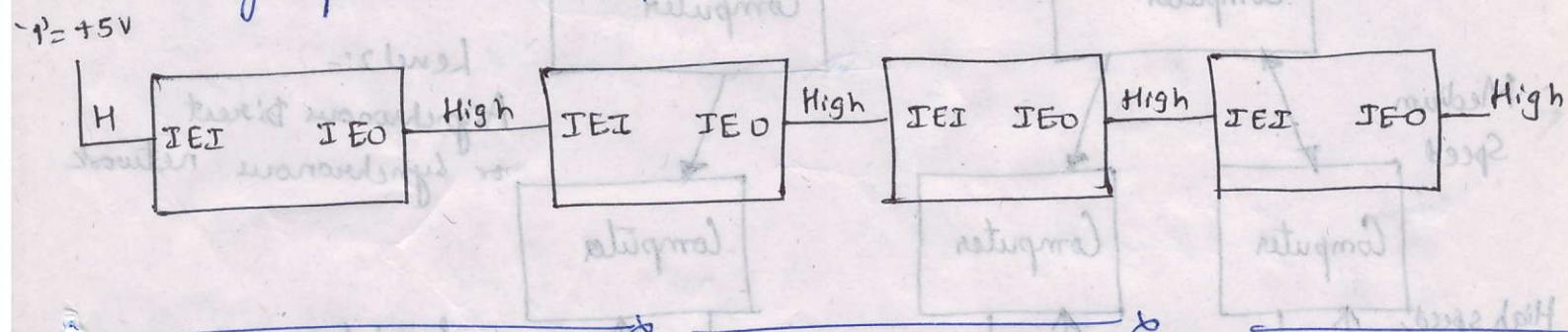




(d) Device 1 servicing completed, 'RET' Instruction Executed, servicing of device 2 resumed



(e) Servicing of device 2 completed and 'RET' Executed



Interrupt response modes (Page no 91)

Multilevel interrupt

Direct memory access

Ref. Text book

(Page no 92)

- Interrupt Element

(Page no 101)

for two devices

• interrupt element is triggered at level threshold

• interrupt occurs at level threshold and pending interrupt is generated. If interrupt is generated at level threshold then it will be triggered at level threshold.

- o trigger threshold is minimum so it will happen before all others

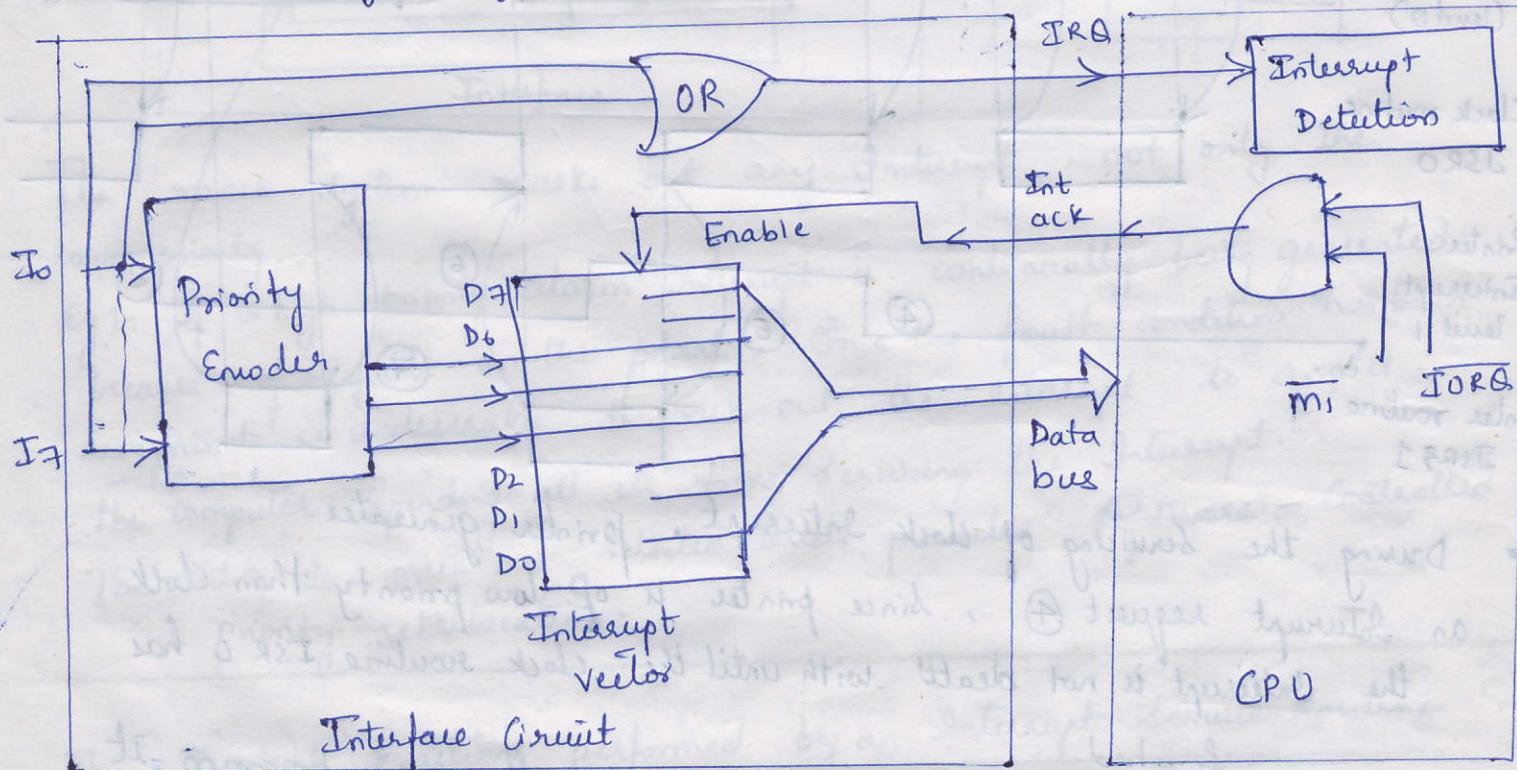
• trigger

Interrupt response mechanism:-

CPU may respond to the Interrupt in a variety of ways:-

1. Transfer control to a specified address - using "call" instruction
2. Load the program counter with a new value from a specified register or memory location
3. Execute a "call" instruction but to an address supplied from the External system.
4. Use an Output signal - an Interrupt Acknowledge to fetch the Instructions from an External device.

Interrupt Vetoing using priority encoding circuit



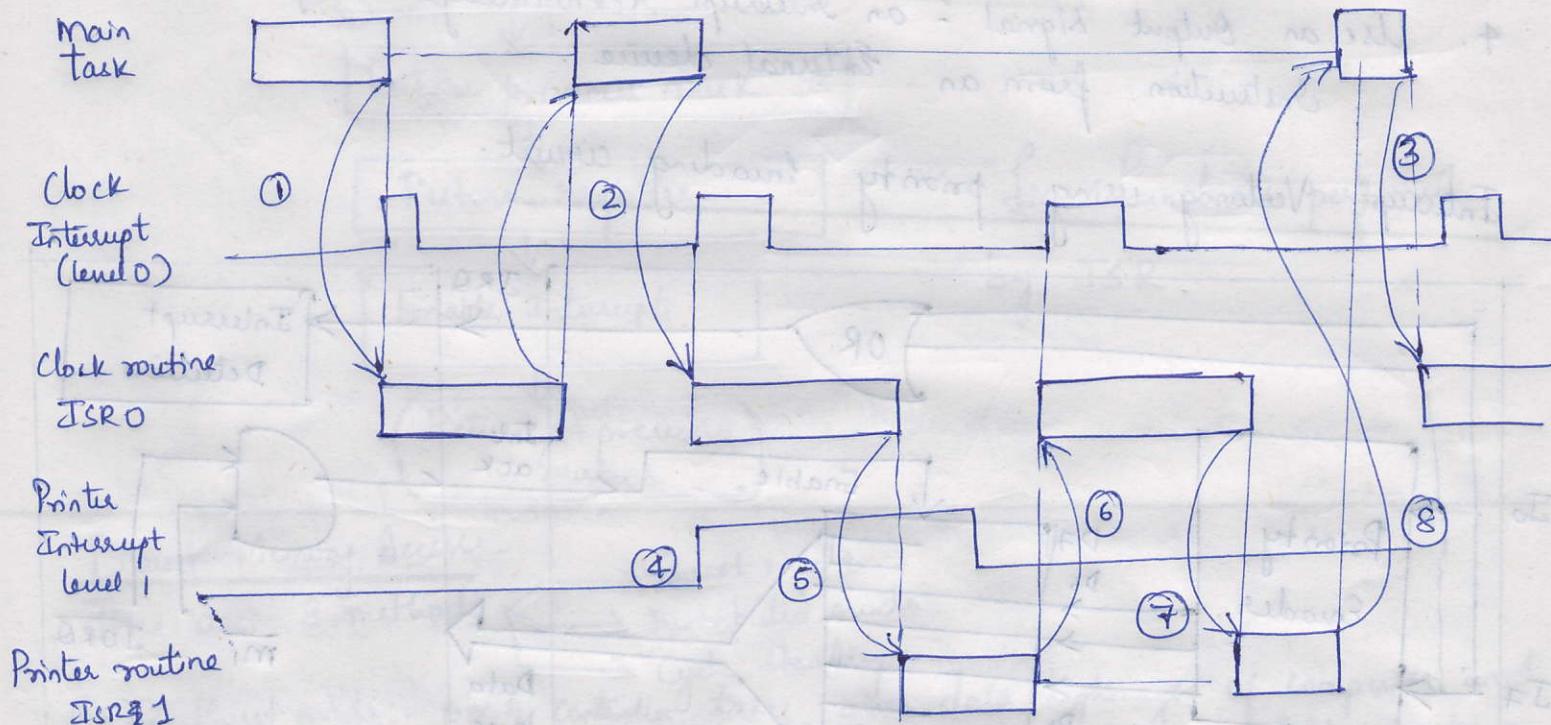
→ In this system an Interrupt Occurring on any line causes the Interrupt line (IRQ) to become active and place a 3 bit code specifying the number of interrupt line which is active on the data bus.

Is In the Event of more than one line becomes active the priority encoder supplies the number of highest - priority Interrupt . lowest number is considered to be of highest priority.

Multilevel Interrupt

Typical structure of Multilevel Interrupt is shown in fig.

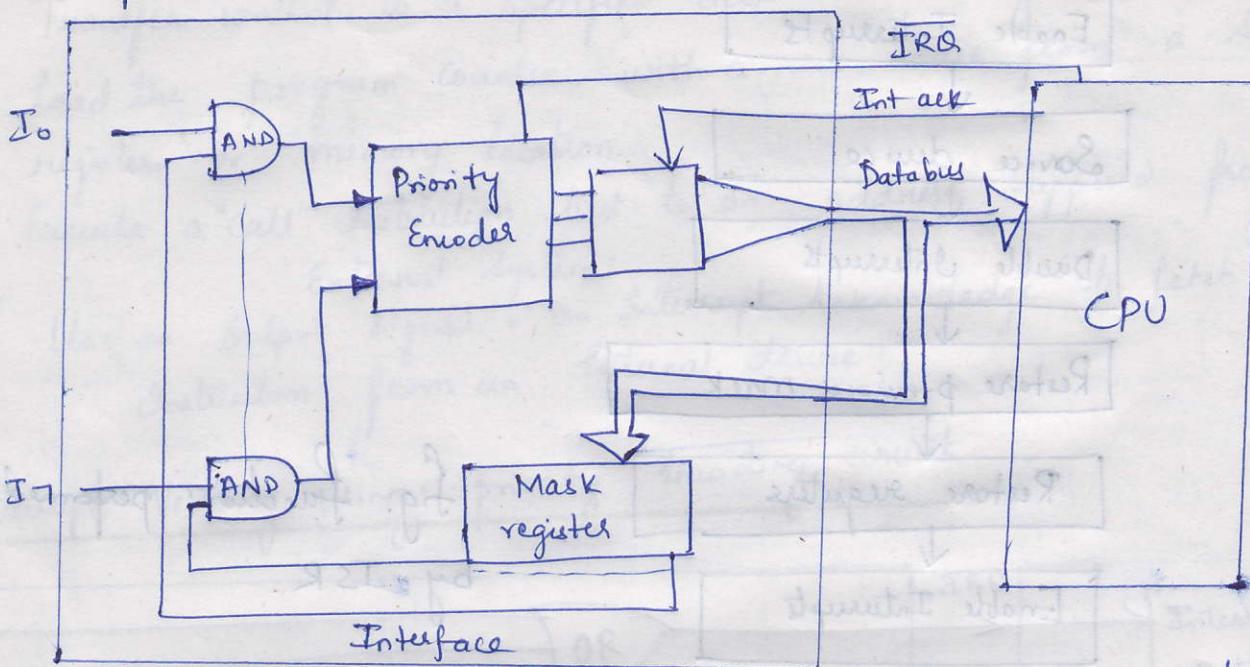
- The main task is interrupted at regular intervals by clock interrupt which is highest priority interrupt (level 0).
- when the interrupt occurs control is passed to clock interrupt service routine (ISR0) - transfers 1, 2, and 3 as in figure.



- During the servicing of clock interrupt, printer generates an interrupt request ④, since printer is of low priority than clock the interrupt is not dealt with until the clock routine ISR0 has finished.
- When this occurs, instead of control returning to the main program it passes to the printer service routine ISR1(5).
- The printer Service routine does not complete before the next clock interrupt, so it is suspended ⑥ while the next interrupt from the clock is dealt with.
- At the termination of the clock routine return is made to the printer ⑦ and finally, when printer ISR finishes, a return is made to main program ⑧.

Interrupt Masking.

An alternative scheme is used to have a mask register which can be loaded from software and to use low-priority interrupt lines.

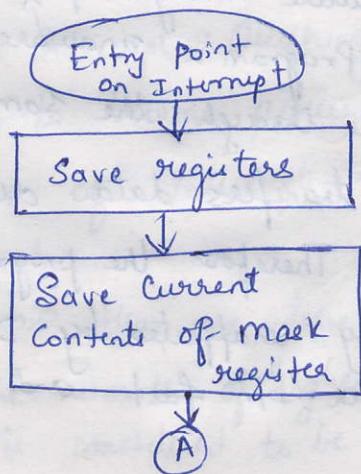


The mask system masks out any interrupt, not only the lower priority.

Eg): if a high priority alarm interrupt is continually being generated because of a fault on the plant; Once the fault condition has been recognised it is desirable to mask out the interrupt to avoid the computer spending all its time servicing the interrupt.

The ability to mask out selected levels provides software - controlled priority re-allocation.

The typical functions performed by an interrupt service routine is as follows.



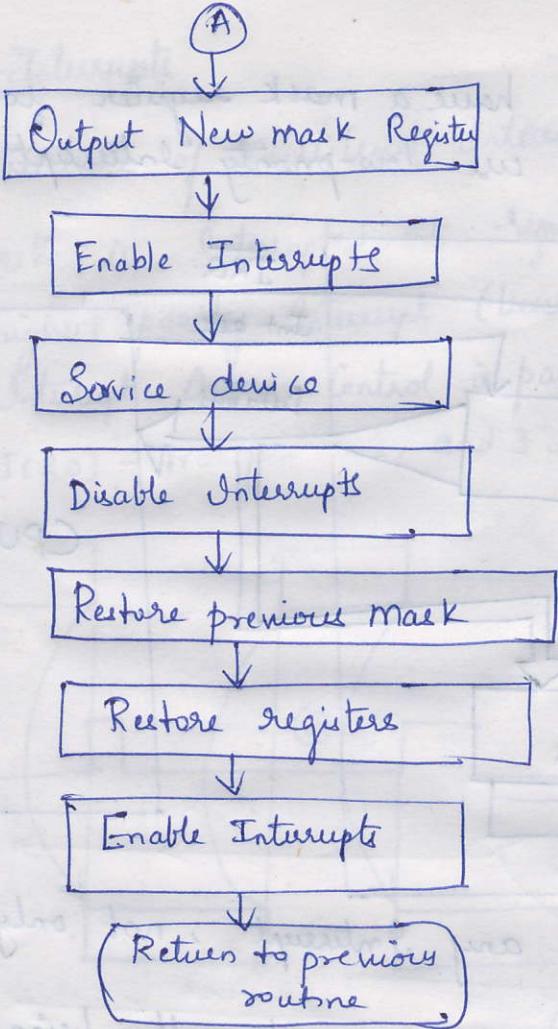
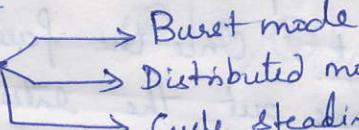


Fig. functions performed by ISR.

Direct Memory Access:-

There are 3 methods



① In Burst mode:- DMA controller takes over data highways of computer and locks out CPU for a period of time necessary to transfer b/w fast & backing memory. This mode can affect the response time of RTS to an external event.

② Distributed Mode:- the DMA Controller takes occasional m/c cycle from CPU control and uses each cycle to transfer a byte of information between fast and backing memory.

In real time systems, use software timing loops, which then affect the time taken to complete timing loop. Program is unaware of m/c cycles used by DMA controller and hence will cycle through the same no. of instructions.

③ Cycle Stealing Method:- only transfers data during cycles when CPU is not using the data bus. Therefore the program proceeds at the normal rate and is completely unaffected by DMA data transfer. This is slowest method of transfer b/w fast and backing store.

Communication :-

The use of distributed Computer Systems implies the need for communication between Instruments on plant and low-level computer between level 1 and level 2 and between level 2 and higher level computers.

The Data transmission links is shown in figure below.

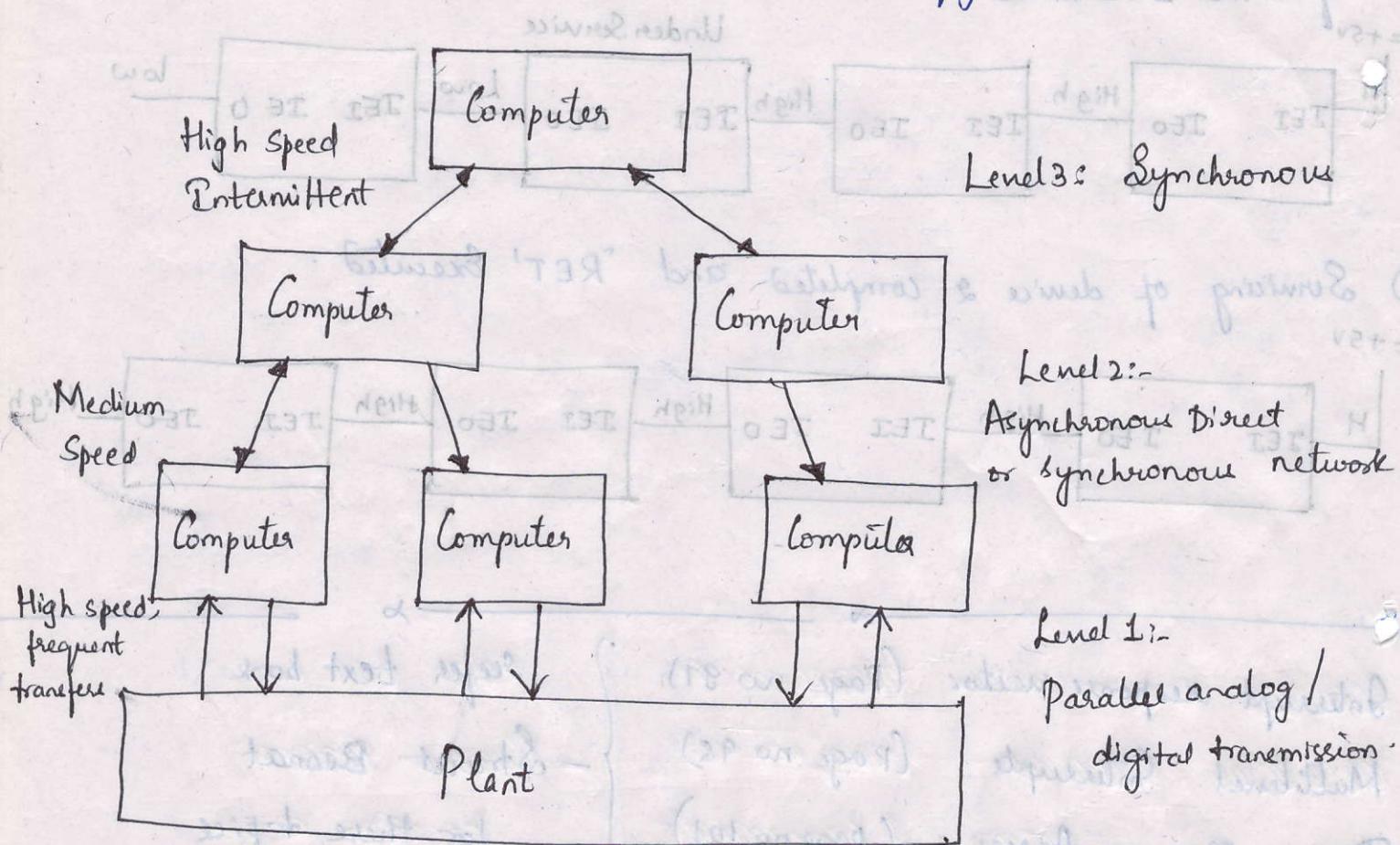


fig. Data transmission links.

- The plant level Communication systems it typically involves parallel analog and digital signal transmission techniques. Since the distances over which communication is required are small.
- At higher levels it is more usual to use serial communication method since, as communication distance extend beyond a few 100 yards.

Serial communication techniques can be characterized in general ways.

1. Mode
 - (a) Asynchronous
 - (b) Synchronous
2. Quantity
 - (a) Character by character
 - (b) Block
3. Distance
 - (a) Local
 - (b) Remote
4. Code
 - (a) ASCII
 - (b) Other

Asynchronous and Synchronous Transmission Techniques.

The most common form of Asynchronous transmission is character-by-character system, which is frequently used for connecting terminals to computer equipment. It is sometimes called 'Stop-Start' system.

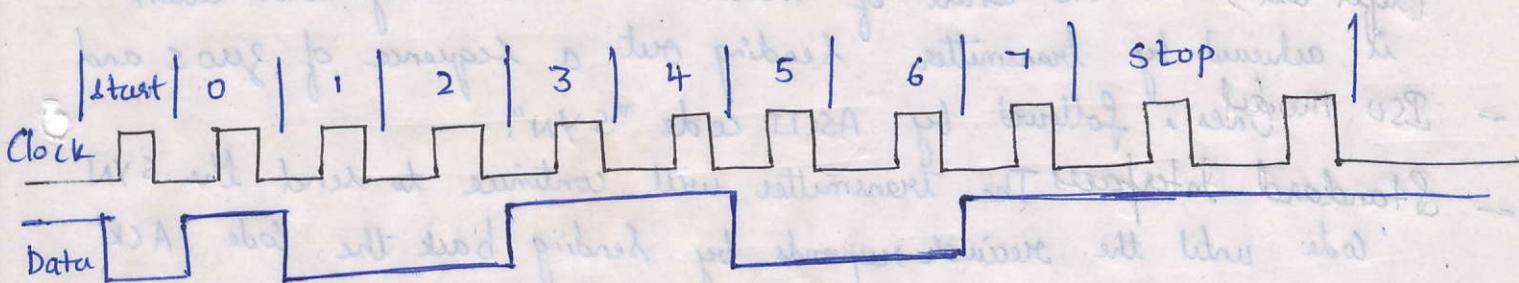


fig. Asynchronous transmission

- In this system each character transmitted is preceded by a 'start' bit and followed by one or two 'stop' bits.
- The 'start' bit is used by the receiver to synchronize its clock with the incoming data. For correct transfer of data the clock and data signals must remain synchronized for time taken to receive 8 data bits and 2 stop bits.
- Advantage of start-stop system is that at lower transmission rates, the frequencies of clock signal generators do not have to be closely matched.

→ Disadvantage is that for each character transmitted (7 bits) 3 or 4 bits of information have also to be transmitted and thus overall information ratio is not very high.

For effective communication, synchronization signal must be transmitted. This information is called protocol.

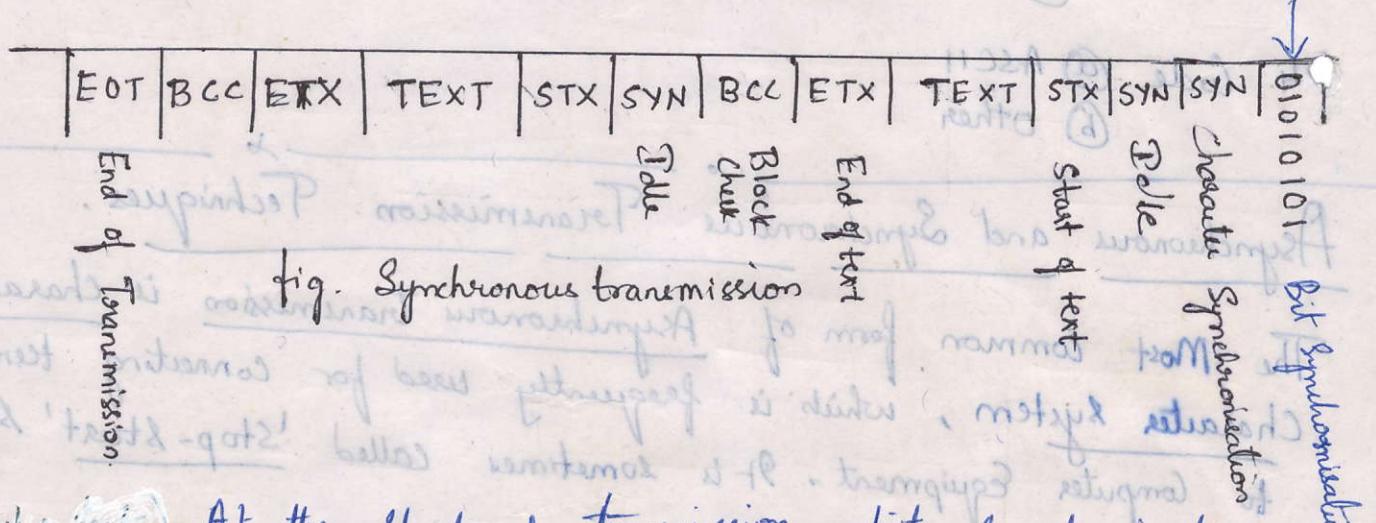


fig. Synchronous transmission

(a) At the start of transmission, bit synchronization is achieved by transmitter sending out a sequence of zeros and ones, followed by ASCII code "SYN".

(b) The transmitter will continue to send the SYN code until the receiver responds by sending back the code 'ACK'.

Once the contact has been established the transmitter will send out SYN characters during any idle period and receiver will respond by sending back 'ACK'.

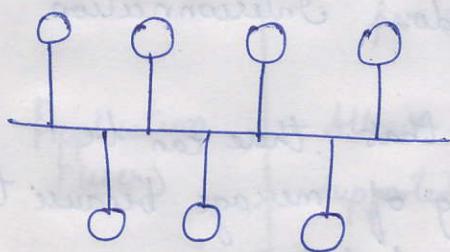
The line will only be completely idle when an EOT (End of Transmission) character has been sent by transmitter.

→ The text is broken up into blocks and each block is preceded by an STX (Start of Text) character and ended by an ETX.

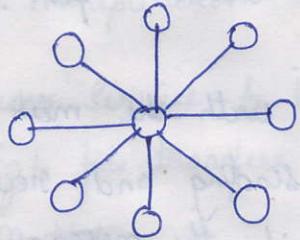
Local and Wide area Networks

Wide area networks operate over a wide geographical area at moderate speeds.

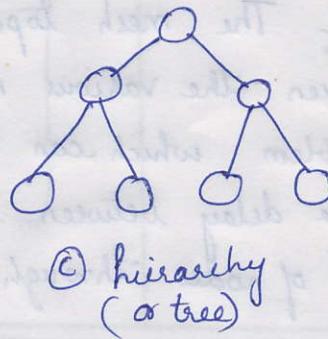
Local area networks make use of wide range of transmission media such as twisted pair, coaxial cable and fibre optics.



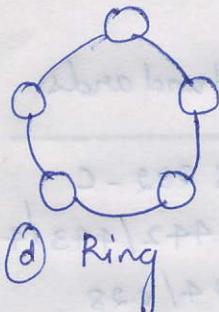
(a) Data bus



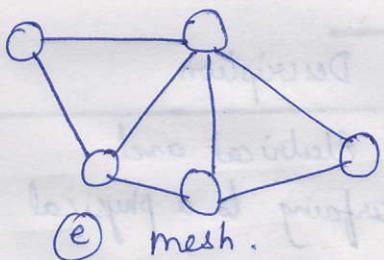
(b) Star



(c) hierarchy
(or tree)



(d) Ring



(e) mesh.

fig. LAN Topologies

Characteristics of Each Topologies are :-

1. Data bus:- Simplest of all LAN Topologies.
↳ This bus is reliable because of its passive nature but there may be a limitation on length of a bus.
↳ It is a broadcast system and hence a packet of data placed on bus is available to all devices.
2. Star:- This network is not widely used.
Data sent to central switch can be forwarded either in broadcast mode, or only to a specified node.
3. Hierarchy:- The system has many of characteristics of Star. Instead of one central switching node, many of nodes have to act as switches.

4. Ring:- This is most popular method.

The ring is typically an active transmission system,

The information placed on a ring network continues to circulate until a device removes it from the ring.

The information is broadcast in sense that it is available to all devices connected to the ring.

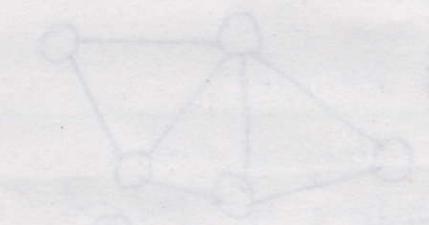
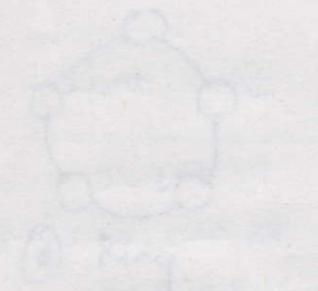
5. Mesh:- The mesh topology allows for random Interconnection between the various nodes.

A problem which can arise with the mesh is that there can be variable delay between the sending and receiving of message because the number of nodes through which the message has had to pass.

The ISO seven-layer model.

Layer	Description	Standards
1. physical	Defines the Electrical and mechanical Interfacing to a physical medium .	RS 232 - C RS 442 / 443 / 449 V. 24 / V. 28 X. 21, X. 26, X. 27.
2. Data link	Establishes Error free paths over physical channel , frame messages , Error detection and Correction. ↳ Ensure proper sequence of transmitted data	ANSI - ADCCP , IBM S D L C , BISYNC X. 25
3. Network	Addressee and route message . Sets up communication paths .	USA D O P - I P X 25, X 75
4. Transport	Provides end - to - end control of a communication session .	USA D O D - T C P IBM S N A DEC D N A

Layers	Description	Standards
5. Session	Establishes and Controls node system-dependent aspects	
6. Presentation	Allows encoded data transmitted via communication path to be presented in suitable formats for user manipulation	
7. Application (User)	Allows a user service to be supported. Eg 1. file transfers, DBMS etc.	FTP JMP FAM.



Star Topology

Characteristics of bus Topology :-

• Data flow format of all bus Topologies is broadcast but there may be some limitation on length of data frame.

• There is no dedicated connection between any two nodes so limitation on length of data frame.

• There is no broadcast limitation hence a greater amount of data placed on bus is available to all devices.

• Bus type network is not widely used in broadcast.

• Data sent to a bus can be forwarded to another bus only via a specified node.

• Bus type network has many advantages over star type network.

OPERATING SYSTEMS

OPERATING SYSTEMS: Introduction, Real-Time Multi-tasking OS, Scheduling Strategies, Priority Structures, Task Management, Scheduler and Real-time Clock Interrupt handles, Memory Management, Code Sharing, Resource Control, Task Co-operation and Communication, Mutual Exclusion, Data transfer, Liveness, Minimum OS Kernel, Examples.

Operating System is a System Software that manages Computer hardware and software resources and provides services for Computer programs.

It is an Interface between a Computer user and Computer hardware. OS is software which performs all basic functions like, File management, Memory management, Process management, Handling Input and Output, resource management etc.

Real Time Operating System (RTOS) is an OS Intended to Serve real time applications that process data used in Embedded systems.

Operating system were developed, to assist operators in running a batch processing Computer.

These systems are developed to support both

Real time systems and Non-real time systems.

General Purpose Operating System

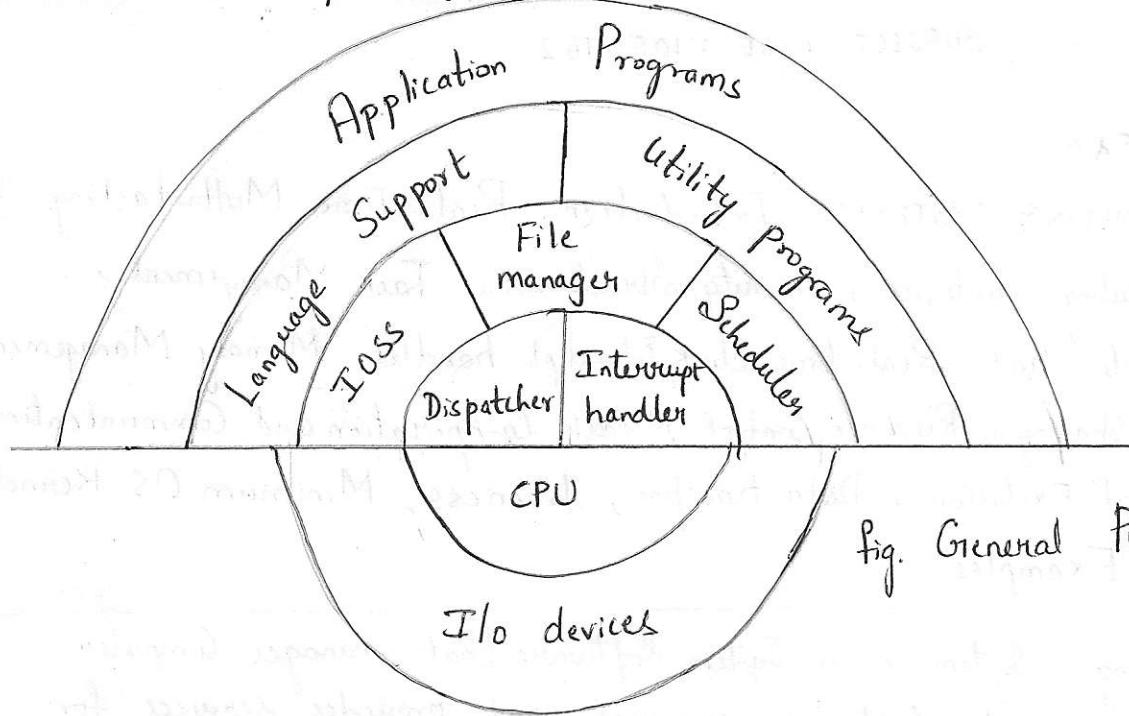


Fig. General Purpose OS

The operating system is used to access hardware and I/O devices. A General purpose Operating system will provide facilities that are not required in a particular operations / applications. So, during the Installation of OS certain features can be Selected or Omitted.

This General structure OS has large no of system software which has following functions.

Dispatcher: - It performs Context Switching ie, save the status of task that is currently using CPU and restores the status of task that is being allocated with CPU Time.

Scheduler: - It Schedules an allocation of CPU time to Each task.

Interrupt handler: - It handles both hardware and Software Interrupts.

I/O SubSystem (IBSS): - It Enables application programmer to perform Input/Output by means of System Calls.

File Manager: - Enables file operations such as file open, file close, file delete, file modify etc.

Utilities: - Supports utility programs such as linkers, loaders, assemblers, debuggers, compilers etc.

Minimal Operating System

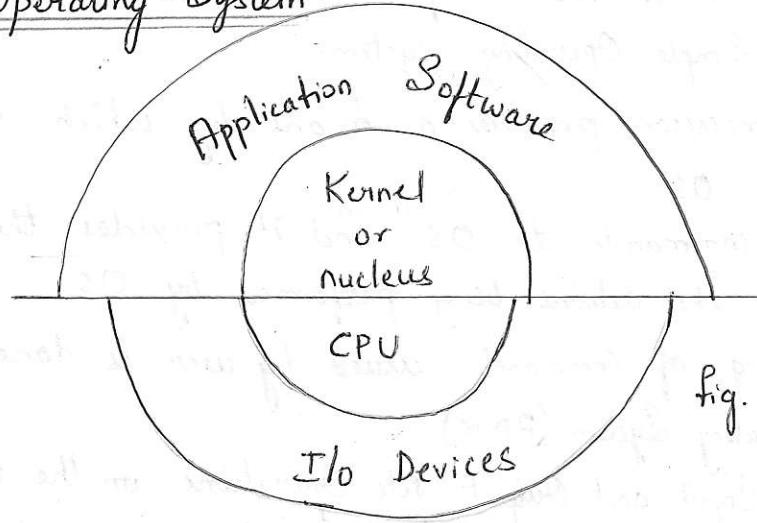


fig. Minimal OS

In, Minimal OS, it provides only minimal Kernel / nucleus, All additional features can be added by application programmer by writing in high level languages.
These types of Kernel are used in Small Embedded Systems.

General Structure of Simple OS

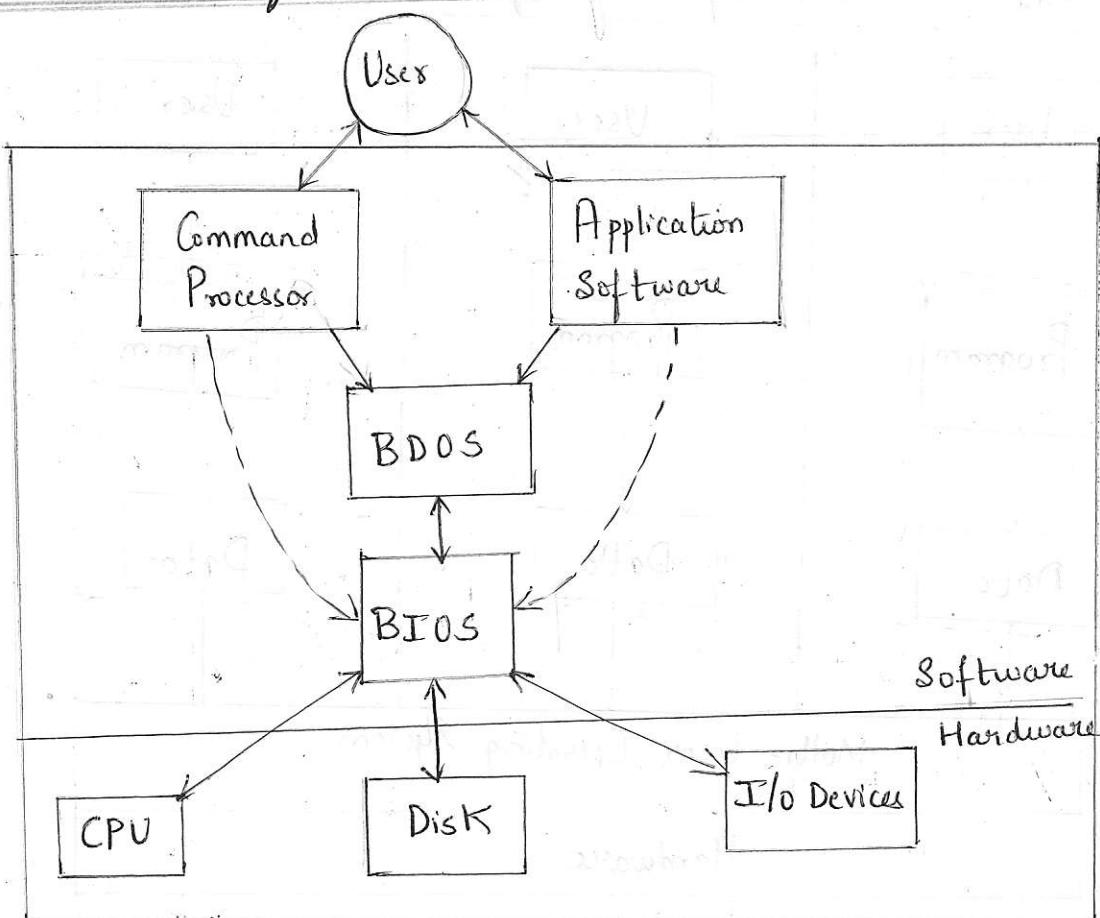


fig. General Structure of a Simple OS

The relationship between the Computer hardware and the User is understood by Simple Operating system.

The Command processor provides a means by which user can communicate with OS

The User issues commands to OS and it provides the user with information about the actions being performed by OS.

The actual processing of Commands issued by user is done by Basic Disk Operating System (BDOS).

BDOS - handles, Input and Output file Operations on the disks.

The Application Software communicate with hardware of system through System Calls which are processed by BDOS.

The Basic Input Output System has various device drivers which manipulate physical devices and vary from one system to another. as it has to directly operate with the hardware.

Real Time Multi User Operating System

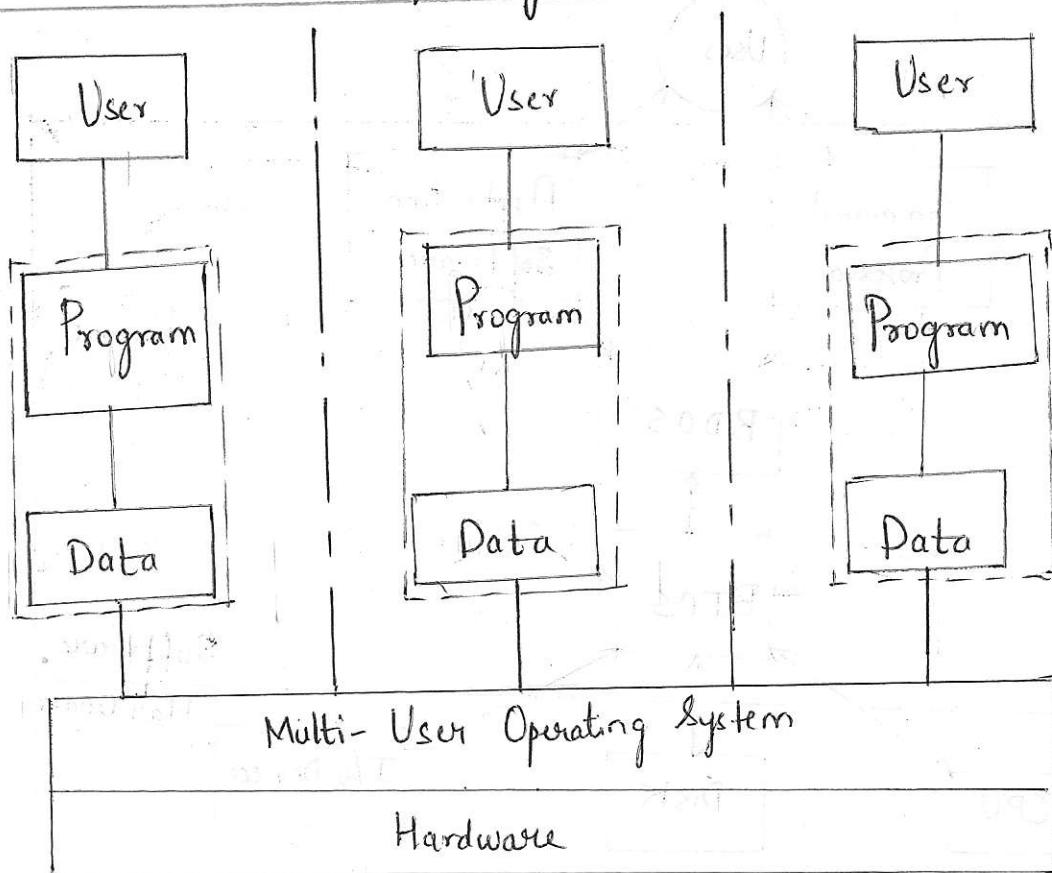


fig. Multi-User OS

The functions of Multi-User OS System are:-

- The OS Ensures that Each User can run a single program considering it has a complete whole system for their program.
- At any point of given instance, it is not possible to predict which user will have the use of CPU.
- The Operating System ensures that one user program cannot interfere with the operations of another user program.
- The Primary concern of Multi-user OS is to prevent one program corrupting another.
- Each user runs in its own protected Environment

Real Time Multitasking Operating Systems.

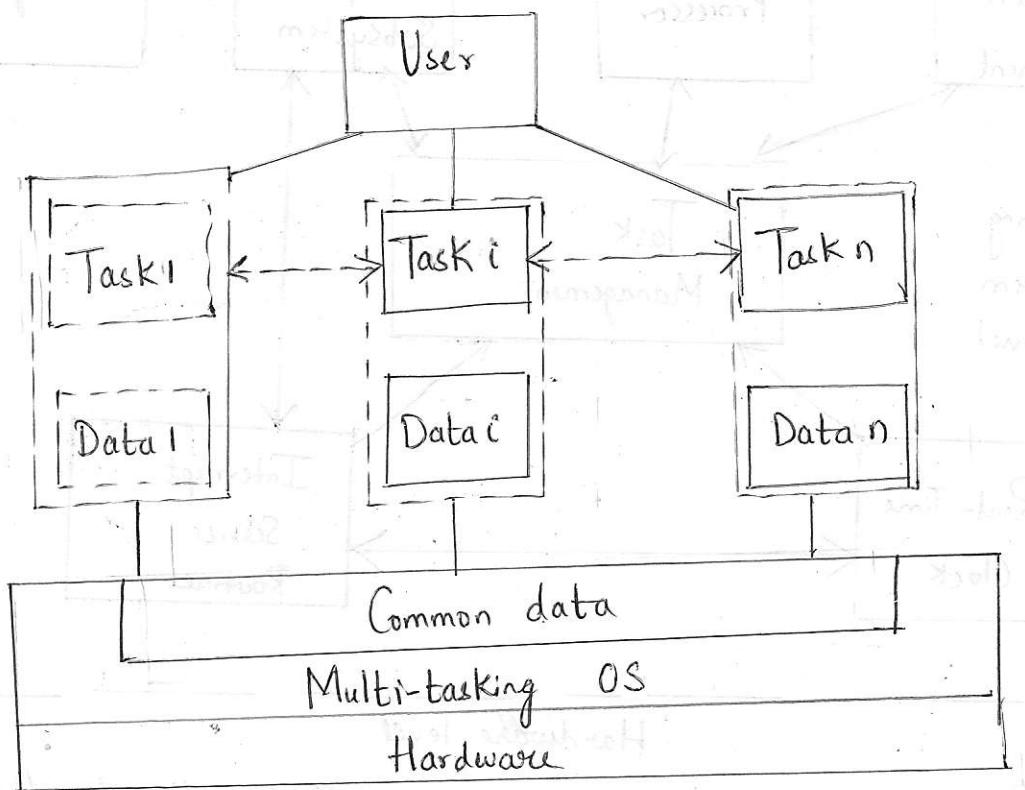
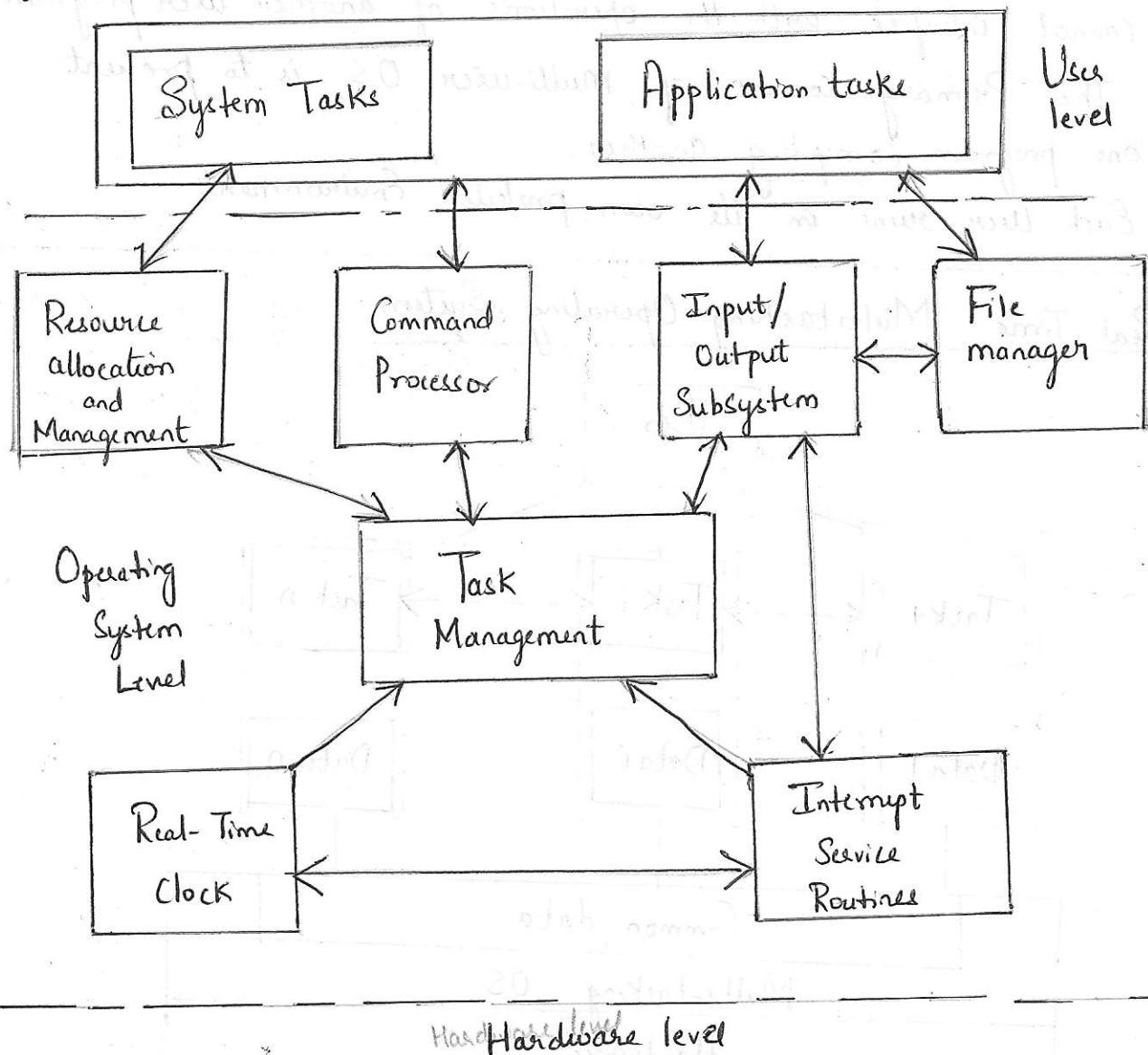


fig. RT Multitasking OS

- In a Multitasking OS, there will be one User and various tasks cooperate to serve the requirements of the User.

- The cooperation between tasks means it should communicate with each other and share common data.
- The task communication and data sharing will be regulated and hence protect data private to that task.

Typical Structure of a Real Time Operating System.



A fundamental requirement of an OS is to allocate the resources of the computer to perform various activities.

A Real time OS has to support the resource sharing and timing requirements of tasks.

The functions can be divided as follows:-

1. Task Management :- The allocation of memory and Processor time to task.
2. Memory Management :- Control of Memory allocation
3. Resource Control :- Control of all shared resources
4. Intertask Communication and Synchronization :- Support mechanism to provide safe communication between tasks and to enable tasks to synchronise their activities.

The overall control of the System is provided by Task Management module, responsible for allocating the use of CPU. This is also referred to as Monitors or Executive Control program.

At the User level, System tasks are shown along with Application tasks, where operations are performed by OS and Utility programs run in memory space allocated to User called as "Working Memory".

Scheduling Strategies:-

There are two basic strategies:-

1. Cyclic
2. Preemptive

Cyclic:-

The cyclic Scheduling allocates the CPU to a task in turn. The task uses the CPU for as long as it wishes. When it no longer requires it, the Scheduler allocates it to next task in the list.

It is an effective strategy for small Embedded systems for which the Execution time for each task run are carefully calculated.

In general this approach is too restrictive since it requires that the task units have similar execution times.

Preemptive

The term preemptive is the possibility that a task will be interrupted - before it has completed a particular task.

It has to perform Context switching i.e., to save the details of the task, since at some time later it will be allocated CPU time and will continue from the exact point at which it was interrupted.

* The simplest form of preemptive Scheduling is to use a time slicing approach also called as Round-robin method.

In this strategy, each task is allocated a fixed amount of CPU time - a specified number of ticks of the clock and at the end of this time it is stopped and next task in the list is made to run.

- * All tasks are given Equal Importance by allocating an Equal share of CPU time.
- * Static priority system :- Task priorities may be fixed
- * Dynamic priority system :- Task priorities may be changed during System Execution which Increases the flexibility of the system.
- * Majority of Existing RTOS uses a priority Scheduling mechanism. Tasks are allocated a priority level and at the end of Time slice, the task with highest priority of those ready to run are chosen and is given control of the CPU.

Priority Structures in an RTOS.

The designer has to assign priorities to the tasks in the system in RTS.

The priority will depend on how quickly a task will have to respond to a particular event.

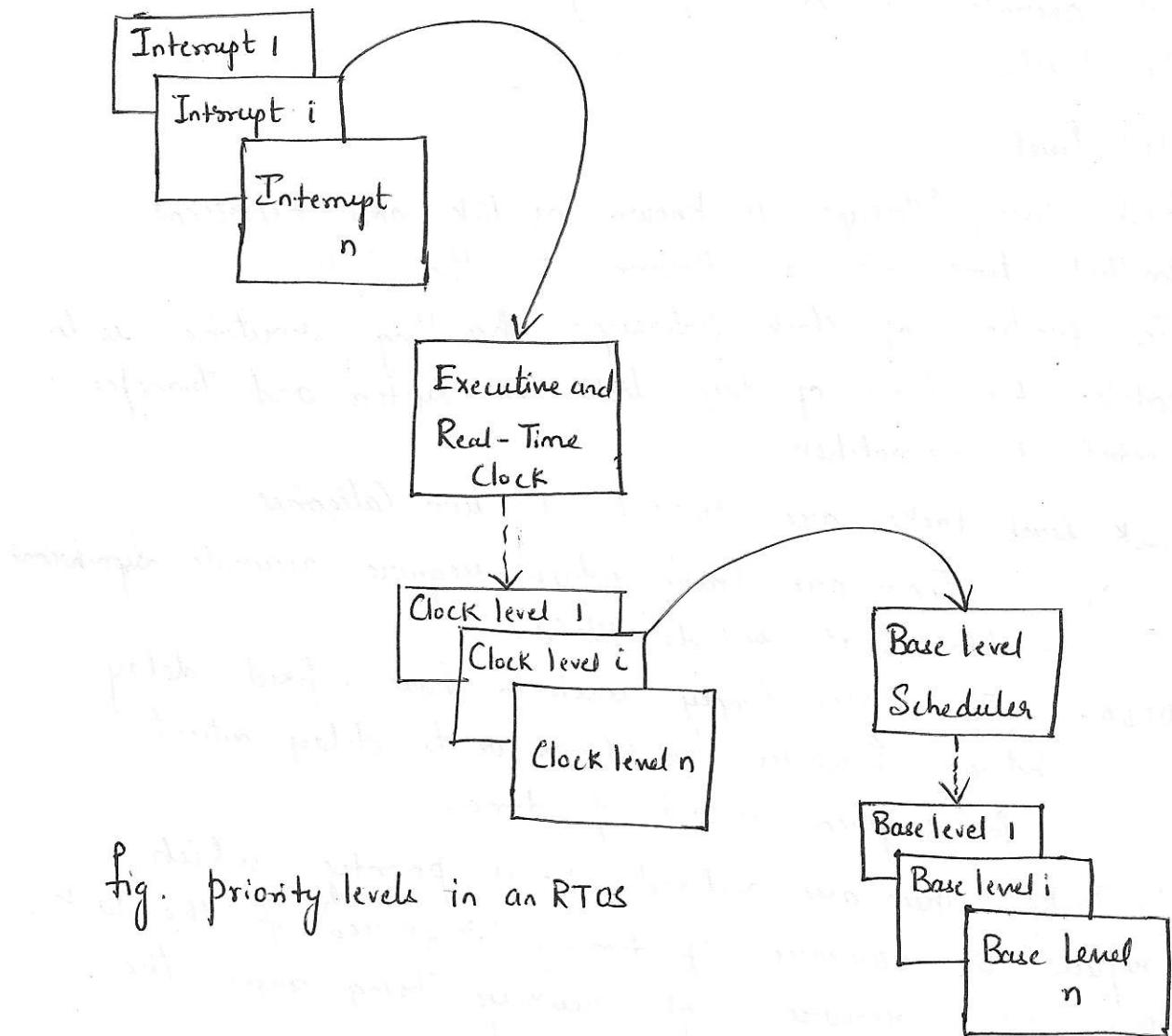


Fig. Priority levels in an RTOS

There are 3 priority levels:-

1. Interrupt Level
2. Clock Level
 - ↳ cyclic tasks
 - ↳ Delay tasks
3. Base Level.

1. Interrupt Level :-

At this level, several routines for the tasks and devices which requires very fast response - measured in milliseconds - Ex: Real Time Clock.

- The Interrupt handling routine does processing, preserves the necessary information and pass it further routines which operate at lower priority level, Either clock or Base level.

2. Clock Level :-

- Each clock Interrupt is known as tick and represent smallest time interval known to the system.
- The function of clock Interrupt handling routine is to update the time of day clock in system and transfer control to dispatcher.

Clock level tasks are divided into two categories :-

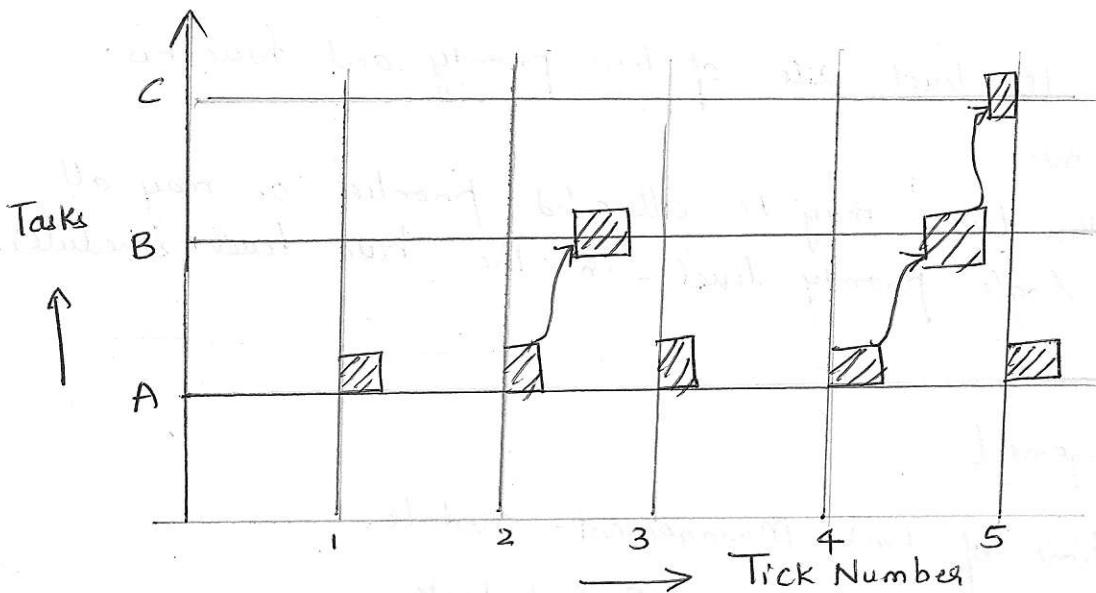
1. CYCLIC:- These are tasks which require accurate synchronization with outside world.

2. DELAY:- These tasks simply wish to have a fixed delay between successive repetitions or to delay activities for a given period of time.

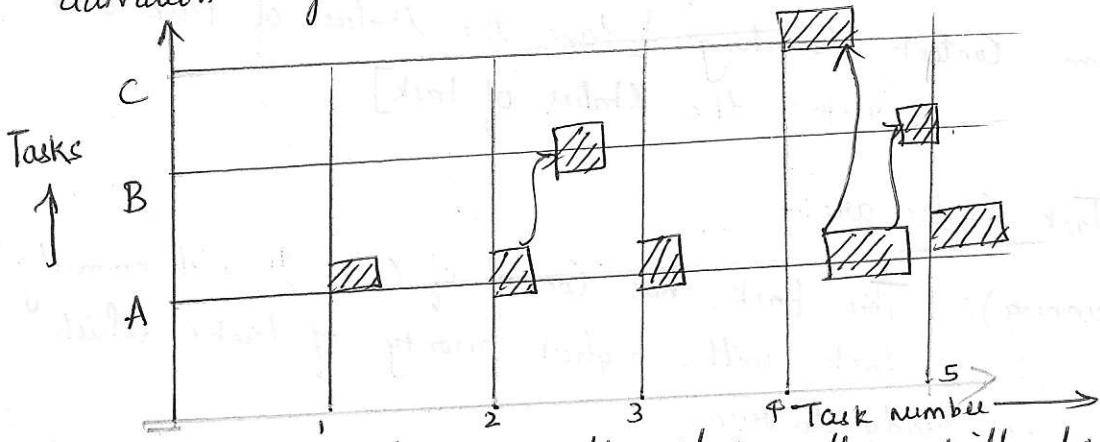
Cyclic Tasks:- These are ordered in a priority which reflects the accuracy of timing required for the task. those which require high accuracy being given the highest priority.

Example:- Three tasks A, B and C are required to run at some, 40ms and 80ms intervals (corresponding to 1 tick, 2 ticks, 4 ticks)

If the task priority order is set as A, B, C with A as highest priority then processing will proceed as shown below.



If the priority order is rearranged to C, A, B, then activation diagram is shown below.



At every fourth tick of the clock, there will be delay in the timing of tasks A and B.

Delay Task :-

When task request a delay its state is changed from runnable to suspended and remains suspended until the delay period has elapsed.

One method of implementing the delay function is to use a queue of task descriptors named DELAYED queue.

This queue is an ordered list of task descriptors, the task at front of queue being that whose next running time is nearest to current time.

Base level:-

Tasks at this level are of low priority and have no deadline to meet.

Tasks at this level may be allocated priorities or may all run at a single priority level - in the base level scheduler.

Task Management

Basic functions of task Management Module.

1. To Keep a record of state of each task
2. To Schedule an allocation of CPU time to each task.
3. To perform Context switching. [Save the status of task and restore the status of task].

Various Task States are:-

1. Active (running):- This task has control of CPU. It will normally be the task with highest priority of tasks which are ready to run.
2. Ready (runnable, ON):- There may be several tasks in this state. The task which are resource required to run task must be available to be placed in ready state.
3. Suspended (Waiting, locked out, delayed):- The execution of tasks placed in this state has been suspended because the task requires some resources which is not available.
4. Existent (dormant, off):- The OS is aware of existence of this task, but the task has not been allocated a priority and has not been made runnable.

5. Non-Existent (terminated):- The OS has not ~~not~~ been made aware of Existence of this task, and is resident in the memory.

Typical Task State diagram.

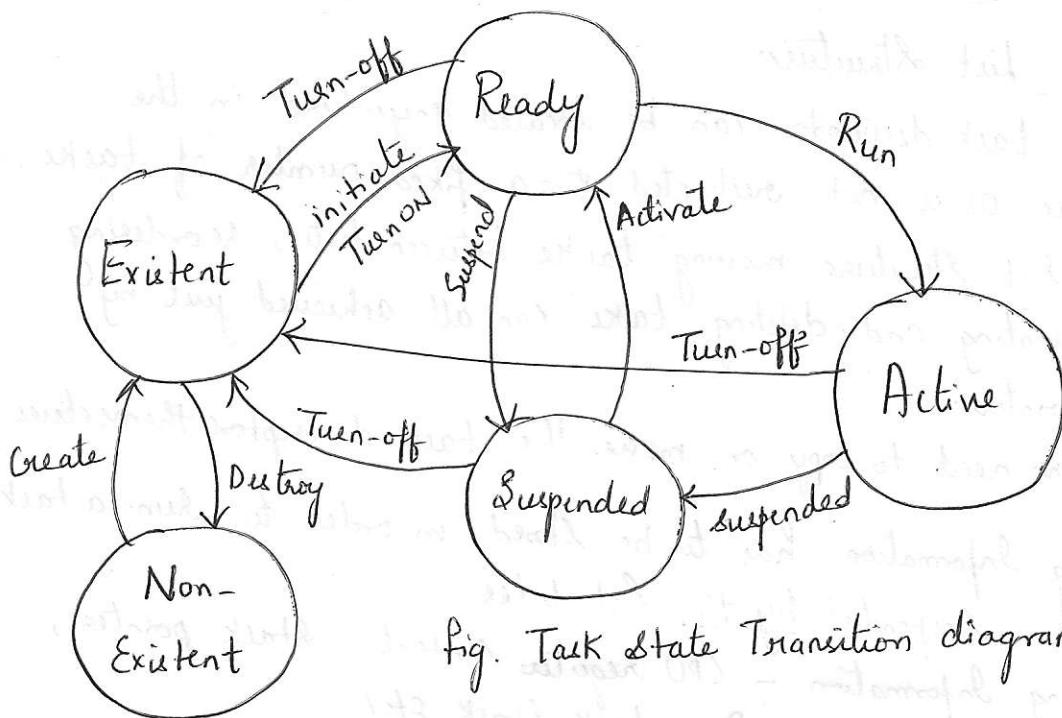


fig. Task state Transition diagram

The status of various tasks may be changed by actions within the OS.

The transition paths between different states vary from OS to OS.

Typical application commands are used for various transitions.

e.g. Turn ON [ID] - Transfer a task from Existent to ready State.

Task Descriptor:-

Information about status of each task is held in a block of memory by RTOS.

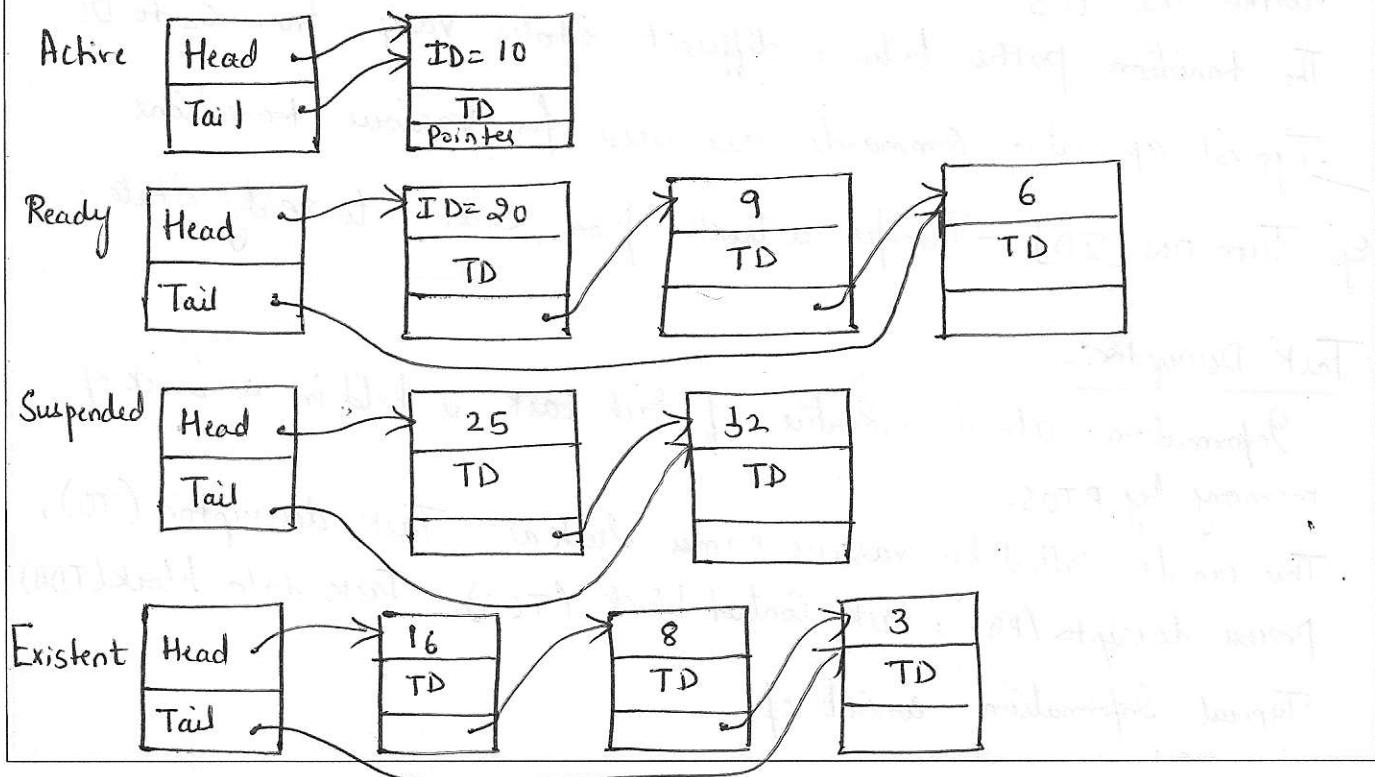
This can be called by various names such as:- Task descriptor (TD), process descriptor (PD), task control block (TCB), Task data block (TDB).

Typical Information consist of:-

- task identification (ID)
- task priority (P)
- Current State of task
- area to Store Information
- pointer to next task in a list.

Advantages of List Structure

- ↳ The actual task descriptor can be located anywhere in the memory hence OS is not restricted to a fixed number of tasks.
 - ↳ With the list structure moving tasks between lists, reordering the lists, creating and deleting tasks can all be achieved just by changing pointers.
 - ↳ There is no need to copy or move the task descriptors themselves.
- The following information has to be stored in order to run a task that has been suspended by the Scheduler.
1. House Keeping Information - CPU register contents, stack pointer, PC, task stack etc.
 2. Task Data - Task general work area.



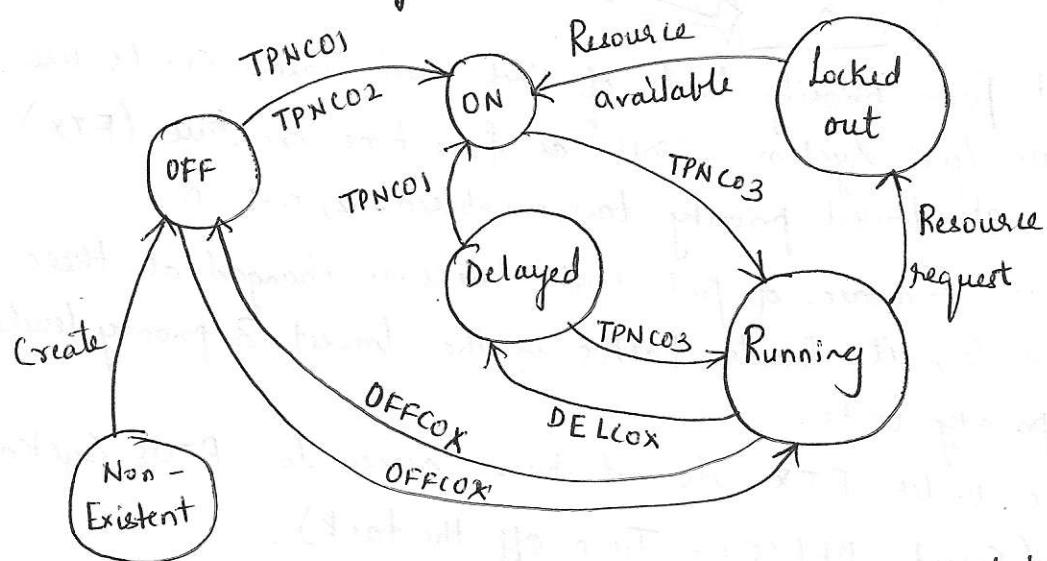
System Commands which change Status of Task.

The range of System commands affecting Task status varies with OS.

RTOS task state transition Commands

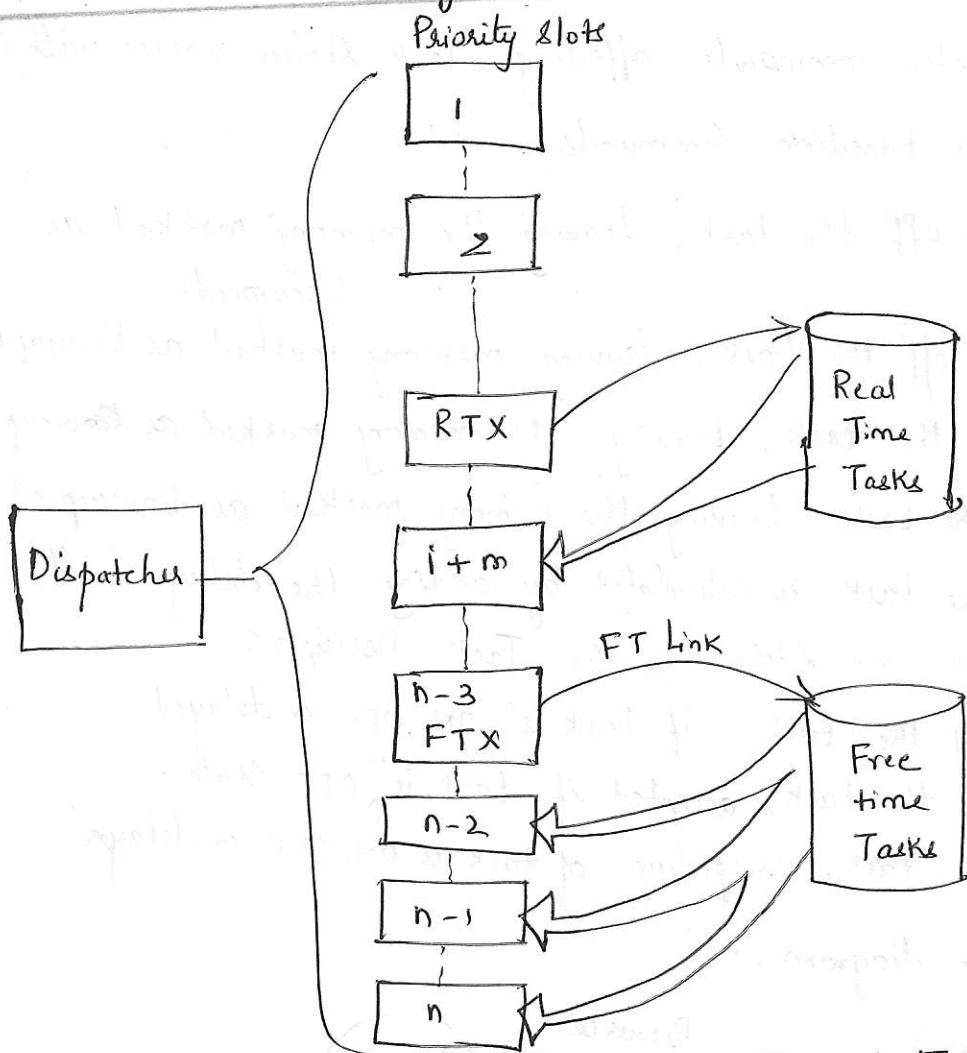
1. OFFC01 - Turn off the task, leaving the memory marked as Occupied
2. OFFC02 - Turn off the task, leaving memory marked as Unoccupied
3. DELC01 - Delay the task, leaving the memory marked as Occupied
4. DELC02 - Delay the task, leaving the memory marked as Unoccupied
5. DELC03 - Delay the task, is calculated by adding the delay to the value of time stored in the Task Descriptors.
6. TPNC01 - Turn ON the task, if task is ON, OFF or delayed
7. TPNC02 - Turn ON the task, accepted if task is in OFF state.
8. TPNC03 - Run the task, irrespective of task is ON, OFF or delayed.

RTOS Task State diagram.



- System distinguishes between tasks which are suspended awaiting passage of time - these tasks are marked as Delayed.
- tasks which are waiting for an event / system resource - marked as Locked Out.

RTOS Task Structure diagram:-

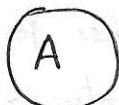


- The lowest four priority levels of clock level Tasks can be used to create a base level system — called as free time Executive (FTX)
- FTX runs at lowest priority task numbers n-2, n-1, n.
- Dispatcher is unaware of fact that tasks are changed at these 3 priority levels, it treats tasks in the lowest 3 priority levels as low priority tasks.
- Tasks run under FTX do not have access to RTOS system commands (Except, DFFC01 - Turn off the task).

RTOS Search for Work by the dispatcher

Entry A:

Real time clock
Interrupt



Save Status of
active task

Priority level
 $= -1$

Inrement
Priority level

Priority
level > max

Task
ON

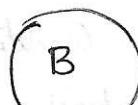
Task in
memory

Restore status
of task

Exit
to task

Entry B:

Task system
Command requests
(Eg. OFFC01, DELC01)



Save status of
active tasks

Priority levels
Current Priority

Request
Transfer of
task to memory

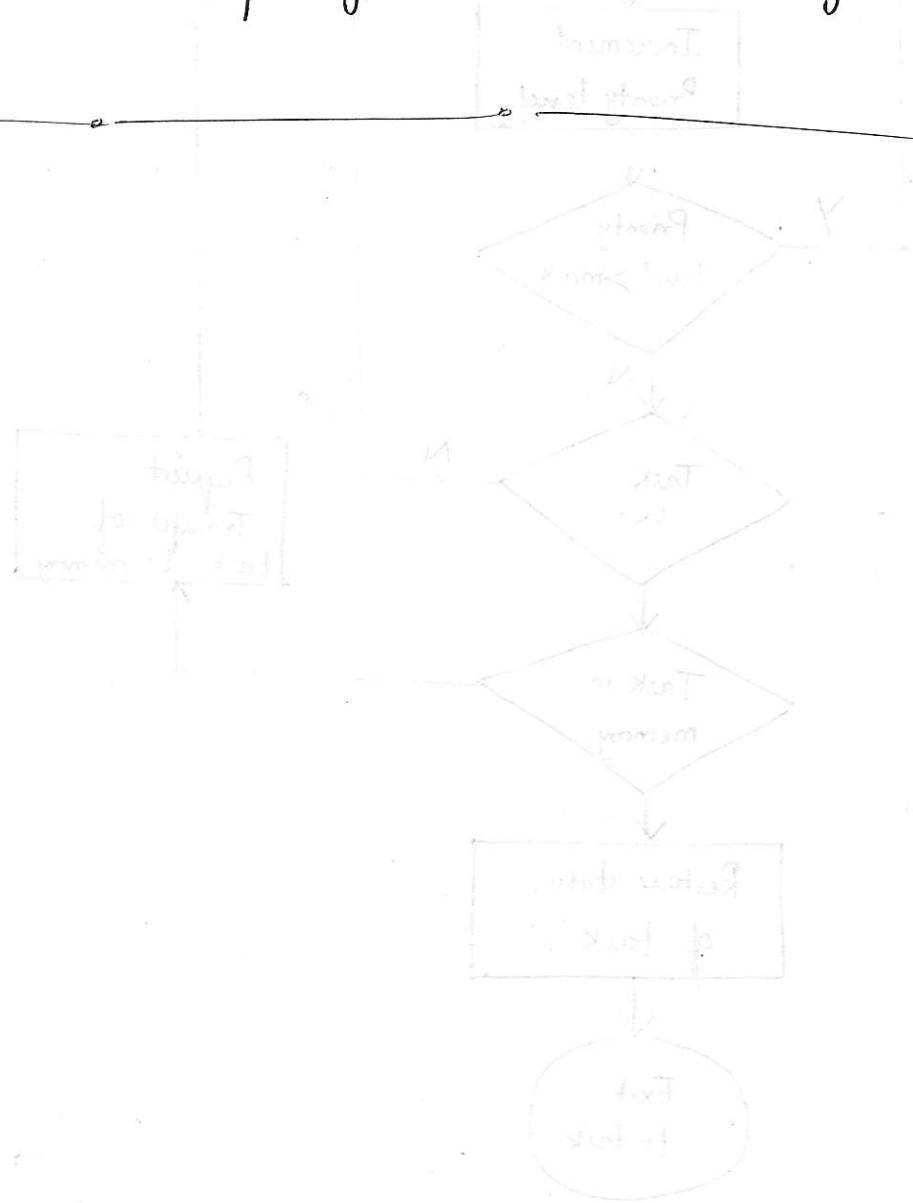
N

N

- The Dispatcher / Scheduler has 2 Entry Conditions.

- RTC Interrupt and any interrupt which Signals the Completion of an I/O request.

2. a task suspension due to a task delaying, completing or requesting an I/O transfer.
- In response to first condition, Scheduler searches for work starting with highest priority task and check each status in priority order. if tasks with a high repetition rate are given a high priority they will be treated as Clock level tasks, i.e. they will run first during each system clock.
 - In response to Second Condition, a search for work is started at the task with next lowest priority to the task which has just been running,



if task is not ready & not finished, add it to priority queue & do a task transfer for task priority 2 & 3

MEMORY MANAGEMENT

With permanently resident software the Memory can be divided as shown in figure below.

2 types:-

- ① Non-partitioned Memory
- ② Partitioned Memory.

→ The User space is treated as One unit and Software is linked and loaded as a single program into the User area.

The Information about the various tasks is Conveyed to the Operating System by means of a Create task Statement.

(Create (TaskID, priority, StartAddress, Workspace)).

The Exact form of statement will depend on the Interface between the high-level language and Operating System.

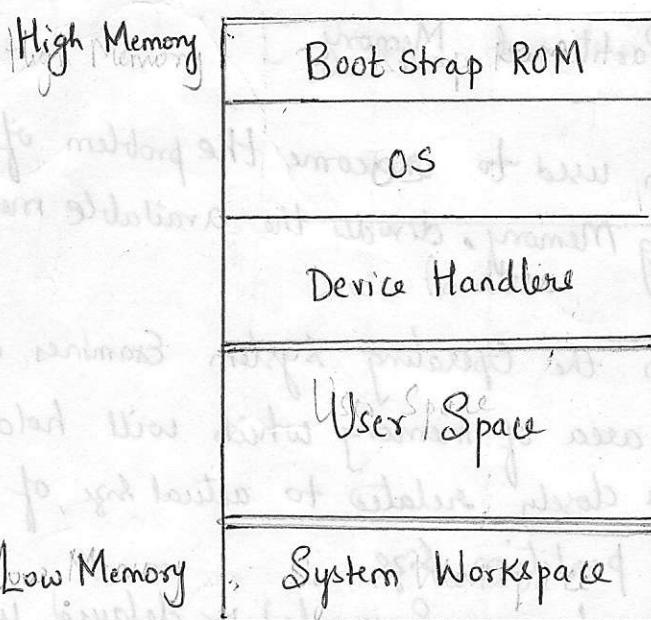


fig. Non Partitioned Memory

→ Partitioned Memory:-

The available memory is divided into predetermined Segments and tasks are loaded individually into the various Segments.

* Divided (par

Operating System and it was frequently extended to allow several tasks to share one partition.

- * The tasks were kept on backing store and loaded ~~processes~~ into the appropriate partition when required
- * The difficulty with this method is, in choosing the best mix of partition sizes.

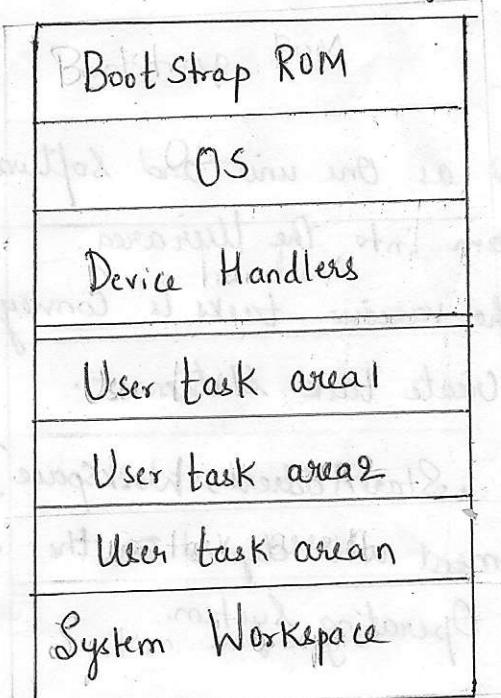


fig. Partitioned Memory

- * A number of methods have been used to overcome the problem of One method is using a floating Memory, divides the available memory into small blocks.
- * When the task is required to run the Operating System examines a map of memory and finds a contiguous area of memory which will hold the task.
- * The area occupied by a task is closely related to actual size of task and not to some predetermined fixed partition size.
- * A task which for some reason becomes suspended or delayed will have the memory area it occupies marked in the memory map as occupied, but available; hence if another task becomes ready then the suspended task can be returned to backing store and ready task loaded into its area.
- * In this type of system information must be held in Task descriptors to indicate if task can be swapped, since for ex - a control task which has to run forever

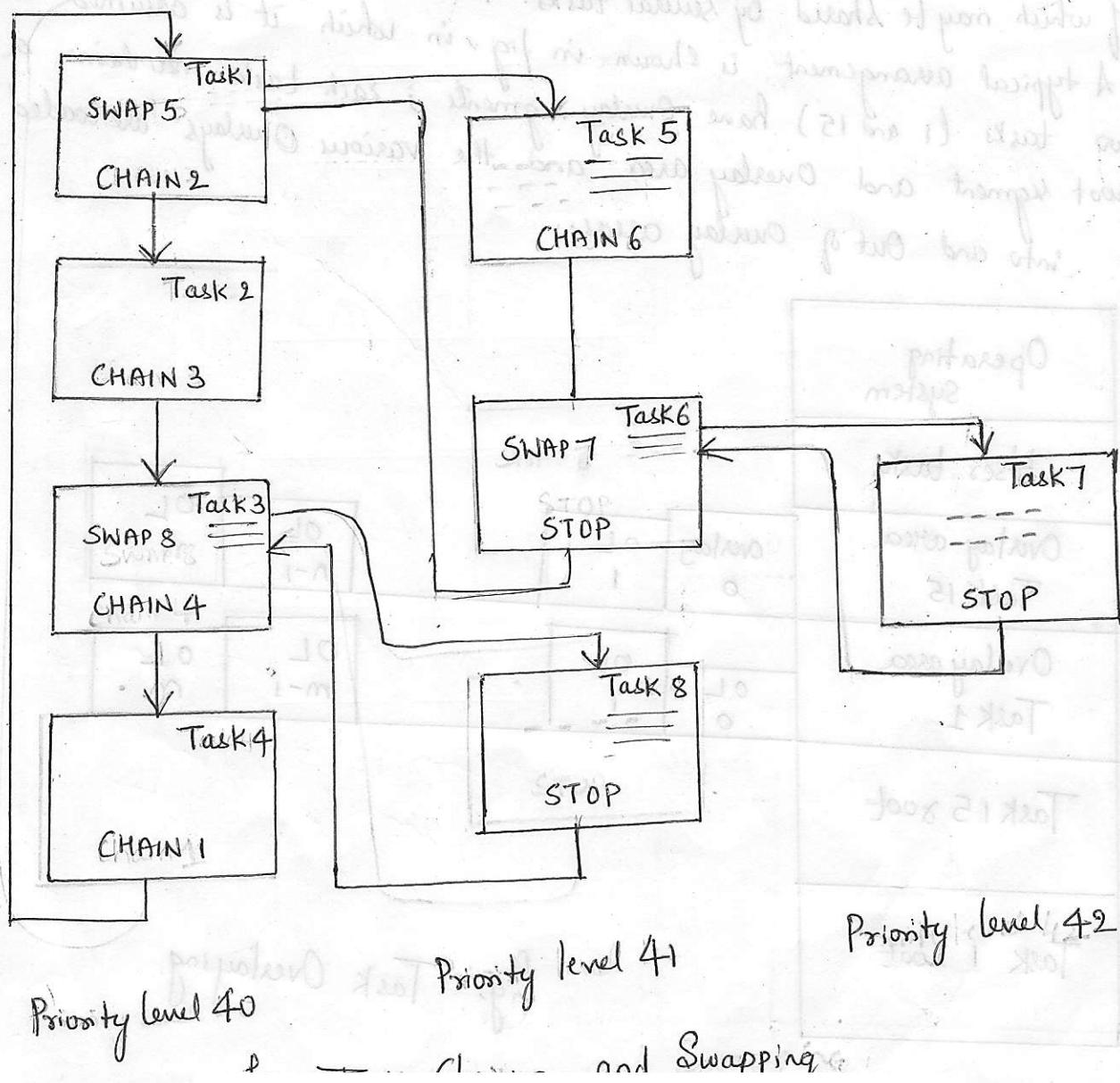
The Dynamic allocation of memory allow the tasks themselves to initiate program segment transfers, Either by Chaining or by Overlaying

Task Chaining:-

In Chaining the task is divided into several segments which run sequentially.

On completion of one segment, the next segment is loaded from memory into the area occupied by previous segment; any data required to be passed is held either on the disk or in a common area of memory.

Task Swapping: It involves one task invoking another tasks the first task is transferred to backing store and second task brought into memory and made available to run.



The procedure for Task chaining and swapping is explained by the Example.

- * Task 1 invokes task 5 by swapping it into priority level 41 and then task 5 chains task 6 into level 41. Task 6 swaps task 7 into level 42.
- * When task 7 terminates the operating system returns control to task 6 and when it terminates control is returned to task 1. It should be noted that task 1 remains suspended until task 6 terminates and similarly task 6 is suspended until task 7 terminates.

(*)

Task Overlaying:-

The difference between task chaining and overlaying is that in overlaying a part of the task, the root task, remains in memory and the various segments are brought into an overlay area of memory.

* In multitasking there may be several different overlay areas each of which may be shared by several tasks.

* A typical arrangement is shown in fig., in which it is assumed that two tasks (1 and 15) have overlay segments; each task maintains a root segment and overlay area and the various overlays are loaded into and out of overlay areas.

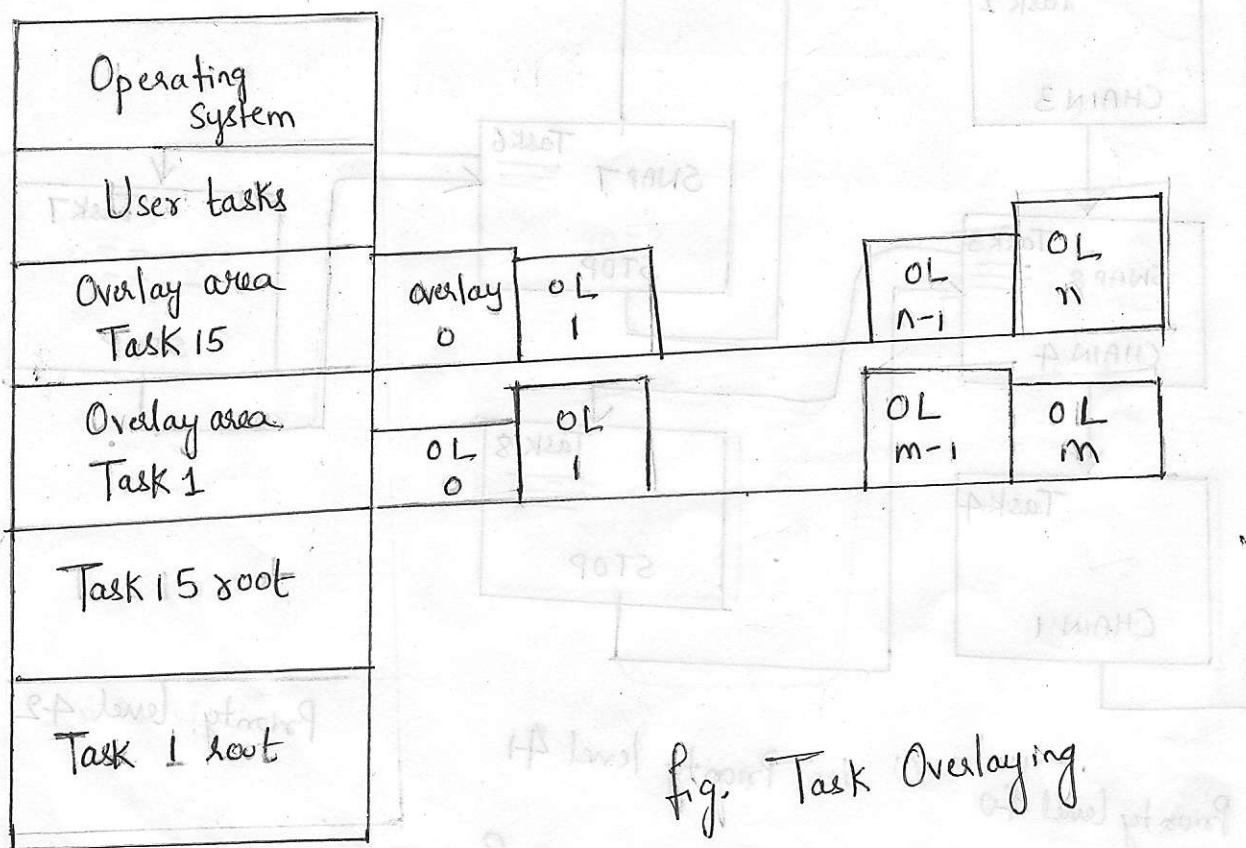


fig. Task Overlaying.

Code Sharing

In many applications the same actions have to be carried out in several different tasks.

In a multitasking system each task must have its own copy of subroutine or some mechanism must be provided to prevent one task interfering with the use of the code by another task.

The problem which can arise are illustrated in figure.

Task A

~~CALLS~~

S-Data for A

Reschedule

Task B

~~CALLS~~

Subroutine

overwrites A's data

Subroutine Contained

data relevant to B.

Reschedule

Reschedule

Two tasks share the same subroutine S. If task A is using the subroutine but before it finishes some event occurs which causes a rescheduling of tasks and task B runs and uses the subroutine, then when return is made to task A, although it begins the subroutine S again at correct place, the values of locally held data will have been changed and will reflect the information processed within the subroutine by task B.

2 Methods can be used to overcome this problem:

- Separately reusable code
- Reentrant code.

Seriously Reusable Code

- Some form of locking mechanism is placed at the beginning of the routine such that if any task is already using the routine the calling task will not be allowed entry until the task which is using the routine unlocks it.
- The use of a lock mechanism to protect a subroutine is an example of the need for mechanism to support mutual Exclusion when Constructing an OS.

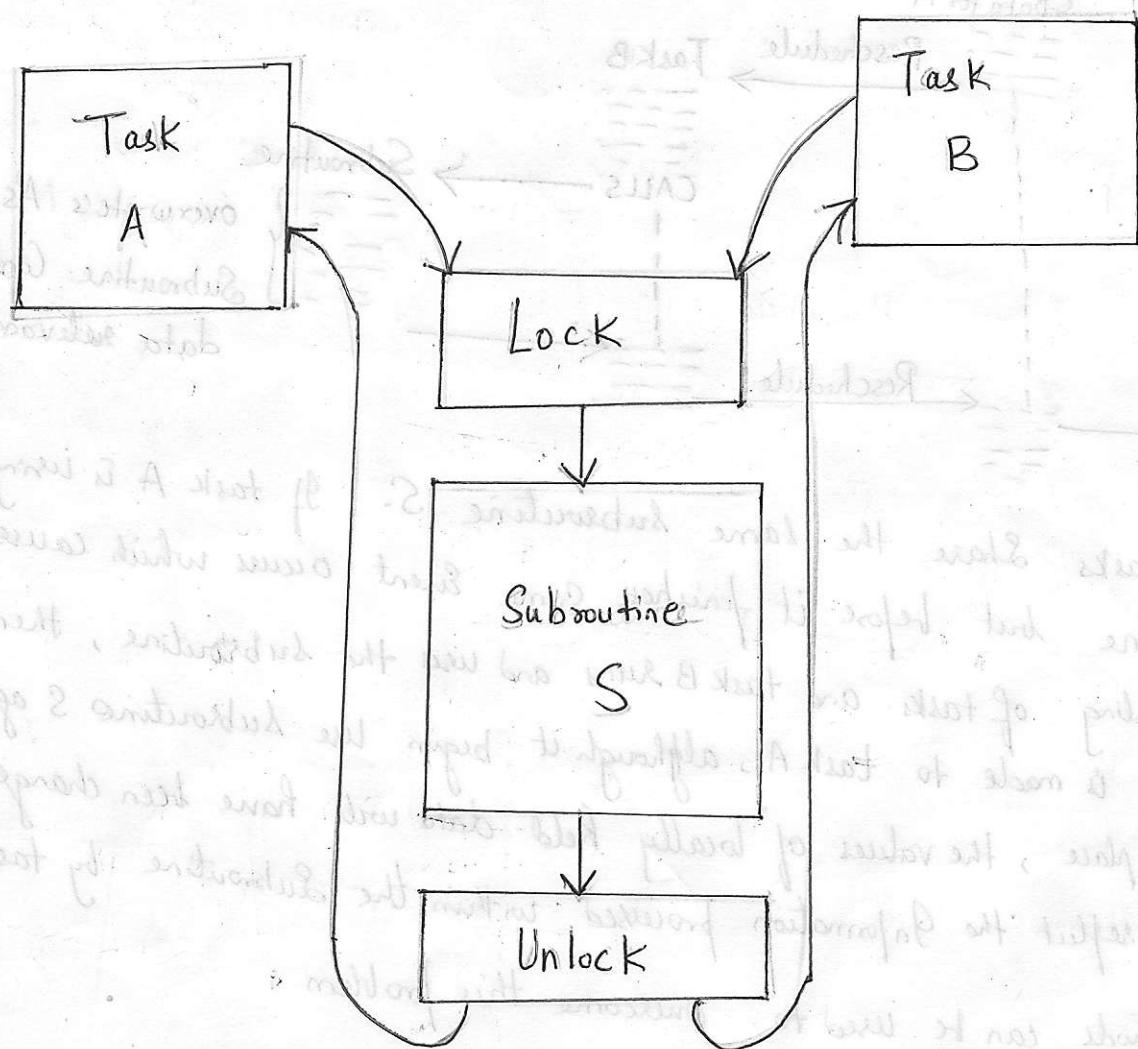


fig. Seriously Reusable Code

Reentrant Code

If the Subroutine can be coded such that it does not hold anything if any data, that is, it is purely code - any intermediate results are stored in the calling task or in a stack associated with the task - then the subroutine is said to be re-entrant.

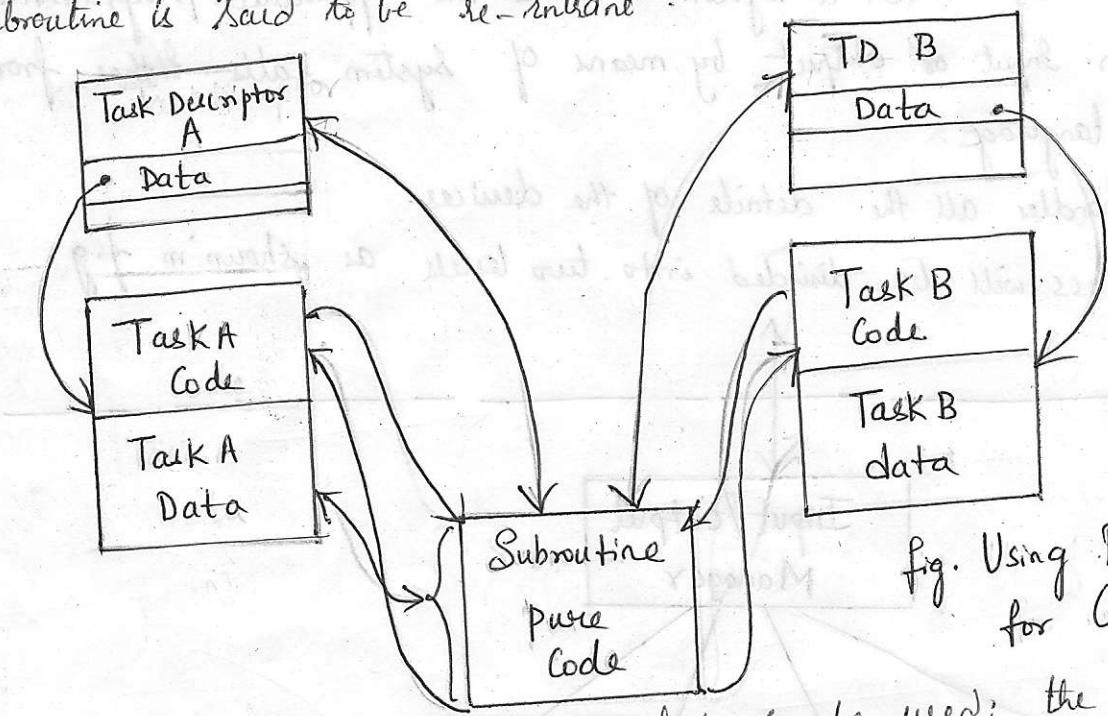


Fig. Using Reentrant Code
for Code Sharing.

Figure shows an arrangement which can be used: the task descriptor for each contains a pointer to a data area - usually a stack area - which is used for the storage of all information relevant to that task when using the subroutine.

All accesses to data by the subroutine will be through the stack and hence it will automatically manipulate the correct data.

Reentrant routines can be shared by several tasks since they contain no data relevant to a particular task and hence can be stopped and restarted at a different point in routine without any loss of information.

Resource Control:- An Example of an Input/Output Subsystem (IOSS)

- * The availability of a well designed and implemented Input/Output Subsystem (IOSS) in an Operating System is essential for efficient programming.
- * The presence of such a system enables the application programmes to perform input or output by means of system calls written from a high-level language.
- * The IOSS handles all the details of the devices.

A typical IOSS will be divided into two levels as shown in fig.

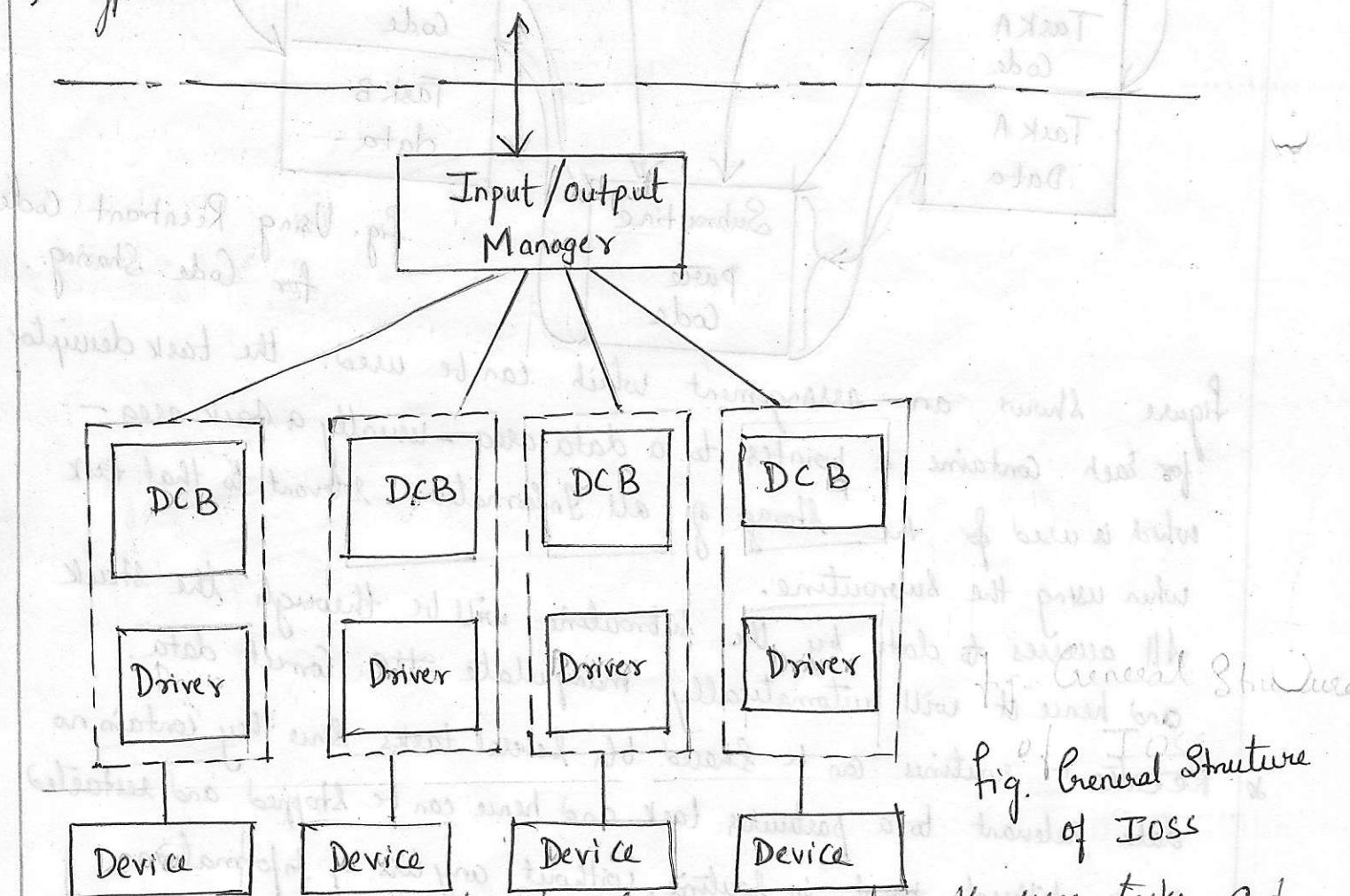


fig. General Structure
of IOSS

- * The IO 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 user task will include:
 - * location of a buffer area
 - * the amount of data to be transferred
 - * Type of data (binary or ASCII)

* The actual Transfer of data between the user task and device will be carried out by the device driver and this segment of code will make use of other information stored in the DCB.

The Detailed arrangement of IOSS is shown below.

Here, a single driver may be shared b/w several devices; however, each device will require its own DCB.

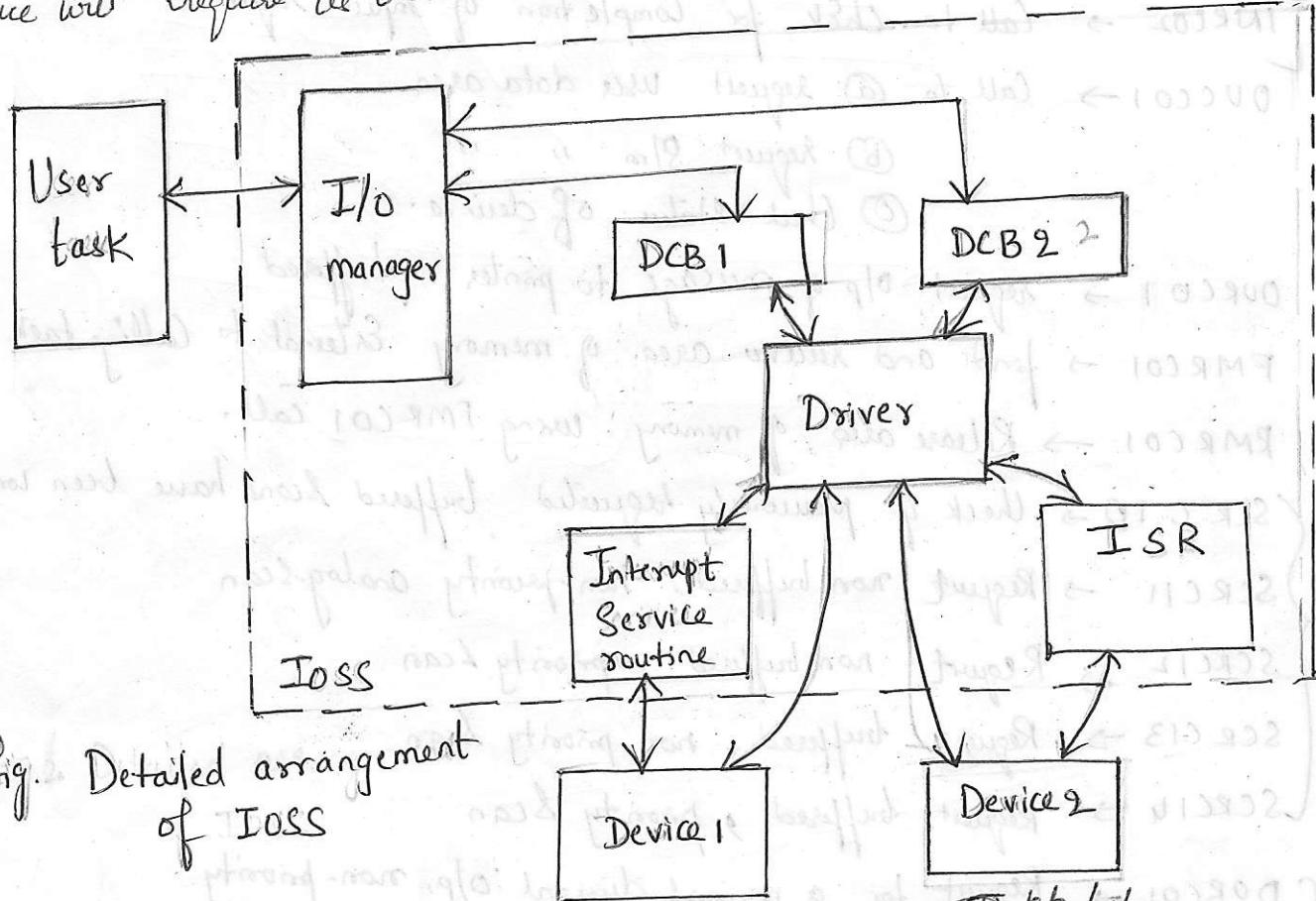


fig. Detailed arrangement of IOSS

Typical DCB will contain the Information shown in Table below.

Physical device name

Type of device

Device address

Interrupt address

Interrupt Service routine address

Device Status

Data area address

Bytes to be transferred

Current byte Count

Binary/A

Device - related Information

Data - related Information

Typical DOS System Commands for RTOS

{ DTRC01 → Disk transfer request - buffered

DTRC02 → Disk transfer request - non-buffered

DTRC03 → Call to check for completion of buffered request

{ INRC01 → Input request from keyboard device - buffered

INRC02 → Call to check for completion of input request

OVCC01 → Call to @ request user data area

(b) Request \$1m " "

(c) Check status of device.

OURC01 → Request O/p of message to printer - buffered

PMRC01 → find and reserve area of memory External to calling task.

RMRC01 → Release area of memory using PMRC01 call.

{ SCR C10 → Check if previously requested buffered scans have been completed

SCR C11 → Request non buffered, non priority analog scan

SCR C12 → Request non buffered, priority scan

SCR C13 → Request buffered, non priority scan

SCR C14 → Request buffered, priority scan

{ DORC01 → Request for a normal digital o/p, non-priority

DORC02 → Request for a normal digital o/p, priority

DORC03 → Request for a timed digital o/p, non priority

DORC04 → Request for a timed digital o/p, priority.

Task Cooperation and Communication

In real time systems, tasks are designed to fulfill a common purpose and hence they need to communicate with each other.

Some of the problems which arise have already been met by considering the Input/Output subsystem and they involve:

- Mutual Exclusion
- Synchronisation and
- Data transfers.

Mutual Exclusion:

A multi-tasking OS allows the sharing of resources between several concurrently active tasks.

The use of some resources is restricted to only one task at a time.

The problem can arise if 2 tasks share a data area and both tasks can write to the data area.

This is illustrated below -

Two SW modules, bottle-in-count and bottle-out-count, are used to count pulses issued from detectors which observe bottles entering/leaving a processing area. 2 modules run independently.

The 2 tasks operate on same variable bottle-count.

→ Module bottle-in-count increments the variable and
Module bottle-out-count decrements it.

The modules are programmed in a high level language & programs language statements are:-

bottle-count := bottle-count + 1; (bottle-in-count)

bottle-count := bottle-count - 1; (bottle-out-count)

At Assembly code level the high-level instructions become:

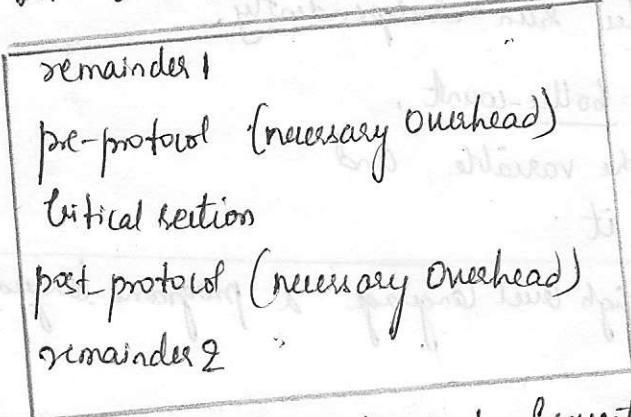
{ bottle-in-count }
LDA, (bottle-count)
ADD 1

{ bottle-out-count }
LDA, (bottle-count)
SUB 1

A-reg	bottle-in-count	Count	bottle-out-count	Arg.
?	LDA, (bottle-count)	10		
10	Context	10		
10	Change forced by OS	10	LDA, (bottle-count) SUB	10 9
10	ADD 1	9	LD (bottle-count), A	9
10	LD (bottle-count), A	9		9
11	11			9

- A reg is loaded with value 10. If OS now reschedules and bottle-out-count runs it will also pick up the value 10, subtract one from it and stores 9 in bottle-count.
- When Execution of bottle-in-count resumes its Environment will be restored and A reg will contain the value 10, one will be added and value 11 stored in bottle-count. Thus the final value of bottle-count after adding one to it and subtracting one from it will be 11 instead of the correct value 10.

In abstract terms mutual Exclusion can be expressed in the form:



Remainder 1 & 2 → represent Sequential code that does not require access to a particular resource or to a common area of mem.

Critical section — is the part of code which must be protected from Interference from another task.

Preprotocol & post protocol — Called before and after the critical sections are

Critical regions

Most of the resources can be used by only one task at a time

and use of resource cannot be interrupted

Such resources are said to be serially reusable and they include certain peripherals \rightarrow shared memory and CPU.

While CPU protects itself against simultaneous use the code that interacts with other serially reusable resources cannot. Such code is called a Critical Region.

If 2 tasks enter the same critical region simultaneously a catastrophic error can occur.

Ex1. Consider 2 C programs, Task-A, Task-B, which are steady running in round robin S/m.

Task-B o/p's the message "I am task-B" and Task-A o/p's the message "I am task-A".

If Task-B interrupts Task-A in middle of priority

Result will be an incorrect o/p:- I am I am task-A task-B.

Ex1. for Mutual Exclusion is shown.

(* Mutual Exclusion problem - use of binary Semaphore*)

VAR

pointerAccess : Semaphore;

Procedure task;

begin

(* remainder 1*)

SemRelease (pointerAccess)

(* if pointer is not available task will be suspended at this point *)

(* pointer available - Critical Section *)

(* do output *)

Release (pointerAccess)

(* remainder

Semaphore.

- The most widely used form of primitive for the purpose of mutual exclusion is the binary semaphore.
- The Semaphore mechanism was first proposed by E.W. Dijkstra in 1968.
- The Binary Semaphore is a condition flag which records whether or not a resource is available.

If binary Semaphore $s = 1$, then the resource is available and task may proceed.

If $s = 0$, then the resource is unavailable and task must wait.

There are only 3 permissible operations on a semaphore.

- ① Initialise : (s : A Binary Semaphore, v : Integer); let semaphore s to value of v ($v \geq 0$ or 1).
- ② Secure (s): If $s = 1$, then set $s = 0$ and allow the task to proceed. Otherwise resume any task that is waiting on semaphore s .
- ③ Release (s): If there is no task waiting for semaphore s then $s = 1$. Otherwise resume any task that is waiting on semaphore s .

For Example, consider a task which wishes to access a printer.

The underlying operations which take place as several tasks attempt to access the same resource are shown.

- Step 1: Binary Semaphore, PrinterAccess, is initialised to the value 1.
 & the pointers to head & tail of Semaphore queue is set to null if created.
- Step 2: Task A performs a Secure (PrinterAccess) operation and Semaphore value is set to 0. Since there was no other task waiting, Task A is ~~allowed~~ allowed to use the resource. Some time later Task A suspends and Task B performs a Secure (PrinterAccess) in Step 3.

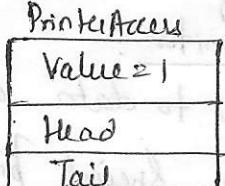
- * If Task C now performs a Secure (PrinterAccess), in Step 4, then the pointer in TD for Task B is filled in address of TD for Task C.

- * when Task B

.. Step 5, Task B

- It is removed from semaphore queue and obtain access to resources.
- At step 6., Task B performs the Release (Pointee Access) operation and hence Task C is allowed to run and at step 7 when Task C performs Release (Pointee Access) the value of semaphore is set to 1.

1. Initialise (Pointee Access, 1)

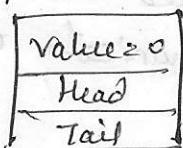


Mutual Exclusion using Binary Semaphore V.T.

Semaphore V.T.

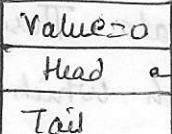
2. Task A enters Queue (Pointee Access)

Pointee Access



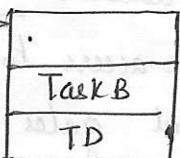
3. Task A suspends, Task B enters Queue (Pointee Access)

PointeeAccess



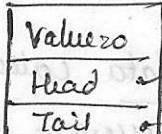
Task B is suspended & placed in

PointeeAccess Queue

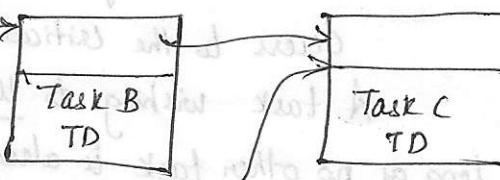


4. Task C runs, attempts Secure (Pointee Access)

PointeeAccess

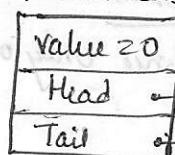


Task C is suspended and added to PointeeAccess queue

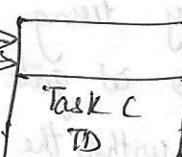


5. Task A runs, release (Pointee Access)

PointeeAccess

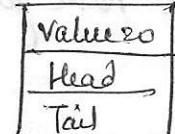


Task B is removed from PointeeAccess queue and placed in Ready queue



6. Task B runs, Release (Pointee Access)

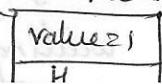
PointeeAccess



Task C is transferred from PointeeAccess queue to Ready queue

7. Task C runs, Release (Pointee Access)

PointeeAccess



Pointee is available to any task.

2. Data Transfer without Synchronisation

RTOS typically support 2 mechanism for the transfer or sharing of data b/w tasks:

- ① Pool

Ex: Tables of values / parameters which tasks periodically consult or update.

Write operation on a pool is destructive

Read operation on a pool is non-destructive.

The most reliable form of mutual exclusion for a pool is to embed the pool inside a monitor.

- ② Channel

→ Supports communication b/w producer and consumer of data. It provides direct communication link b/w tasks, normally on a One to One basis.

The communication is like a pipe down which successive collections of data-messages can pass.

One task is seen as the producer of Information and other as the consumer.

2 Basic Implementation mechanisms for a channel:

- ① Queue (linked list)

- ② Circular Buffer.

Queue:-

Advantage of queue:- the no of successive messages held in channel is not fixed.

→ The length of the queue can grow, the only limit being the amount of available memory.

Disadvantage of queue:- the length of queue increases the access time, that is the time to add and remove items from the queue, increase.

Circular Buffer:-

It uses a fixed amount of memory, the size is defined by the developer.

Condition A.

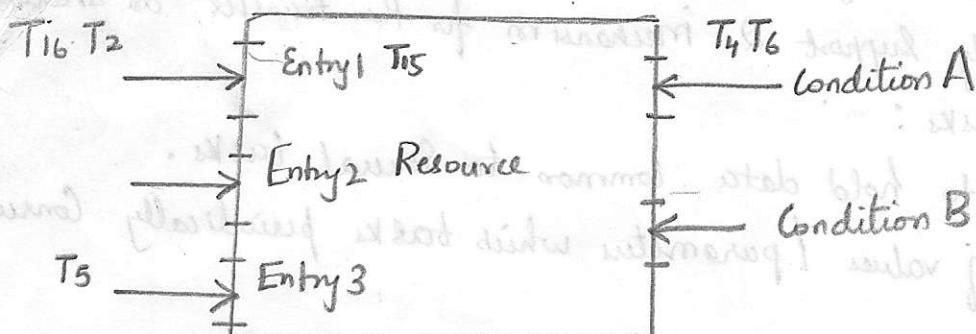


fig. A General Monitor

Intertask Communication

Issues of Synchronisation and Communication \rightarrow divided into 3 areas:-

- Synchronisation without data transfer
- Data transfer without Synchronisation
- Synchronisation with data transfer.

1. Task Synchronisation without Data transfer.

One task wishes to be able to inform another task that an event has occurred. No data needs to be exchanged by the tasks.

The mechanism that enables this to be done is so-called signal. A signal S is defined as a binary variable such that if $S=1$ then a signal has been sent but has not yet been received.

3 permissible operations:-

Initialize : (S : Signal; V : Integer) Set S to value of V (or)

Wait (S) : If $S=1$ then $S:=0$ else suspend the calling task and place it in

the condition queue S .

Send (S) : If condition queue S is empty then $S:=1$ else transfer the first task in the cond n queue to ready queue.

Signal is similar to semaphore, the difference b/w the two is not in the way of implemented but in way in which they are used.

A semaphore is used to secure and release a resource and as such

Call will both be made by one task.

A signal is used to synchronise the activities of 2 tasks.

Liveness :- V-X.

An Important property of a Multitasking real-time system is liveness.

A system is said to possess liveness if it is free from.

Deadlock

Deadlock and

Indefinite postponement.

Deadlock :- It is the condition under which the tasks requiring mutually

Exclusive access to a set of resources both Entry busy wait routines but neither can get out of the busy wait because they are waiting for each other. The CPU appears to be doing useful work & hence the term Deadlock.

Deadlock :- It is the condition in which a set of tasks are in state such that it is impossible for any of them to proceed. The CPU is free but there are no tasks that are ready to run.

Ex. Suppose task A has acquired Exclusive use of resource X & now requests resource Y, but b/n A acquiring X and requesting Y, task B has obtained Exclusive use of Y and has requested use of X. Neither task can proceed, since A is holding X & waiting for Y & B is holding Y and waiting for X.

Indefinite Postponement :-

It is the condition that occurs when a task is unable to gain access to a resource because some other task always gains access ahead of

Minimum Operating System Kernel.

Possible set of functions & primitives for RTOS :-

Functions:-

1. A clock interrupt procedure that decrements a time count for relevant tasks.
2. A basic task handling & context switching mechanism that will support the moving of tasks b/w queues and formation of task queues.
3. Primitive service routines.

Primitives:-

WAIT for some condition

SIGNAL condition & thus release one waiting on the condition.

ACQUIRE exclusive rights to a resource.

RELEASE exclusive rights to a resource

DELAY task for a specified time

WAKE task, that is suspended until the end of its specified cyclic period.

UNIT 8

Software Modelling

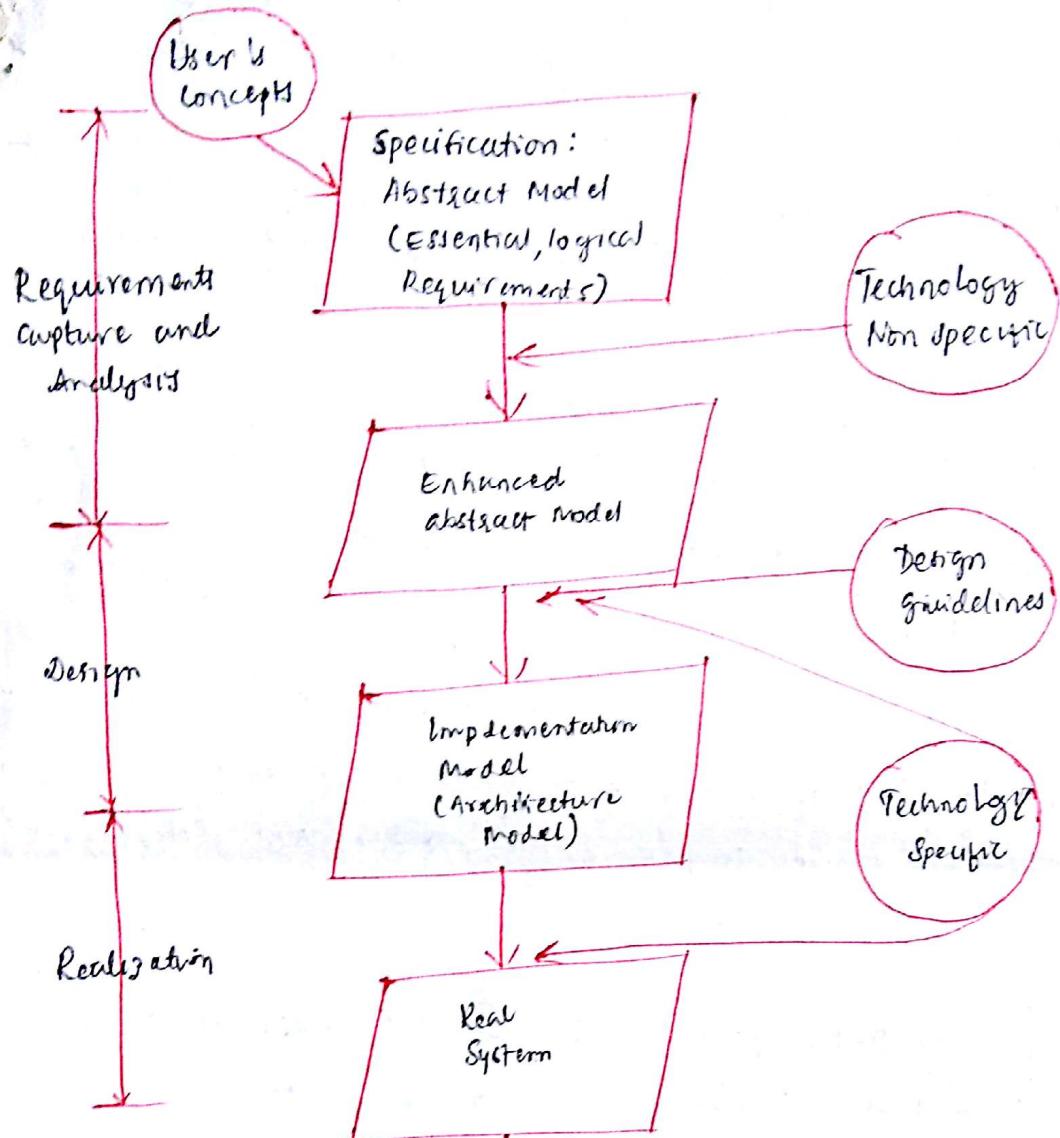


Fig Software Modelling.

- ① The production of a logical or abstract model - the process spec.
- ② The development of an implementation model for a virtual machine from the logical or abstract model. - the process of design
- ③ And the construction of software for the virtual machine together with the implementation of virtual machine on a physical s/m - the process of implementation.

④ The relationship of these three phases are shown in above

fig. 9845

③ Abstract model:

The equivalent of a requirement specification, is the result of requirements capture and analysis.

Implementation model:

This is the equivalent of the system design.

It is the product of the design stages - the architectural design and detailed design.

Implementation:

The process of mapping the implementation model onto the physical hardware, identifying the software module and coding them.

④ Although there is a logical progression from abstract model to implementation model to implemented software, the abstract model, implementation model and deliverable system are produced in three phases overlap.

⑤ The phases overlap because, complex systems are best handled by hierarchical approach

→ Determination of the detail of the lower levels in the hierarchy of the logical model must be based on the knowledge of higher level implementation decisions of

→ higher level decisions determine the requirement specifications for the lower levels in the sm.

UNIT 8RTS Development methodologies:

- * Introduction.
- * Yourdon Methodology
- * Requirement defin" for Drying oven.
- * Ward and Mellor method
- * Hatley and Prabhakar Method.

Yourdon Methodology

- ✓ The Yourdon methodology has been developed over many years.
- ✓ It is a structural methodology based on using dataflow modelling techniques and functional decomposition.
- ✓ It supports development from the initial analysis stage through to implementation.
- ✓ Both Ward and Mellor (1986) and Hatley and Prabhakar (1988) have introduced extensions to support the use of the Yourdon approach for the development of Real time systems.
- ✓ The key ideas of their methodologies are:
 - ① Subdivision of system into activities
 - ② Hierarchical structures.
 - ③ Separation of data and control flows
 - ④ No early commitment to a particular technology.
 - ⑤ Traceability b/w specification, design and implementation.

- Although the original method was developed as a pencil and paper technique, most benefit can be obtained if there is CASE tool support and many software engineering CASE tools now support both Ward and Mellor and the Hatley and Pirkhai versions of Yourdon.
- Examples are Software through pictures and Easy Case plus.

REQUIREMENTS DEFINITION FOR DRYING OVEN

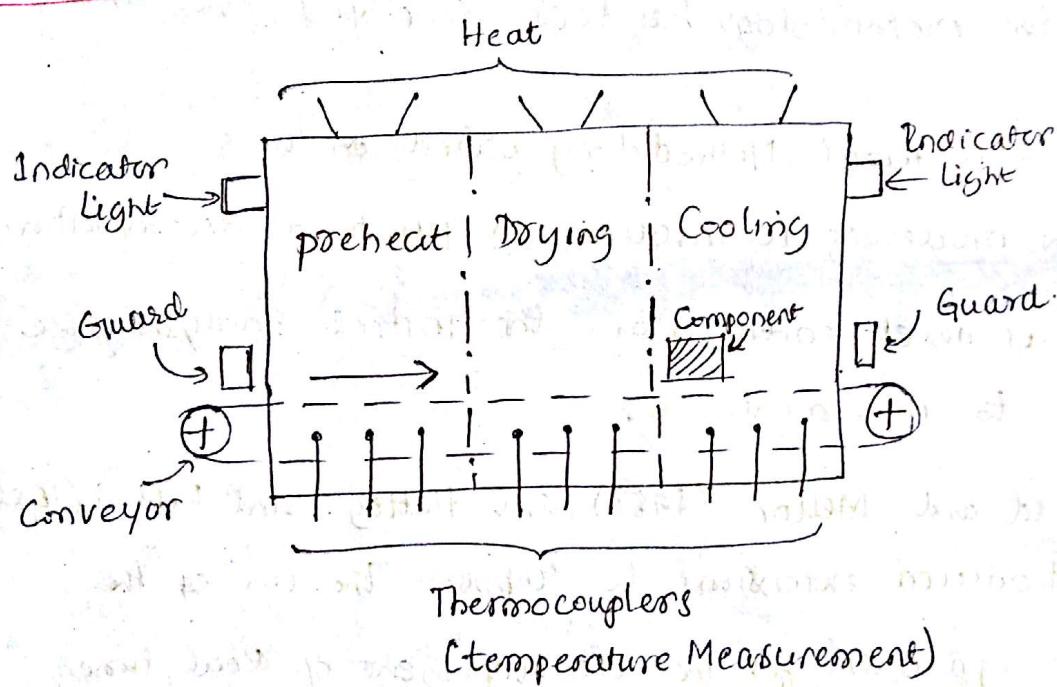


Fig: General Arrangement for Drying Oven.

INTRODUCTION

- Components are dried by being passed through an oven.
- The components are placed on a conveyor belt which conveys them slowly through the drying oven.
- The oven is heated by a three gas fired burners placed at the intervals along the oven. The temperatures in each of the areas heated by the burners is monitored and controlled.

- ④ An operator console unit enables the operator to monitor and control the operation of the unit.
- ⑤ The system is presently controlled by a hardwired control system. The requirement is to replace this hardwired control system with a computer-based system. The new computer based system is also to provide links with the management computer over a communication link.

⇒ The general arrangement of the system is shown in fig.

INPUT/OUTPUT:

- ① The inputs come from a plant interface "cubicle" and from the operator. There will need to be inputs obtained from the communication interface.
- ② Plant Inputs
- ① A thermocouple is provided in each heater area. The heater areas are pre-heat, drying and cooling. The inputs are available as voltages in the range 0 to 10 Vdc at point 1 to 9 on socket j22 in the interface cubicle.
 - ② The conveyor speed is measured by a pulse counting system and is available on pin 3 at socket j23 in the interface cubicle.
 - ③ There are three interlocked safety guards on the conveyor system and these are in-guard, out-guard and drop-guard. Signals from these guards are provided on pins 4, 5, 6 of socket j23. These signals are set at logic HIGH to indicate that guards are locked in place.
 - ④ A conveyor halted signal is provided on pin 1 of socket j23. This signal is logic HIGH to indicate conveyor is Running ~~not~~ ~~guards~~ ~~are~~ ~~locked~~ ~~in place~~.

③ Plant outputs:

① Heater control: Each of the three heaters has a control unit. The input to the control unit is a voltage in the range 0 to 10 which corresponds to no heat output to maximum heat output.

④ Conveyor Startup:

A signal convey-start is output to the conveyor motor control unit.

A second signal convey-stop is also output to the conveyor motor control unit.

The connections are to pins 1, 2, 3 on J10 for convey-start and to pin 5 for convey-stop.

⑤ Guard locks:

Asserting the guard-lock line, pin 8 on J10, causes the guards to be locked in position. And tums on the red indicators on the outside of unit.

⑥ Operator Inputs:

- * The operator sends the command inputs start, stop, Reset, Restart and Pause.
- * The operator can also adjust the desired set points for each area of the dryer.

⑦ Operator Outputs:

The operator VDU displays the temperature in each area, the conveyor belt speed and the alarm status and also the current date and time and the Last operator command issued.

⑧ Communication links

FUNCTIONAL SPECIFICATIONS:

4

① Start up:

- ✓ Starting from the cold, the operator checks that all the guards are closed and issues the "start" command.
- ✓ The guards are locked and if locking is correctly achieved the heaters are switched to ON. Under this condition maximum heat is supplied.
- ✓ The temperature is monitored and when each area reaches a temperature within 20% of its set point control of the heaters is switched to normal.

② Conveyor start up:

When all areas have switched to normal control the conveyor start up sequence is initiated.

- ⇒ The conveyor is stepped through the start up procedure.
Motor pos. 1 is selected and held until speed reaches 0.5 ft/min
then Motor pos. 2 is selected and held until speed reaches 1.5 ft/min
at this point the normal running position 3 is selected.
- ⇒ If the conveyor fails to reach the desired speed within 30 seconds the conveyor is stopped and conveyor fault signal is asserted.
- ⇒ The normally conveyor speed is 8 to 10 ft/min.
- ⇒ If any time speed drops below this for more than 30sec conveyor-alarm signal should be asserted.

③ Temperature Monitoring

The temperature measurement for each area is read at 2 second intervals

If the temperature for an area varied by 5% from the set point for that area the alarm should be asserted.

⑩ Each area is controlled by PID control algorithm.

(Proportional Integral
Derivative controller)

⑪ Conveyor failure :

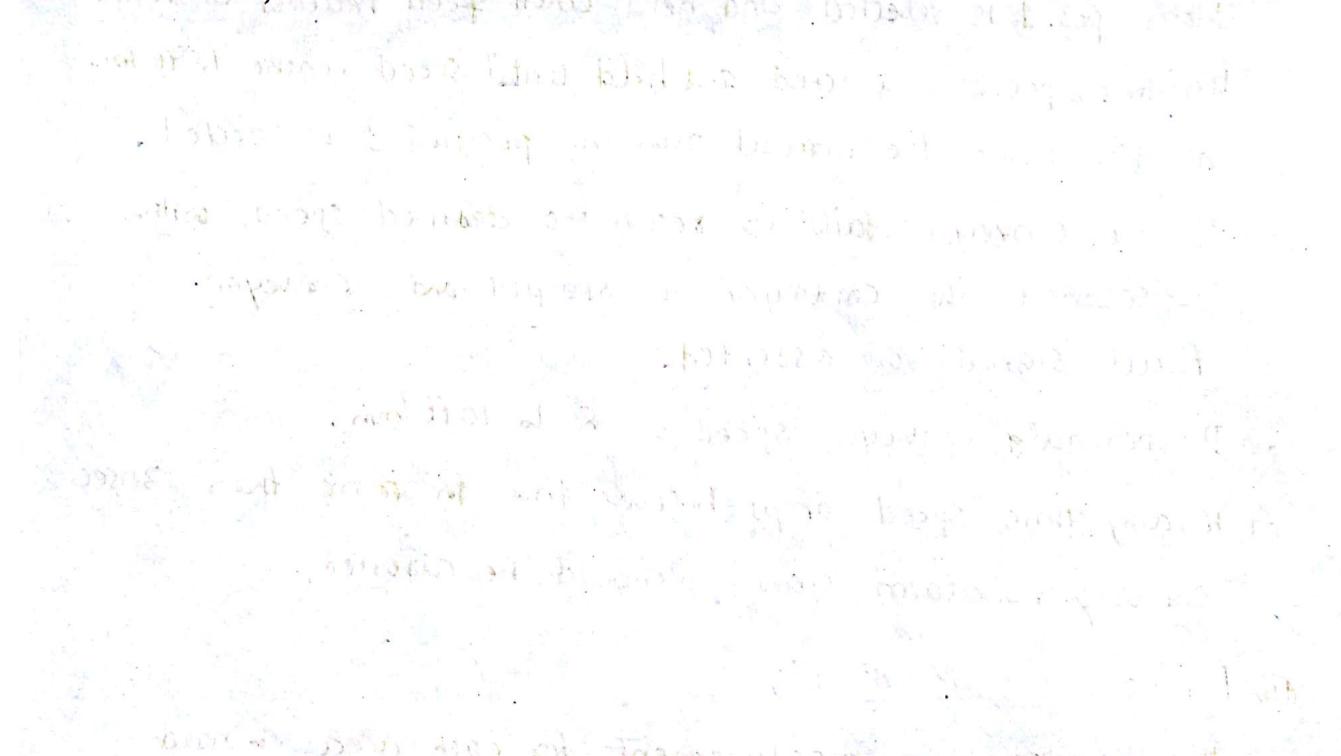
If the conveyor fails to start, the operator can issue a Reset command which closes the whole system.

Or the operator can issue a Re-start command which causes the conveyor first to be stopped and then to enter the full conveyor start-up cycle.

⑫ Conveyor pause

During normal running the operator can issue a pause command. This halts the conveyor. The conveyor can be re-started by the operator issuing the Restart command.

⑬ At any time during running the operator may issue a stop command. It is to turn off heaters and the conveyor.



② The OUTLINE of Abstract modelling approach of Ward and Mellor and Explain. 5

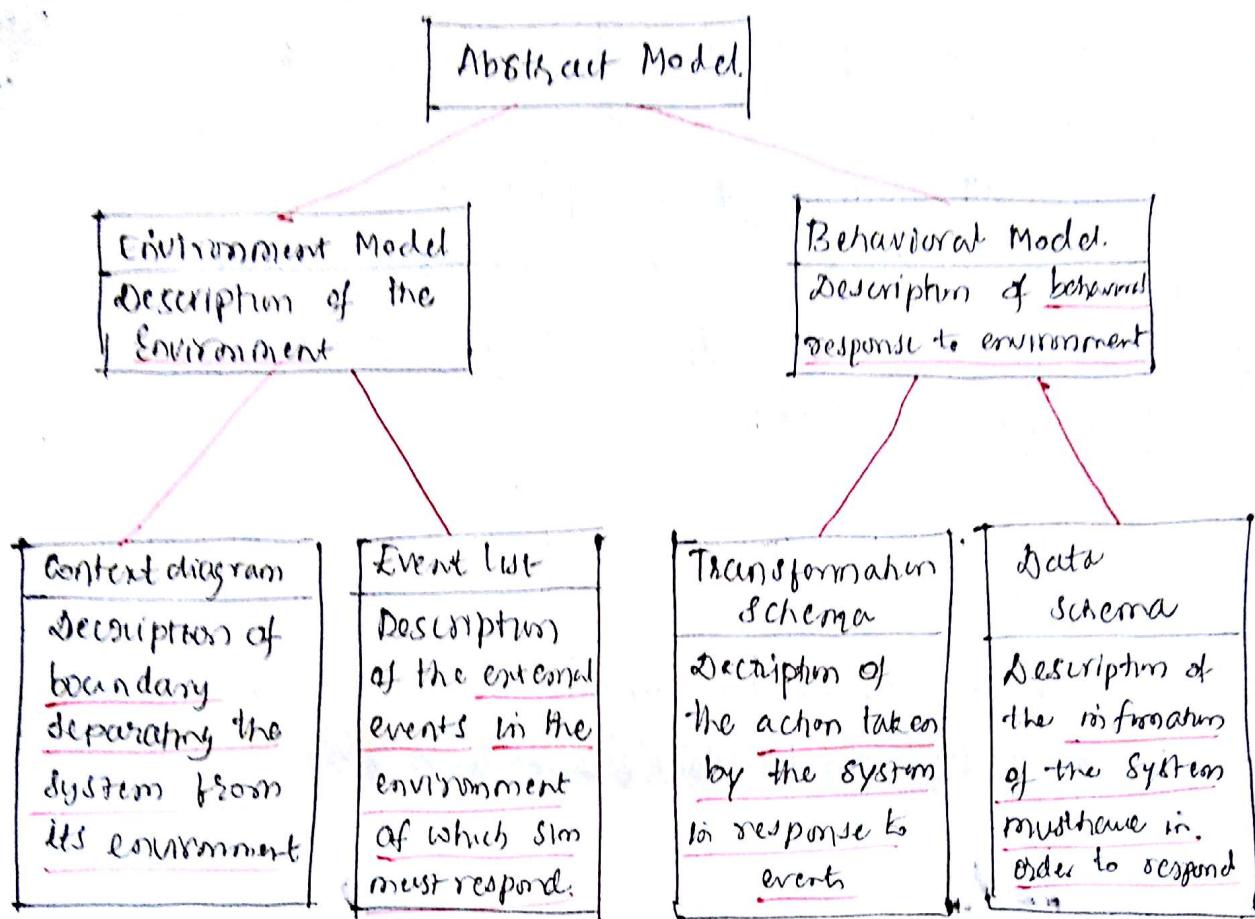


fig: Outline of abstract modelling approach.

- ① The outline of the Ward and Mellor method is shown in fig.
- ② The starting point is to build, from the analysis of the requirements, a software model representing the requirements in terms of abstract entities. The model is called Essential Model.
- ③ It is in two parts
 - Environmental model: describes the relationship of the system being modelled with its environment.
 - Behavioral model: describes the internal structure of the SLM.

- ④ The second stage is the design stage is to derive from the essential model an implementation model which defines how the sim is implemented on a particular technology and shows the (1) allocation of the part of the system to processors, (2) subdivision of activities allocated to each processor into tasks (3) The structure of the code for each task.
- ⑤ The essential model represents what the sim is required to do,
And the implementation model shows ^{How} the system will do what has to be done.

WARD AND MELLOR METHOD

6

- 1. The starting point is to be build, from the analysis of the requirements, a software model representing requirements in terms of abstract entities. This model is called Essential model.
- ✓ It is in two parts.
 - ① Environmental model: Which describes the relationships of the systems being modelled with its environment.
 - ② Behavioral model: Which describes the internal structure of the system.

2. The second stage, the design stage.

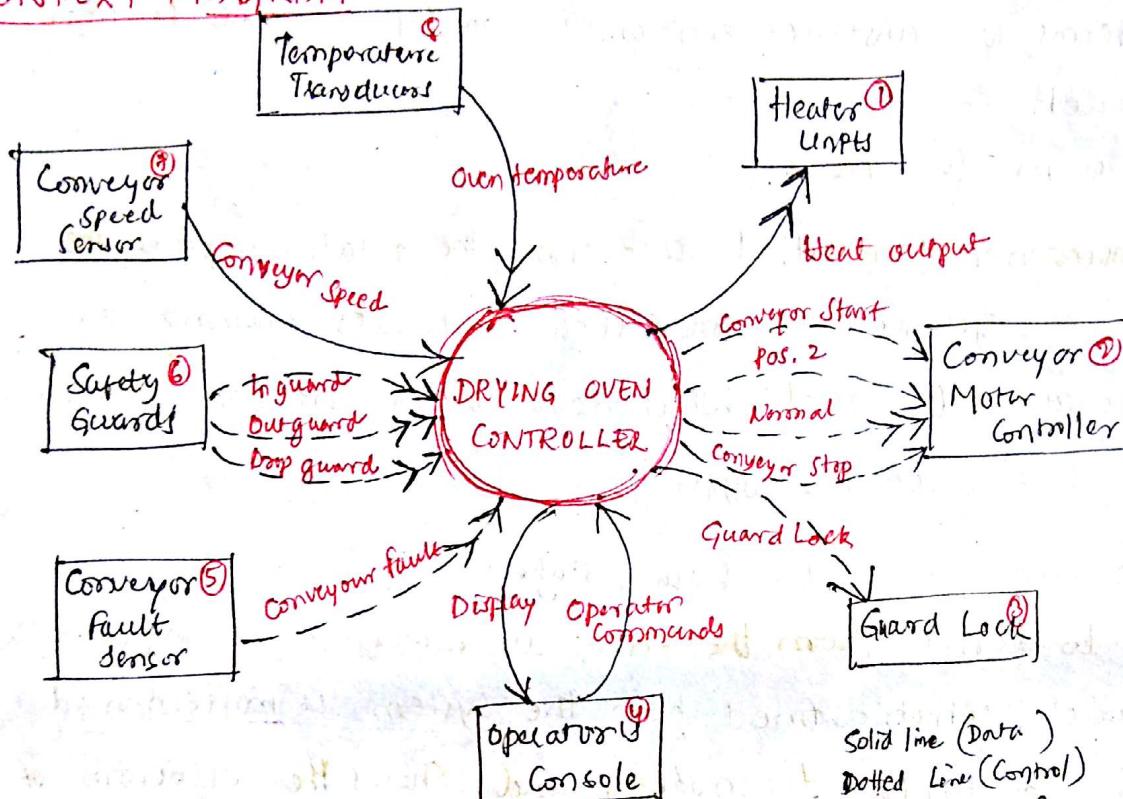
Is to derive from the essential model is an implementation model, which defines how the system is implemented on a particular technology and shows the allocation of parts of the systems to processes, the subdivision of activities allocated to each process into tasks and the structure of the code for each task.

- ↳ The essential model represents what the system is required to do,
And the implementation model shows how the system will do what has to be done.

① The environmental Model

For most real time systems the environmental model will comprise of context diagram and an event list.

CONTEXT DIAGRAM



Solid line (Data)
 Dotted line (Control)
 Double arrow (Continuous)
 Single arrow (Discrete)

- ⑨ The rectangular boxes indicates terminator blocks which are the entities that exists for environment.
- ⑩ The directed lines represent data and control information passing through b/w sim and environment.
- ⑪ Continuous flows are used to represent data or control signals for which there is always a value available to the sim.
- ⑫ Discrete flows are used for the data or control which is generated as separate items.
- ⑬ An analogy that Discrete flows $\xleftarrow{\text{Evt}}$ Connection based on Chapel
 Continuous flows $\xleftarrow{\text{Evt}}$ Connection based on pod.

④ As the context diagram is drawn, all flows should be named and each should be entered into a directory called Data Directory.

It consists of information: Name, Alias, Usage, Content description and other Information.

EVENT LIST:

This is a table which lists all the events that can cause a change in the system and result in a change in an output.

The event list for the Drying oven is as shown below.

Event	Action	Response	Time
① Start	Lock guards	Guard lock	< 0.5s
② In Guard	Start heat up	Set maximum	
③ Out Guard	Cycle	Heat output	< 0.5s
④ Drop guard	when heat normal	Conveyor start	
⑤ Pause	Stop Conveyor	Conveyor stop	< 0.5s
⑥ Conveyor fault	Raise alarm	Conveyor alarm	< 0.1s
⑦ Stop	Close System down	Conveyor stop Heaters off	< 0.5s
⑧ Oven temperature	Do - control	Heat output	Cyclic 1.0s
⑨ Conveyor Speed	Check for normal	Conveyor Alarm	Cyclic 5.0s

① The behavioral model

It shows how the system should respond to events taking place in the environment. Sys's divided into functions.

① first level transformation diagram:

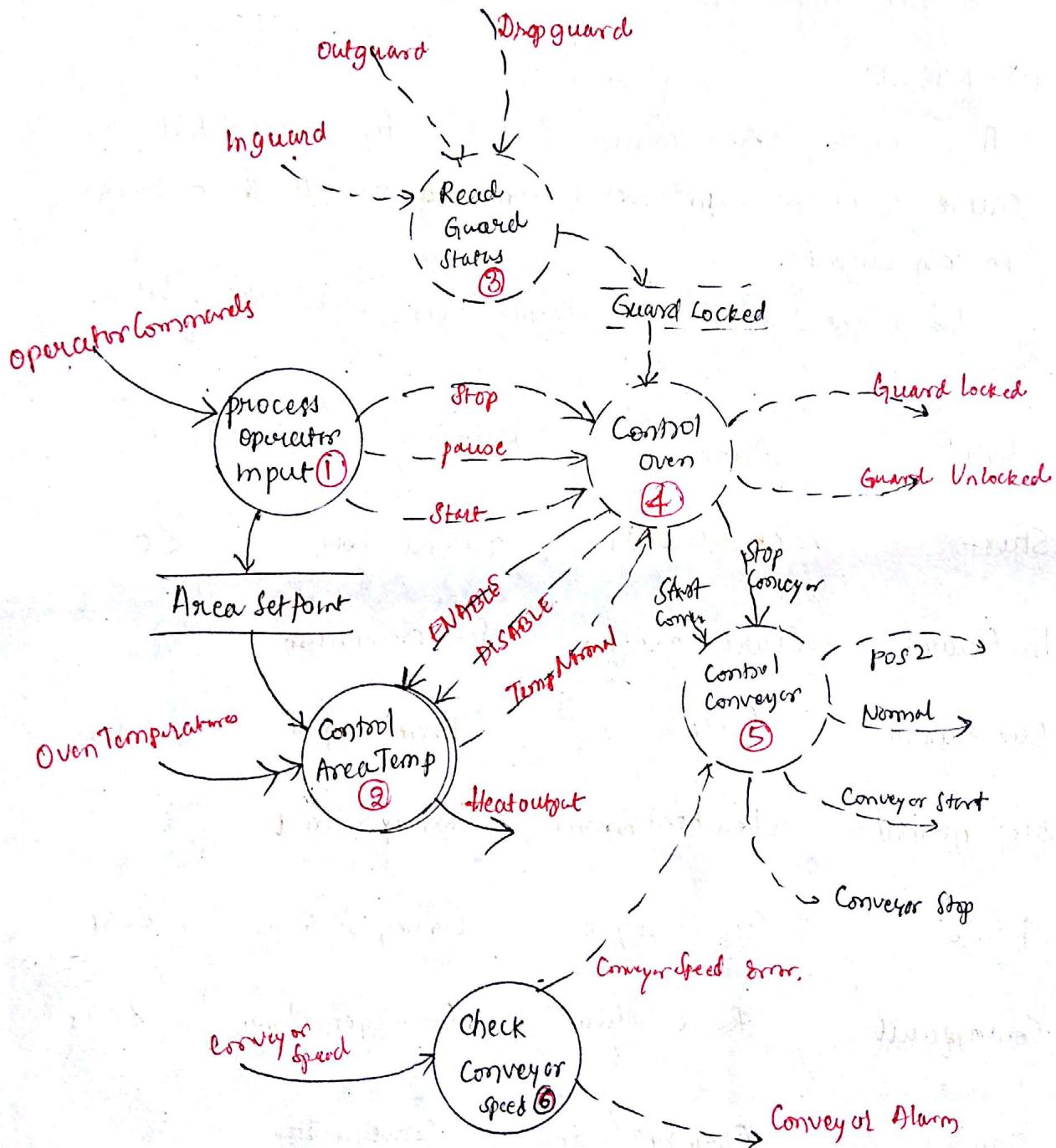


Fig: Level ① transformation diagram.

⇒ The bubbles drawn with solid lines represent data transformations and those drawn with dotted lines represent control transformations.

⇒ The double line round part of the bubble indicates that there are multiple instances of this transformation
a Control Area temp

⇒ The two entities placed between parallel lines labelled Area Set points and Guard locked are respectively a data store and a control store.

Flows entering and leaving the stores are not named as they are assumed to take name of the store.

⇒ Each transformation is given a number, all level 1 transformations have single digit numbers (this indicates that they are level 1)

② Second level transmission diagram

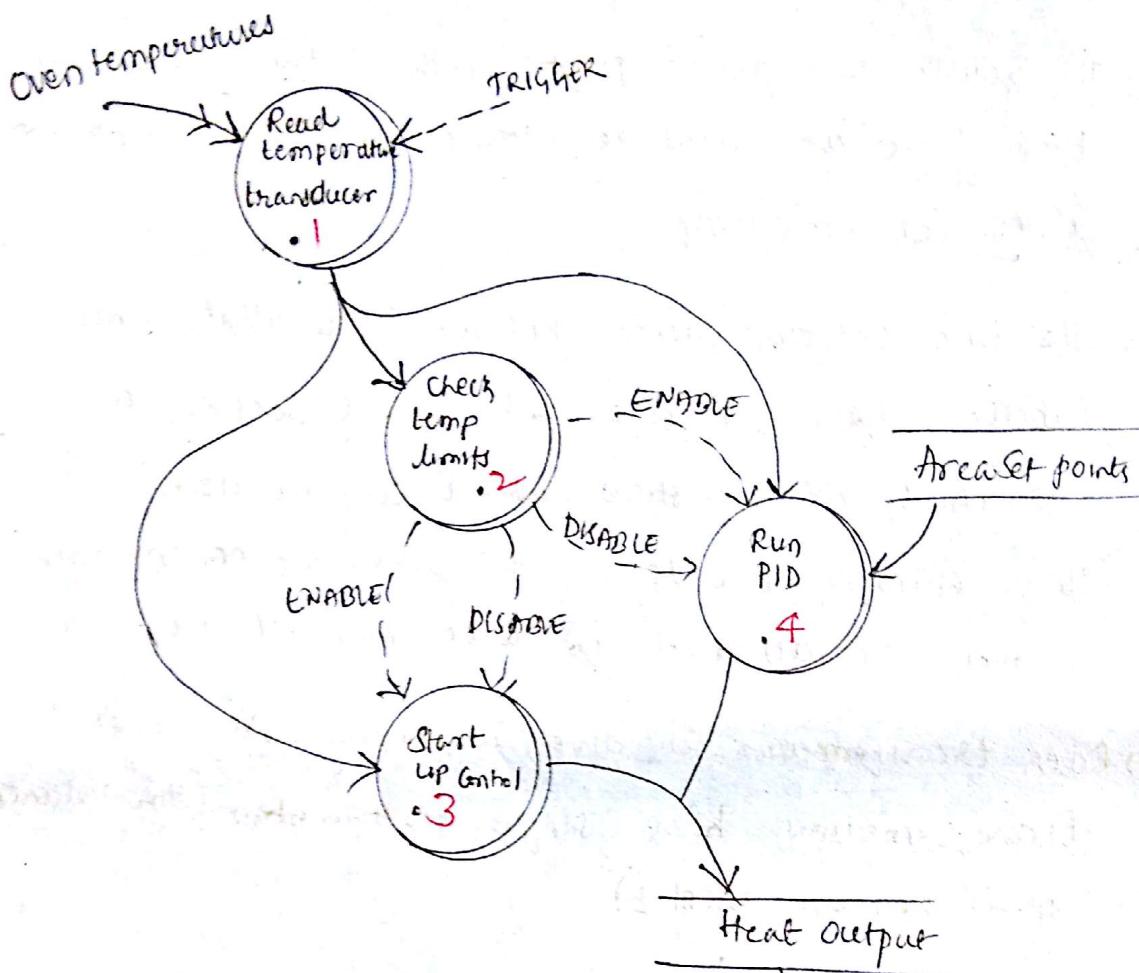


Fig: Level 2 transmission Diagram

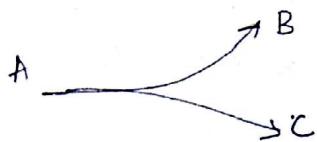
- ⇒ Building the model proceeds by taking each transformation in the level 1 diagram and breaking it into smaller units. for ex: above fig is expansion of ControlAreaTemp.
- ⇒ This diagram contains one new name, TRIGGER. This is a prompt which is used to indicate that the transformation is run once each time when TRIGGER becomes true. It runs in response to a periodic signal or event.
- ⇒ The transformations on this level two diagram are numbered with a full stop in front of the number, this indicates that for full identification Number of the transformation diagram should be added.

Behavioral Model Rules and Convention.

⇒ There are a number of Rules and conventions associated with transformation diagrams

Notation

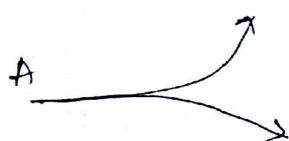
①



Meaning

Here B and C are the two subsets of A which pass on to two diff transformations.

②



All of the A is passed on to two different transformations

③



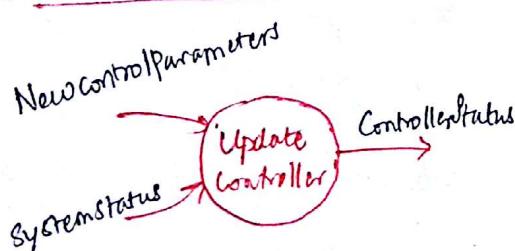
B & C come from two different transformations and they are combined to form a single data flow.

④

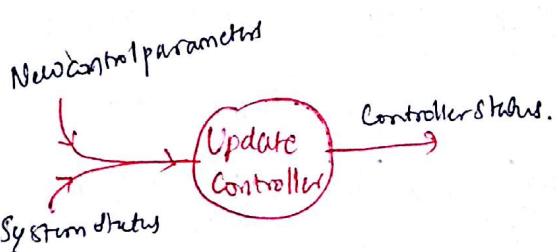


All of A may be provided by either of two diff transformations.

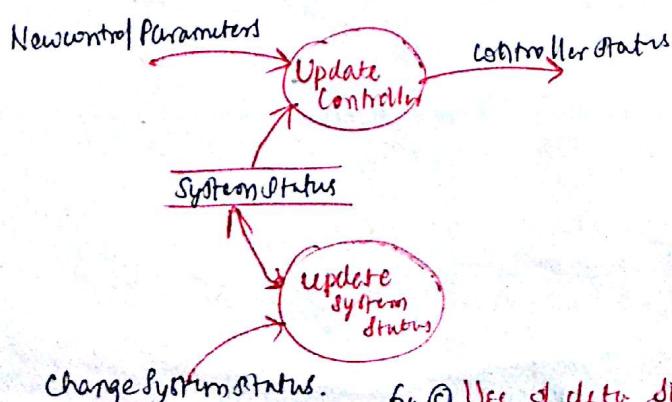
① Data transformations.



fig(a) Ambiguous dia



fig(b) Merged dia



fig(c) Use of data store

① Data transformations with discrete input flows are assumed to be data triggered and hence there can be only one input from another transformation.
The reason for this restriction is to avoid Ambiguity.

- ② Consider the transformation shown in fig ④.
- This can be interpreted, ~~only~~ if both the inputs are present simultaneously or is it to be implemented that one input will be stored until other arrives and the transformation executed?
 - So to avoid this ambiguity the transformation must be represented as shown in either fig ⑤ or ⑥.

③ Fig ⑥ shows the input as a composite ~~flow~~ data flow and the convention is that "the transformation can only be triggered when all the elements of a composite data flow are present."

④ Fig ⑥ A data store is used to hold one of the data flows

The interpretation of this diagram is that

- Updatesystemstatus runs when a transaction changesystemstatus appears. And the system status is held in a Datastore.
 - ✓ Update controller when a transaction Newcontrol parameters appears and it obtains the current system status from Datastore
- This means the data store will have to have the characteristics of pool since it may need many times.

② Control transformations:

There is no restriction on the no. of inputs to a control transformation.

③ Input & output types:

The permitted mixture of input and output flows from the transformation is summarised below

Transform	Inputs	Outputs
Data	Dataflow prompt	Dataflow Control flow
Control	Control flow prompt	Control flow prompt

④ Balancing:

Since transformation diagrams at a given level represent the same information, as the diagram at the next higher level the inputs and outputs must match the inputs and outputs of the higher level diagram.

The process of checking that input and outputs correspond is referred to as balancing.

~~OPF~~ Process Specification

The OPF protocol is a data exchange protocol based on state routing of the domain. It also has an autonomous system.

Process Specifications:

- ① The process specification (PSPEC) is a description of the actions that the data transformation has to carry out.
- ② At some point in the design, the transformation specification will have to be expressed periodically and expressed and often a transformation will be expressed using a procedural notation.
- ③ The most common method is to use a pseudo-code, it can be an informal programming language for e.g:

The PSPEC for a Readtemperature Transducer.

PSPEC 2.1 Read temperature transduced

INPUTS : OvenTemperatures

OUTPUTS: Areatemp

Every 1.0 seconds DO

 read @OvenTemperatures

 convert to internal data representation

 output converted value as Areatemp

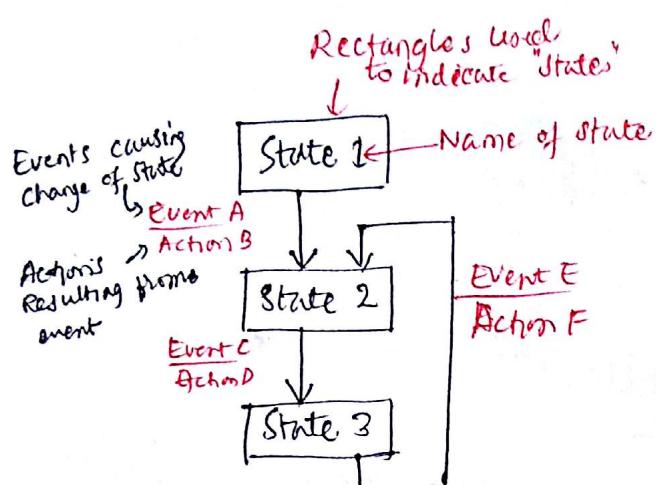
END.

- ④ Most CASE tools provide a means by which PSPEC can be stored in the text files which can be manipulated using a text editor.

* Control Specifications

- ① Control transformations are described using CSPECS.
 the most usual form of a CSPEC is a State Transition diagram (STD) and / or State transition matrix (STM)

② State transition diagram of State transition matrix



Present State	Event A	Event C	Event E	Next State
State 1	State 2			Action B
		State 3		Action C
State 2			Action D	Action E
State 3			State 2	Action F

fig(i) State transition diagram

by(ii) State transition table

Fig(i) & (ii) shows the general form of the STD and STM.

③ Example of a CSPEC

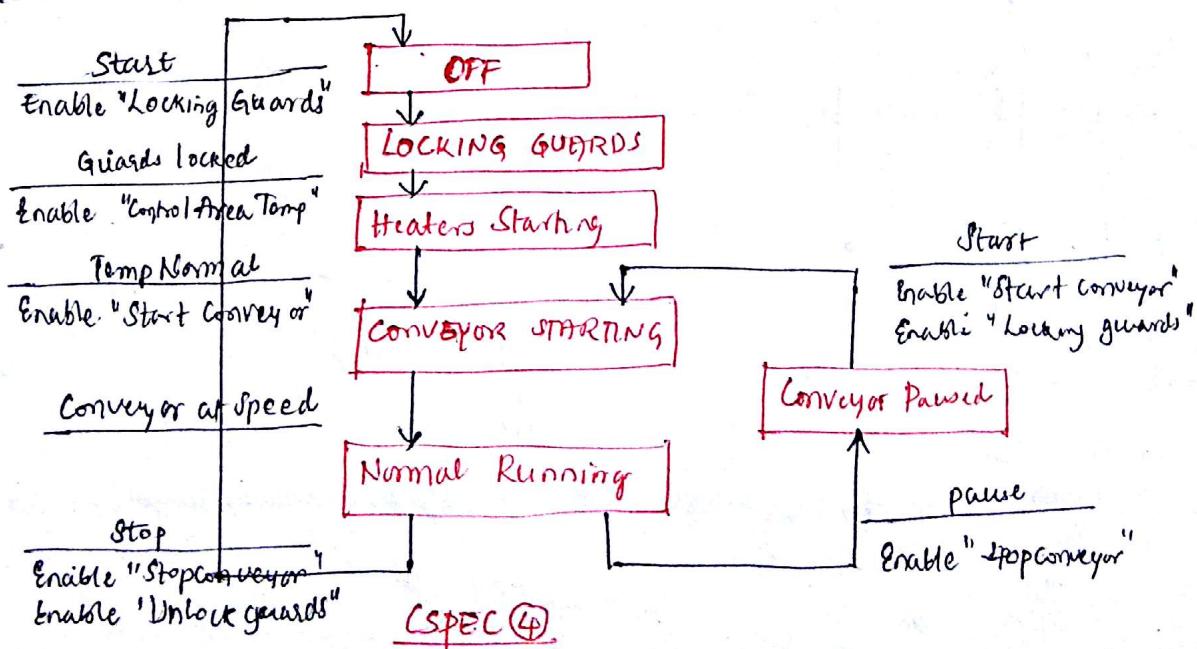


fig: Example of a CSPEC is (State (transition diagram form))

STD of CSPEC

State transition Matrix of CSPEC

	Start	Stop	Pause	Guard locked	Temp Normal	Conveyor at Speed.
OFF	LOCKING GUARDS					
	lock guards					
LOCKING GUARDS				HEATERS STARTING		
HEATER STARTING					Control area temp	
CONVEYOR STARTING					CONVEYOR STARTING	
CONVEYOR PAUSED			OFF	CONVEYOR PAUSED		Start conveyer
NORMAL RUNNING		Stop conveyor unlock guards	OFF	Stop conveyor CONVEYOR PAUSED		
PAUSED	CONVEYOR starting lock guards			Stop conveyor		
CONVEYOR PAUSED	Start conveyor					

- ① The Blank entries in the STM represent non-operational states or undefined states
- ② STM reveals undefined states which frequently represent the exceptions which the designer must take into account.
For eg. In above fig we find that there is only one entry in the column for the event stop. It should be clear that we need to know what to do if the stop occurs when the system is in all other states.
What action should be taken if the event is Temp Normal occurs when in Locking guards. It should be clear that this should not occur and hence must represent a fault in the sys.

Checking the Essential Model.

Ward and Mellor recommend checking the transformation schema of the behavioral model in two ways.

- ① The first is to use the rules for data flow for consistency. This is equivalent to checking the syntax of the program and can be done by hand or by an graphical compiler.
- ② The second level of checking is to determine whether the model can be executed - can it generate outputs from a given set of inputs.

Building the Implementation model

The construction of the implementation model divides into four phases

- ① Enhancing the environmental model
- ② Allocation of processes
- ③ Allocation of Activities to the task for each process
- ④ Definition of the structure of the task.

and also allocation of resources.

Teacher's Signature : _____

Enhancing the model:

Enhancing the model is concerned with;

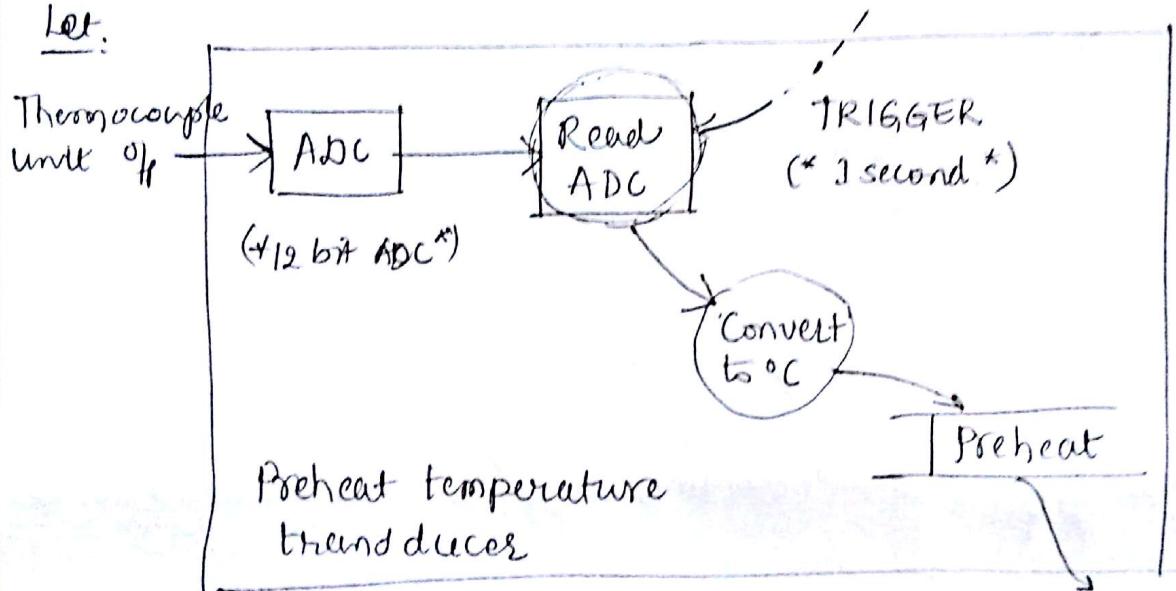
- ① Classifying the boundaries between the system and environment and

Determining what activities the system will carry out and what will be done as part of the environment.

- ② Elaborating data descriptions.

- ③ Adding timing and process activation information.

Let:



Eg: Example of a virtual transducer

- ④ In fig we assumed a 12 bit ~~ADC~~ precision ADC. The data dictionary entry for Preheatcomp will be updated to include information on the resolution of the temperature measurement.

- ⑤ We add the comment (* 1second, ydec*) to it to indicate that the ReadTemperaturetransducer transformation has to be executed or triggered after every 1second.

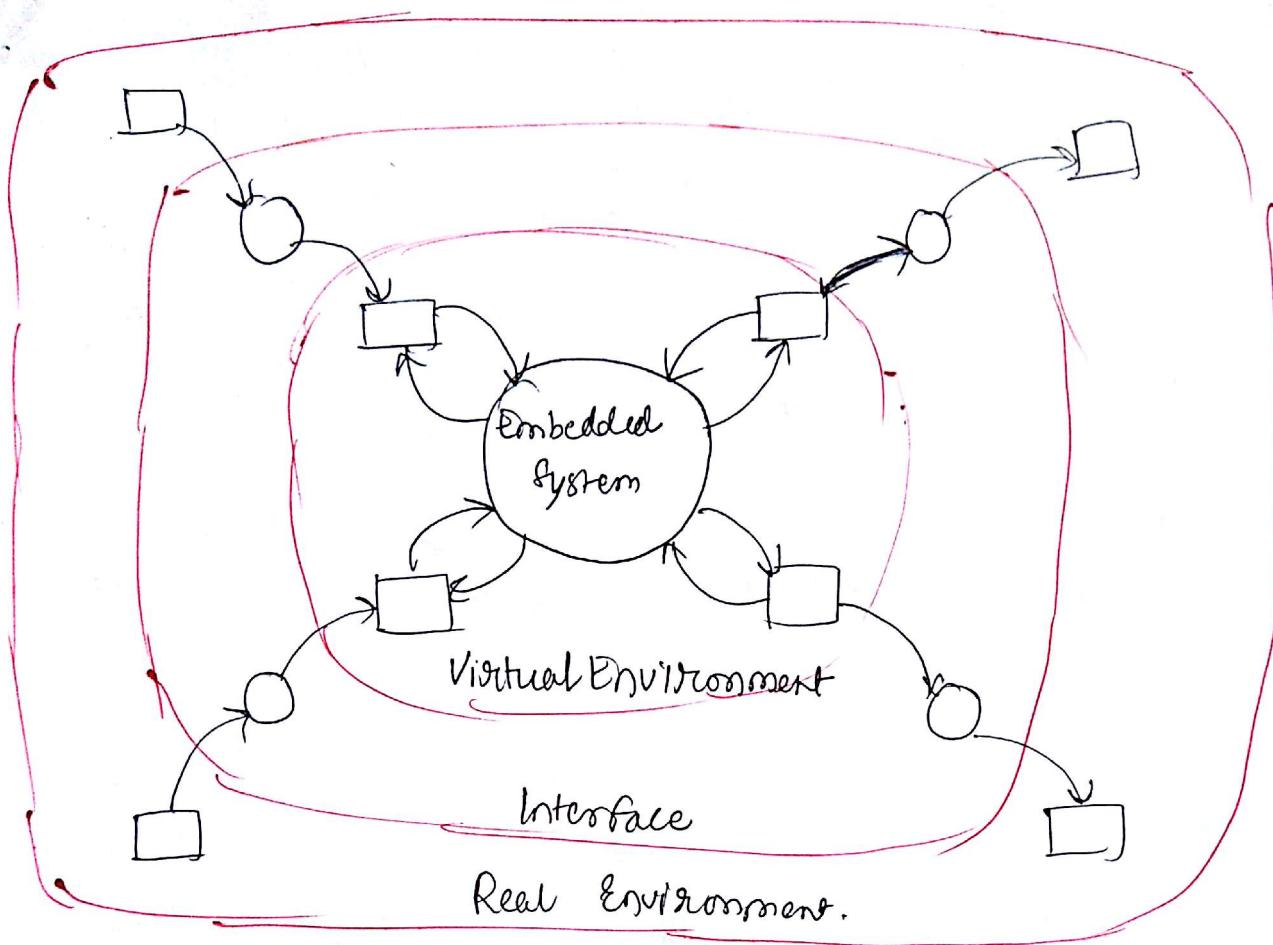


fig: Relationship between Real environment & Virtual environment.

Ward and Mellor regard the process as largely being concerned with providing an interface shell around internal software systems as shown above.

② It separates the functional operations of the interfaces from their electrical and physical manifestation and also serves to hide many of details, of how the functions are implemented.

The implementations may involve both hardware & software.

Allocation of Resources

① The first stage of allocating resources is to decide how many and what type of processing units are required and

how the various functions to be performed are to be allocated to each.

Processing units may be digital computers, logic circuits, analog devices, mechanical devices or human beings

② The next stage is to decide on the task structure and the allocation of the task to the individual processors

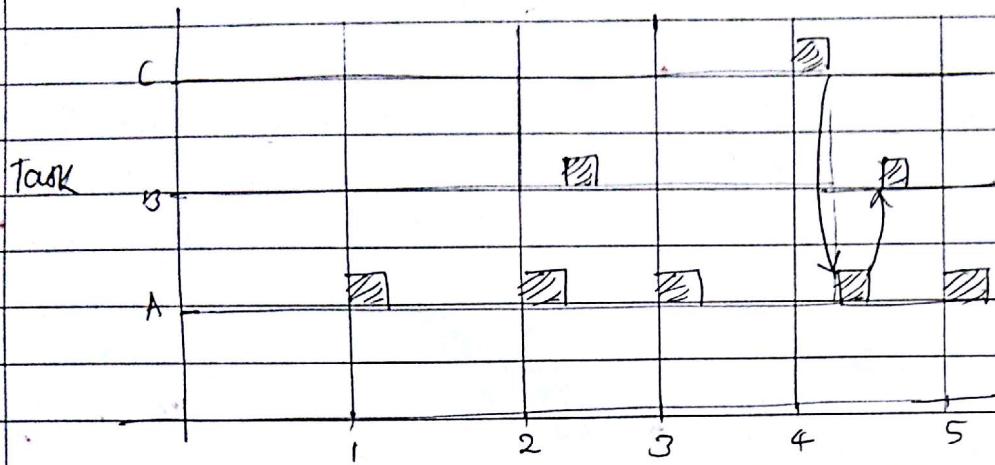
In carrying out this process we need to keep in mind both the standard software engineering design heuristics of

- ① Information hiding
- ② Coupling and cohesion.
- ③ Interface minimisation.

and also some rules of guidance need for RTS.

- ④
 - ① Separate actions into groups according to whether the action is
 - ⓐ time independent
 - ⓑ synchronized
 - ⓒ independent.
 - ② Divide the time-independent actions into
 - ⓐ Hard time constraint
 - ⓑ Soft time constraint
 - ③ Separate actions concerned with the environment from other actions.

with the environment from
Teacher's Signature : _____

(8) Task activation diagramFig: Example of a task activation diagram.

- ☛ The task activation diagram shows the use of the CPU by each task that has to run at a fixed cycle time during each clock cycle (Tick) of real time clock.
- ☛ Using this diagram the effects of task priority and preemption strategies are clearly seen and assessed.

Relationship b/w model and diagrams

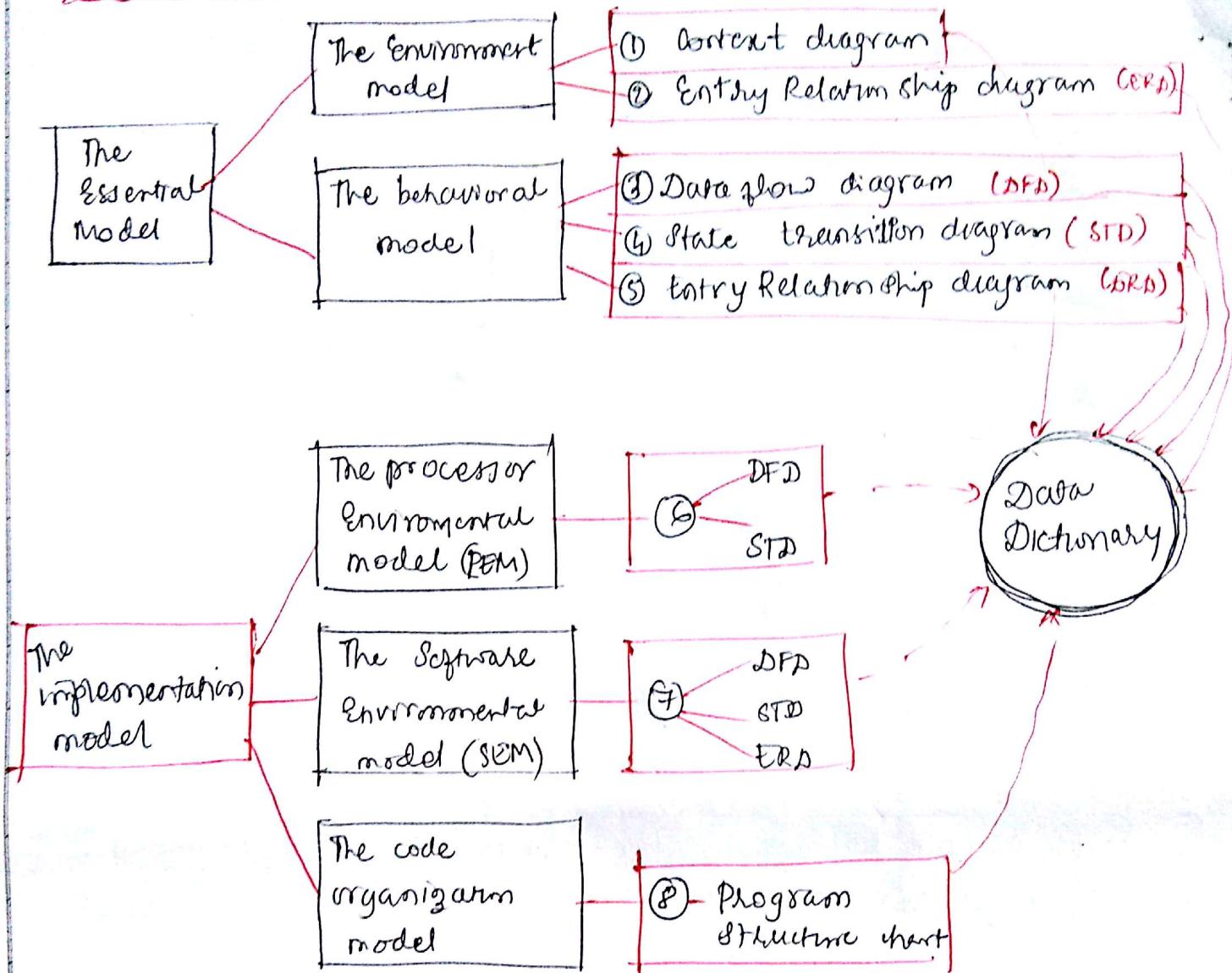


fig Relationship b/w models and diagrams.

- ① The correct use of the method result in documentation. This provides traceability from the physical to abstract specification model.
- ② The type of documentation produced is shown in above fig.

HATLEY AND PIRBHAI METHOD

③ Requirements model.

The basic structure of the requirements model is shown in Fig.

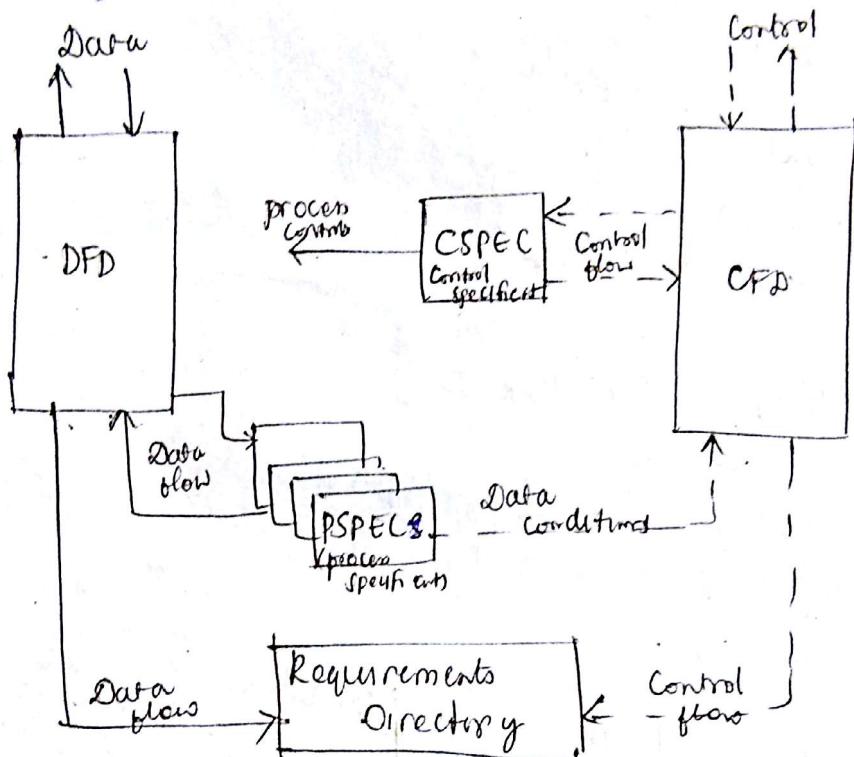
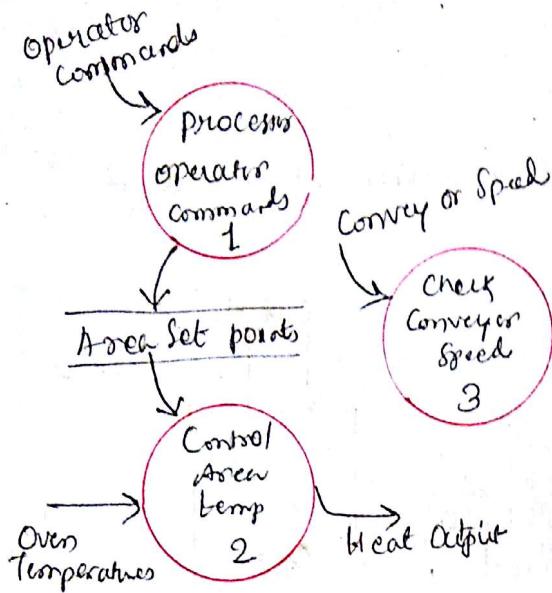


Fig: The structure of requirements model.

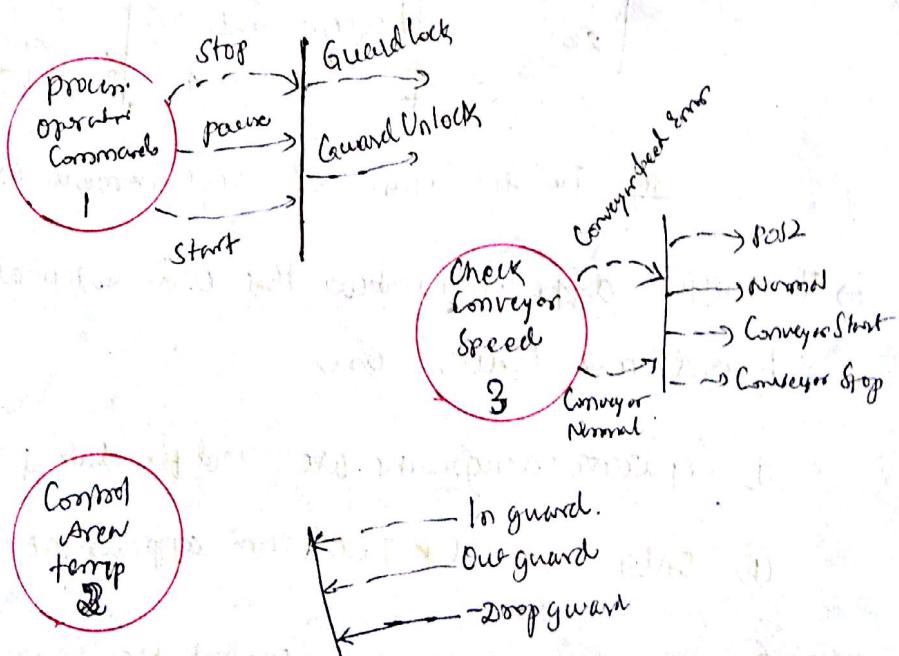
⇒ The major differences b/w this and essential model of Ward and Mellor are.

- (i) Separate diagrams are used for data & control.
- (ii) only one CPSPEC can appear at any given CPD level.
- (iii) All data flows and control flows are shown with a single arrow heads. ↑ The normal assumption here is flow is continuous.
But the distinction b/w continuous flow is determined by the way in which process is activated

Drying oven controller in the Hartley & Pirobhai Notation



DFA O Drying Oven Controller



CDF O Drying Oven Controller

Fig: Hartley & Pirobhai Notation

- ① The 'process transformations' appear on both the DFD and CFD.
 This is because CFD shows , a process can produce a control flow as an output.
 It usually arises as a result of some form of comparison which generates an event.
- ② The CSPEC is represented by a BAR.

Although three BARs are shown they form one CSPEC.
 and because there is only one , it does not need to be named on the diagram.

It takes the number & name of the diagram. CSPEC 0

- ③ The process activation information is held in the CSPEC.

④ Architectural Model

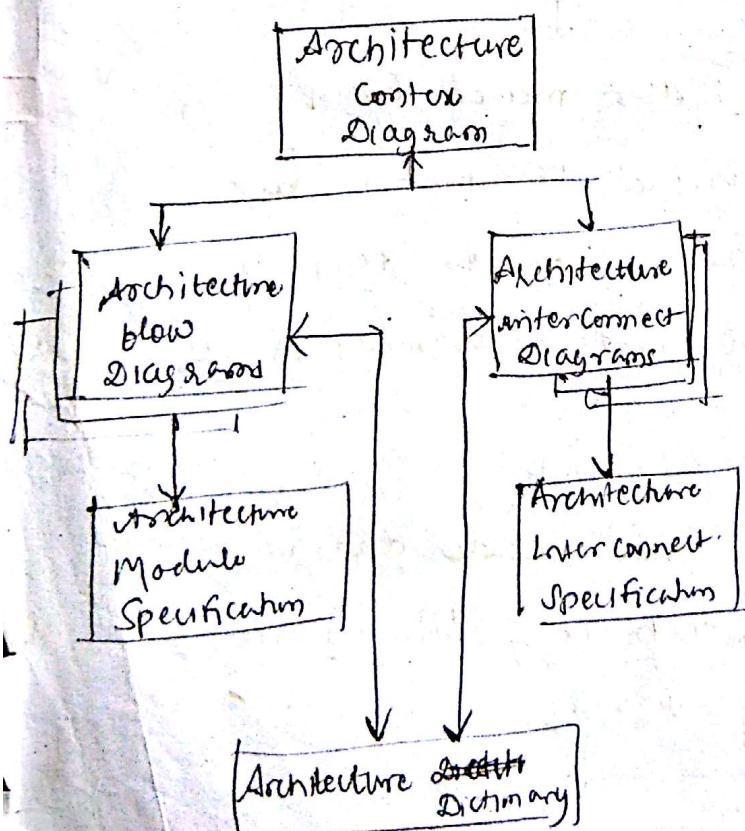
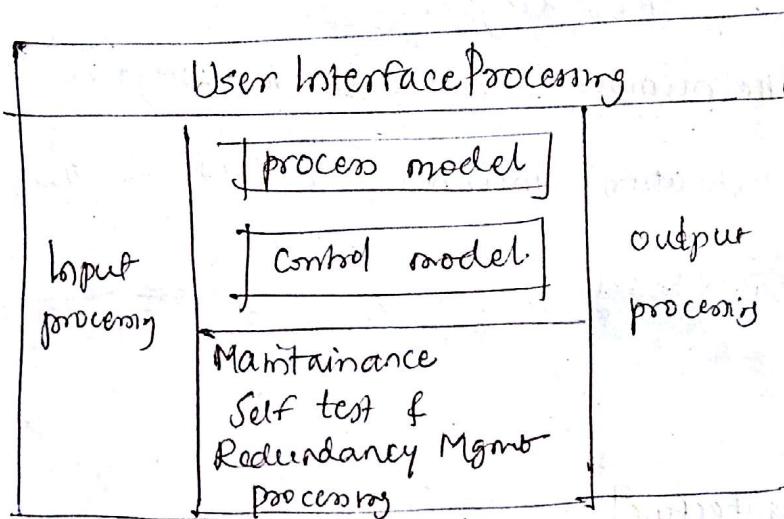


fig Architecture model Components

- ⇒ The general structure of the architecture module is shown in fig. and as with a requirement model it is hierarchical layered structure.
- ⇒ In developing a architecture model a procedure based on using an architecture template is suggested.

⇒ Fig below shows the architecture template



⇒ It is akin to the Ward & Mellor method for enhancing the model by developing virtual terminators but in this case it is suggested that the template be applied at each level in the requirements model Hierarchy.

⇒ The architecture module also includes diagrams showing the interactions technology b/w various technologies elements of the sys.

Difference between Ward and Mellor methodologies and Hatley and Pirbhai methodologies.

Ward and Mellor Methodology

Hatley and Pirbhai Methodology

① Essential model Requirements model

② Implementation model Architecture model

③ Transformation schema Data flow diagram
Control flow diagrams

④ Data transformation Process model

⑤ Control transformation Control model

⑥ Data dictionary Requirements dictionary &
Architecture dictionary,

Comments on Yourdon Methodologies

- ① Both methodologies - Ward and Mellor & Hatley and Pirbhai are simple to learn and have been used widely.
- ② They are founded on well established Structured methods developed by the Yourdon organization.
- ③ Hatley and Pirbhai method is more structured and formalized in its approach.
Its diagrams are less cluttered than Ward & Mellor method.
- ④ The weakness of both methods lies in Allocation of Processes and tasks.

Teacher's Signature : _____

MODULE 5

DESIGN OF RTS

DESIGN OF RTSS - GENERAL INTRODUCTION:

Introduction, Specification documentation, Preliminary design, Single program approach, Foreground / background, Multitasking approach, Mutual Exclusion Monitors.

Design of Real Time Systems:-

The approach to the design of Real Time Computer System can be divided into 2 Sections:-

1. The Planning phase
2. The development phase

Planning phase:-

In this phase, based upon the requirements of the User, the detailed Specification document is produced.

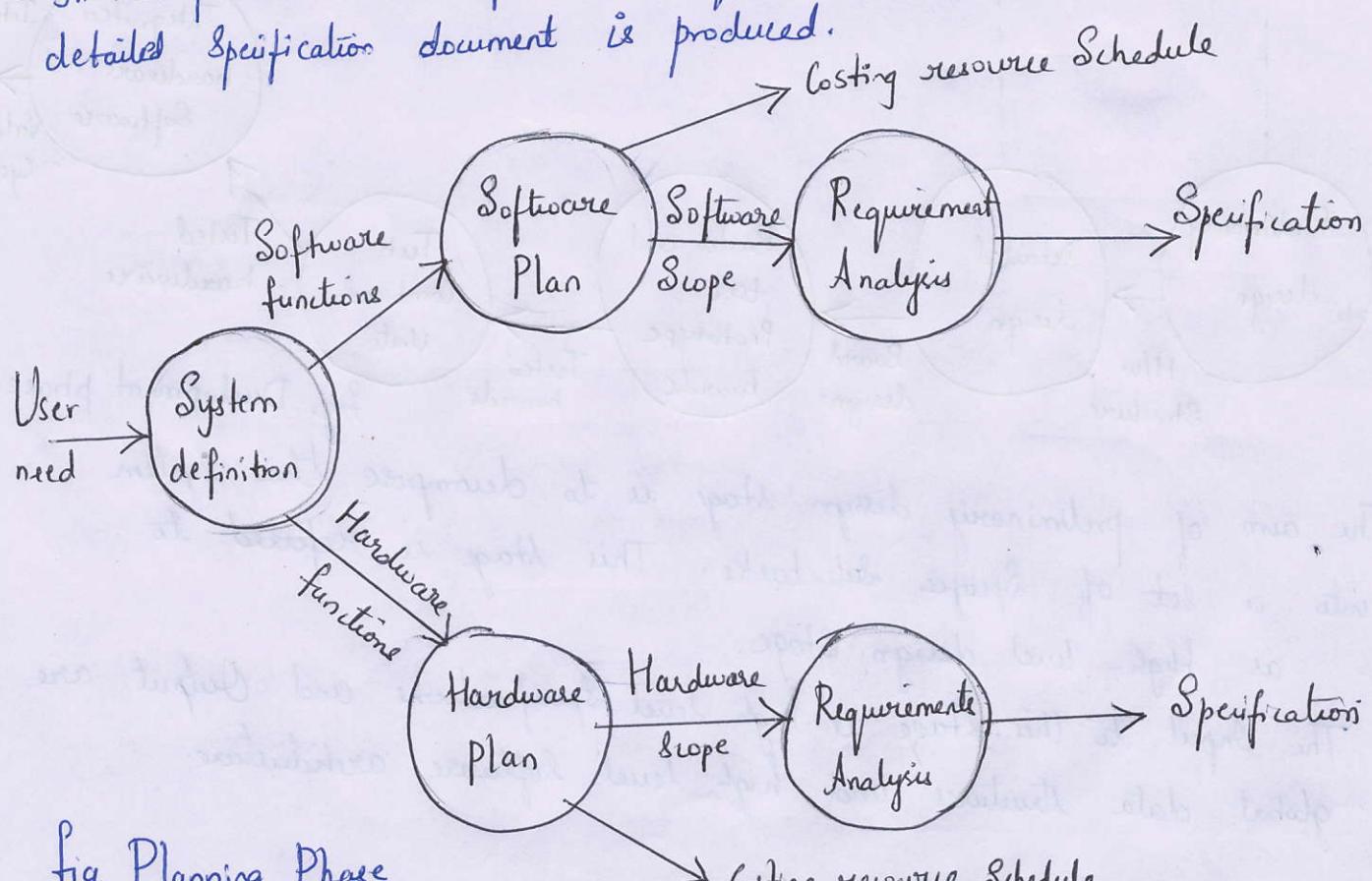


fig. Planning Phase

- ↳ It involves Outplan of resources - Such as people, time, equipment, cost - required to design Real time system.
- ↳ At this stage, preliminary decisions regarding the division of functions between hardware and software will be made.
- ↳ This phase also carries out the preliminary assessment of type of computer system such as a single central computer, hierarchical or distributed systems.
- ↳ The outcome of this stage is Specification or Requirements document.

Development phase:-

The output of this stage is Integrated or Validated System.

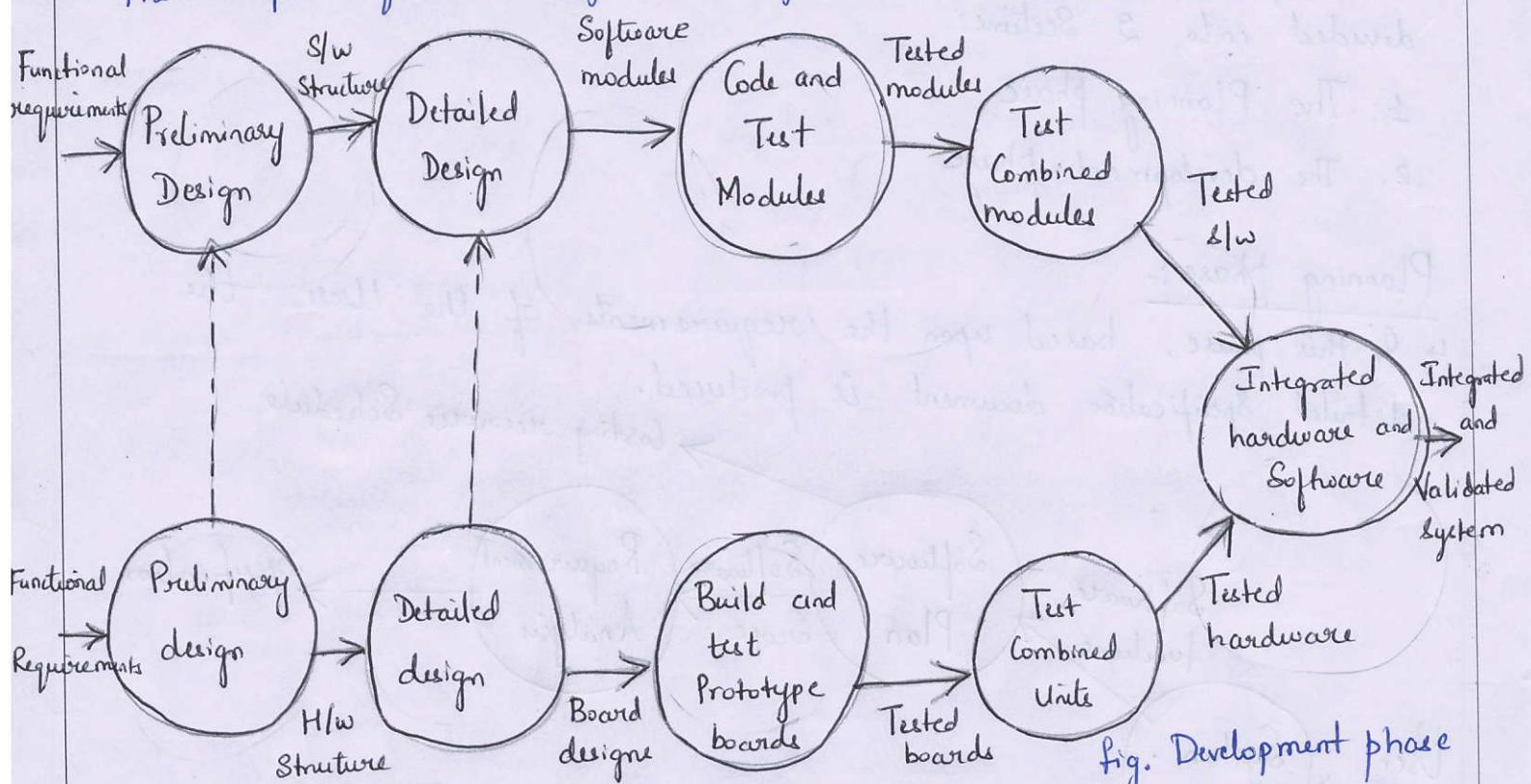


fig. Development phase

- ↳ The aim of preliminary design stage is to decompose the system into a set of specific sub-tasks. This stage is referred to as high-level design stage.
- ↳ The input to this stage is high level specifications and output are global data structures and high level software architecture.

At the end of preliminary design stage, a review of both the hardware and software designs should be carried out.

↳ The detailed design is divided into 2 stages-

① Decomposition into modules - for Software design

② Module Internal design - for Hardware design

The decomposition into modules involves Heuristic rules to aid the designer in making decisions.

This is done using General Methodologies

1. Functional decomposition:- It involves subdivision based on Separation into functions i.e., Each module performs specific function.
2. Information Hiding:- The module division is based on hiding as much as possible of the information used by the module, within the module.
3. Object Oriented :- In this method, the system is divided into entities which contains both data and functions that operate on the data.

For hardware design, it involves questions on board structure of the system such as

- ↳ Are separate boards going to be used for analog / digital Inputs.
 - ↳ Can processor and memory located on one board?
 - ↳ What type of bus structure should be used?
- It also involves, design of the boards.

Specification Document:-

Assuming that the planning phase has been completed and a Specification document has been prepared.

Let us take an example of Hot-air blower Specification.

This document will include the following Information:-

- Introduction
- Plant Interface
- Control
- Operator - Communication
- Management Information
- General Information.

1. Introduction:-

The System Comprises a set of hot air blowers arranged along a Conveyor belt. Several different configurations may be used with a minimum of 6 blowers and maximum of 12.

2. Plant Interface

a) Input from plant:- Outlet temperature : analog signal range 0-10V, corresponding to 20°C to 64°C, linear relationship.

b) Output to plant:- Heater Control:- Analog Signal 0V to -10V -
Linear relationship. (fullheat) (no heat)

3. Control:

A PID Controller with a Sampling Interval of 40ms is to be used.

This Sampling Interval may be changed but not less than 40ms.

The Controller parameters are Expressed to the user in Standard analog form.

The Set point is to be Entered from Keyboard.

4. Operator Communication

- a) Display

The operator display is as shown below.

Set temperature : nn.n°C
 actual temperature : nn.n°C
 error : nn.n°C
 heater Output : nn%.PS

Date : dd/mm/yy
 Time : hh-mm
 Sampling interval : nn ms

Controller Settings

- (i) Proportional gain : nn.n
- (ii) Integral action : nn.nns
- (iii) Derivative action : nn.nns

The above values on the display will be updated Every 5 seconds.

5. Operator Input:

The operator at any time can enter a new set point for the control parameters.

This can be done by pressing the "ESC" Key.

A menu is shown on bottom of display screen.

- | | |
|-----------------------------|-----------------------------|
| 1. Set temperature = nn.n | 4. Derivative action = nn.n |
| 2. Proportional gain = nn.n | 5. Sampling Interval = nn |
| 3. Integral action = nn.n | 6. Management Information |

6. Management Information

It gives the summary of the performance of plant over previous 24 hours will be given.

The summary provides the following information.

- a) Average Error in °C in 24 hour period
- b) For Each 15 minute period:
 - (i) Average demanded temperature
 - (ii) Average Error
 - (iii) Average heat demand
 - (iv) Date and time of Output

General Information:-

There will be a requirement for a maximum of 12 Control Units.

A single display and entry keyboard which can be switched between the units is adequate.

Preliminary Design:-

It consists of 2 types :-
Hardware design
Software design

1. Hardware design:-

There are various possibilities for hardware structure

Some of the arrangements are:-

- Single computer with multi channel ADC and DAC boards
- Separate general purpose Computer on Each Unit
- Separate Computer-based Microcontroller on Each unit linked to single general purpose Computer

Each of these configurations needs to be analysed and evaluated

Some points to consider are:-

Option 1:- System to be able to run with Sampling Interval for control loop of 40ms, Can this be met with 12 units sharing a single processor?

Option 2:- Is putting a processor that include a display and keyboard on each unit an expensive solution??

Option 3:- What sort of communication linkage should be used? A LAN?

Where should the microcontroller be located??

Each Option must be careful analysis and Evaluation in terms of Cost and performance.

2. Software Design

Examining the specification shows that the software has to perform several different functions:-

- DDC for temperature control
- Operator display
- Operator Input
- Management Information
- System shut down and start up
- Clock / Calendar function

Basic Software Modules:-

The various functions and type of time constraints is shown below.

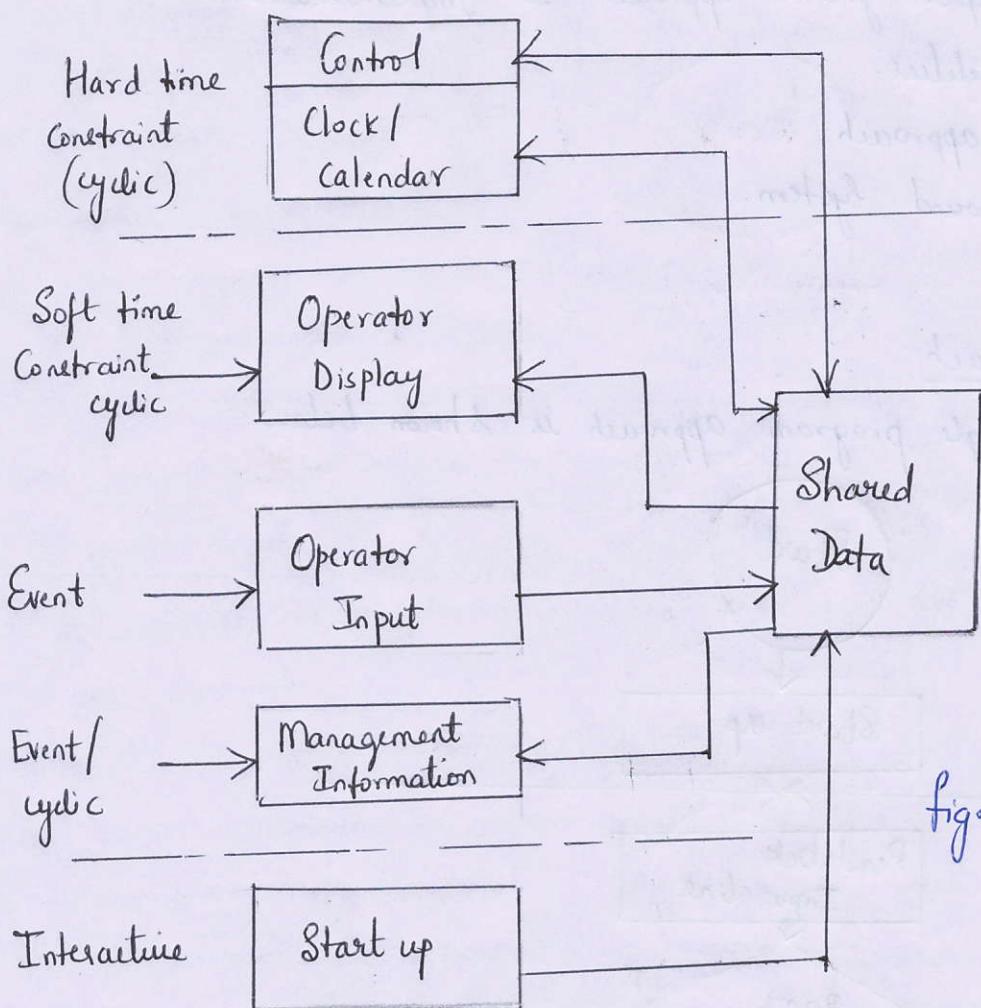


fig. Basic Software modules.

- The Control module - has a hard constraint ie, it must run at every 40ms. This constraint can be relaxed ie, 40ms±time, with an average value over 1 minute. In general this can be simplified as Tstes with an average value.
- The Clock/calendar module must run every 20ms. This constraint can be changed into soft constraint by adding additional hardware in the form of counter which can be read and reset by Clock/calendar module.
- Operator display is specified as a hard constraint that update at every 5sec. however maximum time have been specified say 10 seconds.
- The startup module does not have to operate in real time and hence can be considered as a standard Interactive module

The activities can be divided into sub-problems. These sub-problems will share certain information and design of next stages will depend upon general approach to implementation.

There are 3 possibilities.

1. Single program approach
2. Foreground / background system.
3. Multitasking.

Single program Approach

The flow chart of single program approach is shown below

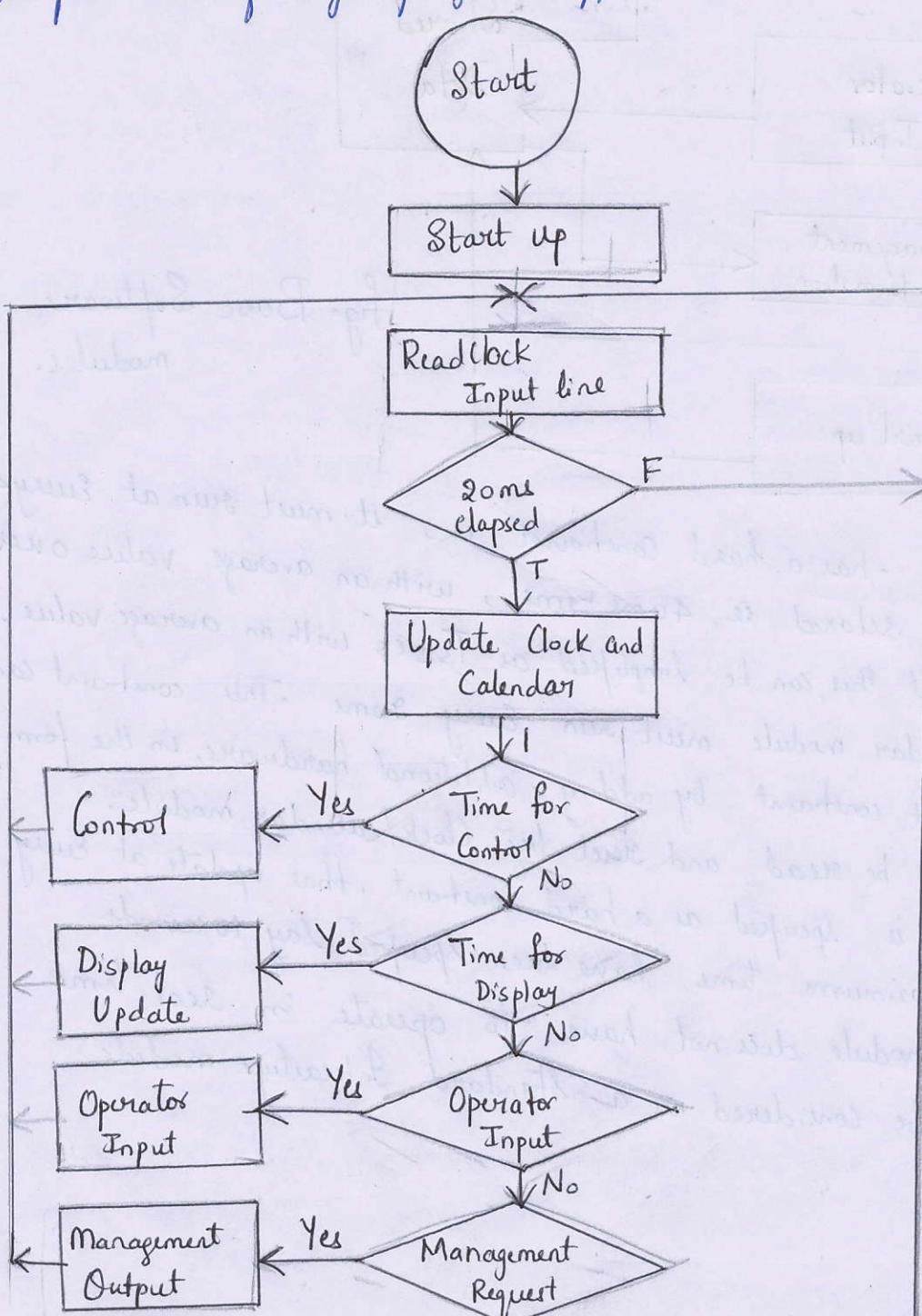


Fig. Single
Program approach

This structure is easy to program, it imposes most severe time constraints -

The requirements that Clock/Calendar module must run at every zone - on all of the module.

In this approach, the clock/calendar module and any one of other modules must complete their operations within zone.

If t_1, t_2, t_3, t_4 and t_5 are maximum computation times for modules Clock/Calendar, Control, Operator display, Operator Input and management Information respectively, then a requirement for system to work can be Expressed as,

$$t_1 + \max(t_2, t_3, t_4, t_5) < \text{Zone}.$$

The single program approach can be used for simple, small systems, such systems are usually easy to test.

Foreground / Background System.

Advantages : less module interaction, less tight time constraint.

The modules with hard time constraint are run in Foreground. The Foreground modules are considered to have highest priority than Background tasks.

The modules with soft time constraint (no constraint) are run in Background. The partitioning of foreground and background usually requires the support of RTOS.

The foreground / Background approach structure is shown in the figure.

→ A requirement for foreground part to work is that,

$$t_1 + t_2 < \text{Zone}$$

where t_1 = maximum execution time for
clock/calendar module.

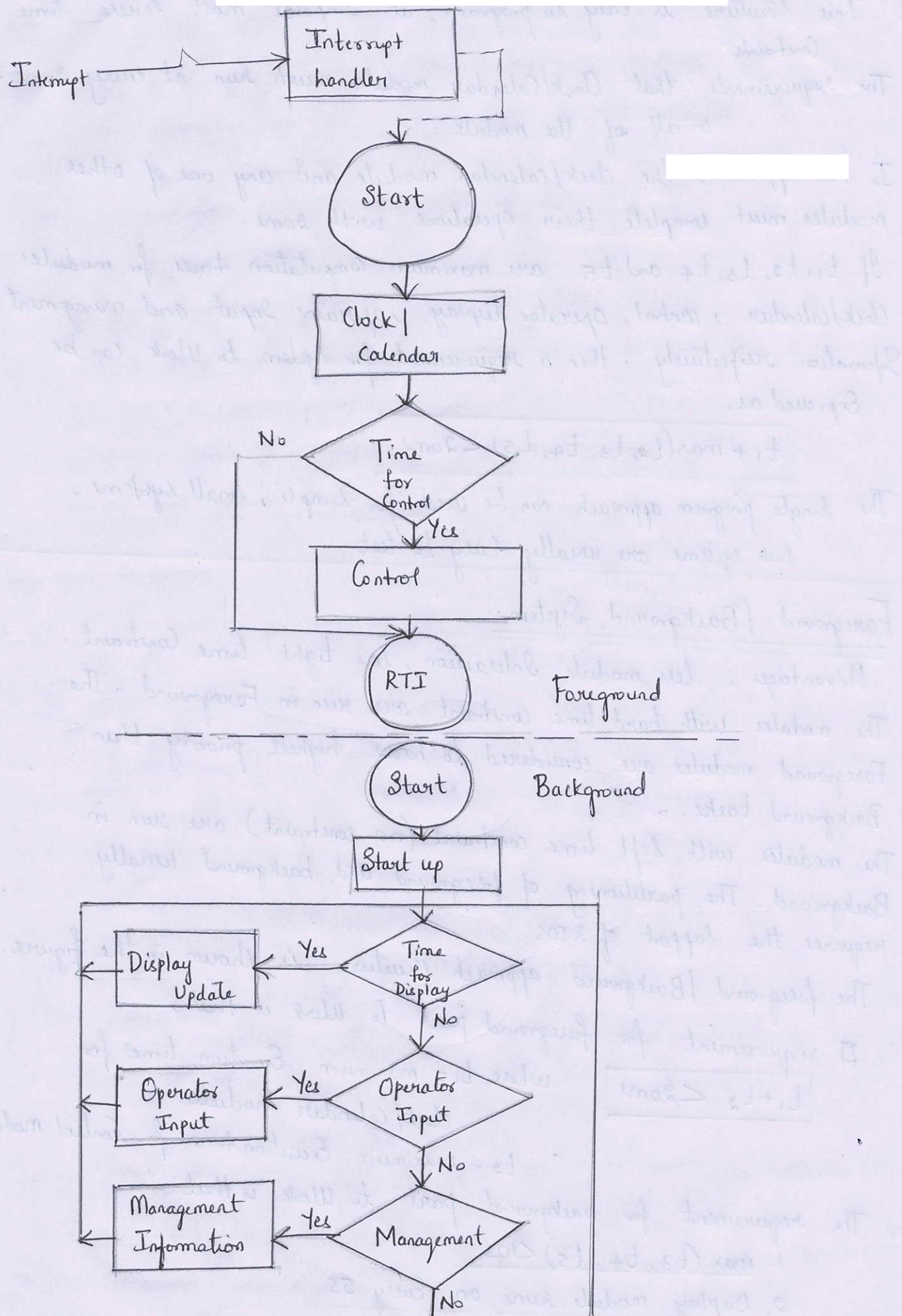
t_2 = maximum execution time for Control module.

→ The requirement for background part to work is that -

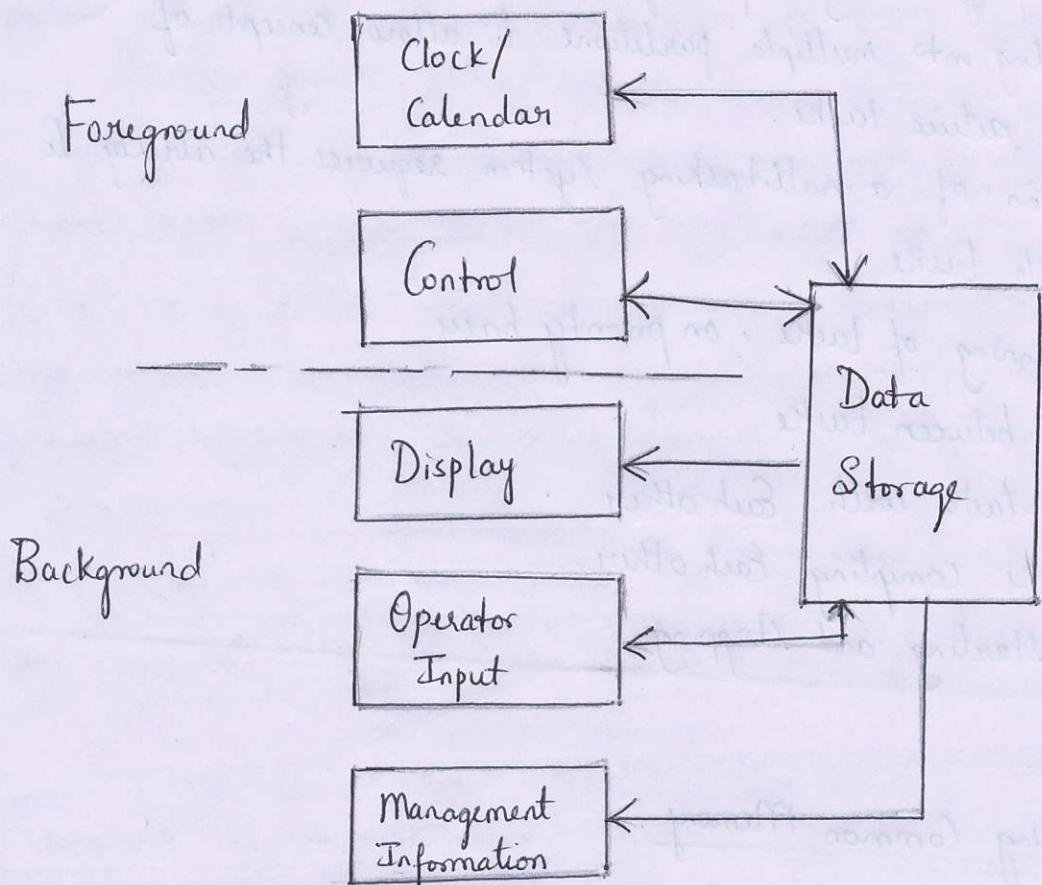
1. $\max(t_3, t_4, t_5) < 10s$

2. Display module runs on Every 5s

3. Operator input responds in < 10s



Software Modules foreground / background system Showing Data Storage.



The foreground / background approach separates the control structure, the modules are linked through the data structures as shown in figure. In single program there was no difficulty in controlling access to shared variables, since only one module was active at any one time. whereas, foreground / background may operate in parallel, i.e., one foreground & one background module may be active at same time.

- The variables can be shared between the Control, display and Operator Input modules without any difficulty since only one module writes to any given variable.

- ↳ The operator input module writes the controller parameters and set point variables
- ↳ The clock / calendar module writes to the date and time variables and
- ↳ Control module writes to plant data variables.

Multitasking approach

The design and programming of large Real time systems can be eased by extending into multiple partitions to allow concept of many active tasks.

The implementation of a multitasking system requires the ability to

- Create Separate tasks
- Scheduling running of tasks , on priority basis
- Share data between tasks
- Synchronize tasks with each other
- Prevent tasks corrupting each other
- Control the starting and stopping

Mutual Exclusion.

Data Sharing using Common Memory.

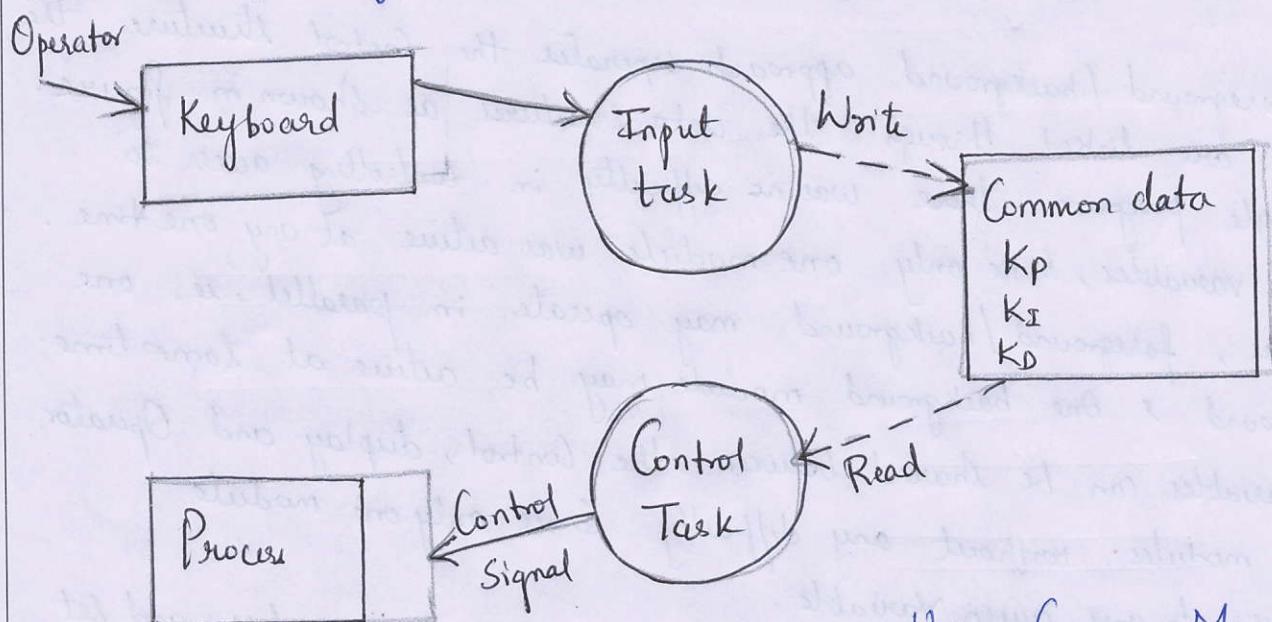


fig. Data Sharing Using Common Memory .

Consider the transfer of information from an Input task to Control task as shown in fig.

The Input task gets the value for proportional gain, Integral action time and derivative action time.

From there it computes the controller parameters K_p , K_i and K_d and these

are transferred to Control task.

This is done by holding parameter values in an area of memory which has been declared as being common and accessible to both tasks. If Input task writes it into common data, without accessing Exclusive rights, there is a danger that Control task will read one new value, say K_p and 2 old values K_d and K_i .

Condition flags :-

D. Explain attempt at Mutual Exclusion using Condition flags with Example.

Condition flag is a method of indicating if a resource is being used or not. It is a flag variable which set to TRUE or FALSE. 0 or 1, set or reset.

If flag is FALSE/0/Reset - then resource is available

If the task sets flag TRUE/1/set - and uses the resource

↳ Tasks A and B are shown in figure, where both share a printer.

It is assumed that the flag variable PrinterInUse, which is set to 1, when the printer is in use and 0 when it is available.

→ Task A checks PrinterInUse flag and finds printer is available, but before it executes next instruction, which would set the PrinterInUse flag to 1, and hence claim the printer. the dispatcher forces a task to & status changes and task B runs.

↳ Task B also wishes to use the printer and checks the flag.

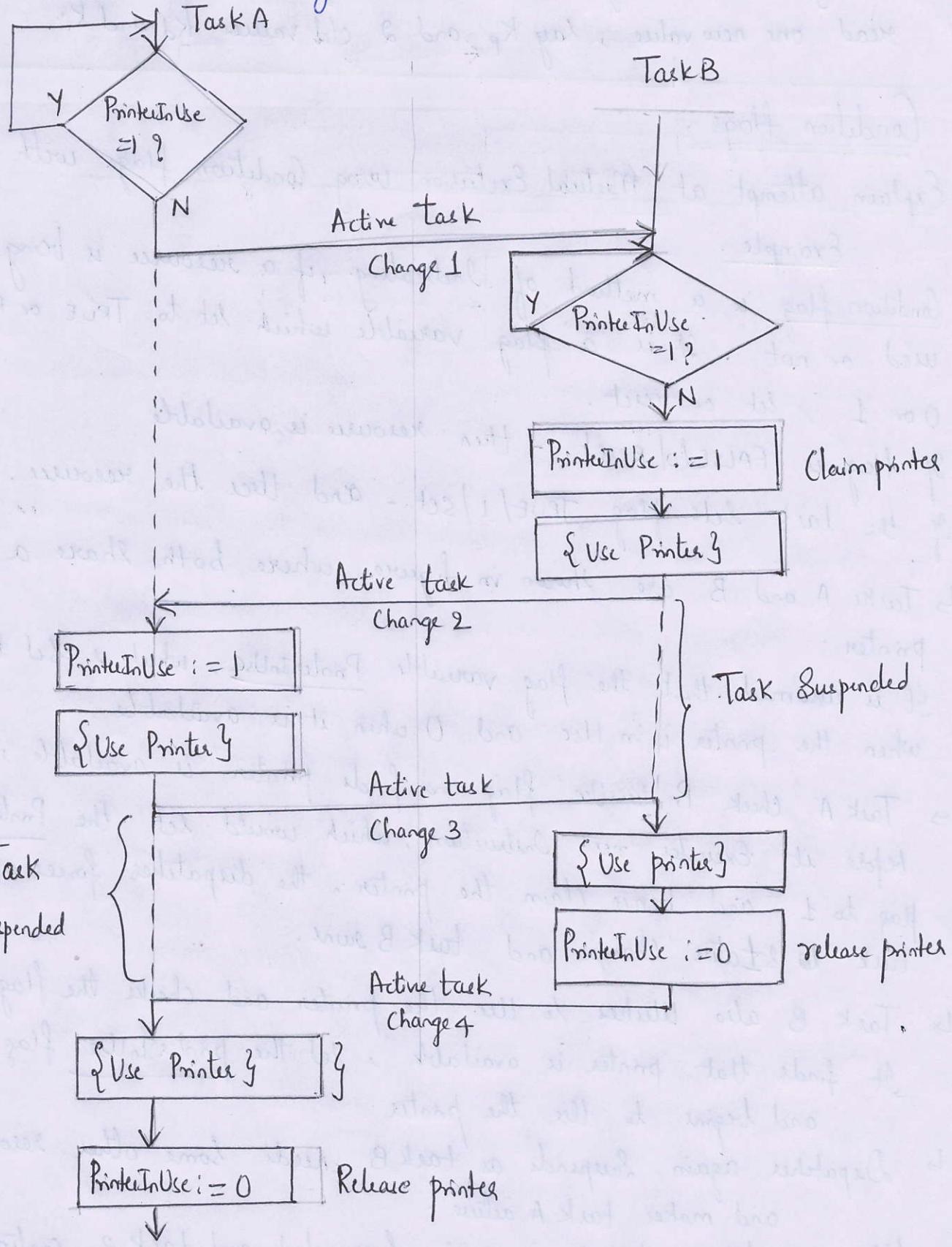
It finds that printer is available, set the PrinterInUse flag to 1 and begins to use the printer.

↳ Dispatcher again suspends as task B needs some other resources and makes task A active.

↳ After some time task A is again suspended and task B continues,

it now finishes with printer and releases it by setting PrinterInUse to 0, making printer available to any task.

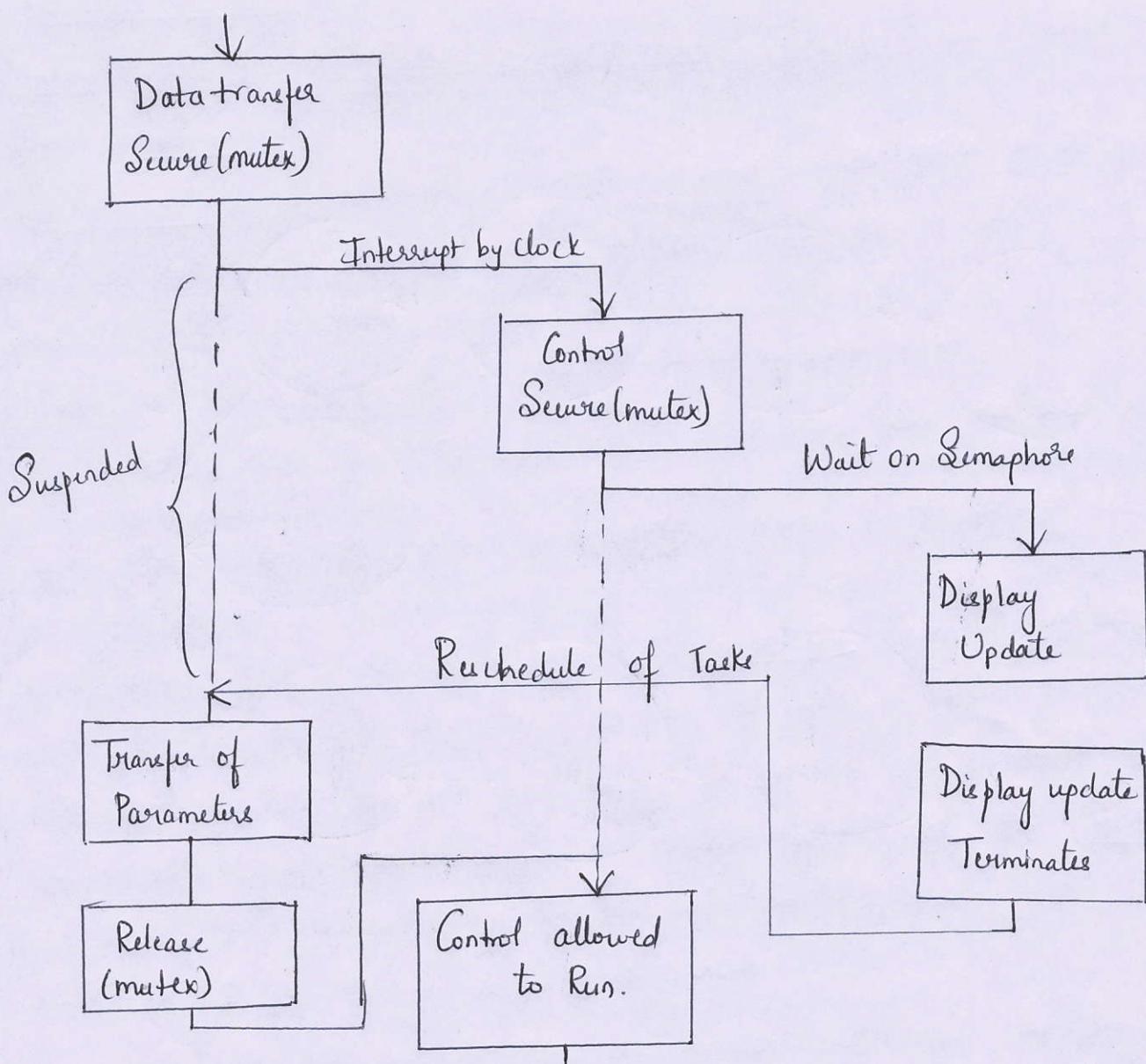
- At task change 4, task A again uses the printer and eventually releases it although it has in fact already been released by task B.



Semaphore

The Semaphore is a Turn flag technique, which gives information about resource is free or in use, and which task is executed next. The problem with this network is that the tasks must run in strict sequence.

- Q. Explain Transfer of Controller parameters - Using Semaphore with Example.



In the figure, the Data transfer Task has just Secured the Semaphore mutex, at which time the clock interrupt forces a rescheduling of tasks and Control task runs.
 → The Control task, will be suspended when it attempts to Execute

Service and reschedule takes place.

- ↳ Display Update task is ready to run, because it is having highest priority than Data Transfer.
- ↳ The task Data Transfer cannot run until Display Update Suspend or finishes running, and Control task cannot run until Display Update has run and release Mutex.