DESIGN OF ELECTRONIC CONTROLLER FOR PROCESS INDUSTRIES

A DISSERTATION
Submitted in partial fulfilment of the
requirements for the award of the degree
of
MASTER OF ENGINEERING
in
MEASURMENT & INSTRUMENTATION

By
R. K. MEHTA

DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE-247 672 (INDIA)
(AUGUST 1979)

C E R T I F I C A T E

Certified that the dissertation entitled "DESIGN OF ELECTRONIC CONTROLLER FOR PROCESS INDUSTRIES" which is being submitted by Shri R.K.Mehta in partial fulfilment of the requirements for the award of the degree of Master of Engineering, in 'Measurement and Instrumentation' of Electrical Engineering of the University of Roorkee, Roorkee, is a record of bonafide work carried out by him under our supervision and guidance. The matter embodied in this dissertation has not been submitted for the award of any other degree or diploma.

This is further certified that he has worked for the period of 12 months from July 1981 to June 1982 for preparing this dissertation at MECON (I) Ltd, Ranchi.

6/8/82

(Dr. S.C. Saxena)
Internal Guide
Reader in Elect.Engg.Deptt.
University of Roorkee
Roorkee

P.S.: A certificate from external guide Sri O.N.Krishnan,C.D.E. MECON(I) Ltd., Ranchi is on following page.

मेकन
मैटलर्जिकल एण्ड इंजिनियरिंग कन्सल्टेन्ट्स (इंडिया) लिमिटेड
राँची–834002, बिहार, इंडिया, तार : मेकन—टेलेक्स : 024—209
METALLURGICAL & ENGINEERING CONSULTANTS (INDIA) LTD.
RANCHI-834002. BIHAR, INDIA, GRAM : MECON—TELEX : 024—209

O.N.KRISHNAN
CHIEF DESIGN ENGINEER.

"TO WHOMSOEVER IT MAY CONCERN"

Certified that the dissertation entitled "DESIGN OF ELECTRONIC CONTROLLER FOR PROCESS INDUSTRIES", which is being submitted by Shri R.K.Mehta of our Organisation in partial fulfilment of the requirements for the award of the degree of Master of Engineering in Electrical Engineering (Measurement and Instrumentation) of the University of Roorkee, Roorkee, is a record of bonafide work carried out by him under my supervision and guidance. The matter embodied in this dissertation has, to the best of my knowledge and belief, not been submitted for the award of any other degree or diploma.

It is further certified that he has worked for the period of 12 months from July,1981 to June, 1982 at MECON (India) Ltd.,Ranchi for preparing this dissertation. The software developed by him, suitable for a typical controller, was successfully demonstrated on the microcomputer available at the Electro-Technical Laboratory of our organisation.

O.N.Krishnan

Ranchi,
July 12, 1982.

(O.N. KRISHNAN)
Chief Design Engineer
METALLURGICAL & ENGINEERING
CONSULTANTS (I) Ltd.
RANCHI-834002

ACKNOWLEDGEMENT

The author with a deep sense of gratitude expresses his sincere thanks to Dr. S.C.Saxena, Reader in Electrical Engineering Department, University of Roorkee for his constant encouragement and invaluable guidance during the dissertation work. The keen interest taken by him throughout the work, the timely suggestions and liberal allowance of the time needed for the critical discussions are most thankfully acknowledged.

The author is highly indebted to Shri O.N.Krishnan, Chief Design Engineer, MECON (I) Ltd., Ranchi for his invaluable suggestions, guidance and constant moral encouragement in preparing this dissertation.

At last, but not least, my sincere thanks go to one and all who have contributed their efforts in preparing this dissertation in one way or other.

R. K. MEHTA

C O N T E N T S

.....

CHAPTER - 1

INTRODUCTION

As the complexity of industrial processes has increased, there has been a consequent increase in the number of process variables (such as temperature, pressure, flow, pH) to be controlled, and it has become increasingly evident that further development would be difficult or even impossible without the aid of devices which would automatically measure and control at least some of these process variables. Automatic control does not replace the human operator but rather supplements him. Automatic control of at least some of the process variables allows him to concentrate upon the parts of the process which require his special skill. In the absence of instruments which directly measure and control the quality of product, it is necessary to control variables such as temperature, pressure, pH etc., at values which experience has shown result in the greatest safety of the operator and the highest quality and quantity of product.

The automatic control can be achieved by different ways and means. The controllers can be classified broadly by two ways-by its principle of control or by its mode of control i.e. the control criterion. If the control action is achieved by way of pneumatic mechanism, it is called pneumatic controller and if liquid is used as means, the controller is known as hydraulic. Similarly electrical and electronic controllers also exist. There is very little

~~little~~ difference between electrical and electronic controllers. The electrical ones use passive components, like resistors, capacitors etc. and hard wired relay logic while semiconductor devices like transistors, ICs are used in the electronic ones. Technological difference between the two may be there but one thing in both of them is common - that is the transmission of signal is electrical. Apart from these controllers, there can also be electro-pneumatic and electro-hydraulic controllers These are the hybrid types.

On the basis of control criterion, the controllers can be classified into proportional type, proportional plus integral type and proportional plus integral plus derivative type. One important class has also been added to this serves and that is adaptive control, which has been discussed later in the dissertation.

Electric and electronic control has got an edge over the others due to its following advantages [2]

- No time lags, no transmission delay
- Linear control response and greater accuracy
- Control medium uneffected by dirt,grease and foreign particles
- Wider ambient temperature operating range and
- Control units and functions easily integrated and adapted to interconnecting systems.

Control of unknown systems has been a challanging and difficult problem for a long time. Once the PID controller structure was invented and its tuning considered, major advances have been hard to comeby. Only recently have significant practicable contributions been made. A large number of new electronic components have been developed during recent years. Among these are the microprocessors and semi-conductors of the types RAM (Random Access Memory) and PROM (Programmable Read Only Memory) which have made it possible to realize desired functions in the form of programs instead of using hard-wired electronics or relay technique. Electronic technology has provided increasingly powerful arrays of programmable control systems that can perform extremely sophisticated operations which would not be practical using pneumatic or electro-mechanical logic devices. The cost [1] of this technology has declined to the extent that a $ 5 microprocessor (1981 price) can now take the place of several cabinets full of moving-part logic devices, costing thousands of dollars, all while enhancing the system's inherent reliability and flexibility for change. Indeed, the cost of electronic logic is usually dwarfted by the cost of the power supplies and power drivers necessary to complete a working system.

Chapter 2 of this dissertation deals with different type of control actions like P, PI, PID etc., and the most modern addition viz.Adaptive controllers classified on the basis of their control media, such as pneumatic, hydraulic, electrical/electronic etc., have been discussed in Chapter 3. The recent trend in electronic controllers viz microprocessor -

based controllers has been dealt-with in a separate section 3.4 of this Chapter, keeping in view of its increasing importance in industry and due to the fact that this is the type of controller the design of which has been discussed in this dissertation. Chapter 4 of this dissertation includes the design of a typical microprocessor-based adaptive position controller. The software program which was developed on the basis of design presented in Chapter 4, and subsequently tested and demonstrated on the available microcomputer has been given in Annexure I.

CHAPTER – 2

CONTROL ACTIONS

The nature of the change in the output from an automatic controller in response to a deviation of the controlled condition from the desired value will depend upon the type of controller. The nature of the action of a controller may be more clearly understood if the thoughts and actions of a highly skilled operator are studied. Suppose he is given the task of controlling the temperature of the water in a tank through which water is flowing at a constant rate. To assist him in his task he is provided with a temperature indicator and recorder which enable him to access the success, or otherwise, of his efforts. The water is heated by passing steam through a heating coil at the bottom of the tank as shown in Fig. 2.0 and the operator varies the flow of steam by means of a valve and attempts to control the water at 80°c [3].

2.1 ON-OFF CONTROL

Suppose the steam-flow control valve has two positions only : Position – I in which the supply of steam is insufficient to maintain the temperature of the outgoing water at 80°C, so that with the valve in this position the water temperature will fall; and position – II in which the supply of steam is more than sufficient to maintain the water temperature at 80°C so that with the valve in this position the water temperature will rise above 80°C. If position I is such that the valve is closed and no-steam flows, the control represents the special case of two-step control namely on-off control.

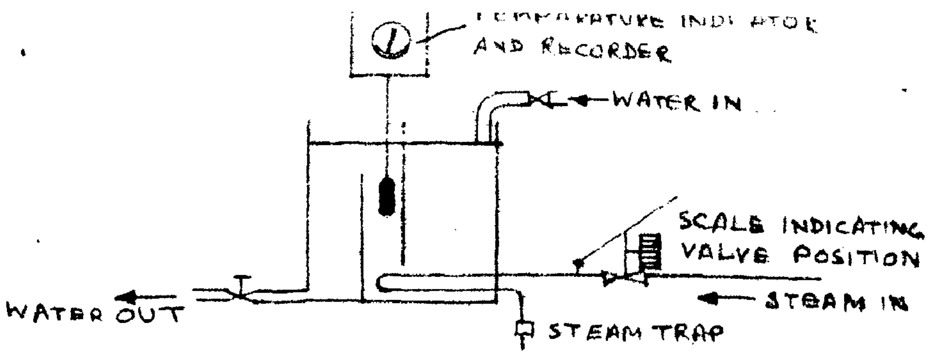


FIG. 2.0 A SIMPLE PROCESS

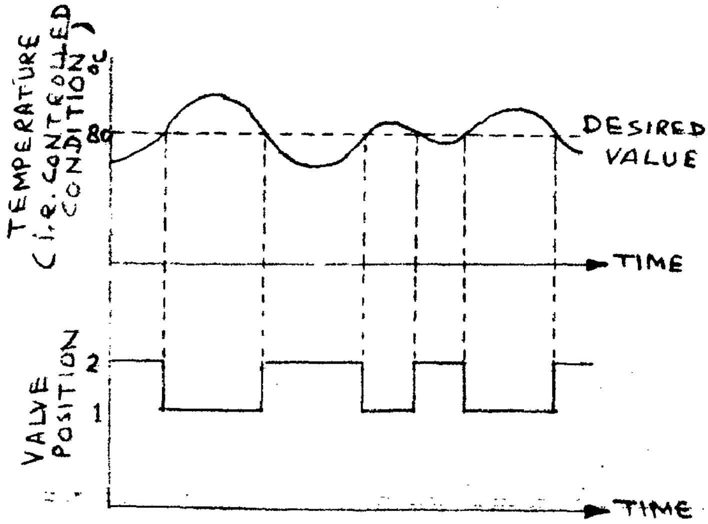


FIG. 2.1. ON-OFF ACTION

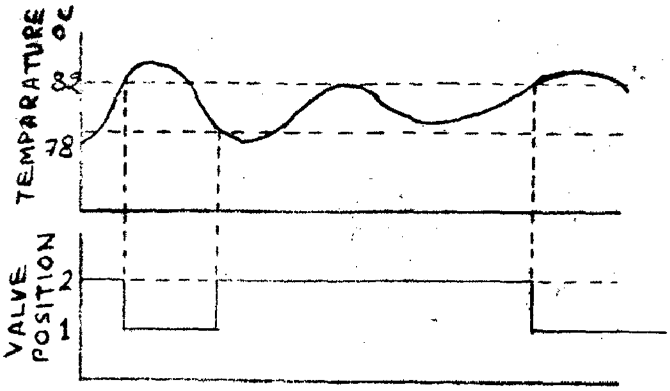


FIG 2.2 ON-OFF ACTION WITH OVERLAP

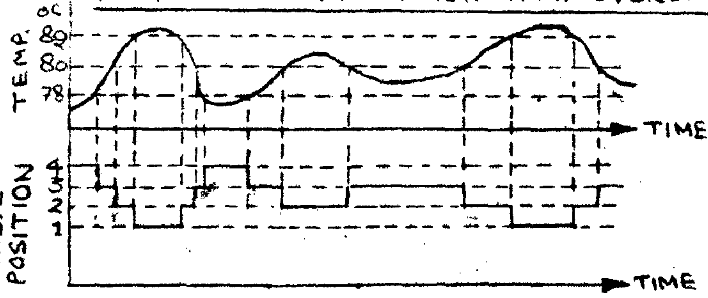


FIG. 2.3 MULTI STEP ACTION

When the temperature of the water is below 80°C, the operator will select valve position - II and the temperature of the water will rise. He will watch the temperature indicator and when it shows that the temperature has reached 80°C, he will move the valve to position (1). It will be immediately obvious that the quality of control will depend upon the accuracy of the temperature indicator. No amount of skill on the part of the operator will compensate for inaccuracy in the measuring unit.

With the valve in position (I), the supply of heat will be insufficient to maintain the temperature of the water and the temperature will fall. Before the temperature of the water can fall, however, all the heat stored up in the steam coil must be given up to the water, so that although the valve is in position (I), the temperature of the water will continue to rise and may reach, say 82°C.

Again, when the temperature of the water has fallen to 80°C, the operator will move the steam valve to position (II), but owing to the fact that an initial supply of heat is used up in heating the steam coil, the temperature will fall to 78°C before it begins to rise again. The position of the valve and the temperature response of the water will be as shown in Fig.2.1 and the response is described as hunting or oscillating.

If the quantity of water flowing through the tank is increased, more heat will be required to maintain the temperature at the same average value, so that the proportion of the time for which the value is in position (II) will be increased, but the operator is still able to maintain an average temperature of 80°C.

The closeness of control that the operator can achieve will depend upon many factors. In the first place it will depend upon hour readily the temperature indicator responds to change in temperature of the water, as he will only take action when the measured temperature crosses the desired value, i.e. it will depend upon the measurement lag.

In the second place, it will depend upon the time it takes him to see that the temperature has crossed the desired value and for him to move the control valve in the appropriate direction, i.e. upon the lags in the controlling and correcting unit.

It will also depend upon the heat capacity of the tank. If the tank contains a large quantity of water the temperature of the water will change slowly, and there will be only a small temperature change while he moves the valve. Also, the heat stored in the heating coil after the steam flow has been reduced will not produce an appreciable rise in the temperature of the water. Similarly, the quantity of heat required will be more to raise the temperature of the water by 1°C. Thus the range through which the temperature will oscillate will be small. Two-step control will, therefore, give good results if the demand side capacity (e.g. the thermal capacity of the tank and water) is many times larger than the supply side capacity (e.g. the thermal capacity of the heating coil) and the lags in the complete loop are small. Statistics show that this form of control is adequate for a large proportion of all control applications.

When the time required for the steam to get into the coil and for the heat to get from the coil to the water is large(i.e. the process has transfer lags) the temperature may oscillate violently if the capacity is small. The valve will, therefore, have to be moved very frequently involving the operator in a great deal of work and resulting in considerable wear on the valve mechenism. The frequency of change of valve position may be reduced, by introducing a hysteresis between the temperatures at which the valve is moved to position (1) and that to position(2). Instead of the operator moving the valve everytime the temperature crosses 80°C, he would move the valve to position (1) when the temperature rises to 89°C and position (2) when the temperature falls to 78°C. This type of control is described as two-step action with overlap, and the position of the valve and the valve of the controlled condition i.e. temperature will be as shown in Fig. 2.2.

If the operator has the choice of more than two positions of the control valve, and he moves the valve into those positions at predetermined values of the temperature, then the controller action which he simulates is described as multistep action, and the position of the valve and the value of the controlled condition will be as shown in Fig. 2.3.

2.2 PROPORTION CONTROL

In order to eliminate the hunting action which may result from ON-OFF control, the operator can move the valve a distance which is proportional to the deviation of the temperature from the derived value. Suppose the valve is provided with a scale having 20 divisions, the valve is half open at 80°C and he moves the valve one division for each degree deviation. The valve will be fully open when the indicator shows 70°C and fully closed at 90°C.

Suppose the temperature is 76°C. The operator will have opened it the additional four divisions as the deviation is 4°C. So that the valve will be 10/20 + 4/20 = 14/20 open. As the temperature rises he will close the valve so that it is 13/20 open at 77°C, 12/20 open at 78°C, 11/20 open at 79°C, and half open at 80°C. Owing to the heat already given to the water, however, the temperature will continue to rise, and the operator will continue to close the valve so that if the temperature reaches 83°C. The valve will be only 7/20 open, and the supply of heat will be insufficient to maintain the temperature which will consequently fall.

As the temperature falls, the valve will be slowly opened so that at 80°C the valve will be half open and at 78°C it will be 12/20 open. This proportional movement of the valve will gradually damp out the temperature oscillations and will result in stable control as shown in Fig.2.4.

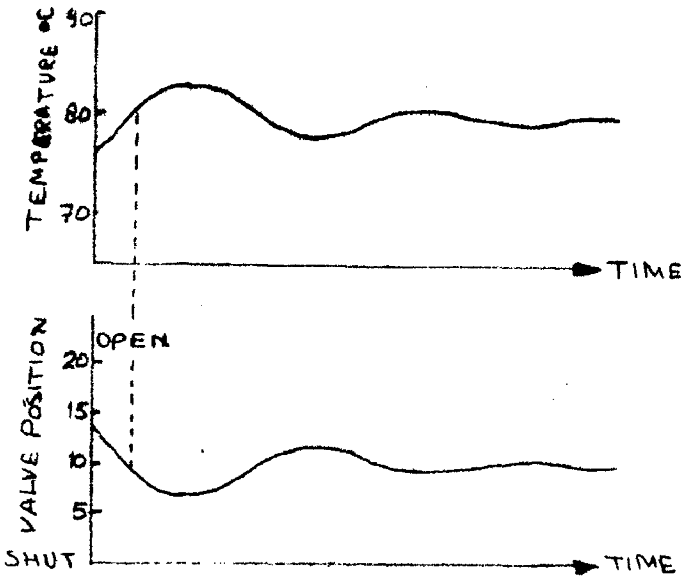


FIG 2.4 PROPORTIONAL ACTION

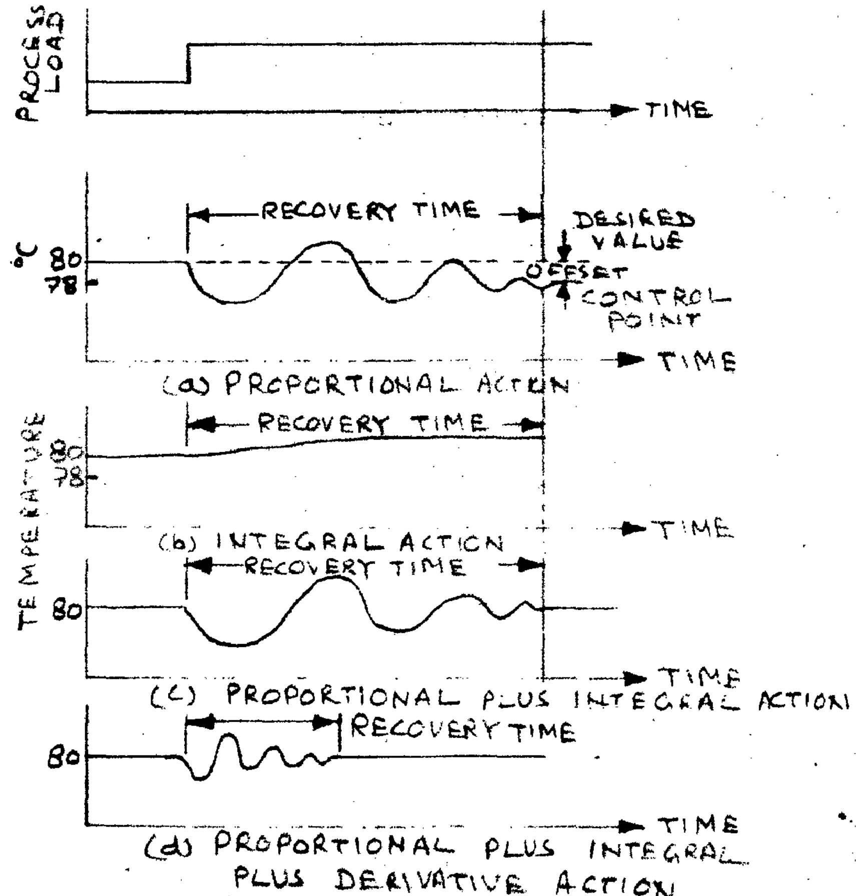


FIG 2.5 EFFECT OF CONTROLLER ACTIONS

The amount the measured value of the controlled condition must change in order that the valve may be moved from the fully closed to the fully open position is called the proportional band. In the case of automatic controllers it is usually expressed as the percentage of the instrument's full scale. If the scale of the instrument is 0 - 100°C and the temperature must change from 70°C to 90°C for the valve to be moved from fully open to fully closed position the proportional band is

2/100 x 100 = 20 percent

The proportional band is made adjustable in order to provide stable control under different process conditions.

Suppose the operator has obtained stable conditions but the flow of water through the tank increases. With the valve half open the supply of heat will be insufficient to maintain the temperature at 80°C. As the temperature falls, the operator will open the valve one division for each degree of deviation. The temperature may oscillate as before but at a lower value, and will finally stabilize out at a temperature, known as the control point, which is below the desired value. This difference between the equilibrium temperature and the desired value is called the 'offset'.

This sustained deviation or offset exists because, owing to the increased flow, a large-quantity of heat must be supplied in order to obtain a steady temperature near to the desired value. Suppose the valve opening required is 12/20. The valve position is determined by the deviation. The deviation must be $-2^{\circ}C$, if the valve is to be opened further 2/20. The temperature, therefore, will after oscillating stabilize out at $78^{\circ}C$, as shown in Fig.2.5(a). In a similar manner, if the flow through the tank is decreased, the heat provided with the valve half open will be more than enough to maintain the temperature at $80^{\circ}C$, and after some oscillations the temperature will stabilize out at a control point which is above $80^{\circ}C$. In other words, there must be an offset if the value is to be maintained in any position other than half open, so that proportional response cannot maintain a constant temperature under conditions which require different valve positions in order to hold the same desired value.

In order that allowance may be made for load changes, many controllers have 'manual reset'. This adjustment enables the operator to adjust the output of the controller when the controlled condition is at the control point and so eliminate offset. When a further load change occurs, however, a further adjustment of the reset control will be required if there is to be no offset. In effect, the manual reset control alters the valve position for the desired value.

Whenever different positions of the correcting element are required for the same value of the controlled condition, offset will be produced and the value of the offset produced will depend upon the extent of the load change and upon the width of the proportional band. In order to keep the offset as small as possible, the band width should be made as narrow as possible, but if it is made too narrow hunting will occur. The permissible width of the proportional band will depend upon the process reaction rate. Small process reaction rates permit the use of narrow proportional bands so that the offset may be negligible and proportional action alone will provide satisfactory control.

In the limiting case, when the process reaction rate becomes extremely small, an extremely narrow or zero proportional band may be used, in other words a two-step action. Two-step action may be regarded as proportional action with an extremely narrow proportional band.

2.3 INTEGRAL CONTROL

Faced with the problem of offset caused by increased flow, the operator will open the valve slowly in order to re-establish the control point at the desired value of 80°C, the longer the deviation persists the more he will open the valve. If he moves the valve an amount which depends upon the size of the deviation and the time for which it persists, or moves the valve at a rate which is proportional to the deviation. This type of action is described as integral action. The effect of integral action upon the control point is shown in Fig.2.5(b).

If the operator could apply both proportional and integral actions simultaneously, the temperature may oscillate as before, but the tendency for the temperature to stabilize at a control point of 78°C owing to proportional action(Fig.2.5(a)) would be eliminated by the integral action, (Fig.2.5(b)) so that the control point would finally concide with the desired value of 80°C as shown in Fig.2.5(c). The most satisfactory combination of these two actions will be achieved if the time taken for the integral action to make the control point concide with the desired value is equal to the time taken for the temperature oscillations owing to proportional control to be damped out.

2.4 DERIVATIVE CONTROL

After very considerable experience of the plant, the operator will become very skilled in its operation and thoroughly understand its behaviour and be able to analyse the record which the temperature recorder is producing. Each time the controlled condition deviates from the desired value, he will take the appropriate action. If the value of the controlled condition is increasing slowly, he will close the control valve the small amount required to counteract the téndency. If the rise is sharp he will close the valve very much more. If the temperature has a tendency to fall he will open the control valve an amount which depends upon the rate at which the temperature is falling.

In other words, in addition to moving the valve in a manner which simulates proportional and integral action, he moves the valve an amount which depends, not upon the size of the deviation, but upon the rate at which the deviation is changing. In moving the valve in this manner he is simulating 'derivative action'. The effect of the addition of dativative action is shown in Fig.2.5(d). It will be seen that the recovery time and size of the disturbance are considerably reduced. It is very important to realise, however, that derivative action cannot be used alone, for derivative action will produce a change in valve position only while the deviation is changing. When the deviation is constant, it does not matter how large it may be, derivative action will produce no change in the valve position.

Derivative action has very important applications in processes which have large lags. It does not require the existance of a large deviation in order to produce a marked valve movement. As soon as the deviation starts, derivative action will apply a controller action which tends to remove the deviation even before it becomes large enough to show up on the control.

2.5 RESPONSES OF THREE TERM CONTROLLERS

2.5.1 Proportional Action

Proportional action is defined as the action of a controller the output signal V of which is proportional to the measured deviation θ. Thus

$$V = - K_2 \theta$$

K_1, the 'proportional action factor' is proportional to

$$\frac{1}{\text{Band width of the controller}} \quad \text{or} \quad \frac{100}{\text{Percentage proportional band}}$$

The negative sign appears in the equation because when θ increases, V increases in the opposite sense.

This type of control results in offset because V will be zero when θ is zero i.e. a sustained deviation is necessary where a valve position other than the preliminary setting is required.

If K_1, the proportional action factor is very high or the bandwidth of the controller is extremely narrow, even a small deviation of the controlled variable from the control point will make the controller output saturated. The sign of V will change whenever θ changes its sign i.e. whenever measured variable crosses control point. Hence it will work as an ON-OFF controller.

2.5.2 Integral Action

The action of a controller, rate of change in the output signal of which is proportional to the measured deviation

$$\frac{dV}{dt} = - K_2 \theta \quad or, \; V = -K_2 \int \theta \, dt$$

where K_2 is the 'integral action factor' of the controller i.e. the change of controller output signal is proportional to the time integral of the measured deviation. Thus for a constant deviation in this type of action the change of output signal will be proportional to the time, i.e. the longer the deviation lasts the greater will be the valve movement produced by the controller.

2.5.3 Derivative Action

The action of a controller, the output of which is proportional to the rate at which the measured deviation changes. Thus

$$V = - K_2 \frac{d\theta}{dt}$$

K_3 is the 'derivative action factor ' of the controller.

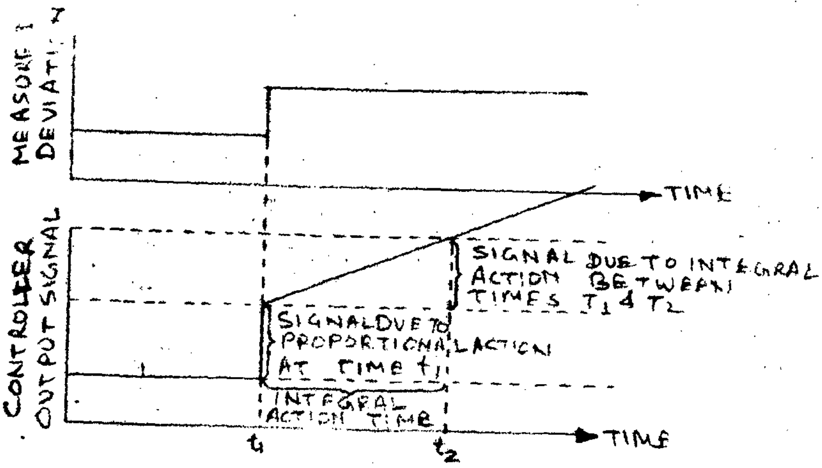


FIG. 2.6 INTEGRAL ACTION TIME

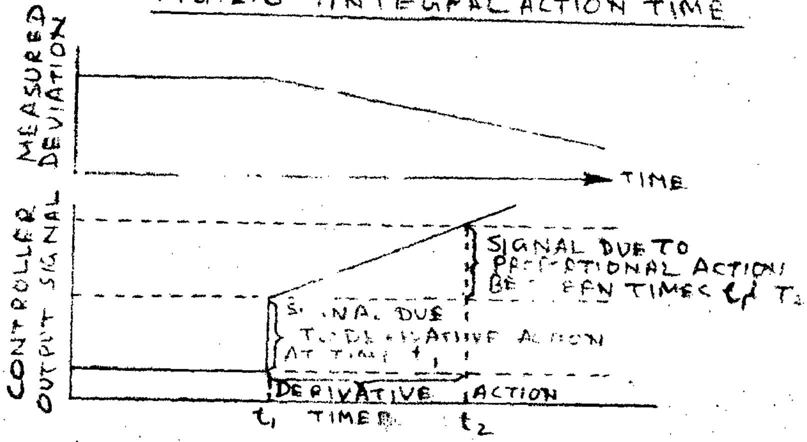


FIG. 2.7 DERIVATIVE ACTION TIME

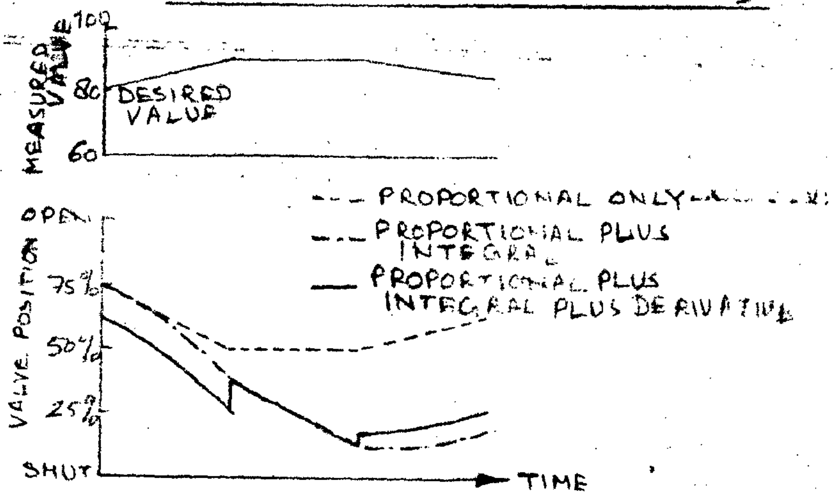


FIG. 2.8 RESPONSE OF A THREE TERM CONTROLLER TO AN ARTIFICIAL DISTURBANCE

When Θ is constant, $d\Theta/dt = 0$, Thus in this type of action no valve movement is produced if the deviation is constant, so that derivative action cannot be used alone as there will be no action however large the deviation, provided it is constant.

2.5.4 Compound Action

A controller action in which the output signal from the controller is the result of more than one operation on the measured deviation.

It should be realized that compound action consists of the simultaneous application of the two or more control actions. Integral action is applied simultaneously with proportional action in order to eliminate 'offset'. Derivative action is applied simultaneously with proportional or with proportional plus integral action in order to reduce the time required for a disturbance to be smoothed out and also to reduce the size of the deviations produced by a disturbance.

Compound actions may be represented by the following equations.

Proportional Plus Integral Action

$$V = -K_1 \Theta - K_2 \int \Theta \, dt$$
$$= -K_1 \left(\Theta + \frac{K_2}{K_1} \int \Theta \, dt\right)$$

The integral action time t_1 of the controller is defined as the time interval in which the part of the output signal due to integral action increases by an amount equal to the part of the output signal due to proportional action when the deviation is unchanging. If θ is constant, owing to proportional action

$$V = - K_1\theta$$

Owing to integral action

$$V = -K_2\int\theta\, dt$$

If these actions are equal at a time t_1 see

$$-K_1\theta = -K_2\theta\int_0^{t_1} dt = - K_2\theta\, t_1$$

$\therefore$ Integral action time,

$$t_1 = \frac{K_1}{K_2}$$

$$V = - K_1(\theta + \frac{1}{t_1}\int\theta\, dt)$$

Proportional Plus Derivative Action

$$V = -K_1\theta - K_3\frac{d\theta}{dt}$$

$$= -K_1(\theta + \frac{K_3}{K_1}\frac{d\theta}{dt})$$

The derivative action time t_2 of a controller is defined as the time interval in which the part of the signal due to proportional action, in a controller having proportional plus derivative action, increases by an amount equal to the part of the output signal due to derivative action, when the deviation is changing at a constant rate.

Suppose deviation is changing at a constant rate. The derivative action will immediately produce a change in the output signal of $-K_3(d\theta/dt)$ which will remain constant as the deviation is changing at a constant rate.

Suppose the proportional action signal is equal to the deviative action at a time t_2. At time t_2

$$\text{Deviation} = \int_0^{t_2} \frac{d\theta}{dt}\, dt = \frac{d\theta}{dt}\int_0^{t_2} dt$$

$$= t_2 \frac{d\theta}{dt} \text{ (as } \frac{d\theta}{dt} \text{ is constant)}$$

$\therefore$ At time t_2, $-K_1 t_2 \frac{d\theta}{dt} = -K_3 \frac{d\theta}{dt}$

$\therefore$ Derivative action time $t_2 = \frac{K_3}{K_1}$

$$V = -K_1(\theta + t_2 \frac{d\theta}{dt})$$

Proportional plus Integral plus derivative Action

Where the controller produces three controller actions simultaneously,

$$V = -K_1\theta - K_2 \int\theta\, dt - K_3 \frac{d\theta}{dt}$$

$$= -K_1(\theta + \frac{1}{t_1} \int\theta\, dt + t_2 \frac{d\theta}{dt})$$

The response of this type of controller is shown in Fig. 2.8.4~~.3.~~

2.6 ADAPTIVE CONTROL

Adaptive control is one in which automatic and continual measurement of the process to be controlled is used as a basis for the automatic and continuing self-design of the control system [4]. Adaptive Control systems are characterized by the inclusion of a group of components specifically inserted for the automatic and frequent measurement of process dynamics (or at least specific aspects of process dynamics) and another group for the automatic adjustment of controller characteristics.

Since adaptive control involves a measurement of process dynamics in at least some form, a computer is frequently required, hence, adaptive control systems are often specific types of computer controlled systems.

2.6.1 The Essential Components of Adaptive Control

Two components which always appear in some form in adaptive control are (1) identification and (2) actuation. Identification refers to the measurement of the dynamic transfer characteristics of the process to be controlled and actuation signifies the generation of an appropriate actuating signal as the process input.

The identification problem, of essential importance in any approach to control-system design, becomes a central element of adaptive control since adaptivity implies automatic, frequent and rapid solution of the identification problem. Once the identification problem is solved, the results of

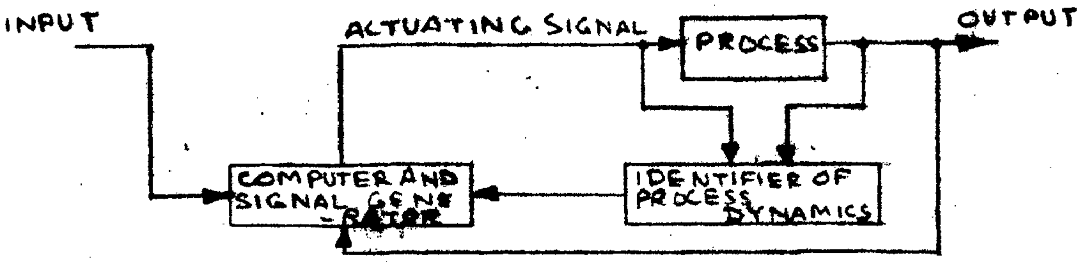


FIG 2.9 ADAPTIVE CONTROL WITH DIRECT ACTUATION

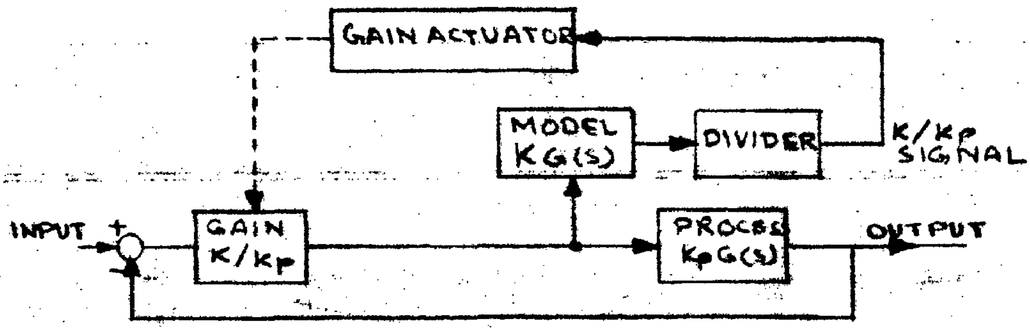


FIG. 2.10 ELEMENTAL ADAPTIVE SYSTEM

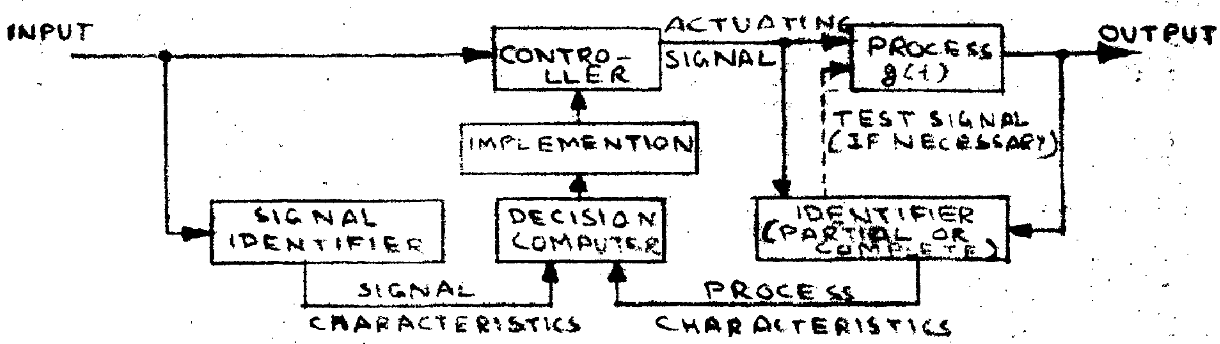


FIG. 2.11 GENERAL ADAPTIVE SYSTEM

identification are utilised to establish automatically the controller transfer characteristics and hence the way in which the system error is modified before it is used to drive the process.

If the overall system is to be operable in the case of removal (or failure) of the adaptive section, the existence of a conventional feedback control system upon which the adaptive section is superimposed is certain-ly desirable. An adaptive control with direct actuation has been shown in Fig.2.9. As the actuating signal is generated by a computer directly from observations of the input signal, the system response, and the measured process dynamics. In such a system, the actuating signal may bear no simple relation to system error, but rather may represent the computer's best estimate of the actuating signal required to drive the output to the desired value in the minimum time. This generalization of the actuation problem represents a natural extension of the concepts inherent in the prototype sampled-data systems, the optimum bang-bang systems, and the final-value systems.

2.6.2 The Essential Nature of Adaptive Control

The configuration of Fig.2.10 represents the simplest conceivable adaptive system. In this system, the only variable parameter of the process is the gain K_p, which is measured by comparison of the process and model output signals (if there are long periods with no significant normal operating signals, an extra identification signal can be inserted at the input of the process and model). The gain of the controller is adjusted automatically to the value of K/K_p, so that regardless of K_p, the open-loop transfer function is the unvarying $KG(s)$.

This elemental adaptive system is obviously linear in so far as the relation between input and output is concerned, at least to the extent that the adaptive section responds instantaneously and accurately such linear time-variant performance is realized, however, only by the inclusion of properly selected nonlinearities.

Thus, an overall adaptive system may be linear or nonlinear, but the usual realizations of the adaptivity involve the inclusion of nonlinear elements within the system.

2.6.3. The General Adaptive System

Figure 2.11 is one possible block diagram of the desired generality. The elements of the system are -

(1) The identifier, in which the identification problem is solved in the form associated with the particular type of adaptivity.

(2) The signal identification, in which the properties of the input signals are determined as a basis for the selection of performance or optimization criteria.

(3) The decision computer, in which the results of both process and signal identification are utilized to determine the required controller characteristics or the ~~/~~ necessary actuating signal.

(4) The implementation, or the equipment required to implement the decision of the decision computer.

(5) The controller, usually combined with the implementation equipment, in which the input is modified to yield the required actuating signal.

The general representation of Fig. 2.11 is unrealistic in terms of practical adaptive control systems for the two reasons - (1) most actual processes are exceedingly complex involving many loops and numerous inputs and outputs, and (2) any practical adaptive system must, because of the limitations imposed by cost, size, weight, and reliability considerations, involve a combination or om-ission of several of the blocks indicated in the figure.

2.6.4. Learning in Adaptive Systems

The system described in Fig. 2.11 fail, however, to implement the learning characteristic of the human controller. In learning to drive an automobile on icy roads, the human being adapts in the following way. The new driver,operating with a desire to reach his destination in a minimum time, in spite of slippery road conditions, attempts an initial speed while measuring the road conditions by injecting small disturbances on the steering wheel. When a slight skid occurs, the driver notes the road conditions (or car dynamics) and reduces speed. Gradually, over days or years of experience the driver ac quires a knowledge of the functional relationship if the maximum allowable speed (under the constraint that the probability of skidding during a specified time interval should be less than a fixed small quantity) to the car dynamics. Once this threshold-speed function is known, the driver is experienced on icy roads and future years bring only a slight modification of the knowledge. Acquisition of the required knowledge depends upon the experience or upon the learning mechanism.

The above example represent simple learning process within the scope of feasible instrumentation. In this problem, learning consists in determining the functional relation among forward speed, the probability of skidding, and measured automobile dynamics (or environmental conditions). In this case as well as in the other manifestations of the learning process, the possibility of introducing learning into the decision computer of the adaptive control system opens an entirely new horizon for novel and improved control system performance.

The major promise of the adaptive concept lies in the possibility of introducing a simple learning mechanism within the adaptive part of the system. Once learning is combined with adaptivity, the control system approaches the flexibility and capabilities of human controllers in more significant tasks.

CHAPTER-3

PROCESS CONTROLLERS

3.1 PNEUMATIC CONTROLLERS

In a pneumatic control system, compressed air is supplied to the controlling element. As the value of the measured variable changes, the pneumatic output of the controlling element also changes. A flapper-nozzle mechanism provides the means of controlling the pneumatic output. By adding different elements to this basic mechanism, various control actions can be achieved, as discussed below.

3.1.1. Proportional Control

Refer figure 3.1. A source pressure P_s is fed into nozzle chamber through a fixed restriction. Usually this supply is obtained from a seperate compressed air system. A nozzle opening, which is longer than fixed orifice allows the air to escape. Movement of flapper varies flow in nozzle which in turn varies pressure in nozzle chamber. This is used as output loading pressure P_L.

Adding a bourdon tube -

The bourdon tube acts as pressure sensor. As input pressure P_2 decreases, the bourdon tube movement acts on flapper which closes the nozzle accordingly causing increase in P_L. In practice, the fixed pivot on the flapper is replaced by a bellows (Fig.3.2). P_L is tapped and fed into an adjustable three way valve. The part of

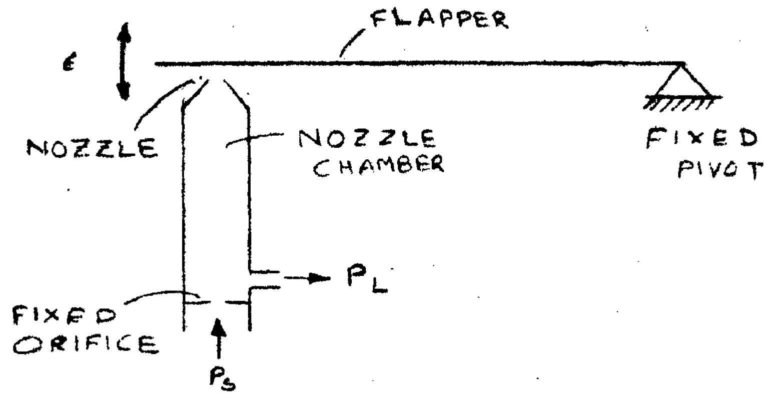


FIG 3.1 BASIC NOZZLE-FLAPPER AMPLIFIER

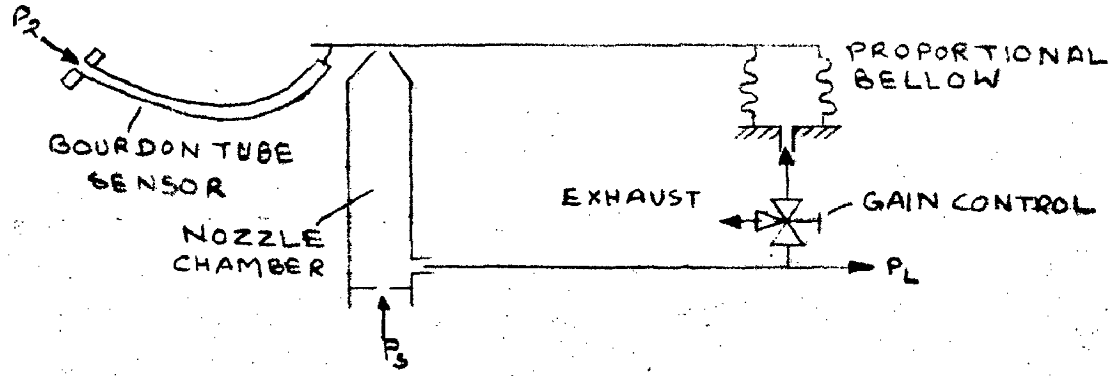


FIG. 3.2 PNUMATIC PROPORTION CONTROLLER

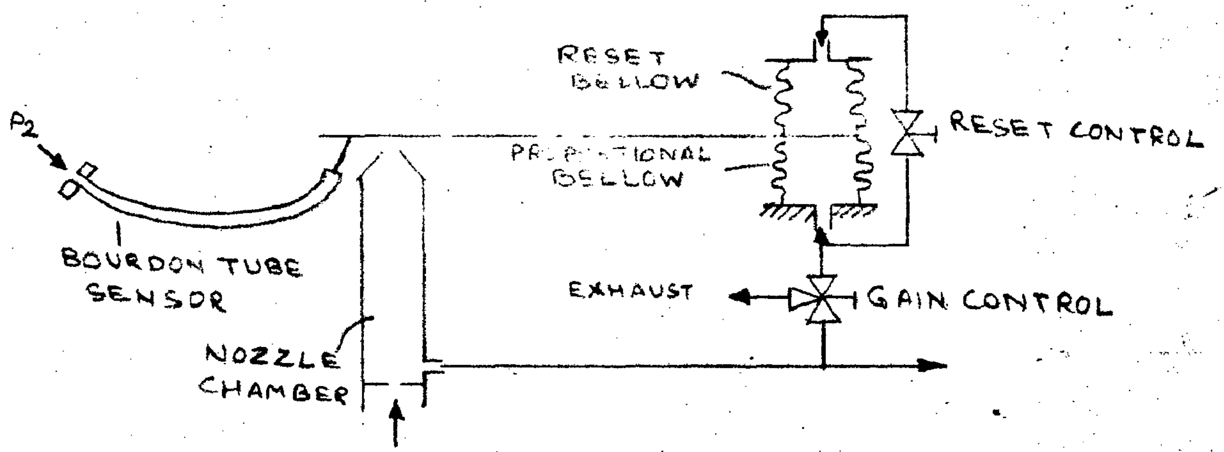


FIG. 3.3 PROPORTION PLUS RESET CONTROLLER

P_L that is sent to bellows can be varied by adjusting the valve. The three way valve acts as splitter. Part of output signal going to it, is fed to the bellows and the remainder is exausted to atmosphere.

Negative feed-back-

Any change in P_2 changes P_L. Part of this is fed to bellows in such a way to reduce the effect of P_2 upon P_L. This is known as negative feedback and is used to reduce overall sensitivity or gain of the mechanism. The greater is negative feedback ~~is~~, greater is the decrease in gain of mechanism. Adjustment of 3-way valve is called gain or proportional band control.

When the load change is large, the controlled pressure P_2, after settling down, differs from the set point. This difference is called offset. The amount of offset can be decreased by increasing the gain of the system. If the gain is very high, offset will become negligible.But the high gain can cause instability in the system. The way to solve instability problem in proportional controller is to adjust the gain to a lower value, by increasing negative feedback i.e. widen the proportional band setting. However it will result offset. This offset can effectively be reduced by adding reset control to the mechanism.

3.1.2. <u>Reset Control</u>

In Fig.3.3, another bellows (reset bellows), opposite to that in Fig.3.2.(proportional bellows) with operating characteristics matching those of later, has been added. If the pressure in these two bellows are equal, each one cancels the effect of other and the system again enjoys high gain. Before the change in pressure can reach reset bellows, it passes through adjustable needle valve, known as reset control. Sole purpose of this valve is to introduce a time delay. The more the restriction adjusted into valve, the longer it will take a change in pressure to register in reset bellows.

Positive feed-back

Signal that goes to reset bellows is called positive feed-back. The more is the value of positive feed back, the more is the increase in sensitivity or gain of controller. In the steady state condition the positive and negative feed back signals cancel each other. In the transient condition, however, the reset restriction temporarily delays the positive feedback signal, so that the negative feedback can achieve its gain cutting effect.

If the gain in steady state condition is high enough, the amount of offset is so small that, for all practical purposes, P_2 is returned to normal position. High gain of amplifier is sacrificed only during transient condition to keep the system stable As transient condition decay, reset action slowly increases the gain and by the time steady-state condition reaches, the high gain leads to steady-state performance, Fig.3.4.

Controller tuning -

To obtain best results, restriction of reset control must be adjusted properly. The reset signal must be delayed just long enough to match the recovery characteristic of process under control

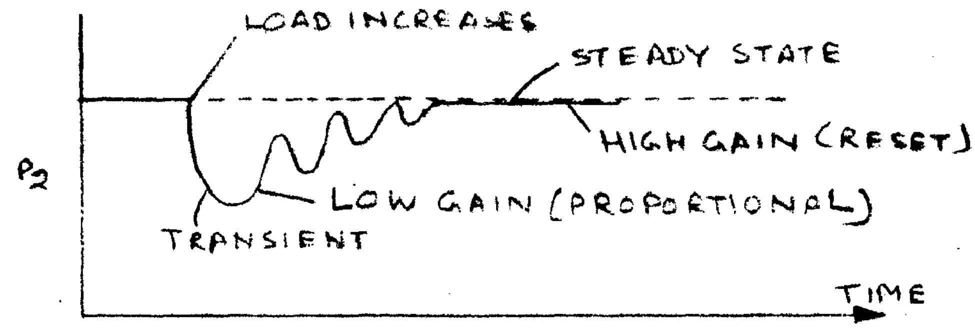


FIG 3.4 CONTROL PERFORMANCE WITH PI CONTROLLER

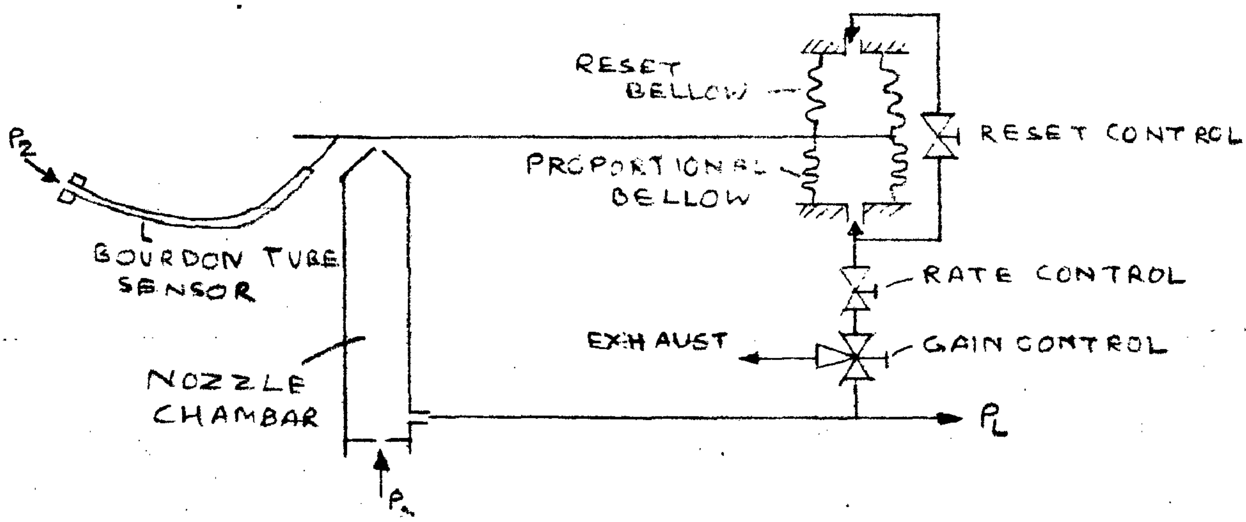


FIG. 3.5 THREE MODE CONTROLLER

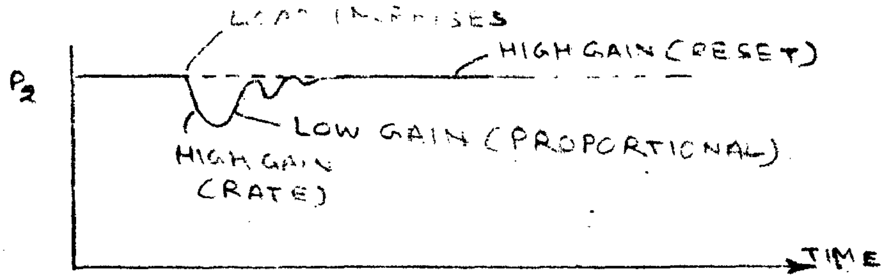


FIG. 3.6 PERFORMANCE OF THREE MODE CONTROLLER

If gain increases too rapidly, the system will become unstable and if it decreases too slowly, the system will be sluggish and no effective control will be there. Proper adjustment of the control is called controller tuning.

3.1.3. Rate Control

As seen above, immediately after disturbance, gain is low to achieve system stability but this causes poor transient responce. Transient responce can be significantly improved by delaying the gain cutting effect just long enough to start the system responding to load disturbance, but not so long as to make the system unstable Fig.3.5 shows the addition of another valve restriction in pressure line, feeding both bellows. Like reset control, this valve also can be adjusted to provide proper time delay, before gain cutting effect begins. Once the pressure change has had time to bleed across the valve restriction, the controller acts as described for two-mode control.

The introduced valve is called rate control because its effect is influenced by rate at which load disturbance occurs. Fig.3.6 shows three mode control.

3.2 HYDRAULIC CONTROLLERS

Hydraulic controllers are also available which provide the three control responces. Essentially, the hydraulic controller resembles the pneumatic controller except that the system must remain completely closed. Jet pipes and pistons, or four way valves, are

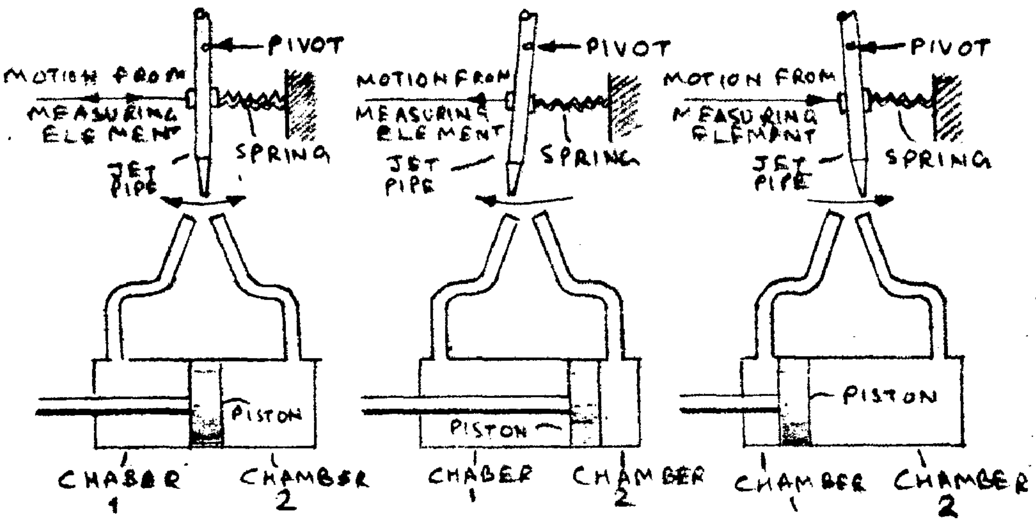


FIG. 37 HYDRAULIC CONTROLLER WITH JET PIPE

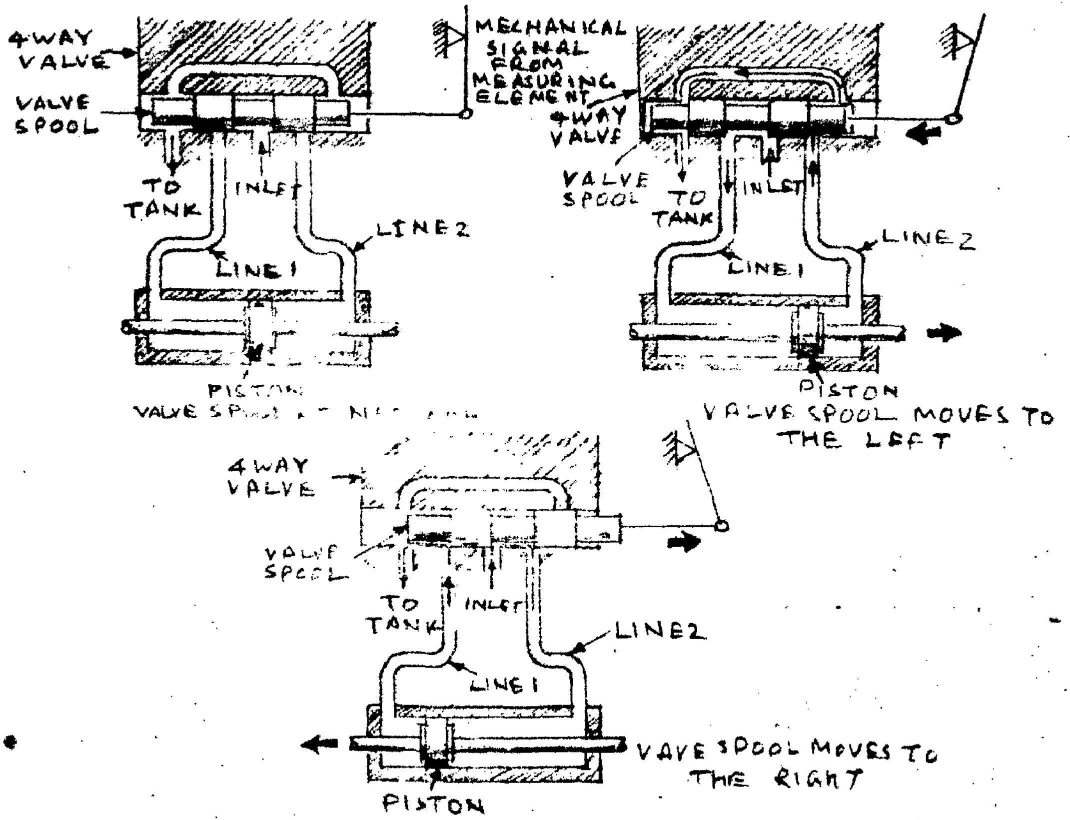


FIG. 38 HYDRAULIC CONTROLLER WITH FOUR WAY VALVE

used in place of the flapper-nozzle and air relay used in pneumatic controllers (5).

The jet pipe resembles the pneumatic nozzle. It directs a fluid stream into either of two receiving chambers of a double acting cylinder. When the jet pipe is moved to the left by the measured variable as shown in Fig.3.7, more fluid enters chamber 1 than enter chamber 2. This causes the pressure in chamber 1 to increase, moving the piston to the right. The piston movement can be used to position a final element.

The four-way valve can be used in the same manner as a jet pipe. See Fig.3.8 when the value spool is moved to the left, the fluid path to Line 1 is opened and the path to line 2 is closed. The piston moves to the right. Similarly, when the valve spool is moved to the right, the fluid path to line 1 is closed, and the path to line 2 is opened. The piston then moves to the left. The action of the four-way valve resembles the action of the jet pipe system.

On-off Action is accomplished in a hydraulic controller by very rapid movement of the piston to either extreme position when the measured variable deviates a small amount from set point. The jet pipe or valve spool is positioned at neutral to establish the set point at a particular valve of the measured variable.

Proportional Action requires a feed back signal from the piston to the jet pipe or spool. Proportional action is provided by the addition of a feed back linkage from the piston. When the jet pipe is moved to the right, as shown in Fig.3.9, because of a chang

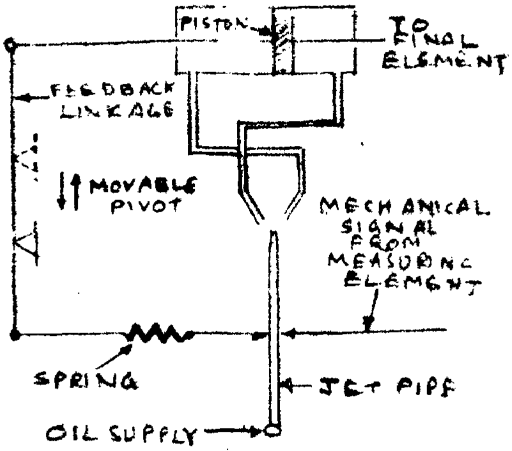


FIG. 3.9 PROPORTIONAL ACTION IN A JET PIPE HYDRAULIC CONTROLLER

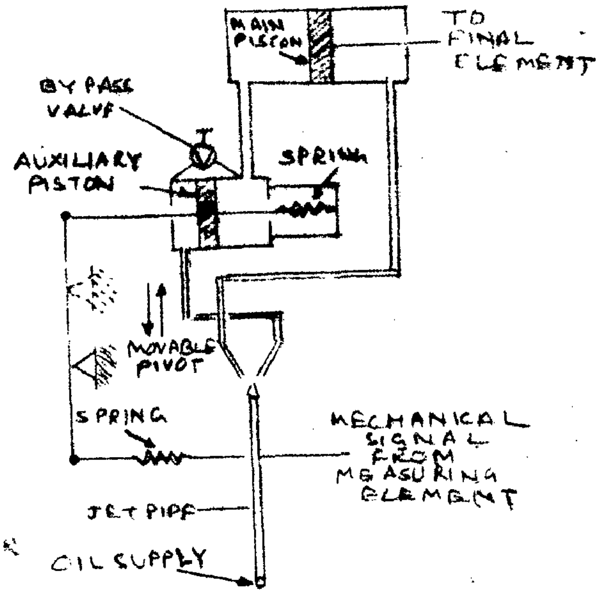


FIG. 3.10 PROPORTIONAL PLUS RESET ACTION IN A JET PIPE HYDRAULIC CONTROLLER

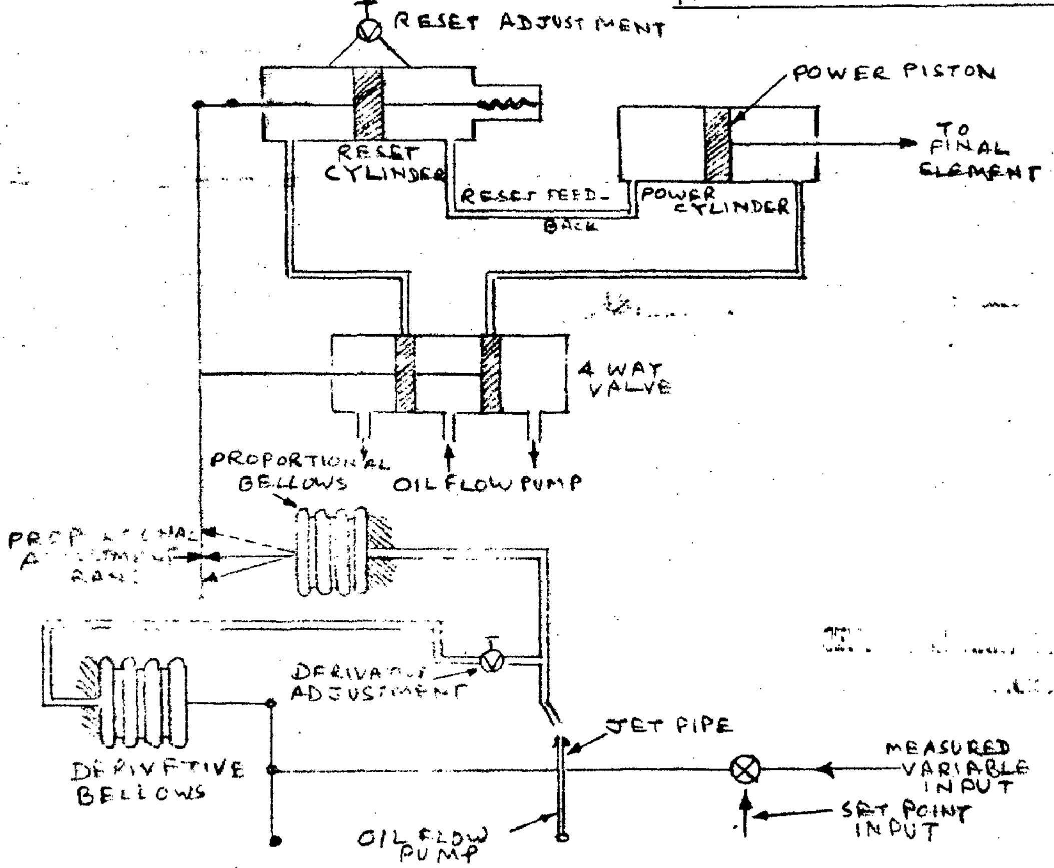


FIG. 3.11 PROPORTIONAL PLUS RESET PLUS DERIVATIVE ACTION IN A HYDRAULIC CONTROLLER

in the value of the measured variable, the piston moves to the right. The feed back linkage is attached to the piston. It brings the jet pipe back to neutral position which stops the movement of the piston. Thus there is a piston position for each value of the measured variable. Changing the location of the pivot provides proportional band adjustment. This regulates the amount of piston motion required to restore the jet pipe to its neutral position.

Automatic Reset Action can be added to the proportional controller by modifying the feedback signal. Fig.3.10 illustrates a jet pipe hydraulic controller with proportional plus reset control action. The addition of an auxiliary piston with a bypass provides the reset action. When the jet pipe is moved to the right, both the main piston and the auxiliary piston move to the right. The feedback linkage is attached to the auxiliary piston. This linkage returns the jet pipe to its neutral position. Because of the bypass, the auxiliary piston returns toits mid position, moving the jet pipe to the right again. This causes motion of the main piston in the original direction.

Derivative Action can be added to a hydraulic proportional plus reset controller. In the schematic of a controller shown in Fig.3.11, a four-way valve is used to provide the power to the reset and power cylinders. The proportional bellows provides the actuation of the four way valve. There are two seperate feedback systems, one for reset action and a second for rate action. The reset feedback signal is provided by the power cylinder. The jet pipe sends a signal to the proportionalbellows when the value of the measured variable deviates from set point. Reset action allows the power piston to move as long as the proportional bellows receives this signal. The adjustable restriction on the bypass line of the reset cylinder determines the reset time.

The linkage from the derivative bellows provides the derivative feedback signal. The derivative bellows moves the jet pipe back to a neutral position. The adjustable restriction between the jet pipe and the derivative bellows determines the rate time. The derivative action delays the feedback to the jet pipe as long as necessary in the particular process. The proportional bellows is able to operate longer with derivative action in the controlling system. Proportional band adjustment is made at the point where the proportional bellows acts on the lever assembly which connects the four-way valve spool to the reset piston.

9.3 ELECTRIC/ ELECTRONIC CONTROLLERS

Electric controllers (5) for proportional, proportional plus reset, and proportional plus reset plus derivative actions can be divided into two types -

1. The null-balance type in which there is an electrical feedback signal to the controller from the final element.

2. The direct type in which there is no electrical feedback signal. This is also called feed forward control.

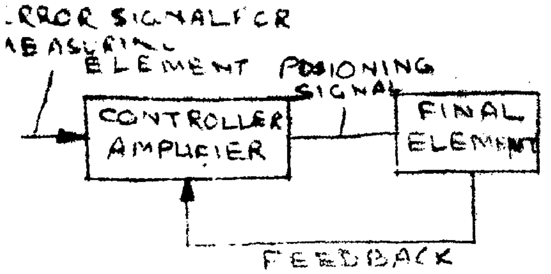


FIG 3.12 SCHEMATIC OF AN ELECTRICAL NULL BALANCE CONTROLLER

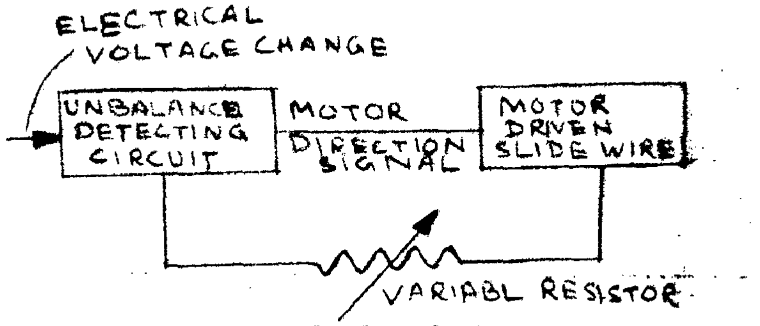


FIG 3.13 PROPORTIONAL ACTION is provided by addition of a variable resistor, which allows adjustment of the proportional band

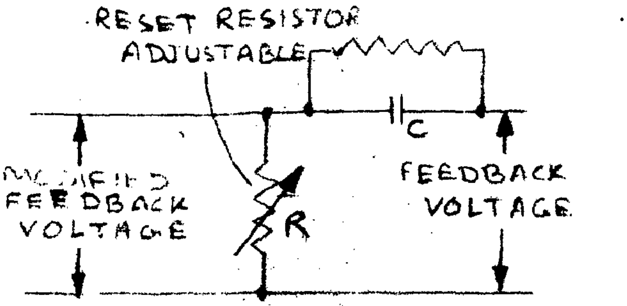


FIG 3.14 THE FEEDBACK SIGNAL IS MODIFIED TO PROVIDE PROPORTIONAL PLUS RESET ACTION

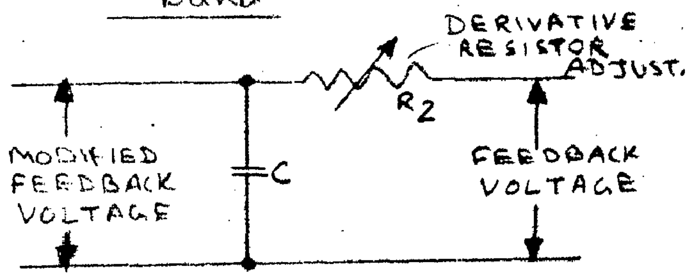


FIG 3.15 THE FEEDBACK SIGNAL IS MODIFIED TO PROVIDE PROPORTIONAL PLUS DERIVATIVE ACTION

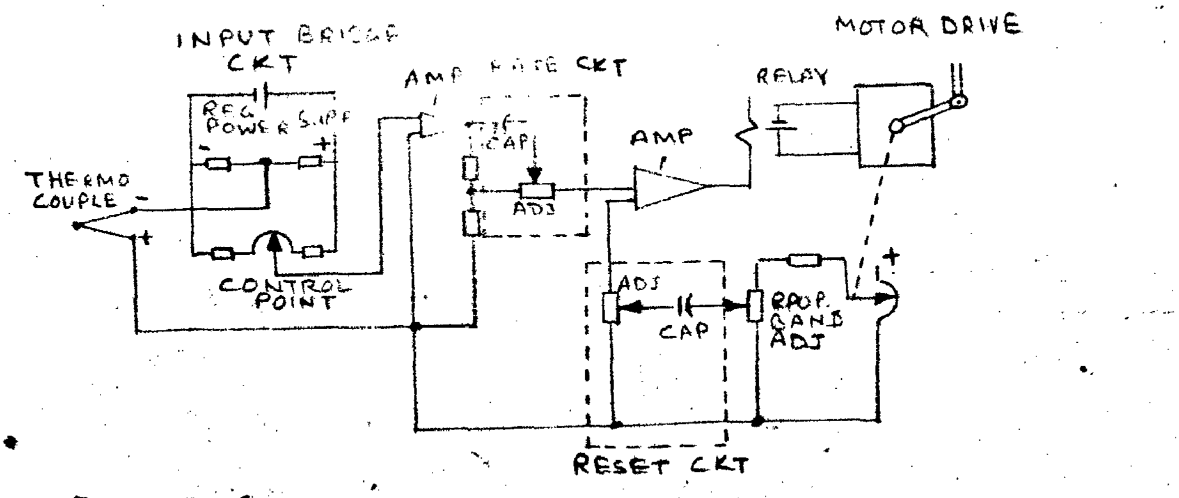


FIG. 3.16 AN ELECTRONIC TEMPARATURE CONTROLLER WITH PROPORTIONAL PLUS DERIVATIVE (RATE) PLUS RESET ACTION

Fig. 3.12 is a diagram of an electrical null-balance controller. An electrical null-balance controller provides the various control actions by modifying the feedback signal. This is done by adding properly combined electrical resistances and capacitances to the feedback circuit, just as restrictions and chambers are added in the pneumatic circuits.

A sensitivity, or gain, adjustment is the only addition to the controller required to achieve proportional action. This permits the adjustment of the proportional band. The sensitivity adjustment x is made by inserting a variable resistor in the feedback line. This provides for regulating the magnitude of the feedback signal. The feedback signal depends on the position of the final element. The amount the final element must move is now established by the setting of the variable resistor. This movement produces electrical balance in the controller. See Fig.3.13.

The feedback signal is modified by the addition of a resistor - capacitor arrangement to provide proportional plus reset action. See Fig. 3.14.

Any change in the feedback voltage from the final element slidewire causes a current to flow into the capacitor C. This is where it is stored as a voltage. The capacitor is then said to be charged. This results in a voltage drop across resistor R. For a current to continue to flow through R, the capacitor must remain charged. Hence, it must continue to receive current. It receives current by a continuing change of the feedback voltage. The slide wire of the final element and the final element itself must, therefore, continue to change position to produce the necessary voltage change. The voltage continues to change as long as there is an unbalance signal from the controller. The reset action ceases when the error signal is eliminated and the value of the measured variable is at set point. The setting of the resistor R determines the rate at which reset action proceeds.

The feedback signal is modified with an additional resistor-capacitor net work to provide proportional plus derivative action. See Fig. 3.15. With this arrangement, any change in the feed back voltage causes the capacitor C_2 to draw either more or less current, whichever is required to delay the effect of the change in feedback voltage. The final element can move more than it would if only proportional action were used.

The setting of the resistor R_2 determines the rate at which the capacitor is charged. Therefore, the modified feedback voltage varies with the rate the final element slidewire changes the feedback voltage. The final element slidewire position changes as the value of the measured variable changes. The modified feedback voltage changes with the rate the measured variable changes. Derivative action is this accomplished.

Fig. 3.16 shows a null balance electric controller with proportional plus derivative plus reset action. The controller includes a control bridge, a feedback bridge, a detector-amplified relay, and a power motor. The rate and reset networks are also shown.

The power motor positions the sliders of two different slidewires. One slide wire is in the control bridge, and the other is in the feedback bridge. In some electric controllers are power motor can also be used to position the final element.

The input signal of the detector is the difference in voltage between the control bridge and the feedback bridge. This signal is sufficiently amplified so that the relay is actuated. The relay causes the power motor to run in a direction which repositions the sliders. This reduces the input signal to zero.

The control bridge contains the proportional band adjustment. This is the sensitivity adjustment. The adjustment determines the relationship between the input signal to the control bridge and the resultant output signal of the control bridge.

The feed-forward electric controller operates on a different principle. There is no feedback signal from the final element. Fig.3.17 is a schematic of a typical feed-forward electronic controller with proportional plus reset plus derivative action. This controller operates from a 0 to 10 volt dc input signal. If the primary element does

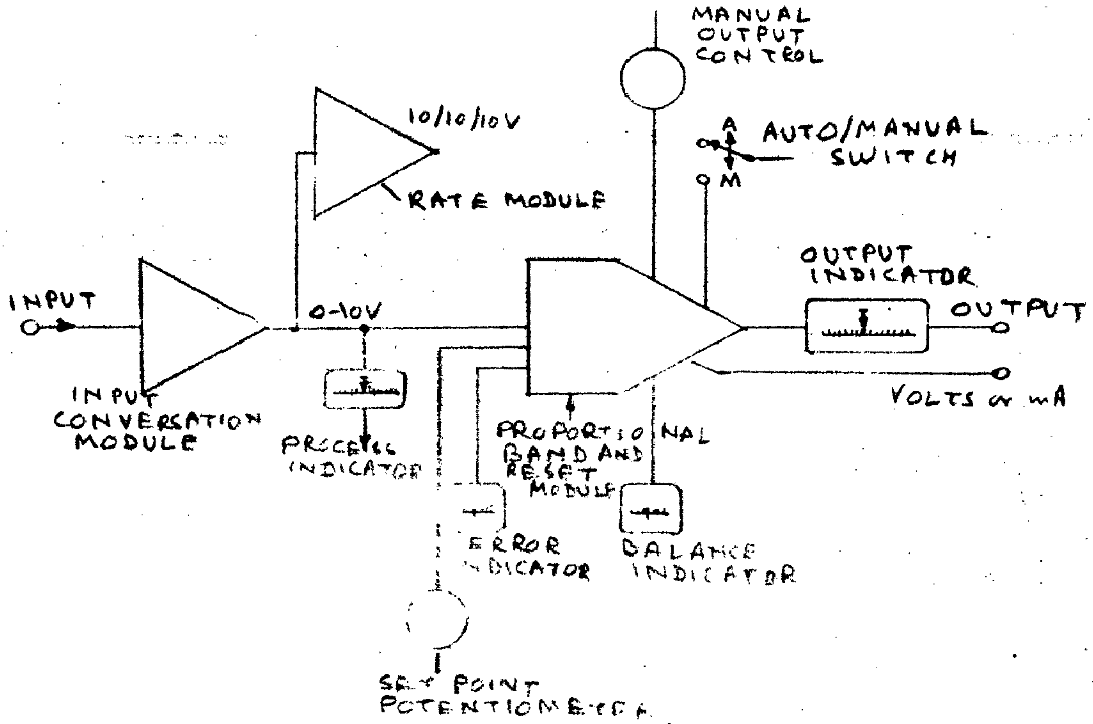


FIG. 3.17 A DIRECT-TYPE ELECTRIC CONTROLLER USING PROPORTIONAL PLUS DERIVATIVE PLUS RESET ACTION

not provide an input, the conversion module provides the operating voltage.

The percentage of total input voltage is shown on the process indicator. The matching set point voltage of 0 to 10 volts is introduced by a precision potentiometer. Any difference between the input signal voltage and the set point voltage enters the proportional plus reset module. The percentage of total input voltage is also shown on an error indicating meter.

Proportional band adjustment is accomplished by a precision potentiometer which regulates the output of the amplifier in the proportional plus reset module. The regulated output of the amplifier is further changed by a precision potentiometer in the reset RC network to provide reset action. For rate response the input signal voltage is modified by an RC network before entering the proportional plus reset module to provide derivative action. The output of the derivative module is then proportional to the change rate of the input signal voltage. The output meter indicates the percentage of total output voltage being produced.

A balance-indicating meter is provided to facilitate switching to manual control. The manual adjuster must be positioned so that the meter is at the null position when it is switched from automatic to manual control. A capacitor in the instrument circuit eliminates the need for manual adjustment when the meter is switches from manual to automatic This feature of a meter is called automatic bumpless transfer.

3.4 MICROPROCESSOR BASED CONTROLLERS

Microprocessor based controllers are the most modern-type of electric controllers. Due to their increasing importance and due to the fact that this is the type which has been considered later in the dissertation, we shall discuss them in this separate section.

The industrial control system is a dilemma, a contradiction, and the downfall of many a high technology entrepreneur, control at once demands the very latest capabilities and also rejects them. Control equipment designed over twenty years ago sells side by side with the latest designs. Reliability is the reason or sometimes really the execuse for not updating [6].

Until very recently, electronic process controllers were analogs of earlier pneumatic-mechanical controllers. They did little beyond offering the advantages of electrical signal transmission. Solid state replacements for relays promised more reliability, but required a different order of maintenance expertise when they did fail.

The primary onslaught of technology against the controls market, however, was in form of the electronic digital computer. For control, as for everything else, the computer was the ultimate problem solver. It could (and still can) do everything short of supplying the final actuator power to move the process. It could replace everything else in the system but the primary sensors and the output power controllers.

But even the computer had large problems. It was not reliable enough to commit a process to. It cost too much. New people would be needed to operate and maintain it. Every input had to be converted to digits. And on, and on. So even with all the wonderful inferential calculations and comparisons that could be made with a computer, it took the large-scale-integrated technology of the microprocessor before computer control really caught hold.

Today it seems there is hardly a control product of any kind that does not somehow incorporate or rely on a microprocessor. They are not only in controllers, and communications links, and terminals for the operator, and actuators. They are even in the central minicomputers, making them more capable and more reliable.

So it seems that the microprocessor has finally made all that great electronics technology available to control and apparently is even making control technology driven. According to a manufacturer of microprocessor based distributed controllers, 'We have found the advent of microprocessor technology to be qualitatively different from many of the previous new developments. We can recall situations where we were market-driven and technology limited. That is, our marketing people were defining user needs, and then working with development and engineering people to determine whether the technology existed to meet those needs. Today, we find more technology available in the microprocessor than can be used intelligently. The marketing task has become a matter of determining how this technology can be employed to deliver real user benefits'.

3.4.1 How to Design Microcomputer into Control Systems

Having decided that a given system should be microcomputer controlled - and the only valid reasons for using a microcomputer is either to give a product enhanced features or to cut costs - the first thing that needs to be done is to sit down and list all the functions to be performed [7].

Once the listing has been prepared, we are ready to scan microcomputer literature - specifications, user manuals, programming manuals, etc.

Draw a Flow Chart

With some idea of what microcomputers are available and what the system must do, we can prepare 'high-level' flow chart. Such a chart is really like the function listing prepared previously, but now it is in a flow-chart form, making it easier to visualize the process flow.

Now we should attempt to come up with a rough system cost. To this end, draw up a rough block (or interconnection) diagram of the system using several major microcomputer candidates. The reason for obtaining a cost estimate at this early stage is obvious: if it turns out too high, the project can be dropped before a significant expenditure of time and/or money.

Sketch a Block Diagram -

Once the block diagram has satisfied that the system is within our budget, the next step is to interconnect the blocks on a functional level. In doing this, we must keep track of the I/Os that have been used and must continuously examine necessary functions in terms of microcomputer capabilities. For instance, if we have to do a job in, say, 100 μs and it will take some 20 steps in a 15 μ machine, we immediately can know that we are in trouble - we either have to look for another microcomputer or do the required functions in hardware.

Speed vs Word Length

The overall design of a microcomputer - based control system boils down to a trade-off between speed and word length. Suppose, for instance, we have to control a motor by monitoring its speed at all times. How many times per seconds is it necessary to sample its speed and give a command. Applying basic sampled data théory tells us that the sampling frequency must be at least twice (and practically, at least five times) the highest frequency of the sampled signal. A calculations and I/O functions must be performed at this rate, Can our microcomputer execute all the necessary instructions in that time interval. From this analysis we can estimate all the 'tight-spots' that really determine the speed versus word length trade-off.

At the same time, memory size is highly important, particularly in view of the fact that most single-chip microcomputers come with some fixed amount of RAM and ROM.

Learn the Micro Architecture

After going through these exercises we come out with a particular microcomputer. We must now get familiar both with the instruction set and the architecture.

Once we are familiar with the I/Os, the architecture, and the instruction set, we can take the block diagram developed earlier and convert it into a very detailed, schematic-like diagram, including everything . At this point the hardware design is being completed, and the diagram should clearly pinpoint what must be controlled.

In addition to this a detailed flow chart must be created which should also include sequence of events and their timing.

Programming -

There are two basic ways to program - straight linear programming (one program for the whole job), or by partioning and subroutines. Straight linear programming is sometimes more efficient, but it is more difficult to program.

An efficient program is written by combining hardware and software, knowing limitations of the machine, and knowing what can be done outside more efficiently or at a lower cost.

Software development can be done in one of the three following ways -

(1) In-house using a timesharing service or an in-house software development system.

(2) By using a vendor-supplied software development system.

(3) By using a software consultant.

Trial Run

Typically, we first try to run the program and it does not run. Here is where an accurate and detailed flow chart will pay off. By looking at it, we will see at a glance just what conditions should be present or absent at any point in the program.

Now comes the hard part - the real-time, 'true-life' system execution. At this point we may discover that certain real-world conditions were not taken into account during program development and program must be modified.

Getting the system into production

Till now we have made two assumptions -

(1) that we have a real-world system, and

(2) that we have operated it in a real life environment.

One or both of these assumptions might be wrong. So we can go to a prototype boards. We can take the prototypes and operate them in the field. This process will uncover any real-world problems.

After this last step we can start talking to the microcomputer vendor about developing the mask. The first engineering sample deliveries must be tested again. This is to confirm that the vendor implemented our program as we told him.

CHAPTER-4

DESIGN OF µP-BASED ADAPTIVE CONTROLLER

4.1 INTRODUCTION

When a microprocessor or low end minicomputer is used in developing an adaptive or self-tuning controller for a process with relatively fast dynamics, the combination of low computing speed and high sampling rate will limit the complexity of feasible algorithms. Thus complex adaptive control algorithms are not easily implemented. An attractive alternative is an adaptive controller which substitutes table look up for computation.

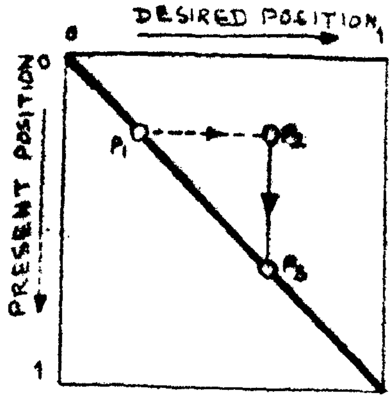
A major problem encountered in the development of control algorithms which make use of data storage and conditional operation arisesfrom the fact that the mathematics involved in a rigorous analysis of the performance of these algorithms is extremely complex. To avoid this complexity a relatively simple, intutive development of a self-tuning control scheme (8) is presented in section 4.2. Before discussing this scheme in the following sections we shall have a preliminary overall look at the controller to be designed.Let us suppose that the process has a moving part, position of which is to be controlled as per the set value. There will be a position sensor which will be sensing the current position and converting it into electrical signal. This signal after processing willbe fed into the microcomputer based controller. The set value will be fed through a keyboard. If there is any difference between present position and the set value, the controller will take the corrective action through some device e.g. d.c. motor. The corrective action of the controller will be based on the look up table which will continuously be updated.

In section 4.3, essential elements of a microcomputer are discussed, then a typical microcomputer MCT-1, on which the software was developed and tested, has been described. Architecture of 8080A CPU and its instruction set has also been included. In section 4.4 actual algorithm used, structure of the look-up table, zero-speed detection etc. has been discussed. Interconnection of various blocks in order to achieve complete controller has been discussed in section 4.5. The software program which was developed and tested has been given in annexure-1.

4.2 ADAPTIVE POSITION CONTROLLER

Suppose we have a table of ' how to move' data organised by ' present position' and ' desired position'. When the table is filled, the entries show how to move from each present position to each desired position. Situations where present position equals desired position correspond to points on the diagonal such as P_1 (diagram FIG 4.1). The entries on the diagonal show how to stay at the present position (8).

A set point (i.e. desired position) change corresponds to a move off of the diagonal from P_1 to P_2. The entry at P_2 shows



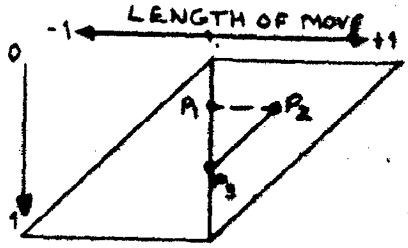


FIG. 4.2 MODIFIED TABLE ORAGANISATION

FIG. 4.1 A SIMPLE POSITION MOVE

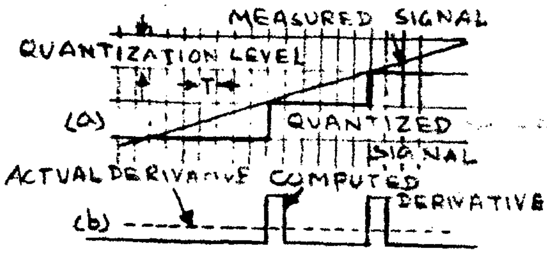


FIG. 4.3 DERIVATIVE CALCULATION

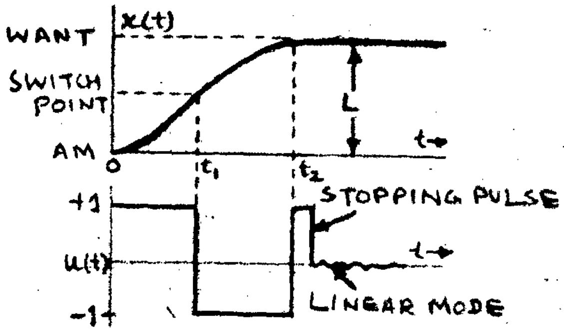


FIG 4.5 SEQUENCE OF CONTROL ACTIONS

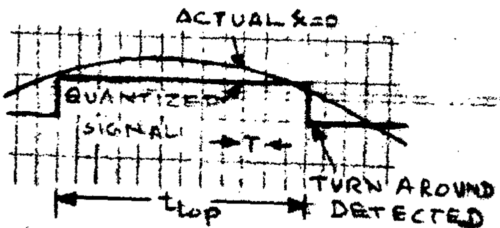


FIG 4.4 ESTIMATION OF ZERO VELOCITY POINT

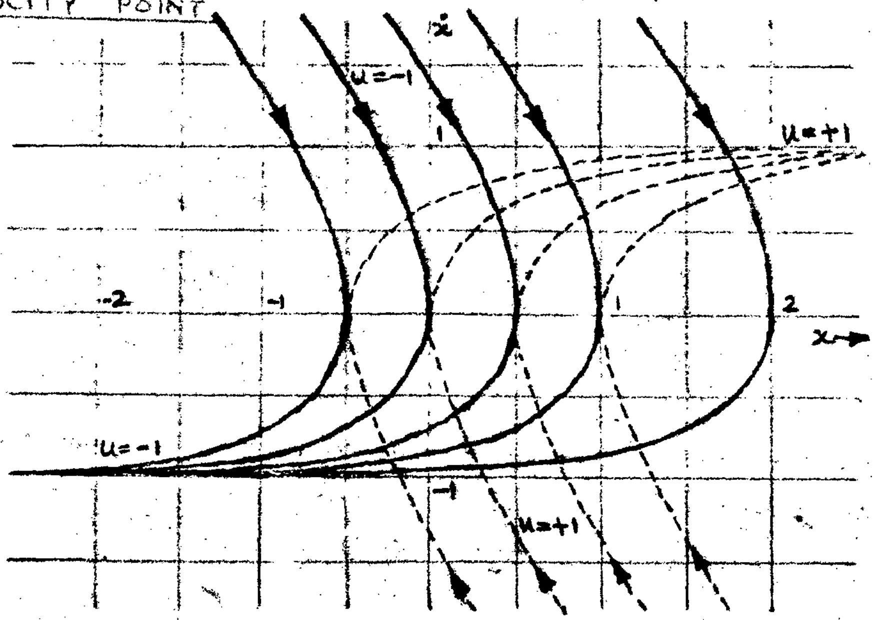


FIG 4.6 PHASE PLANE PORTRAIT FOR $\ddot{x}+\dot{x}=\pm 1$

to make the desired response - a return to the diagonal with ...ent position equal to the new set-point. This corresponds to ... move from P_2 to P_3.

When the table is full, there is information on how to ...form all possible moves. Assuming that this information is ...rect, we can perform any move within the constraints of our ...c design assumptions.

...1. Trial and Error

Suppose, however, that the table is only sparsely popu...ed with known data and that we are requested to perform a move ... which there is no data in the table. A simple strategy is to ... one or several known moves which are 'near' to the desired ... in some sense and then extrapolate from them.

As we make moves, we gain information required to fill ... table. If the control actions based on an extrapolation from ...rest neighbors proves successful (i.e. we get to the desired ...tion) then this becomes the actual data for the previously ...own move.

If the actions were unsuccessful, we have also gained ... information. We now know how to get to wherever we ended up, ... the entry in the table corresponding to this actual move may ...filled for future use.

If the table contains data for a requested move, we ... still check the actual final position against desired positio...

in order to verify table accuracy. If the move is unsuccessful there are two courses of action. 1) the entry may have been based on bad data or the system may have changed, in this case we wish to correct or adapt the controller by updating the entry, or 2) the failure may be a spurious result due to noise or a disturbance, in this case we do not want to change the entry. Since we generally cannot tell which is true, there must be a trade off between speed of acquisition (learning speed) and noise rejection.

So far we have assumed that a request to perform a move for which there is no data in the table will lead to a search for the nearest moves for which there is data. Since searching for the nearest neighbors will be time consuming, and since we would like to begin the move as soon as possible after the request is received, we will not perform the search when the request is received. Instead, each unknown position in the table will be provided with links to its nearest known neighbors.

These links need only be updated when a previously unknown move becomes known- when this occurs, all entries in the table are examined to see if the new move is closer than any of their present nearest neighbors. If so, the appropriate links are changed to refer to the new move.

The actual control program will be driven by interrupts from a real time clock. When this routine completes a move, successful or not, it placed data concerning the actual move in a first-in/first out queue. A con-current non-interrupt routine removes data from the queue and updates the table.

4.2.2. Content of the Table

In order to facilitate development of a table based self-tuning scheme, the control method used should exhibit several important properties -

_ To minimize the table size, the control action for each move should be describable with a minimum data.

_ To facilitate extrapolation from the nearest neighbors the stored move data should change in a relatively smooth way over the entire table.

_ The move data must be chosen so that it can be reliably determined by the computer from measurements made during normal system operation.

This last requirement is perhaps the most difficult to satisfy. If we are allowed to apply steps or other test signals to the open loop system, it may be quite easy to find parameter values which give the best control. However we should like to determine these parameter values from closed loop operating data in order to improve or optimize performance on line. This requirement renders many control methods unsuable since they tend to mask the system's characteristics.

For example, if we use measurements of peak overshoot and ringing. However, since our measurements have only finite precision it will now be rather difficult to obtain an accurate assessment of system performance under the present control so that further optimum tuning may take place.

4.2.3 Bang-bang control

A simple control which performs well with respect to the above requirements is bang-bang or contactor control. This is also time optimal for many systems.

Let us consider a simple controlled process modelled by

$$\ddot{x} + \dot{x} = u(t) \qquad (4.1)$$

where $x(t)$ is position and $u(t)$ is a bounded input in the range $-1 \leq u \leq 1$. The fastest way to move from a initial position $x(o)=x_o$ to final position $x(t_f) = 0$ is to make use of the maximum force available, that is, to accelerate towards zero until the ' last ' possible movement' and then applyfull braking force in order to come to rest at $x(t_f) = 0$.

The Fig.4.6 shows the phase plane portrait of the process described by equation (4.1) for $u = 1$ and $u = -1$. It is clear that we can move x_o to 0 by applying $u = -1$ until we intersect the $u = +1$ trajectory which passes through the origin, and then applying $u = +1$ to follow this trajectory into the origin. Since for any initial condition the switching of control polarity will occur at the trajectories which lead into the origin, these trajectors are called the switching curves for the system.

The equations of these curves may be determined from equation (4.1). By integration of eqn.(4.1) for u constant we obtain

$$x = ut - A + A e^{-t} \qquad (4.2)$$

$$\dot{x} = u - A e^{-t} \qquad (4.3)$$

where $A = u + \dot{x}_o + 2 x_o$. Eliminating t between equation (4.2) and equation (4.3) gives equation (4.4)

$$x = -\dot{x} - u\left[\ln\left(\frac{u-\dot{x}}{u-\dot{x}_o}\right)\right] - \dot{x}_o - 2 x_o \qquad (4.4)$$

The switching curve is obtained by setting $x_o = \dot{x}_o = 0$. It is

$$x = -\dot{x} + \text{sign}(\dot{x})\left[\ln(1 + |\dot{x}|)\right] \qquad (4.5)$$

(Note- when $\dot{x}$ is +ve, u is -ve and when $\dot{x}$ is -ve, u is +ve).

Although this equation is not simple, it might be possible to evaluate it in real time if x and $\dot{x}$ are given. A more serious problem is that with more general processes the switching curve depends in a complex way on process parameters. Thus errors in modelling the process may lead to significant errors in the switching curve equations, giving rise to poor performance. In addition, in a digital realisation it is difficult to get a good value for $\dot{x}$ from measurements of x. For these reasons, we would prefer to avoid using the derivative in complex calculations.

In a positioning system, it is not unrealistic to require that $\dot{x}_f = \dot{x}(t_f) = 0$. While this places a restriction on the capabilities of the system, there are many applications, such as numerically controlled machine tools, where a tool head or work piece must come to rest before an operation can commence. This type of application will be assumed.

Using this assumption, the switch point for our example process may now be determined from equation (4.4). Let x_o be greater than zero and $\dot{x}_o = 0$. Then for the trajectory with $u = -1$ in equation (4.4) we obtain

$$x = -\dot{x} + \ln(1 + \dot{x}) - 2x_o \qquad (4.6)$$

For the trajectory with $u = +1$ and $x_o = 0$ we obtain

$$x = -\dot{x} - \ln(1 - \dot{x}) \qquad (4.7)$$

The switch point is the value of x at which these trajectories intersect. Eliminating $\dot{x}$ between equation (4.6) and equation (4.7) gives equation (4.8)

$$x_{sw} = -\ln\left(1 + \sqrt{1 - e^{2x_o}}\right) + \sqrt{1 - e^{2x_o}} \qquad (4.8)$$

While this derivation assumes a final position of zero, it can easily be extended to arbitrary final conditions by a change of co-ordinates. The important point is that in all cases the switch point is now a function only of the initial and final positions. It is these switch points which will be stored in the self-tuning tables.

For more complex processes the bang-bang solution and switching formulae are more complex. However, as is common in control system practice, we will assume that our process is dominated by a pair of poles and thus can be approximated as a second order system. Further, the ' stop-pulse' described later in this chapter can be viewed as a correction to accommodate higher order dynamics of the process.

4.2.4 Implementation

Since the control used is bang-bang and the process is approximated as a second order system, there will be only two transitions of the control signal for each move. Moreover, the second transition always occurs when the velocity $\dot{x}$ is zero so there is only one piece of data to be stored for each move in the table, the location of the first switch point. To facilitate extrapolation this data will be stored as the ratio of the distance at which switching occurs to the total distance to be moved.

For linear systems, the switch ratio will depend only on the length of the move. Thus for systems which are ' almost linear' we want our measure of nearness to consider same length moves more favourably than same origin moves. This corresponds to a search for nearest neighbors in the form of an expanding ellipse whose major axis is parallel to the diagonal of the table. As this organization makes the search and computation of distance to neighbors messy, the organization of the table will be changed to place all moves of the same length in the same column of the table. The result is a rhomboidal table where the row is determined by the starting position, and the column is determined by the length of of the move. Diagonal ellipses in the original table now become vertical ellipses which are more easily searched.

Let P_i be the starting position for a move, and L_i be the length of the move. Then the distance (i.e. nearness) in the table between moves (P_i, L_i) and (P_j, L_j) is defined as

$$D = \left[W_P \, (P_j - P_i)^2 + W_L \, (L_j - L_i)^2 \right]^{1/2} \qquad (4.9)$$

where W_P and W_L are weighting factors which control the shape of the constant distance ellipses. For example, W_L greater than W_P, will favour moves of the same length over moves with the same starting position in the linkage assignment.

For a requested move from AM to WANT we define L = WANT- AM (L can be +ve or -ve) and obtain the position at which switching will occur as equation (4.10)

$$\text{SWITCHPOINT} = \text{AM} + \text{Table (AM,L) XL} \qquad (4.10)$$

for a known move, and equation (4.11)

$$\text{SWITCHPOINT} = \text{AM} + \text{Table (LINK (AM,L)) XL} \qquad (4.11)$$

for an unknown move, where LINK(AM,L) point to the known move nearest to (AM,L) as discussed earlier.

A move now consists of applying an accelerating force until SWITCHPOINT is reached, and then applying a braking force until motion ceases. At the end of each move, the interrupt routine puts the data necessary for table updates in a queue read by the update routine. The interrupt routine then checks position to determine whether or not the move was successful. If the error is not small, ENDUP and WANT are used for another table look up and move.

If the error between ENDUP and WANT (or AM and WANT) is small, continuing the use of bang-bang control will cause 'Chattering'. This type of operation is usually undesirable and it is preferable to use a gentle form of control when the error becomes small. In this program we enter a linear mode of operation, specifically PID control.

When a set-point change occurs, or a disturbance causes the error to become large, we leave the linear mode and perform another table based move. In some cases, this dual mode control can lead to limit cycling in and out of linear mode. In order to avoid this, hysteresis is added to the linear control boundary. That is, the error must be less than e_{in} to enter linear mode, and greater than e_{out} to leave linear mode where e_{out} is greater than e_{in}.

4.2.5. Measurement of derivative

We have assumed that we could tell when motion had ceased, i.e. when $\dot{x} = 0$. In reality, this is not a trivial problem due to the limited precision of digital measurements.

The usual approximation to the derivative is

$$\dot{x} = (1/T_s)(x_i - x_{i-1}) \qquad (4.12)$$

If the resolution of the measurement of x and the sampling period T_s are appropriately related, this approximation performs quite well. However, this approximation encounters difficulties when $x_i - x_{i-1}$ is less than the amplitude quantization of x during the period T_s.

To see this, consider the analog signal x with the quantized version $\bar{x}$ superimposed as shown in the figure. The analog signal is a ramp with derivative shown by the dotted line in the figure 4.3. The quantized signal is a staircase rather than a ramp, with the effect that the derivative approximation in equation (4.12) yields a series of pulses at the points where the ramp signal crosses the quantization boundaries. The effect of such a pulse derivative on control can range from satisfactory to destabilizing.

The reason that equation (4.12) gives such poor performance is that we are sampling faster than the signal can cross the quantization boundaries. Decreasing the sampling rate of the entire control algorithm would help, but may not be desirable due to effects on other portions of the computation.

As an alternative we can use the approximation

$$\dot{x}_i = \frac{1}{n\,T_s}\left(x_i - x_{i-n} \right) \tag{4.13}$$

where n = 1, this is identical to equation (4.12). For larger n, the effect is to increase the time between the samples used in the difference. While the same difference could have been obtained by using a (locally) reduced sampling rate in calculating equation (4.12), the approximation in equation (4.13) has the advantage that we get a new value for the derivative every T_s seconds rather than every $n\,T_s$ seconds.

However, it should be noted that the derivative computed in equation (4.13) is (approximately) the derivative at time $t-\frac{1}{2}nT_s$. Hence there is a limit on how large we can make n due to the delay (and error) introduced. To avoid these effects nT_s should be on the order of five or ten times smaller than the shortest significant system time constant.

4.2.6. The Stopping Pulse

In the bang-bang control described we must detect the point of zero derivative to terminate the control action. Unfortunately, even the scheme outlined fails when the derivative becomes sufficiently small.

Since full braking force is applied even when the error and its derivative are small, the effect of continuing the braking force after the derivative goes to zero is to cause the system to reverse direction or ' turn around'. As we cannot eliminate the delay and consequent late switching, we attempt to compensate for it.

One method is to keep track of the time (number of samples) that the measured variable has been at the present quantization level. When we detect that the variable has changed

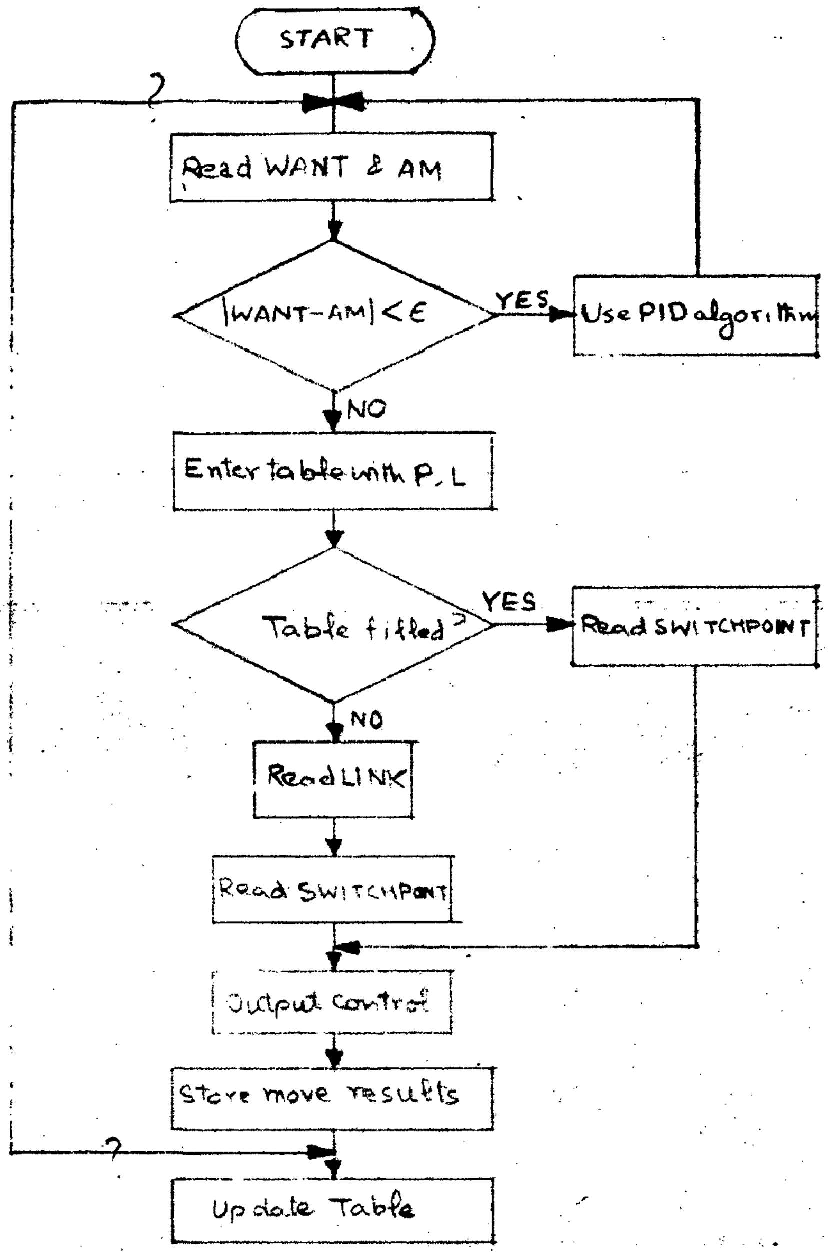


FIG 4.7 FLOW DIAGRAM OF ADAPTIVE POSITIONING CONTROLLER

directions, we have a value for the time spent at the peak quantization level. If the system is reasonably well behaved, we can assume that the actual zero velocity point occurred at the midpoint of this interval. We then use this estimate to apply a ' stopping pulse' of the opposite polarity and of duration equal to half the time at the peak quantization level. While some overshoot is inherent in this scheme, it is typically on the order of a few quantization levels and usually insignificant. If no stopping pulse is used, the effect of the late switching is more serious, since the next control action must cope with the (unknown) nonzero initial velocity.

The entire sequence of control actions comprising a move can now be diagrammed as shown in Fig.4.7. At t =0, the switchpoint is calculated from present position AM and length of move L = WANT-AM using equation (4.10) or equation (4.11). When the position reaches SWITCHPOINT at t_1, the control is switched from maximum accelerating force to maximum decelerating force. At t_2, turn around is detected and the stopping pulse width calculated. At the end of the stopping pulse, information concerning the move is placed on the update queue, for processing by the concurrent update routine. A decision is then made on whether to initiate another table-based move or to enter PID mode.

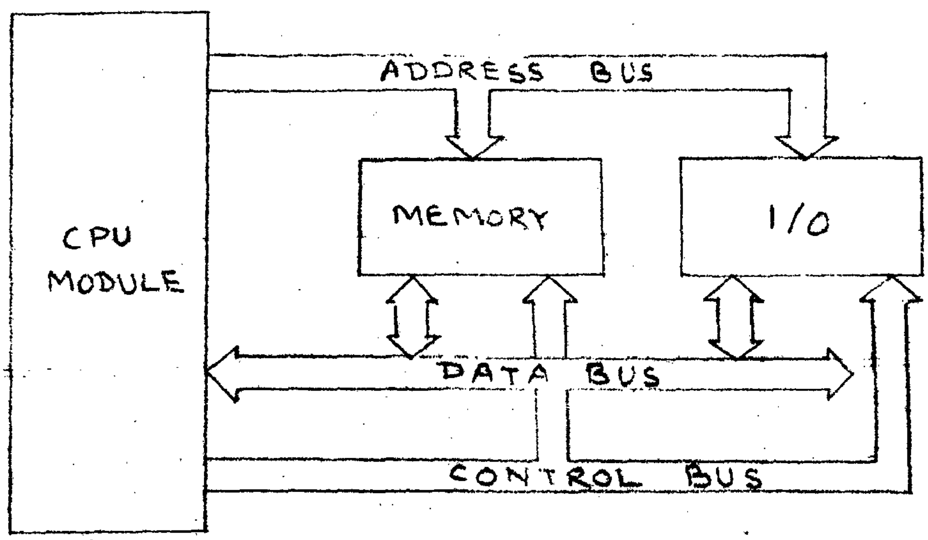


FIG. 4.8 ARCHITECTURE OF MICROCOMPUTER

4.3 MICROCOMPUTER

4.3.1 Microcomputer System (9)

A typical microcomputer consists of -

(a) Central processor unit (CPU)

(b) Memory

(c) Input/Output (I/O) ports

The memory serves as a place to store instructions, the coded pieces of information that direct the activities of the CPU, and data, the coded pieces of information that are processed by the CPU. A group of logically related instructions stored in memory is referred to as a program. The CPU ' reads' each instruction from memory in a logically determined sequence, and uses it to initiate processing actions. If the program sequence is coherent and logical processing the program will produce intelligible and useful results.

The memory is also used to store the data to be manipulated as well as the instructions that direct that manipulation. The program must be organised such that the CPU does not read a non-instruction word when it expects to see an instruction. The CPU can rapidly access any data stored in memory, but often the memory is not large enough to store, the entire data bank required for a particular application. The problem can be resolved by providing the computer with one or more input ports.The CPU can address these ports and input the data contained there. Theaddition of input ports enables the computer to receive information from external equipment at high rates of speed and in large volumes.

A computer also requires one or more output ports that permit the CPU to communicate the result of its processing to the outside world. The output may go to a display, for use by a human operator, to a peripheral device that produces ' hard copy', such as a line-printer, to a peripheral storage devices, such as a floppy disk unit, or the output may constitute process control signals that direct the operations of another system, such as an automated assembly line. Like input ports, output ports are addressable. The input and output ports together permit the processor to communicate with the outside world.

The CPU unifies the system. It controls the functions performed by the other components. The CPU must be able to fetch instructions from memory, decode their binary contents and execute them. It must also be able to reference memory and I/O ports as necessary in the execution of instructions. In addition, the CPU should be able to recognize and respond to certain external control signals, such as INTERRUPT and WAIT requests. The functional units within a CPU (8080A) that enable it to perform these functions are described in chapter 4.33.

4.3.2 A Typical Microcomputer

The MICROCOMPUTER TRAINER Model MCT-1, on which the soft-ware was developed and demonstrated, is a microprocessor

(Intel 8080A) based microcomputer. It is intended to highlight various operations of microprocessors while executing programmes down to machine cycle level. The program can be executed at two speeds, (a) computer speed in RUN MODE and (b) Operator's speed in STEP MODE i.e. either an instruction cycles or machine cycle at a time. This provides a thorough insight into the working of the microprocessor. This knowledge of microprocessor is essential while designing programmable process controllers, signal processing, telecommunication equipment, measuring instruments, consumer products, traffic controllers and interfacing for a peripheral devices like paper tape punch, paper tape reader, teletype, floppy disc controllers, cassette systems, intelligent terminals etc.

This microcomputer trainer provides all facilities to write software, debug, interrupt programmes, acquire computing concepts and peripheral interfacing through sequence of events associated with program execution. The microcomputer makes use of an ergonomically designed keyboard and controls. The keyboard acts as an integrated input device through which programmes are entered, checked and executed.

Technical Features

— 8-bit microprocessor-based system (8080A CPU)

— Fully operational and ready to use

— Key debounce by software

— Check for program entered

— Software controlled display of MEMORY and ADDRESS

- Run-mode for program execution at clock speed
- Single instruction step execution
- Single machine cycle step execution
- Can execute any part of program at clock speed and rest in step-mode.
- 1 K Byte of memory includes executive program and R/W memory
- Provision for memory expansion up to 4K on board, the rest outside the system
- Noise-protected RAM
- Counting feature for manipulating external operation for system design
- Program execution from any desired location
- Test signal observation at I/O connector
- Operates on Octal format
- Machine cycle display
- Status indicator
- Displays DATA, ADDRESS
- Built-in I/O facilities
- Provision for external peripheral interface
- Reserve (bounceless) key 'R' for future use

<u>Specifications</u>

- Microprocessor	-	Intel 8080A
- Word size	-	8 bits
- Capable of Addressing	-	256 I/O lines
- Number of channels used for internal operations	-	8

- Clock rate - 1.3 microseconds
- Number of instructions - 78
- Memory

512 bytes of EPROM is provided of which 256 bytes consist of executive program, the rest left for the user. 512 bytes of RAM is provided for loading programs. Memory expansion facilities are provided ~~I/O Port.~~

- I/O Port

Four 8 bit parallel I/O data lines are available on different ports. 8 bit parallel buffered data bus is available on this I/O connector.

Test signals provided are -

INSTRUCTION FETCH, MEMORY READ, MEMORY WRITE, HALT ACKNOWLEDGE, STACK READ, STACK WRITE, INPUT READ, OUTPUT WRITE, INTERRUPT ACKNOWLEDGE, INTERRUPT ACKNOWLEDGE WHILE HALT, HOLD, SYNC, STROBE, DBIN, READY, INTERRUPT ENABLE, INT, OSC.

8 numbers of plated through printed circuit boards.

Power requirement - 220 volts, 50 Hz

Power dissipation - 21 watts

Weight - 6 kgs

Dimensions - 420 x 555x 122 mm

4.3.3. Architecture of the 8080A CPU

The 8080A CPU consists of the following functional units -

- Register array and address logic
- Arithmetic and logic unit (ALU)
- Instruction register and control section
- Bi-directional, 3-state data bus buffer

Registers

The register section consists of a static RAM array organised into six 16-bit register.

- Program counter (PC)
- Stack counter (SP)
- Six 8-bit general purpose registers arranged in pairs, referred to as B,C; D, E; and H,L.
- A temporary register pair called W,Z.

The program counter mountains the memory address of the next program instruction and is incremented automatically during every instruction fetch. The stack pointer maintains the address of the next available stack location in memory. The stack pointer can be initiated to use any portion of read write memory as a stack. The stack pointer is decremented when data is ' pushed' onto the stack and incremented when data is ' popped' off the stack (i.e. the stack grows ' downward').

The six general purpose registers can be used either as single registers (8-bit) or as register pairs (16 bit). The temporary register pair, W,Z, is not program addressable and is only used for the internal execution of instructions.

Eight-bit data bytes can be transferred between the internal bus and the register array via the register-select multiplexer. Sixteen-bit transfers can proceed between the register array and the address latch or the incrementor/decrementer circuit. The address latch receives data from any of the four register pairs and drives the 16 address output buffers (A_0-A_{15}), as well as the incrementer/decrementer circuit. Theincrementer/decremeter circuit receives data from the address latch and sends it to the register array. The 16-bit data can be incremented or decremented or simply transferred between registers.

Arithmatic and logic Unit (ALU)

The ALU contains the following registers -

- An 8-bit accumulator
- An 8-bit temporary accumulator (ACT)
- A 5-bit flag register, zero, carry, sign, parity and auxiliary carry
- An 8-bit temporary register (TMP)

Arithematic, logical and rotate operations are performed in the ALU. The ALU is fed by the temporary register (TMP) and the temporary accumulator (ACT) and carry flip-flop. The result of the operation can be transferred to the internal bus or to the accumulator, the ALU also feeds the flag register.

The temporary register (TMP) receives information from the internal bus and can send all or portions of it to the ALU, the flag register and the internal bus.

The accumulator (ACC) can be loaded from the ALU and the internal bus and can transfer data to the temporary accumulator (ACT) and the internal bus. The contents of the accumulator (ACC) and the auxiliary carry flip-flop can be tested for decimal correction during the execution of the DAA instruction.

Instruction Register and Control

During an instruction fetch, the first byte of an instruction (containing the OP code) is transferred from the internal bus to the 8-bit instruction register.

The contents of the instruction register are, in turn, available to the instruction decoder. The output of the decoder, combined with various timing signals, provides the control signals for the register array, ALU and data buffer blocks. In addition, the outputs from the instruction decoder and external control signals feed the timing and state control section which generates the state and cycle timing signals.

Data Bus Buffer

This 8-bit bidirectional 3 state buffer is used to isolate the CPU's internal bus from the external data bus (D0 through D7). In the output mode, the internal bus content is loaded into an 8-bit latch that, in turn, drives the data bus output buffers. The output buffers are switched off during input or non-transfer operations.

During the input mode, data from the external data bus is transferred to the internal bus. The internal bus is precharged at the beginning of each internal state, except for the transfer state.

4.3.4 Instruction Set

Memory used in 8080A is organised in 8-bit bytes. Each byte has a unique location in physical memory. That location is described by one of a sequence of 16-bit binary addresses. The 8080A can address upto 64K (K = 1024, or 2^{10}, hence 64K represents the decimal number 65,536) bytes of memory, which may consist of both random-access, read-write memory (RAM) and read-only memory (ROM) which is also random-access.

Data in the 8080A is stored in the form of 8-bit binary integers -

DATA WORD

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

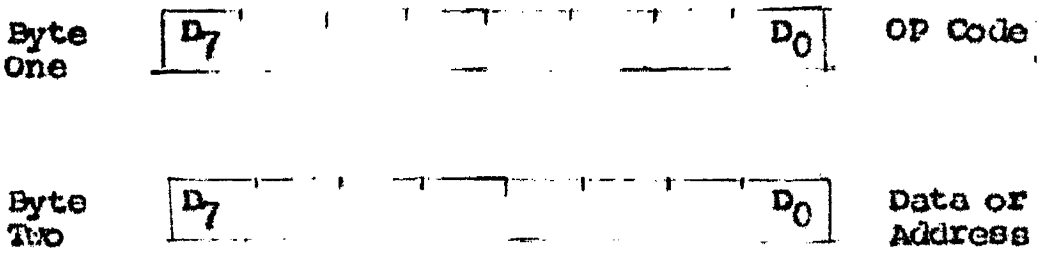
MSB LSB

In 8080A, BIT 0 is referred to as the Least Significant Bit (LSB), and BIT 7 (of an 8-bit number) is referred to as the Most Significant Bit (MSB).

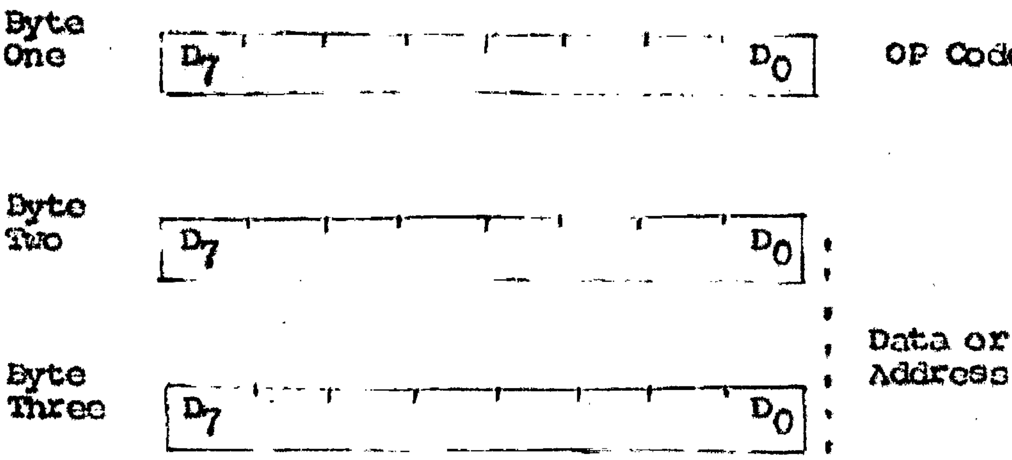
An 8080A programme instruction may be one, two or three bytes in length. Multiple byte instructions must be stored in successive memory locations; the address of the first byte is always used as the address of the instruction. The exact instruction format will depend on the particular operation to be executed.

Single Byte Instructions

Two- Byte Instructions



Three-Byte Instructions



<u>Addressing Modes</u>

The 8080A has four different modes for addressing data stored in memory or in registers -

- Direct - Bytes 2 and 3 of the instruction contain the exact memory address of the data item (the low order bits of the address are in byte 2, the high-order bits in byte 3).

Register - The instruction specifies the register or register pair in which the data is stored.

Register - Indirect The instruction specifies a register pair which contains the memory address where the data is located (the high order bits of the address are in the first register of the rpair the low order bits in the second)

Immediate - The instruction contains the data itself. This is either an 8-bit quantity or a 16-bit quantity (least significant byte first, most significant byte second).

A branch instruction can specify the address of the next instruction to be executed in one of two ways -

Direct - The branch instruction contains the address of the next instruction to be executed.(Except for 'RST' instruction, which is a special one-byte call instruction)

Register - Indirect The branch instruction indicates a register-pair which contains the address of the next instruction to be executed.

Condition Flags

There are five condition flags associated with the execution of instructions on the 8080A. Each is represented by a 1-bit register (or flip-flop) in the CPU. A flag is set by forcing the bit to 1; it is reset by forcing the bit to 0. An instruction affects a flag in the following manner.

Zero - If the result of an instruction has the value 0, this flag is set, otherwise it is reset.

Sign - If the most significant bit of the result of the operation has the value 1, this flag is set, otherwise it is reset.

Parity - If the module 2 sum of the bits of the result of the operation is 0, (i.e. if the result has even parity), this flag is set, otherwise it is reset (i.e. if the result has odd parity).

Carry - If the instruction resulted in a carry (from addition), or a borrow (from subtraction or a comparison) out of the high-order bit, this flag is set, otherwise it is reset.

Auxiliary - Carry If the instruction caused a carry out of bit 3 and bit 4 of the resulting value, the auxiliary carry is set, otherwise it is reset. This flag is affected by single- precision additions, subtractions, increments, decrements, comparisons and logical operations.

Instruction Set

The 8080A instruction set is grouped under five different functional headings, as follows -

(1) Data Transfer Group - This group of instructions transfers data to and from registers and memory. Condition flags are not affected by any instruction in this group.

MOV r_1, r_2	(Move register)	One byte
MOV r, M	(Move from memory)	One byte
MOV M, r	(Move tomemory)	One byte
MOV r, data	(Move Immediate)	Two byte
MOV IM, data	(Move to memory immediate)	Two byte
LXI rp, data 16	(Load register pair immediate)	Three byte
LDA addr	(Load accumulator direct)	Three byte
STA addr	(Store accumulator direct)	Three byte
LHLD addr	(Load H and L direct)	Three byte
SHLD addr	(Store H and L direct)	Three byte
LDAX rp	(Load accumulator indirect)	One byte
STAX rp	(Store accumulator indirect)	One byte
XCHG	(Exchange H and L with D and E)	One byte

(2) Arithmatic Group

This group of instructions performs arithmatic operations on data in registers and memory. Unless indicated otherwise, all instructions in this group affect the zero, sign, parity, carry and Auxiliary carry flags according to the standard rules.

In all subtraction operations, thecarry flag is set to one to indicate a borrow and reset to zero to indicate no borrow.

ADDr	(Add Register)	One byte
ADD M	(Add Memory)	One byte
ADI data	(Add immediate)	Two byte
ADC r	(Add Register with carry)	One byte
ADC M	(Add memory with carry)	One byte
ACI data	(Add immediate with carry)	Two byte
SUB r	(Subtract Register)	One byte
SUB M	(Subtract memory)	One byte
SUI data	(Subtract immediate)	Two byte
SBB r	(Subtract Register with borrow)	One byte
SBI data	(Subtract ~~memory~~ immediate with borrow)	One byte
SBB M	(Subtract memory with borrow)	One byte
INR r	(Increment Register)	One byte
INR M	(Increment memory)	One byte
DCR r	(Decrement Register)	One byte
DCR M	(Decrement memory)	One byte
INX rp	(Increment register pair)	One byte
DCX rp	(Decrement register pair)	One byte
DAD rp	(Add register pair to H andL)	One byte
DAA	(Decimal Adjust Accumulator)	One byte

(3) <u>Logical Group</u>

This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags. Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Auxiliary Carry, and carry flags according to the standard rules.

ANA r	(AND Register)	One byte
ANA M	(AND memory)	One byte
ANI data	(AND immediate)	Two byte
XRA r	(Exclusive or Register)	One byte
XRA M	(Exclusive or memory)	One byte
ORA r	(OR Register)	One byte
ORI data	(OR immediate)	Two byte
ORA M	(OR memory)	One byte
CMP r	(Compare Register)	One byte
CMP M	(Compare memory)	One byte
CPI data	(Compare immediate)	Two byte
RLC	(Rotate left)	One byte
RRC	(Rotate right)	One byte
RAL	(Rotate left through carry)	One byte
RAR	(Rotate right through carry)	One byte
CMA	(Complement accumulator)	One byte
CMC	(Complement carry)	One byte
STC	(Set carry)	One byte

(4) Branch Group

This group of instructions alter normal sequential program flow. Condition flags are not affected by any instruction in this group. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if thespecified branch is to be executed. The conditions that may be specified are as follows -

NZ	-	not zero (Z = 0)
Z	-	zero (Z = 1)
NC	-	no carry (CY = 0)
C	-	carry (CY = 1)
PO	-	parity odd (P = 0)
PE	-	parity even (P = 1)
P	-	plus (S = 0)
M	-	minus (S = 1)

JMP addr	(Jump)	Three byte
J condition addr	(conditional jump)	Three byte
CALL addr	(Call)	Three byte
Ccondition addr	(Conditional call)	Three byte
RET	(Return)	One byte
R condition	(Conditional return)	One byte
RST n	(Restart)	One byte
PCHL	(Jump H and L indirect - move H and L to PC)	One byte

(5) Stack, I/O, and Machine Control Group

This group of instructions performs I/O, manipulates the stack and alters internal control flags. Unless otherwise specified condition flags are not affected by any instructions in this group.

PUSH rp	(Push register pair)	One byte
PUSH PSW	(Push processor status word)	One byte
POP rp	(Pop register pair)	One byte
POP PSW	(Pop processor status word)	One byte
XTHL	(Exchange stack top with H and L)	One byte
SPHL	(Move HL to SP)	One byte
IN port	(Input)	Two byte
OUT port	(Output)	Two byte
EI	(Enable interrupts)	One byte
DI	(Disable interrupts)	One byte
HLT	(Halt)	One byte
NOP	(No operation)	One byte

4.4 SOFT WARE

Considering the potential of table top microprocessor trainer system available for the purpose of development of software for Adaptive Position Controller, some changes have been made in the original algorithm presented in section 4.2. These changes have been listed below -

(1) This has been presumed that the failure of a particular move is not due to any spurious result caused by noise or a disturbance and hence no provision has been made to check the speed of acquisition (i.e. learning speed) and noise rejection.

(2) The links have not been provided for each unknown position in the table with its nearest known neighbor. Instead the nearest known neighbor will be searched as and when it is required. This, however, will give rise to time required for a particular move, in the initial stage of learning (hence not time optimal) but it will work satisfactorily once the controller has started learning. Moreover, if the speed of microcomputer system is quite high, there will not be any problem even in the initial stage of learning. This way we shall be avoiding the complexity in the algorithm.

(3) Instead of placing the data concerning the actual move in a first in/first our queue after the completion of a particular move and updating the table after the completion of controller job, we will be updating the table immediately after the move is completed.

(4) Hystericis has not been provided to avoid cycle in and out of PID control. However this feature can be included in the software if the features of the individual parts of the complete system as well as the process are known.

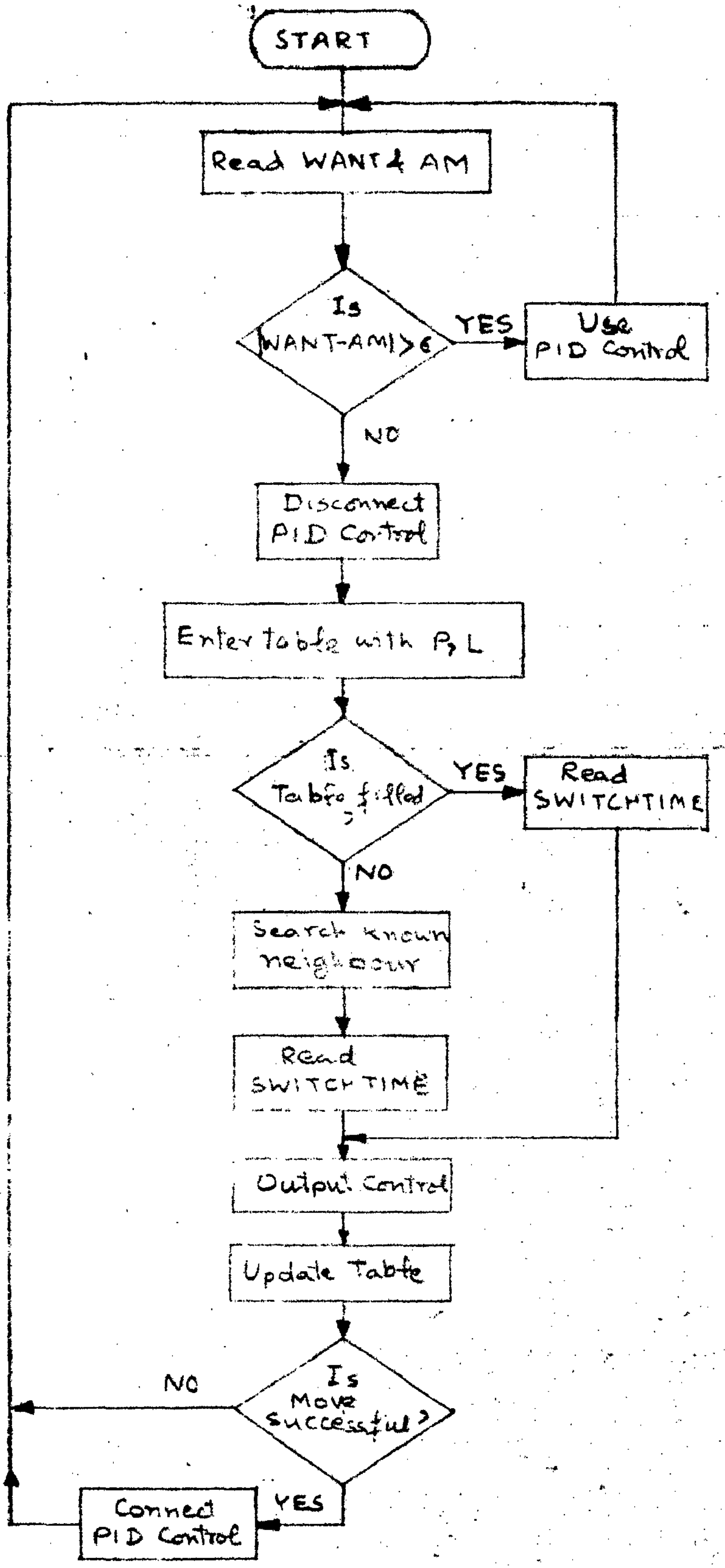


FIG 4.9 MODIFIED FLOW DIAGRAM

The flow chart incorporating these changes has been shown in Fig.4.9.

4.4.1 <u>Algorithm</u>

(1) Read set value (or WANT) from keyboard and current position (or AM) from input port 003. If there is any error (or if error is more than e_{limit}) go to point (2) otherwise repeat the point (1).

(2) Disconnect PID control (output 000 at port 006)

(3) Determine P and L from WANT and AM. Also find out whether L is negative. Compute LO address of the required switch ratio.

(4) Find out if table is filled. If yes, go to point (6), otherwise go to point (5) for searching the nearest entry.

(5) Move toward lower end of table by one from computed address. If table filled, go to point (6), otherwise move towards upper end of table by one from computed address. If table still not filled, repeat the process but shifting by two towards lower or upper ends. Go on increasing this length of shift till filled position is reached. In this process if a particular end is reached, do not search the filled position in that direction any more. If both the ends have reached, it indicates that the table is not having any entry. So interrupt the programme.

(6) Read the switchratio. Multiply it ~~is~~ with L. Add this (or subtract if L is -ve) to AM. This is switchpoint.

(7) If -ve move, go to point (12), otherwise to point (8).

(8) Output 111 at port 004. Read current position from port 003. Compare it with the switch point. If both are not equal repeat point (8), otherwise go to point (9).

(9) Output 222 at port 004.

(10) Call zero speed detection subroutine

(11) Output 111 at port 004. Go to point (16).

(12) Output 222 at port 004. Read current position from input port 003. Compare it with the switch point. If both are not equal, repeat point (12), otherwise go to point (13).

(13) Output 111 at port 004

(14) Call zero speed detection subroutine

(15) Output 222 at port 004.

(16) Wait for half of the zero speed duration measured during zero speed detection.

(17) Output 000 at port 004.

(18) Read current value from input port 003 and compare it with set value (WANT). If move not successful, go to point (21), otherwise go to point (19).

(19) If table was not filled, fill it with the used switch ratio.

(20) Connect PID control (output 377 at port 006). Go to point (1).

(21) Determine P and L assuming END UP as WANT. Compute address.

(22) Compute the new switch ratio = $\frac{\text{Used switch ratio X L}}{\text{END UP - AM}}$

(23) Update the table by computed switch ratio

(24) Go to point (1).

4.4.2. Look-Up Table

Fig. 4.10 shows the table configuration. Positionsare assumed to be 0 through 7. First three LSBS B_0, B_1 and B_2 represent position P or the row. Next three bits B_3, B_4 and B_5 represent length of move L or the column. Seventh bit B_6, represent the direction of move. It is 1 for position move and 0 for negative move.

Lo address can be computed from P and L as follows

B_2 B_1 B 0 ⟶ Present position P.

B_5 B_4 B_3 ⟶ (A) If +ve move subtract 1 from L

(B) If -ve move subtract 1 from L and compliment

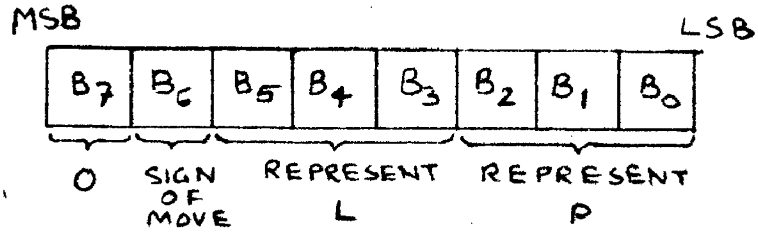
B_6 ⟶ 1 for + ve move

0 for -ve move

B_7 ⟶ 0

Hi address is taken as 005 (Octal)

Field of table | Hi address 005
| Lo address 017- 160



LENGTH OF MOVE L / P POSITION	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7
0														
1														
2														
3														
4														
5														
6														
7														

FIG 4.10 Table

The switch ratios in the table shall be stored in integers only and their unit shall be 0.01. The care has been taken in the software programme in making use of the switch ratio while computing the switch time.

4.4.3 Zero Speed Detection

The current position will be read from input port 005. After a finite delay, the same port will again be read. Go on repeating this till the two readings are same. At this juncture the speed is detected to be zero. Now go on counting time period till the two readings are again unequal (i.e. speed is nonzero). This time-period is the period during which the speed was computed to be zero. Half of this time-period shall be the duration of the stopping pulse.

4.5 INTERCONNECTION OF VARIOUS BLOCKS

Having discussed the software for the adaptive position controller and the microcomputer details, we will now take up how the microcomputer, having all the necessary software behind it, will be connected with other support facilities in order to constitute a complete controller.

Let us assume that the process has a moving part, the position of which is to be controlled as per the set value. There will be a position sensor (see Fig.4.11), which will sense the position of the moving part and will convert it into electrical signal. This signal will now be processed in signal processor.

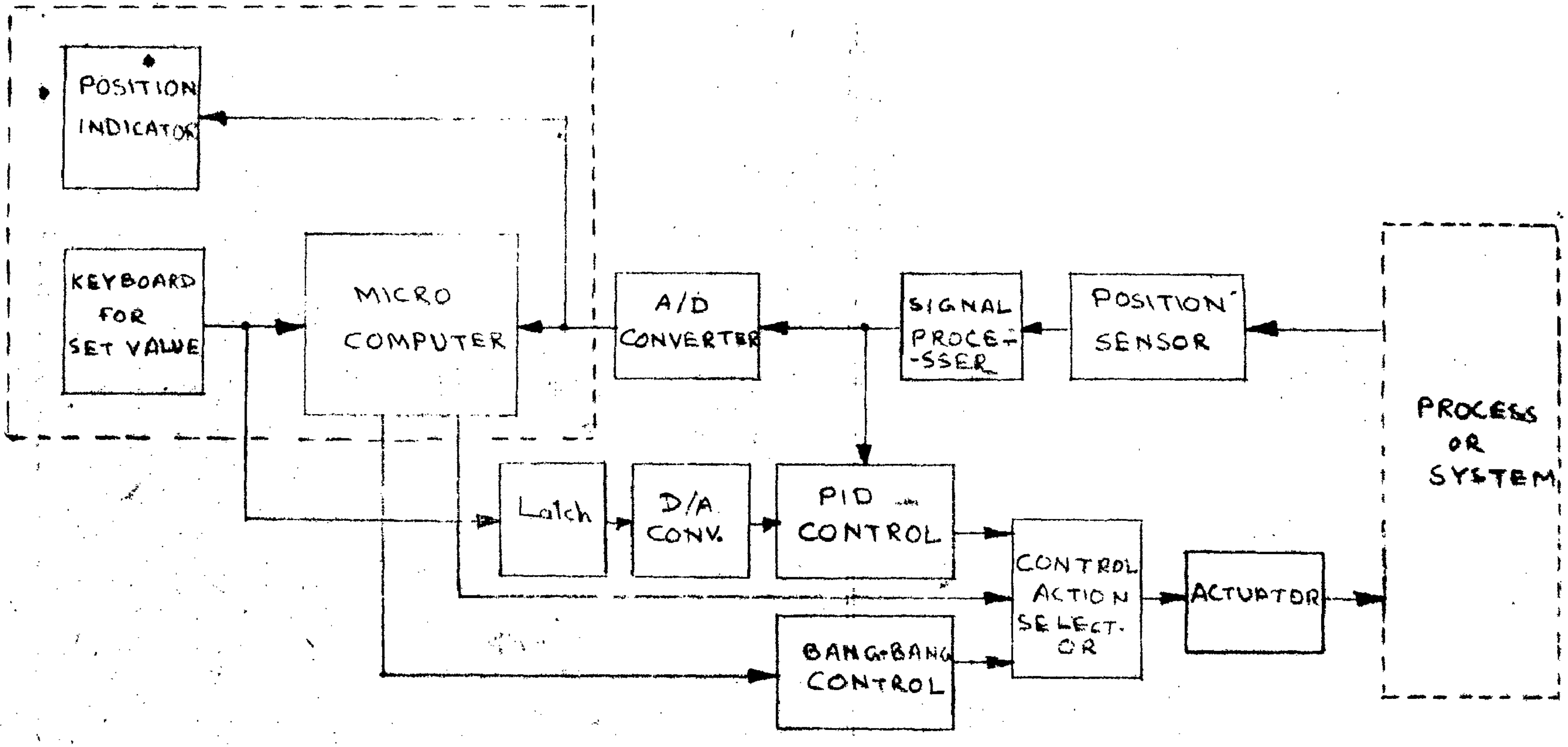


FIG. 4.11 COMPLETE ADAPTIVE POSITION CONTROLLER

A/D convertor will convert this processed analog signal into the digital one. The current position (digital signal) will be provided at the input port of the microcomputer. Simultaneously the current position will be displayed at one of the display port of microcomputer. Set value, i.e. the required position, can be set by the keys available on key board. Now the microcomputer will find out if there is any error in the position. The error may be due to change in the set value or load value or due to change in the process dynamics. If the error is more than the predetermined value e_{in}, the control will enter into bang-bang control. A command to control action selector will be given. If one of the output port (port 001) is 000, the bang-bang control will be selected. The necessary commands, through another output port (port 004), will be given to bang-bang control in order to achieve the set value.

If output is 111 - Apply full force in forward direction

If output is 222 - Apply full force in reverse direction

If output is 000 - Apply no force

For controlling the position, there may be for example, a ±dc motor. By applying +ve, -ve or zero voltage, we can achieve the above functions.

If the error between the set value and the current position is not at all, or is less than e_{in} or it has been made less than e_{in} through bang-bang control, then the microcomputer will output 377 on output port 001. Thus giving command to control action selector to switchover to PID control. In this condition the signal processor will feed the current position to the PID controller, and the set value will be fed by keyboard through latch and D/A converter. Now PID controller will provide the control to the process. The output of PID controller will be in analog form and hence the voltage to the motor will also be in analog form, corresponding to the small error in the position. In the steady-state condition, PID control will provide zero voltage.

Current position and the set value will continuously be monitored by the microcomputer. Whenever the error crosses e_{out}, the control again will be transferred to the bang-bang control.

4.6 DISCUSSION

Computational effort has been kept to a minimum by making use of the data storage and conditional operation capabilities of the digital computer. This keeps the algorithm simple and frees the processor from extensive calculations. This permits the use of a microcomputer which might have limited computational speed. The algorithm also has the ability to tune itself to the process and follow changes in the process without requiring detailed modeling or tuning intervention.

Bang-bang control was selected because of its time optimal performance and because it allows a simple updating of the self-tuning table from measurable data. The primary disadvantage is that bang-bang control is rather 'unforgiving' of errors or delays in the switchpoint locations, as maximum force is applied even near the switchpoints.

While the bang-bang control implemented here is time optimal only for second order systems. It gives reasonable performance for higher order systems which have a dominant pair of poles. The algorithm tunes to give zero error and zero velocity at the setpoint. Higher order systems will thus generally possess some non zero higher derivatives at the setpoint. For systems with dominant second order poles, these derivatives will be small and the linear mode (PID) should be able to bring them to zero without undue position or velocity error, although the responce will no longer be time optimal.

CHAPTER - 5

CONCLUSIONS

or Introduction?

?

Computers are shrinking in both size and price. This is happening as a direct result of advances in very large scale integration techniques. Existing computer architectures shrink. ? Then new applications for computers become possible which were previously unfeasible [10].

In industrial control systems, the shrinking computer has already made it possible for analog process controllers to be implemented in digital circuits, first by sharing a single microcomputer's cost to eight or more controllers. This has developed so far that it is now practical to include one or even two dedicated computers per controller. These developments have led in turn to distribution of these microcomputer-based controllers from the central control room into the plant on control data highways. Distribution of control is increasing for many reasons, including system reliability and reduced wiring costs.

Although process control in the early years started with centralised main frame computers, now with fast development of microelectronics coupled with its cost effectiveness, the distributed control concept is well established.

This means the deployment of more number of 'satellite' or remote computers located close to the process-task. A central or host computer is connected with these satellite puters. The central computer can execute an optimization ting using linear programming and other techniques with goal of maximizing profit. The program can usé the data ained from the satellites to determine the optimum value key operating parameters in accordance with an objective ction and plant operating constraints.

About 50 percent of industrial control loops require intervention of a human operator in order to add an element 'intelligence' to the system. One of the ways in which the erator displays this intellegence is by adapting his behaviour changes in the process as he recognises them. He learns from process [11].

But the advent of cheap computing power in the form of microprocessor and other aids, has meant that more and re of the adaptive functions normally allocated to the human erator, can now be given to the machine. However, despite availability of this computing power in theory, the practi- l implementation of adaptive control systems has been very nited for a number of reasons.

REFERENCES

1. Control Engineering Aug. 1981; Kenneth Crater; Systems approach reduces microprocessor to pneumatic's interfacing costs; pp.86.

2. M.E.Stephen and R.P.Alonsa; Standard Instrumentation Questions and Answers, Vol.II; 1962; pp.307.

3. E.B.Jones; Instrument Technology, Vol.3; 1970; pp 46, 58-72.

4. McGraw-Hill; E.Mishkin and L.Braun; Adaptive Control Systems; 1961, pp 1 to 19.

5. F.W.Kirk and N.R.Rimboi; Instrumentation; 1975; pp 269 - 276.

6. Control Engineering May 1979; Editorial; Control and Microprocessor Technology; pp 47.

7. Control Engineering May 1979; William Bottari; How to Design single chip microcomputers into Control Systems; pp 69-72.

8. Control Engineering May 1981; J.L. Hartman; Adaptive Position Controller Avoids Complex Algorithms; pp 71-74.

9. INTEL Corporation, October 1979; MCS-80/85, Family Users Man ual; pp 4.2 - 4.3.

10. Control Engineering August 1981; Editorial; The Shrinking Computer and industrial control; pp 61.

11. Process Engineering June 1980; Adaptive Control - a Case for its Implementation.

ANNEXURE-1

PROGRAM

Point	Address Hi	Address Lo	Octal Code	Mnemonic	Comments
1	2	3	4	5	6
S1	004	000	046	MOVIH	
	004	001	005	005	
Loop 1	004	002	315	CALL	Read Keyboard (set value)
	004	003	315	315	
	004	004	000	000	
	004	005	062	STA	
	004	006	016	016	Set value stored in 005-016
	004	007	005	005	
S2	004	010	107	MOV B,A	
	004	011	315	CALL	Read current position + display in 003
	004	012	232	...	
	004	013	005	...	
	004	014	117	MOV C,A	
	004	015	220	SUBB	
	004	016	312	JZ	
	004	017	002	Loop 1	
	004	020	004	004	
	004	021	315	CALL	Address P - in C; computation L - in B; Dis000- -ve move; Dis 100- +ve move
	004	022	200	Address	
	004	023	005	Computation	

Annexure-1 (Contd.)

1	2	3	4	5	6
	004	024	315	CALL	
	004	025	325	325	
	004	026	005	005	
	004	027	000	NOP	
	004	030	036	MOVIE	If E remains 000- Table is filled for reqd. move and need not be updated is case of successful move.
	004	031	000	000	
	004	032	176	MOV A,M	
	004	033	247	ANA A	
	004	034	3 2	JNZ	
	004	035	130	57	Jump if table filled for reqd. move
	004	036	004	004	
	004	037	036	MOVIE	E is 111- nearest entry to be searched
	004	040	111	111	
	004	041	325	PUSH D	
	004	042	305	PUSH D	
	004	043	026	LXI D	Lower and upper ends of Table
	004	044	017	017	
	004	045	160	160	
	004	046	034	INRE	
	004	047	006	MOVIB	
	004	050	001	001	

Annexure-1 (Contd.)

1	2	3	4	5	6
	004	051	175	MOV A,L	
Loop2	004	052	220	SUB B	
	004	053	157	MOV L,A	
	004	054	272	CMP D	Check if lower end of table has reached
	004	055	016	MOV IC	If C remains 000- lower end has reached
	004	056	000	000	
	004	057	372	JM	
	004	060	072	53	Jump if lower end has reached
	004	061	004	004	
	004	062	000	NOP	
	004	063	016	MOVIC	
	004	064	111	111	If C is 111 - lower end has not reached
	004	065	176	MOV A,M	
	004	066	247	ANA A	
	004	067	302	JNZ	
	004	070	126	56	Jump if table search is over
	004	071	004	004	
S3	004	072	175	MOV A,L	
	004	073	200	ADD B	
	004	074	200	ADD B	
	004	075	157	MOV L,A	

Annexure-1 (Contd.)

1	2	3	4	5	6
	004	076	273	CMPE	Check if upper end of table is reached
	004	077	372	JM	
	004	100	111	S4	Jump if upper end has not reached
	004	101	004	004	
	004	102	171	MOV A,C	
	004	103	247	ANA A	
	004	104	302	JNZ	
	004	105	117	S5	Jump if lower end has not reached but upper end has reached
	004	106	004	004	
	004	107	373	EI	
	004	110	166	HLT	HLT if table is completely blank
S4	004	111	176	MOVA, M	
	004	112	247	ANA A	
	004	113	000	NOP	
	004	114	302	JNZ	
	004	115	126	S6	Jump if table search is over
	004	116	004	004	
S5	004	117	175	MOV A,L	
	004	120	220	SUB B	
	004	121	157	MOV L,A	
	004	122	004	INR B	
	004	123	303	JMP	

Annexure-1 (Contd.)

1	2	3	4	5	6
	004	124	052	loop 2	
	004	125	004	004	
S6	004	126	301	POP B	
	004	127	321	POP D	
S7	004	130	062	STA	
	004	131	014	014	Switching ratio stored in 005-014
	004	132	005	005	
	004	133	325	PUSH D	'
	004	134	046	MOVIH	'
	004	135	000	000	'
	004	136	120	MOVD,B	'
	004	137	137	MOVE, A	'
	004	140	257	XRA A	' Switch ratio XL
	004	141	203	ADDE	' Result in H and L
	004	142	376	CPI	' Integers in H
	004	143	144	144	' and Decimal figures in L
	004	144	372	JM	'
	004	145	155	S8	'
	004	146	004	004	'
	004	147	044	INRH	'
	004	150	326	SUI	'
	004	151	144	144	'
	004	152	303	JMP	'

Annexure-1 (Contd.)

1	2	3	4	5	6
	004	153	142	Loop 4	
	004	154	004	004	
88	004	155	025	DCRD	
	004	156	302	JNZ	
	004	157	141	Loop 3	Switch ration XL
	004	160	004	004	
	004	161	157	MOV L,A	
	004	162	321	POP D	
	004	163	042	SHLD	
	004	164	161	161	
	004	165	005	005	
	004	166	172	MOV A,D	
	004	167	247	ANA A	
	004	170	312	JZ	
	004	171	236	89	Jump if -ve move
	004	172	004	004	
	004	173	171	MOV A,C	
	004	174	204	ADD H	
	004	175	147	MOV H,A	
	004	176	076	MOV IA	
	004	177	111	111	
	004	200	323	OUT	
	004	201	004	004	

Annexure-1 (Contd.)

1	2	3	4	5	6
Loop5	004	202	315	CALL	In 003
	004	203	232	...	
	004	204	005	...	
	004	205	274	CMPH	
	004	206	372	JM	
	004	207	202	Loop5	
	004	210	004	004	
Loop6	004	211	315	CALL	
	004	212	060	060	
	004	213	006	006	
	004	214	275	CMPL	
	004	215	372	JM	
	004	216	211	Loop6	
	004	217	004	004	
	004	220	076	MOVIA	
	004	221	222	222	
	004	222	323	OUT	
	004	223	004	004	
	004	224	315	CALL	Zero speed detection
	004	225	267	Zero speed	
	004	226	005	Detection	

Annexure-1 (Contd.)

1	2	3	4	5	6
	004	227	076	MOVIA	
	004	230	111	111	
	004	231	323	OUT	
	004	232	004	004	
	004	233	303	JMP	
	004	234	314	S11	
	004	235	004	004	
S9	004	236	175	MOV A,L	
	004	237	247	ANAA	
	004	240	312	JZ	
	004	241	250	S10	
	004	242	004	004	
	004	243	076	MOVIA	
	004	244	144	144	
	004	245	225	SUBL	
	004	246	157	MOVL, A	
	004	247	044	INRH	
S10	004	250	171	MOV A,C	
	004	251	224	SUB H	
	004	252	147	MOV H,A	
	004	253	076	MOVIA	
	004	254	222	222	
	004	255	323	OUT	

Annexure-1 (Contd.)

1	2	3	4	5	6
	004	256	004	004	
	004	257	000	NOP	
	004	260	044	INRH	
Loop7	004	261	315	CALL	In 003
	004	262	232	...	
	004	263	005	...	
	004	264	274	CMPH	
	004	265	362	JP	
	004	266	261	Loop 7	
	004	267	004	004	
	004	270	064	INRL	
Loop8	004	271	315	CALL	In 005
	004	272	060	060	
	004	273	006	006	
	004	274	275	CMPL	
	004	275	362	JP	
	004	276	271	Loop 8	
	004	277	004	004	
	004	300	076	MOVIA	
	004	301	111	111	
	004	302	323	OUT	
	004	303	004	004	
	004	304	315	CALL	Zero speed detection

Annexure-1 (Contd.)

1	2	3	4	5	6
	004	305	267	Zero speed	
	004	306	005	Detection	Zero speed detection
	004	307	000	NOP	
	004	310	076	NOVIA	
	004	311	222	222	
	004	312	323	OUT	
	004	313	004	004	
S11	004	314	054	INRL	
Loop9	004	315	315	CALL	
	004	316	277	Time	Time delay
	004	317	000	Delay	
	004	320	055	DCRL	
	004	321	302	JNZ	
	004	322	315	Loop 9	
	004	323	004	004	
	004	324	076	MOVIA	
	004	325	000	000	
	004	326	323	OUT	
	004	327	004	004	
	004	330	072	LDA	
	004	331	016	016	Load set value
	004	332	005	005	

Annexure-1 (Contd.)

2	3	4	5	6
004	333	147	MOV H,A	
004	334	315	CALL	In 003
004	335	232	...	
004	336	005	...	
004	337	274	CPMH	
004	340	000	NOP	
004	341	302	JNZ	
004	342	366	S12	Jump if move not success ful
004	343	004	004	
004	344	173	MOV A, E	
004	345	247	ANAA	
004	346	312	JZ	
004	347	163	S17	If zero table not to be updated so jump to initial point
004	350	005	004	
004	351	072	LDA	
004	352	015	015	Load address of reqd. entry
004	353	005	005	
004	354	157	MOV L,A	
004	355	046	MOV,1,H	
004	356	005	005	
004	357	072	LDA	
004	360	014	014	Load switch ratio
004	361	005	005	
004	362	167	MOV,M,A	

Annexure-1(Contd.)

1	2	3	4	5	6
	004	363	303	JMP	
	004	364	163	S17	
	004	365	005	005	
S12	004	366	107	MOV B,A	
	004	367	046	MOV IH	
	004	370	005	005	
	004	371	000	NOP	
	004	372	171	MOV, A,C	
	004	373	220	SUBB	
	004	374	312	JZ	
	004	375	006	S13	
	004	376	005	005	
	004	377	315	CALL	Address computation
	005	000	200	Address	
	005	001	005	Computation	
	005	002	315	CALL	Switch ratio computation
	005	003	000	000	
	005	004	006	006	
	005	005	167	MOV M,A	
S1 3	005	006	072	LDA	
	005	007	016	016	
	005	010	005	005	
	005	011	303	JMP	

Annexure-1 (Contd.)

1	2	3	4	5	6
	005	012	010	S2	
	005	013	004	004	
S17	005	163	076	MOVIA	
	005	164	377	377	
	005	165	323	OUT	
	005	166	002	001	
	005	167	303	JMP	
	005	170	000	000	
	005	171	004	004	
	005	325	062	STA	
	005	326	015	015	
	005	327	005	005	
	005	330	257	XRAA	
	005	331	323	OUT	
	005	332	001	001	
	005	333	311	RET	
					Address Computation Subroutine
	005	200	372	JM	
	005	201	213	S14	
	005	202	005	005	
	005	203	107	MOV B,A	
	005	204	026	MOVID	
	005	205	000	000	

Annexure-1 (Contd.)

1	2	3	4	5	6
	005	206	075	DCRA	
	005	207	057	CMA	
	005	210	303	JMP	
	005	211	221	S15	
	005	212	005	005	
S14	005	213	026	MOVID	
	005	214	100	100	
	005	215	170	MOV A,B	
	005	216	221	SUBC	
	005	217	107	MOV B,A	
	005	220	075	DCRA	
S15	005	221	007	RLC	
	005	222	007	RLC	
	005	223	007	RLC	
	005	224	346	ANI	
	005	225	070	070	
	005	226	262	ORAD	
	005	227	261	ORAC	
	005	230	157	MOVL,A	
	005	231	311	RET	
	005	232	000	NOP	IN 003 Subroutine
	005	233	305	PUSH B	
	005	234	365	PUSH PSW	

Annexure-1(Contd.)

1	2	3	4	5	6
	005	235	333	IN	
	005	236	003	003	
	005	237	107	MOV B,A	
	005	240	315	CALL	
	005	241	335	Time	Time Delay
	005	242	005	Delay	
	005	243	333	IN	
	005	244	003	003	
	005	245	270	CMPB	
	005	246	312	JZ	
	005	247	262	516	
	005	250	005	005	
	005	251	016	MOVIC	
	005	252	062	062	
Loop10	005	253	315	CALL	
	005	254	277	Time	
	005	255	000	Delay	
	005	256	015	DCRC	
	005	257	302	JNZ	
	005	260	253	Loop 10	
	005	261	005	005	

Annexure-1 (Contd.)

1	2	3	4	5	6
S16	005	262	361	POP PSW	
	005	263	170	MOV A,B	
	005	264	301	POP B.	
	005	265	311	RET.	
	005	266	000	NOP	
	005	267	333	IN Zero speed Detection Subroutine	
	005	270	005	003	
Loop11	005	271	147	MOV H,A	
	005	272	315	CALL	
	005	273	277	Time	
	005	274	000	Delay	
	005	275	333	IN	
	005	276	005	003	
	005	277	274	CMPH	
	005	300	302	JNZ	
	005	301	271	Loop 11	
	005	302	005	005	
	005	303	056	MOVIL	
	005	304	000	000	
Loop12	005	305	054	INRL	
	005	306	315	CALL	
	005	307	277	Time	
	005	310	000	Delay	

Annexure-1(Contd.)

1	2	3	4	5	6
	005	311	333	IN	
	005	312	003	003	
	005	313	274	CMPH	
	005	314	312	JZ	
	005	315	305	Loop12	
	005	316	005	005	
	005	317	257	XRAA	
	005	320	205	ADDL	
	005	320	037	RAR	
	005	322	311	RET	
	005	335	323	OUT	
	005	336	000	000	
	005	337	315	CALL	
	005	340	277	Time	
	005	341	000	Delay	
	005	342	311	RET	

Annexure-1 (Contd.)

1	2	3	4	5	6
	006	000	345	PUSH H	
	006	001	325	PUSH D	
	006	002	052	LHLD	
	006	003	161	161	
	006	004	005	005	
	006	005	174	MOVA,H	
	006	006	247	ANA A	
	006	007	175	MOVA,L	
	006	010	026	MOVID	
	006	011	000	000	
	006	012	152	MOV L,D	
	006	013	312	JZ	
	006	014	030	S19	
	006	015	006	006	
Loop13	006	016	306	ADI	
	006	017	144	144	
	006	020	322	JNC	
	006	021	024	S18	
	006	022	006	006	
	006	023	024	INRD	

Annexure-1(Contd.)

1	2	3	4	5	6
S18	006	024	045	DCRH	
	006	025	302	JNZ	
	006	026	016	Loop13	
	006	027	006	006	
S19	006	030	137	MOVE, A	
Loop14	006	031	173	MOV A,E	
	006	032	220	SUB B	
	006	033	137	MOVE,A	
	006	034	302	JNC	
	006	035	046	S20	
	006	036	006	006	
	006	037	172	MOV A,D	
	006	040	326	SUI	
	006	041	001	001	
	006	042	372	JM	
	006	043	052	S21	
	006	044	006	006	
	006	045	127	MOVD,A	
S20	006	046	054	INRL	
	006	047	303	JMP	
	006	050	031	Loop14	
	006	051	006	006	

Annexure-1(Contd.)

1	2	3	4	5	6
S21	006	052	175	MOVA, L	
	006	053	321	POPD	
	006	054	341	POPH	
	006	055	311	RET	
	006	060	305	PUSH B	IN 005 Subroutine
	006	061	365	PUSH PSW	
	006	062	333	IN	
	006	063	005	005	
	006	064	107	MOV B,A	
	006	065	315	CALL	
	006	066	277	Time	
	006	067	000	Delay	
	006	070	333	IN	
	006	071	005	005	
	006	072	270	CMPB	
	006	073	312	JZ	
	006	074	262	S22	
	006	075	006	006	
	006	076	016	MOVIC	
	006	07[illegible]	06[illegible]	062	
Loop15	006	100	315	CALL	
	006	101	277	Time	
	006	102	000	Delay	

Annexure-1(Contd.)

1	2	3	4	5	6
	006	103	015	DCRC	
	006	104	302	JNZ	
	006	105	100	Loop 15	
	006	106	006	006	
822	006	167	361	POP PSW	
	006	110	170	MOV A, B	
	006	111	301	POP B	
	006	112	311	RET	