A STUDY OF A SPECIAL PURPOSE AUTOMATIC OPTIMIZER

by

WILLIAM LAWRENCE WRIGHT

B.A.Sc., University of Waterloo, 1963

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

<u>MASTER OF APPLIED SCIENCE</u>

in the Department of

Electrical Engineering

We accept this thesis as conforming to the

required standard.

Members of the Department

of Electrical Engineering

THE UNIVERSITY OF BRITISH COLUMBIA

August, 1965

ABSTRACT

Small special purpose digital computers (SPDC) could be used to control processes for which the cost of general purpose digital computers is prohibitive. This thesis describes a SPDC to optimize a process for which an exact mathematical model does not exist. The SPDC could use any of the empirical or trial and error methods originally designed for hand calculations or for use on a large general purpose digital computer. The methods discussed in this thesis are gradient search, direct search and random search.

The overall operation of a SPDC is described in detail using logic block symbols. From the knowledge gained in building and testing the computer, improvements in circuitry and search strategy are suggested.

The logic and circuitry used in a SPDC depends on the nature of the process to be controlled. This is illustrated in the thesis by the description of the optimization of a flotation process.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ACKNOWLEDGEMENT

Grateful acknowledgement is given to Northern Electric Co. for a fellowship awarded in 1963 and to the National Research Council for financial support from Block Term Grant A68.

The author would like to thank the supervisor of this project, Dr. E. V. Bohn, and all other staff members and graduate students of the Electrical Engineering Department for their assistance during the course of this study. The author is also indebted to W. K. Gowans of The Consolidated Mining and Smelting Co. for suggesting the example in Chapter V.

A STUDY OF A SPECIAL PURPOSE AUTOMATIC OPTIMIZER

1. INTRODUCTION

In the last few years there has been an increase in the number of computers used in industrial controls,[1] and an accompanying expansion of the literature about them.[2] Since many companies are now actively investigating computer use, an even greater increase is anticipated. Despite this widening interest and a general lowering of computer and computer component prices, there are still many processes to which optimal control techniques cannot be applied because of the high cost of available general purpose computers relative to the expected increase in returns due to optimal control.

The object of this thesis is to outline a special purpose low cost digital controller for on-line optimal control. This type of unit could be used to drive a single loop or small process to a steady state optimum. The controller is also applicable to the quasi-static process where the optimum operating point varies with uncontrolled plant parameters which change slowly relative to the dominant response time of the process. This digital controller could also be used in conjunction with a high speed analog computer to control dynamic processes as suggested by Bohn.[3, 4]

Alone, or in conjunction with the high speed analog computer, the action of the digital controller will be the same. The process, or plant, would be such that a mathematical model cannot be completely prepared because of the complexity of the process. Even if a mathematical model can be prepared, it would be too complex to be solved analytically and is complicated further by time varying parameters. The optimum is determined by experimenting with the process, sometimes referred to as the direct method of optimizing.

The changes made by the controller in the controlled variables are based on past responses of the performance function P. Various strategies have been developed to find the optimum under these conditions. The decision of what strategy to instrument would be based on what is known of the process and performance function surface. The Quarie controller and OPCON are commercial controllers using this technique.[5,6,7,8] An analog computer of this type has been suggested by Feldbaum.[9,10]

There are several well-documented search strategies for maximizing mathematical functions, usually intended for use on a large general purpose computer. This thesis describes how standard components could be used to instrument one such search strategy for direct process optimization. The general purpose digital computer has a large minimum cost because the same control and arithmetic units are required regardless of computer size. As the digital controller described here requires no program flexibility and works at speeds comparable to the process time constants, a very inexpensive unit should be possible. This thesis shows how the performance of the circuits in the controller may change the basis for choosing a search strategy, and concludes with a description of an industrial process which might use such a controller.

2. CONTROL STRATEGIES

The term "control strategy" is used to describe the procedure by which the control parameters of a process are varied in order to operate the process at the desired optimum. For simplicity, the optimum operating point will be considered a minimum of P. The performance function surface will be considered unimodal (one minimum), a realistic assumption as the multimodal case becomes unimodal once the lowest minimum has been found. Some authors suggest handling the multimodal case by repeated use of a control strategy designed for the unimodal case using different parameter values for starting points.[11] The "no-error" cases where the control parameters and P are assumed to be known exactly will be considered here, as their principles are simple and the control strategies based on them can be used even if the assumption is not quite true. Process noise and instrumentation error which exist in most industrial cases might necessitate use of a variation (using a statistical technique[12]) of the "no-error" strategies. The controlled system as suggested by Bohn[3,4] can be considered error free.

Again for simplicity and ease of description, the process will be considered as shown in Figure 2-1 with X and Y as two controlled parameters, v as one uncontrolled parameter, and the performance function P as the output to the control computer.

The control computer must vary the inputs X and Y in discrete steps as the effect of changing a variable is not immediately measurable because of time lags in the process. Each new control parameter setting must be held until the static response to the change can be measured.

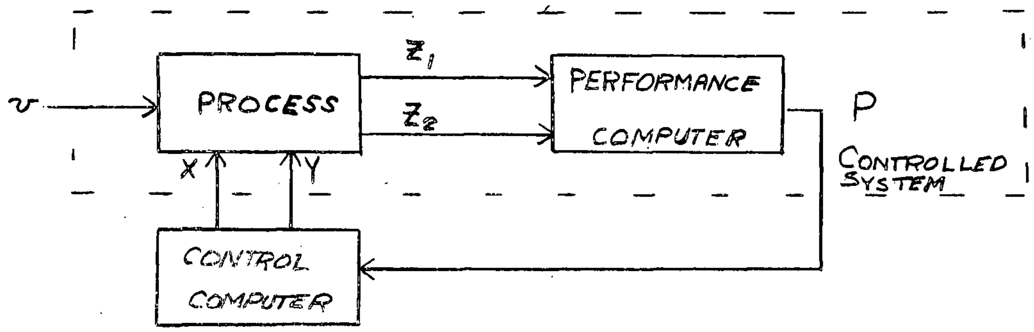


Figure 2-1 Block Diagram of Direct Optimizing

2.1 Nonsequential Strategies

In nonsequential strategies the entire performance function is systematically explored. Each new setting of X and Y is independent of past responses. This type of strategy, while useful for multimodal performance surfaces, is not as applicable to the problem outlined in the introduction as the sequential strategies.[13]

2.2 Sequential Strategies

Sequential strategies are the types of control strategies best suited to a special purpose digital computer (SPDC) with limited memory capacity. Sequential strategies are often referred to as "hillclimbing" because the physical ascent of a hill involves similar decisions.[14]

In sequential strategies, each new setting of X and Y is dependent on one or more past responses to step changes in X and Y. Thus the changes in X and Y cannot be predetermined as in nonsequential strategies because they depend on the unknown performance surface and on the starting values of X and Y.

There is extensive literature on sequential control strategies and much overlapping of ideas and terms. In general, the sequential strategies may be divided into three main types: gradient, direct, and random search.

Gradient methods explore the variations of P with respect to parameter changes to find the local gradient or direction of greatest slope of the P surface in the neighbourhood of the starting point and move the control parameters in this direction. In the univariate gradient method all variables except one are held constant and this one is varied to obtain an improvement in P. Thus X_1 would be held and Y varied until Y_2, the value of Y which gives the largest possible P (X_1, Y), is found. Then Y is held at Y_2 and X is varied, etc. One of the many possible variations is to start varying X if P (X_1, Y_2) is not found in a specified number of steps of Y. The univariate technique works quite well as long as the control parameter axes are parallel to the axes of the contour surfaces of P.[15] Even with an advantageous alignment of axes, this method requires many more moves to arrive at the minimum than other methods which move all parameters simultaneously.

Another gradient strategy similar to the univariate method but which makes simultaneous steps in all control parameters is called relaxation. The local P surface is explored to find the gradient direction and then all control parameters are stepped simultaneously to move the operating point in this gradient direction. These steps are continued until no further improvement in P (local minimum) is possible in this direction and then a new local gradient is found. After the first step there is no reason to assume the moves are still in the direction of the local gradient and this is one of the factors that lead to many variations on this general strategy. For example, the change ΔP_N in P for the Nth step could be monitored and if ΔP_N becomes less than a predetermined percentage of ΔP_1 in the particular direction being moved, a new gradient direction could be found immediately rather than continuing on to find the local minimum. A variation that would not require the storage of ΔP_1 is to use either the finding of the local minimum or the Nth control parameter step as the signal to find a new local gradient.

Perhaps the best known of the gradient methods is "steepest descent" which lets the N of the previous variation be 1. Thus a new local gradient is found after each step. This enables the maximum advantage to be gained from each step of the parameters once the gradient direction is known. If finding the gradient direction is very time consuming, the relaxation method may find the minimum in a much shorter time than steepest descent. In minimizing a mathematical function, the gradient direction is found by calculating the derivatives of the function by any suitable method. With the process as outlined in the introduction, these derivatives cannot be calculated. The gradient is found by sequentially changing each control a known amount, δX, waiting any required dead time or until all transient effects have settled and then measuring the resulting δP_X. Then assuming $k \delta P_X \approx \delta P_X / \delta X$, a move can be made in the gradient direction by simultaneously stepping the control parameters an amount $\Delta X = -k \delta P_X$, $\Delta Y = -k \delta P_Y$.

In most industrial processes the required delay, T_p, between stepping δX and the measurement of δP_X is so much greater than any computation time required by the SPDC that, using a steepest descent strategy, a large percentage of search time is spent finding the gradient direction rather than improving the operating condition of the process. But on any irregular P surface the steepest descent method may arrive at the optimum operating condition in shorter time than the relaxation method. It should be pointed out that any direction of search which improves P can be made the steepest descent direction by a proper change of scale of the control parameters.[11,16,17] This indicates a basic problem of all gradient methods; moving in the gradient direction can be no better or no worse than moving in any other direction. If the contour lines of P are circles, the gradient direction at any operating point will lead directly to the minimum, but, if the P contours are any shape other than circular, there may be very little correlation between the gradient direction at an operating point and the direction from the same operating point to the minimum.

Direct search[18] or trial-and-error[17] strategies are in no way dependent on a gradient direction. The control parameters are stepped in any direction which the previous responses have shown to improve P. One type of direct search is to make simple exploratory moves of each control parameter and use the improvement or lack of improvement of P (no interest in relative sizes of δP, only in sign of δP) as a guide to direction of search. This exploration is followed by a "pattern" step in the direction indicated by the exploration. For any P surface having noncircular contours (which would seem to include many industrial processes), the direct search strategies should be just as good if not better than any gradient method. One advantage of direct search strategies for use in a SPDC is that they tend to use repeated identical arithmetic operations with a simple logic.

In random search strategies the control parameters are stepped in a randomly chosen direction about a base operating point. If P is improved at any chosen operating point, it becomes the new base point for continued random moves. It might seem obvious to prefer a procedure which makes some use of previous responses in planning the next move but a random search strategy is efficient at optimizing very irregular P surfaces. Some tests have shown random strategies better than systematic control strategies for a process with few inputs (less than three) if the process can be described by algebraic equations.[15] With a process as outlined in the introduction where time T_p is required after each parameter step before δP

can be measured, the random strategies do not perform as well as the gradient methods.[14]

All the basic sequential strategies so far considered will operate, with various degrees of success, using only a single step size throughout the entire search for the optimum. A much improved performance (fewer steps to converge to the optimum) is possible if adjustment of step size is used. In this substrategy, the usually suggested pattern is to use coarse steps in the initial stages and finer steps in the latter stages of the search to improve resolution and reduce overshoot and hunting losses. There is a large number of criteria for changing step size and most modify the initial step size depending on the success of the previous step taken (if P was improved, the step was a success). The effectiveness of any search strategy is dependent on this initial step size.[20] The "best" initial step size will depend on the particular process under consideration. For an unknown process, an adjustment of step size is very important to correct any initial poor choice of step size.

One possible strategy might be to double the step size if the last step was a success. This could result in rapid convergence to the optimum in the initial search but would overshoot the optimum. In the gradient strategies the steps are proportional to the previous response, $\Delta X = -k\,\delta P_X$, and k can be reduced as the minimum is approached. This could be done by making k smaller for each negative ΔP (an overshoot). If k is only made smaller than the initial k, the controller could spend a long time searching in an area of relatively flat P.

Another possibility is to use the frequency of overshoot as a step size criterion.[12] For example, if four overshoots occur in sequence, the step size is considered too large; if four successes occur in sequence, the step size is considered too small. This type of step size adjustment is used in direct search strategies where the size of ΔP is not measured. The step size is adjusted by a predetermined amount dependent only on the success of the previous step.

Constraint handling is another substrategy that supplements any of the basic sequential strategies such as gradient search or direct search. A constraint is usually a physical consideration which prohibits certain areas of the P surface in which the operating point cannot or should not exist. "Hard constraints" such as maximum temperature or maximum flow obtainable cannot be violated. "Soft constraints" are those which should not be violated as they represent areas of operation where undesirable effects start to appear.

The constraint may be a known function of the system parameters, such as $XY \geq C$, or some measure might have to be taken directly to sense constraint violation. Two possible methods for handling constraints are "hemstitching" and "riding the constraint".[21] In "hemstitching", as used in a basic gradient search, the control parameters are moved in the direction of the gradient of P if the constraint is not violated. Once the constraint is violated the parameters are moved in the direction of the gradient of the constraint. Like all search techniques considered in this thesis, there are many variations on this idea, but basically the parameters are adjusted to improve P until the constraint is violated and then the parameters are adjusted with the aim of satisfying the constraint. Hemstitching requires the constraint to be violated. This violation is not acceptable in any controller empirically optimizing an industrial process with hard constraints. In "ride the constraint" method the gradient direction of P is always found, but in a zone within one step of the constraint (assuming the constraint is a known equation) only a limited parameter step is allowed so that the operating point will tend to follow the constraint line and the constraint is never violated. This nonviolation of the constraint assumes that the process variables (e.g. temperature) follow closely, without overshoot, the corresponding control set point (e.g. thermostat setting). This is part of the overall assumption that, to be effectively optimized, a system must be closely controlled. Unlike hemstitching, the "ride" method does not oscillate about the constraint. This would seem to make a search strategy with a "ride" substrategy converge to the optimum in fewer steps than one with a hemstitching substrategy.

For any particular process, some control strategies will be better than others. The choice of what basic control strategy (and substrategies) to implement in a SPDC will depend on the knowledge of the P surface, and on the speed and accuracy required by the control computer. Many detailed comparisons of search strategies over various types of P surfaces are available.[13,15,20]

2.3 Details of a Specific Search Strategy

Assume that it is required to operate a process as in Figure 2-1 at a minimum of P and satisfy the constraint $XY \geq C$, and that, based on the knowledge of the process, a relaxation search has been chosen. The search is to be divided into two stages, the initial or coarse stage to quickly locate a neighbourhood of the optimum, and a final or fine stage to obtain and track the optimum operating point. If ΔP_K, the change in P resulting from the K^{th} step in the coarse search, is less than $\Delta P_1/A$ of the same gradient direction, a new gradient direction is to be found. The step size is to be made smaller with each overshoot and, if the coarse search is at its smallest step size, an overshoot causes the change from coarse to fine search. In the fine search stage, the control parameters are to be stepped sequentially n times in the local gradient direction using the smallest possible step which will still produce a recognizable δP.

For convenience in the instrumentation the choice $A = 2$, $n = 7$ is made and only two step sizes k and k/2 will be allowed. While this arbitrary choice may not create the best possible gradient search, it is suitable to test the ability of simple SPDC to carry out a specific search strategy. This search is very general and should be able to optimize, to some degree, any type of P surface. It also has the advantage, for this thesis, of being complicated enough that, in instrumenting a SPDC to carry it out, the type of problems and considerations of instrumenting any search strategy by simple logic units should be encountered.

Once the search strategy is chosen, the next step in designing a SPDC is to specify the sequence of operations within the search. The flow diagrams in Figures 2-2 to 2-4 show one possible organization of operations enabling the control computer to carry out the required search. The search is shown broken down into subroutines.

The command subroutine checks whether to continue in coarse search, alternating subroutines mode 1 and mode 2, or to change to mode 4, the subroutine for fine search. When control is transferred from the fine search subroutine back to the command subroutine, the position of switch #2 (S2) determines whether fine search will be continued or the entire search procedure will be repeated (i.e. return to coarse search). If at any time during the search the constraint is violated, control is immediately transferred to mode 3, the constraint handling subroutine. This is shown by a separate flow diagram labelled "priority".

Mode 1 is the subroutine which determines the local gradient direction. X and Y are sequentially perturbed δX and δY; the size and sign of the resulting δP_X and δP_Y are stored.

Mode 2 simultaneously changes X and Y amounts $\Delta X = -k\delta P_X$ and $\Delta Y = -k\delta P_Y$. The step size, k or k/2, is chosen by the control subroutine and δP_X and δP_Y are provided by mode 1. These steps are continued until ΔP is negative or

$\Delta P_K < \frac{1}{2}\Delta P_1$.

Mode 3 is the constraint handling subroutine. Since the constraint is a known equation of the two control parameters, it is possible to handle this constraint in the same manner as an equality constraint when solving a set of algebraic equations, that is, to reduce the number of variables. As long as $XY = C$, Y is made a dependent variable. The constraint equation gives the relation between ΔX and ΔY and the regular search is continued. This is a variation of the "ride the constraint" substrategy, mentioned in section 2.2, as the search can move along the constraint but never cross it.

Mode 4, the fine search stage, takes seven steps sequentially, stepping X and Y amounts δX and δY. The directions of the steps are determined by the polarity of the previous δP_X and δP_Y. These step sizes are used in both mode 1 and mode 4. They should be the smallest step which produces a recognizable change in P.

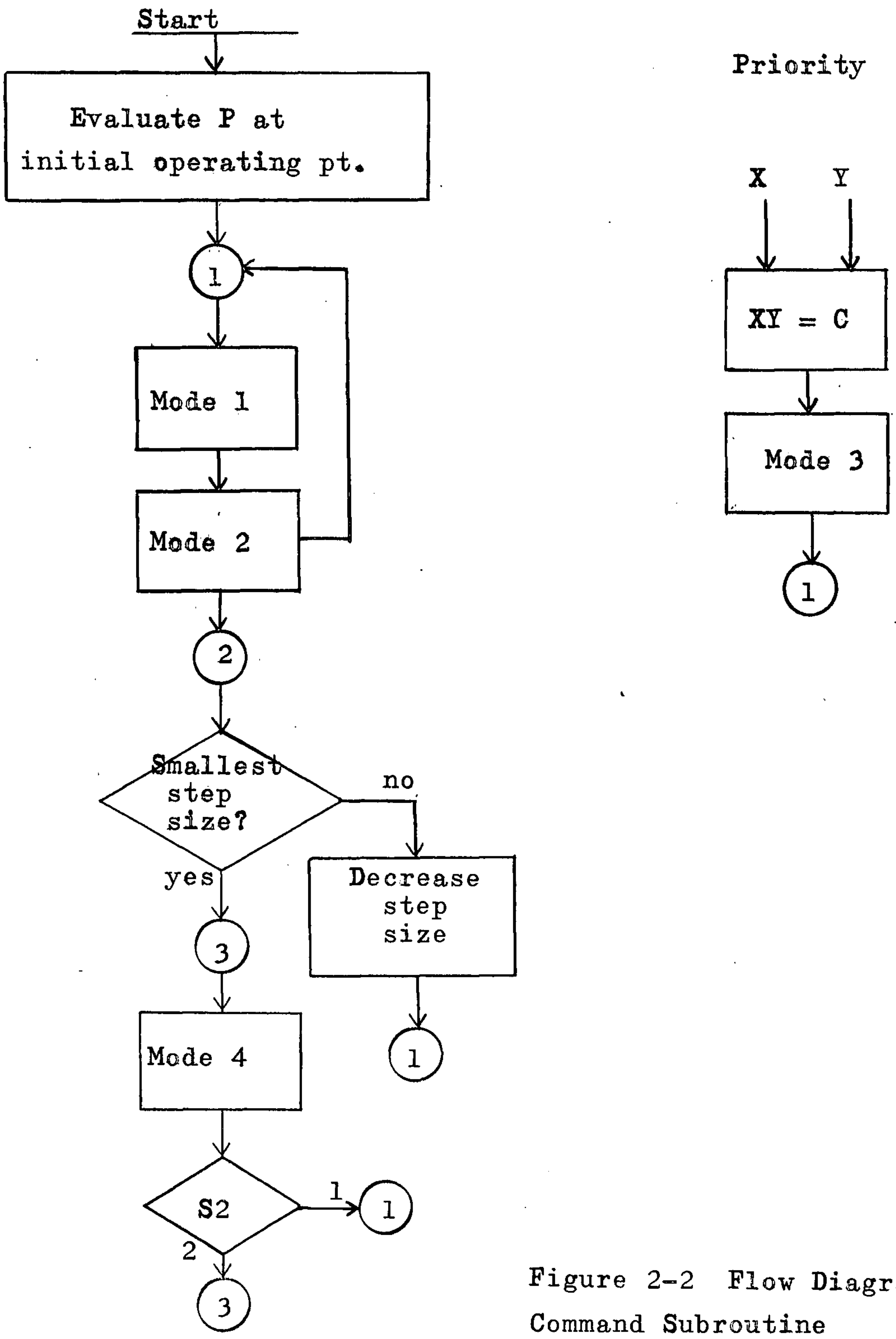
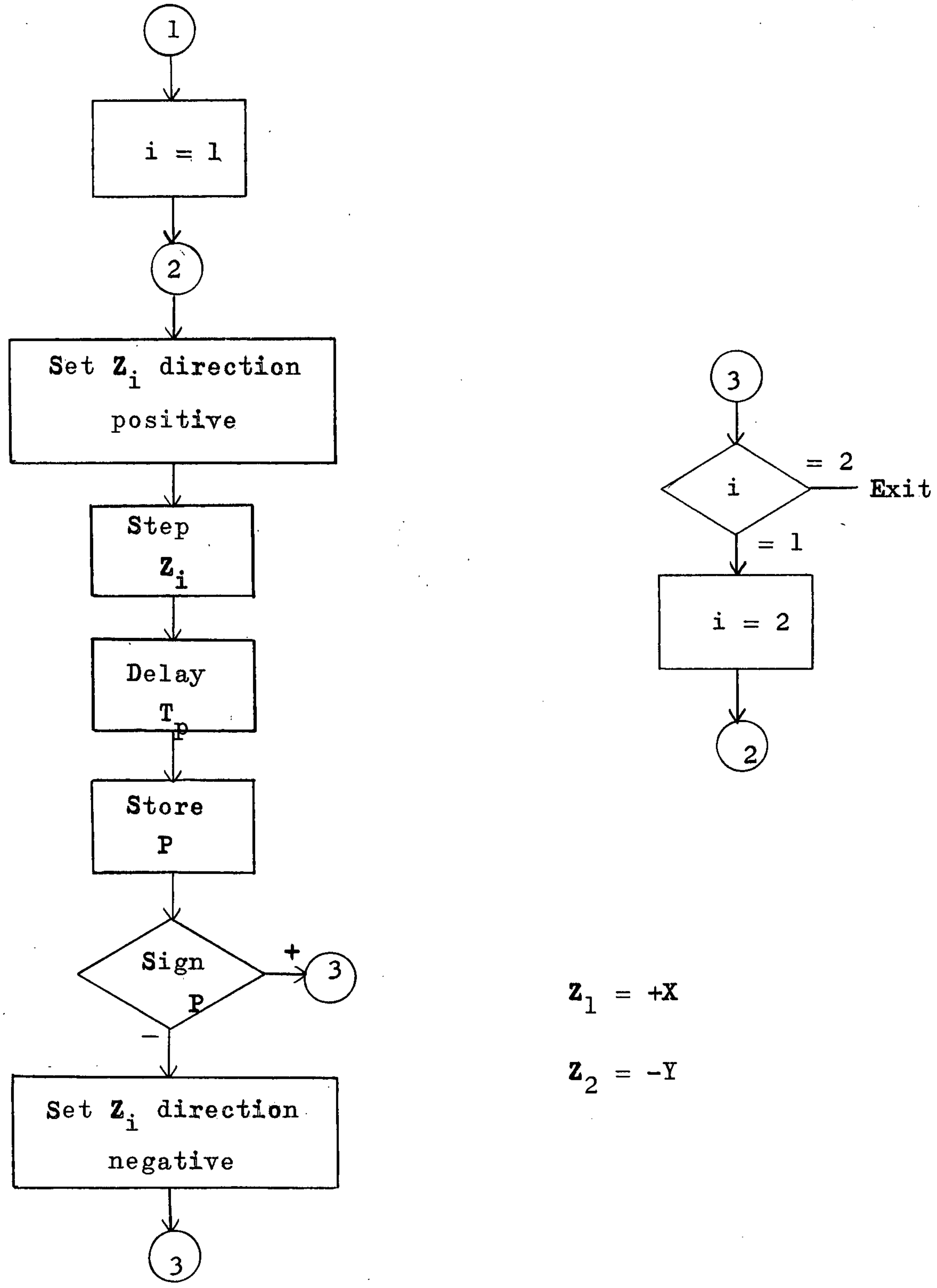


Figure 2-2 Flow Diagram of Command Subroutine



Z_1 = +X

Z_2 = −Y

Figure 2-3 Flow Diagram of Mode 1

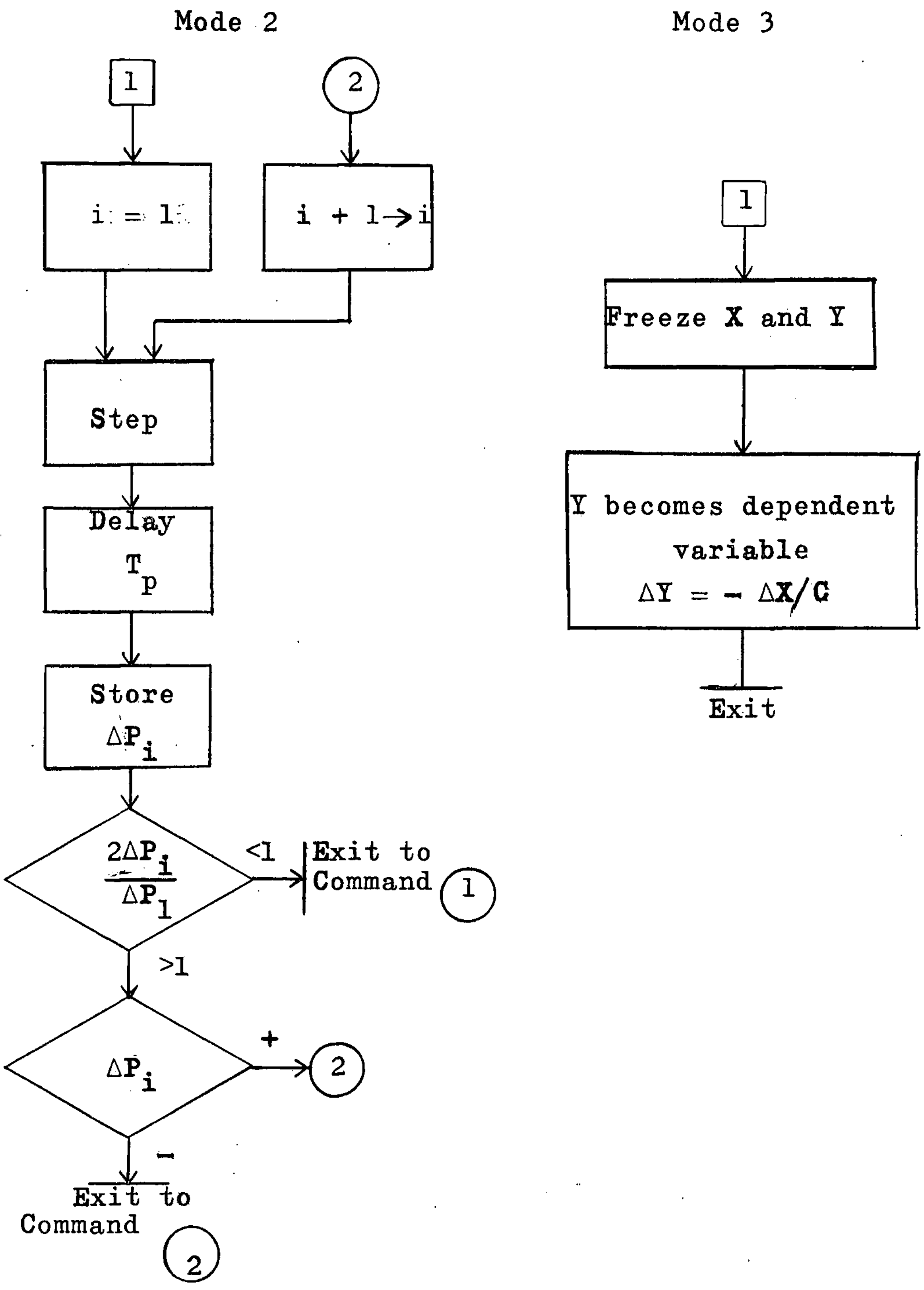


Figure 2-4 Flow Diagram of Mode 2 and Mode 3

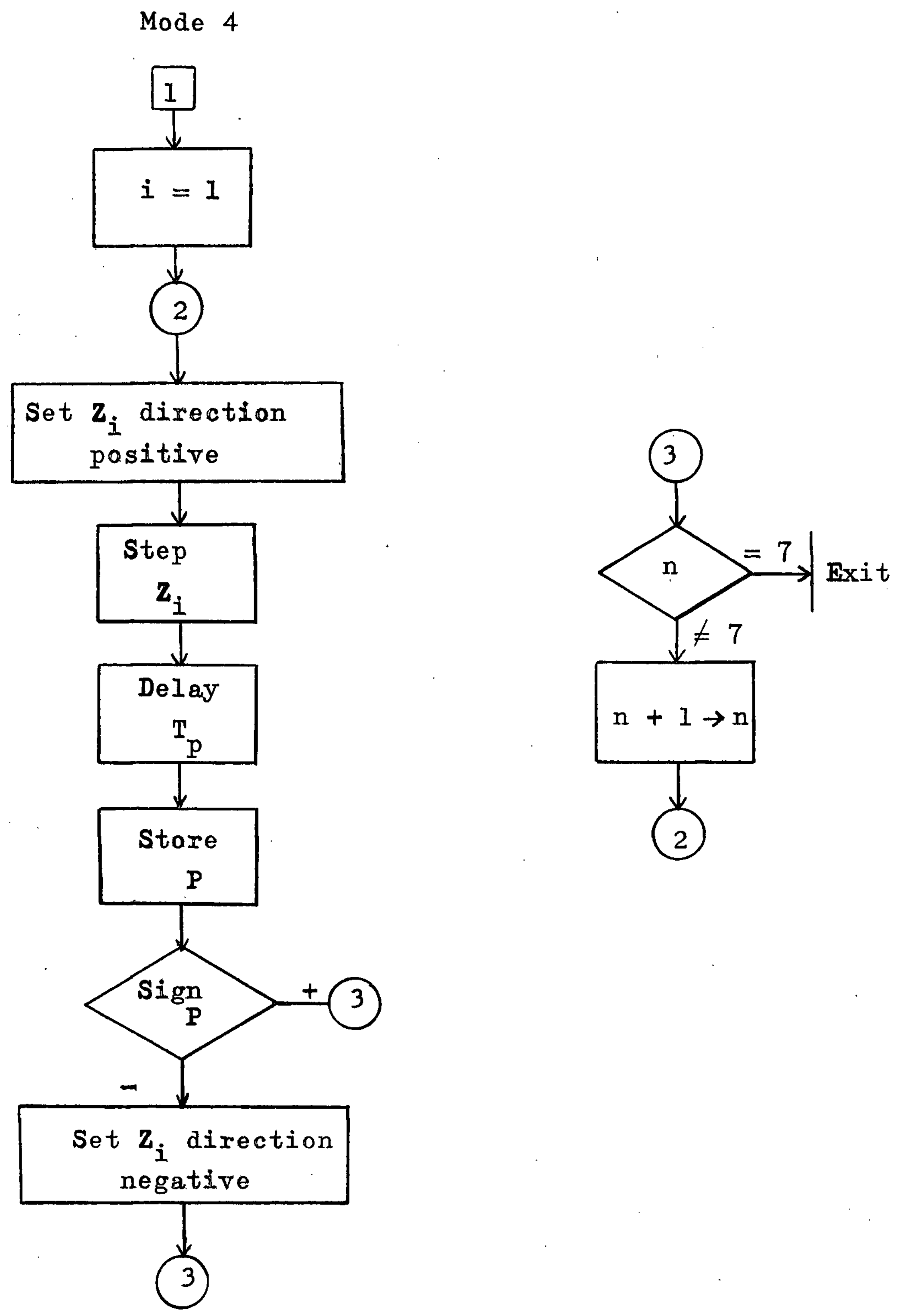


Figure 2-5 Flow Diagram of Mode 4

3. THE COMPUTER

A very inexpensive special purpose digital computer can carry out the search described in section 2.3. The computer design explained in this section uses standard digital logic units such as gates, monostables and flip-flops. The discontinuous nature of empirical optimization and the lengthy storage times required would seem to favour digital rather than analog techniques as would the compatibility of a SPDC with future overall plant digital control or telemetry. A computer using a hybrid of analog and digital techniques could carry out the same search; it might even be argued that the computer described is a hybrid as the outputs of the computer are analog signals obtained from potentiometers driven by step motors. These motors, readily available commercially, are driven by the pulses of the digital equipment. The motors are bidirectional with each input pulse causing a 15^o angular displacement of the motor shaft.

Figure 3-1 shows a very simple outline of the computer. The main control unit, labelled Control, co-ordinates the activities of the auxiliary units which are shown as smaller blocks. The computer is termed asynchronous as the controller starts the next operation when the smaller blocks have signalled completion of the current operation rather than waiting a fixed time for the smaller blocks to carry out their operations. P_- is a negative voltage representing the performance function fed into the control computer by the controlled process, and X and Y are the controlled parameters of the process.

To start the search for the optimum operating point, X and Y are set to some initial values within the limits of the constraint and, when P_- has reached its steady state value, P_+ (Figure 3-1) is adjusted to equal P_-. Here P_+ is an internal positive voltage of the computer which is periodically updated to equal the magnitude of P_-. The SMP block is very much like a sample and hold unit with P_- as an input and P_+ as an output. As used by the computer, the output of SMP is the change needed to update P_+.

The change in P_+ is either δP_X, δP_Y or ΔP depending on whether it was caused by δX, δY or ΔX and ΔY. If the change is δP_X, it is stored in the CCX block; if it is δP_Y, it is stored in the CCY block; and if it is ΔP, it is stored in the CCP block.

Upon signal from the Control block, CCX generates a number of pulses proportional to δP_X to step X, CCY generates a number of pulses proportional to δP_Y to step Y. While the circuitry of CCP is much like that of CCX and CCY, its function is different. CCP stores ΔP_1 in a counter and compares each succeeding ΔP with ΔP_1. Upon the result of this comparison the search either stays in mode 2 or returns to mode 1.

The block SMX has X as output and the various changes δX and ΔX as inputs. The δX input comes directly from the Control block and ΔX is routed by the Control block to SMX from CCX. SMY has exactly the same function with Y instead of X. The details of all the blocks are shown in Figures 3-2 to 3-7. The symbols used are explained in Appendix A.

3.1 Control Block

Figures 3-2 and 3-3 are details of the Control block. Figure 3-2 shows the circuitry for subroutines mode 1 and mode 4. It can be seen from comparing Figures 2-3 and 2-5 that these two modes are very similar and, as the computer cannot be in both these modes at the same time, the same equipment is used for both modes.

The main components of the Control block and their functions are as follows:

FFA — selects mode 1 (state 1) or mode 2 (state 2);

FFG — selects coarse search (state C) or fine search (state F);

FFB — selects, in mode 1 only, whether to change X by a fixed δX (state X) or to change Y by a fixed δY (state Y);

MC — sets correct logical states of the Control block flip-flops to establish the controller in mode 1;

MB — produces time delay of T_p seconds.

With X and Y at their initial values and P_+ updated, the search routine is initiated by a pulse on all the lines marked S in the figures (start pulse). The computer must first carry out mode 1 as shown in Figure 2-2.

In mode 1 the computer is to:

— move X one step, δX, wait T_p seconds and store δP_X;

— move Y one step, δY, wait T_p seconds and store δP_Y;

— go to mode 2.

Referring to Figure 3-2, the S pulse sets FFG to C and sets MC to 1. The MC pulse sets FFA to 1 and sets flip-flops within SMX and SMY so that any pulses to G53 (Figure 3-4) and the corresponding gate for Y, G54, will step X in a positive direction and Y in a negative direction. The computer is now in mode 1. The MC pulse also sets FFB to X and MD to 1. The MD pulse passes through G2 to G53 and causes X to be stepped by a fixed amount δX. T_p seconds later the MB pulse (store) causes SMP to update P_+ to the value of P_-. The change needed to update P_-, δP_X, is stored by the CCX block (Figure 3-5). The decision to store this change in P_- as δP_X rather than δP_Y or ΔP_1 is based upon the states of FFA and FFB. The sequence is now repeated with Y and is initiated by the end of storage pulse (ES).

The ES pulse from SMP signals that P_- has been updated. This ES pulse passes through G31 and sets MF. The MF pulse changes the state of FFB (in this case from X to Y), passes through G30 to set MA, and through G11 but not G1. The MA pulse now passes through G6 as FFB has been set to state Y. The G6 pulse causes Y to be changed an amount $-\delta Y$. The ES signal generated when δP_Y has been stored passes through G31 and sets MF. The MF pulse does not pass through G30 to continue the mode 1 subroutine; instead, it passes through G11 and then through G1 to set FFA to 2. This setting of FFA to state 2 sets the computer to mode 2 subroutine (Figure 3-3).

Continuing the explanation of Figure 3-2, the operation of the circuitry for mode 4 is the same as that given for mode 1 except that the MC pulse is originated by G50 (SMP) and FFG must have been set to F by G26 (Figure 3-3). In mode 4 the computer is to move a total of seven fine steps in X and Y. The directions of the X and Y steps are determined by the polarity of the previous δP_X and δP_Y.

The MC pulse originated by G50 sets FFA to 1, determines the direction of the next X and Y steps, and sets FFB to X and MD to 1. The MD pulse passes through G2 stepping δX and in mode 4 this G2 pulse is counted on the seven counter which will now have a count of one. The MB pulse causes SMP to update P_+ T_p seconds after X has been stepped δX. The ES pulse passes through G31 and sets MF. The MF pulse passes through G30 to continue mode 4.

The G30 pulse sets MA which causes a step δX but the sign (or direction) of this δX is not preset by MC as it was in mode 1. The sign of this δX is determined by the response (sign) of the previous δP_X.

When the counter has received seven pulses, the next MF pulse does not pass through G30 but passes through G34 to set MK. This MK pulse can either continue the mode 4 subroutine or start the entire search routine depending on the position of S2.

The control circuit for mode 2 is shown in detail in Figure 3-3. In mode 2 the computer is to simultaneously step $\Delta X = -k\,\delta P_X$ and $\Delta Y = -k\,\delta P_Y$, wait T_p seconds and record ΔP. If $2\Delta P_k < \Delta P_1$ or if ΔP is negative the computer is to return to mode 1. When G1 sets FFA to 2 (Figure 3-2), MG is set to 1. The MG pulse clears the δX and δY counters of CCX and CCY and sets MH, MH, MB, and FFE are arranged to allow clock pulses through G20 for T_B seconds. The MB shown is the same one shown in Figure 3-2 and is repeated only to make the diagram less complicated. T_p seconds after MB is set, the MB pulse causes SMP to update (store) P_+ in order to find ΔP. A clock rate of 100 c/s was used in the computer constructed.

CCX passes a number of pulses, proportional to δP_X, to G25. These pulses are passed through G25, or are divided by two, depending on the state of FFH, to SMX where X is changed an amount directly proportional to the number of pulses passed through G25. Equivalent pulses are simultaneously passed through G22 to change Y. The first overshoot signal from SSP during mode 2 sets ML which sets FFH to 1. FFH in state 1 means that one half of G19 pulses pass through G25 cutting the step size ΔX in half. The second overshoot during mode 2 passes through G26 to set the Control block to fine search (FFG to F).

ME and G10 are part of the circuitry required for mode 1 but are shown in Figure 3-3 because of limited space in Figure 3-2. An overshoot signal during mode 1 (negative δP_X) is passed through G10 to SMX so that the direction of ΔX will be opposite to that of δX.

3.2 The SMX and SMY Blocks

Figure 3-4 shows SMX, and SMY has exactly the same circuit layout. The function block labelled "Stepping Motor" is explained in Appendix A. The change in X due to one step of the stepping motor is δX. The sign of the change depends on XFFD. The MC pulse at the start of every mode 1 subroutine sets XFFD to "up", but if δP_X is negative a pulse from G10 passes through G5 and changes the state of XFFD.

3.3 The CCX, CCY, and CCP Blocks

Figure 3-5 shows CCX, and CCY has exactly the same circuit layout. The counter and the digital to analog converter functional blocks are explained in Appendix A. A number of pulses, proportional to the magnitude of δP_X, from G45 passes through G16 and is stored in the counter. Upon a signal from the Control block the CCX block is to produce a number of pulses proportional to the stored δP_X.

G16 limits the number of pulses counted to 15 to prevent a 16th pulse from setting the counter to zero. In an actual process the maximum ΔX would be determined by physical limitations, not counter size.

In mode 2, to step ΔX, G20 of the Control block passes clock pulses which are passed through G19 and then through G25 (Figure 3-3) to SMX and CCX. When the count is the same in both counters of CCX, the output of the comparator, C4, becomes 0 and no further pulses pass through G19. Thus, the number of pulses passed by G19 is exactly equal to the count stored in the δP_X counter. The G19 pulses could be made equal to some ratio of the δP_X count by adjusting the resistance network at the input to G1.

The CCP block is shown in Figure 3-6. It is very much like CCX. The input to comparator C3 is arranged so that the output of C3 becomes 0 if the count on the ΔP_K counter is less than one half the count on the ΔP_1 counter. G14 and G42 both prevent any count over fifteen. The pulses proportional to ΔP_1 and ΔP_K are provided by SMP. If $\Delta P_K \geqq \Delta P_1/2$ the G48 pulse from SMP passes through G49 to set MG (Figure 3-3) and thus continues the mode 2 subroutine.

The two D/A converters of the CCX,CCY and CCP blocks have outputs of opposite polarity. This sign difference allows the output of the D/A units to be compared by simple algebraic addition in a resistance network, with the algebraic sum being amplified by a difference amplifier. The sign difference is the result of different flip-flops being used in the two counters.

3.4 The SMP Block

The SMP block is shown in Figure 3-7. T_p seconds after either X or Y is changed, the MB pulse sets FFN to 0 which allows clock pulses through G39. G39 pulses pass through G40 or G41, depending on the sign of $P_+ - P_-$, and adjust the size of P_+ to equal the size of P_-. The output of G57 is a number of pulses proportional to the difference between P_+ and P_- when FFN is set to 0. The G57 pulses pass through G45, G9, G42, or G43, depending on the state of FFA and FFB of the Control block.

If $P_+ - P_-$ is positive, an overshoot, the G41 pulses set FFO to 0. This state of FFO stops any G48 pulse from continuing the mode 2 subroutine. MD resets FFN to 1, T_s seconds after it was set to 0. This resetting of FFN to 1 is the ES signal to G31 of mode 1 subroutine or, if passed through G37 and G48, this resetting will continue the computer in mode 2. The G31 pulse sets FFP to 1 so that only the first $P_+ - P_-$ (ΔP_1) for any particular mode 2 subroutine is passed through G43.

The time constant, T_s, of MO is adjusted to ensure P_+ can be updated for any possible $P_+ - P_-$. Even though this is a convenient method for obtaining an ES signal, T_s would be difficult to predict for an unknown P surface and the G31 pulse for a real process may have to be originated by the condition of $P_+ = P_-$ after the MB pulse.

The circuitry enabling the Control block to carry out mode 3, the constraint handling subroutine, was not built and tested, but a suggested circuit is shown in Figure 3-8. This circuit would only work for the constraint $XY \geq k$ and is shown as an addition to SMX (Figure 3-4) and SMY. When $XY = C$, no G54 pulses can move Y. Instead, the G53 pulses stepping X are divided by C and passed to step Y. Moving X one step of the stepping motor might create problems if $C < 1$, but the gearing in SMX could be changed so that δX becomes a number of pulses divisible by C. Any changes in X and Y could be kept in opposite directions by letting any overshoot in mode 1 change the state of XFFD (Figure 3-4) and YFFD. This would require a counter and could result in loss of accuracy due to accumulative error. A better method would be to drive Y as a position servo.

as shown in Figure 3-9.

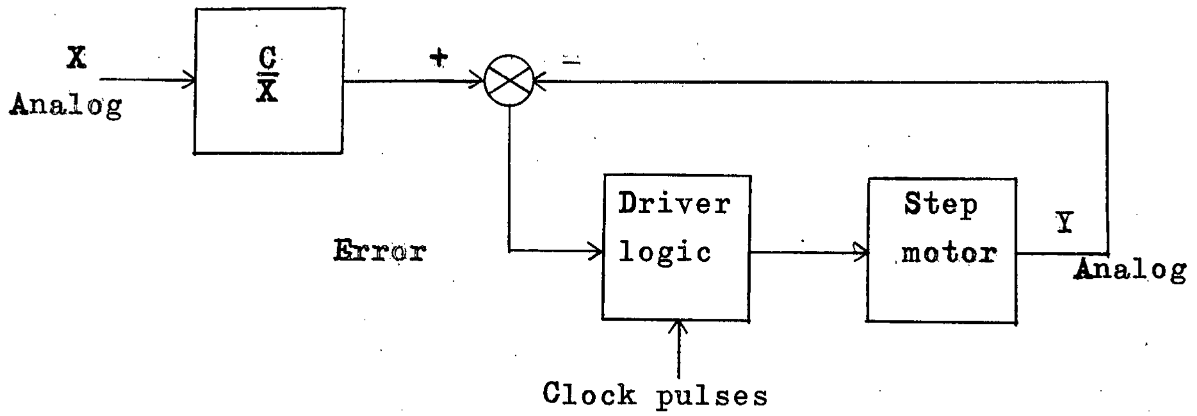


Figure 3-9 Constraint Circuit with No Accumulative Error

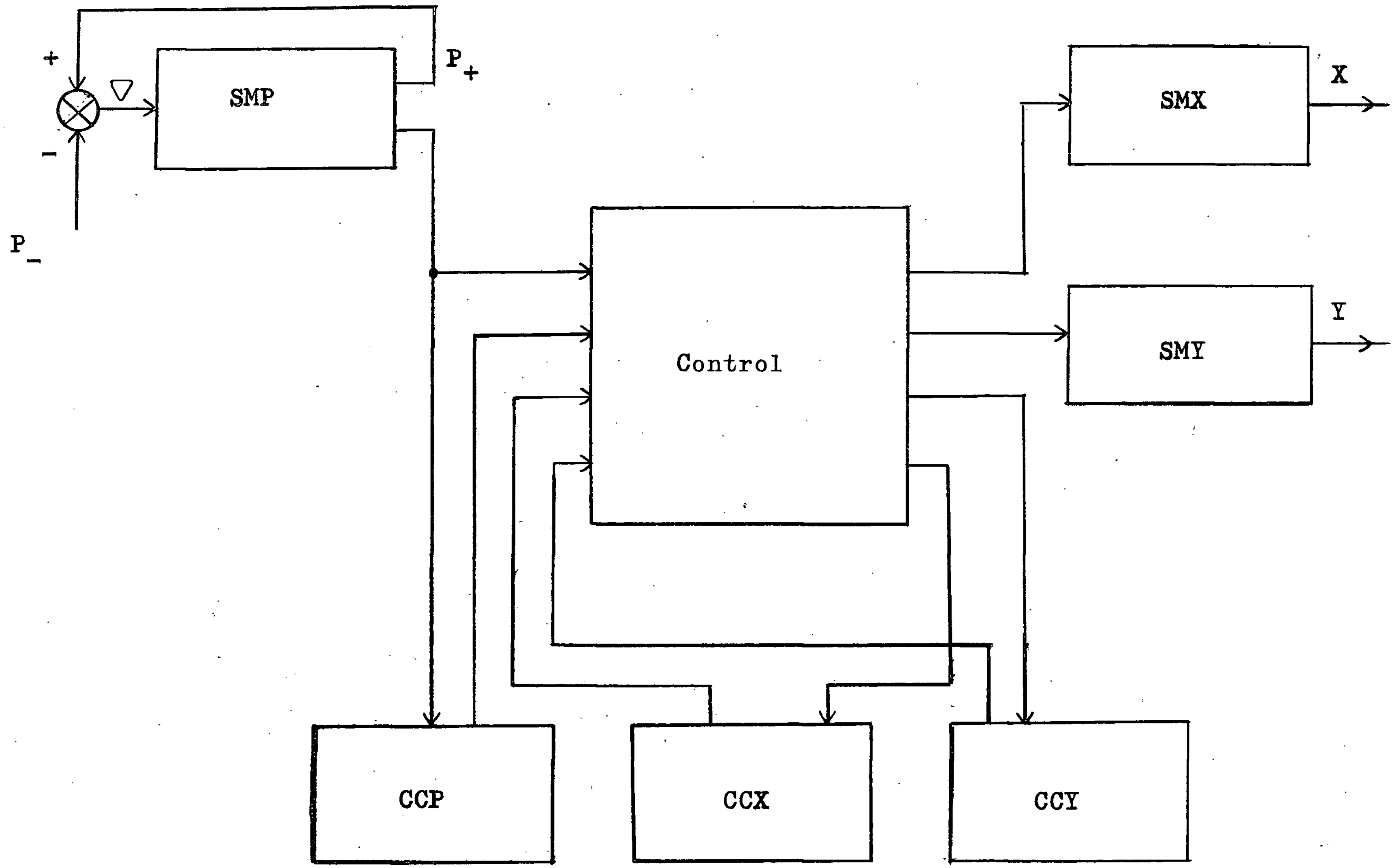


Figure 3-1 General Computer Outline

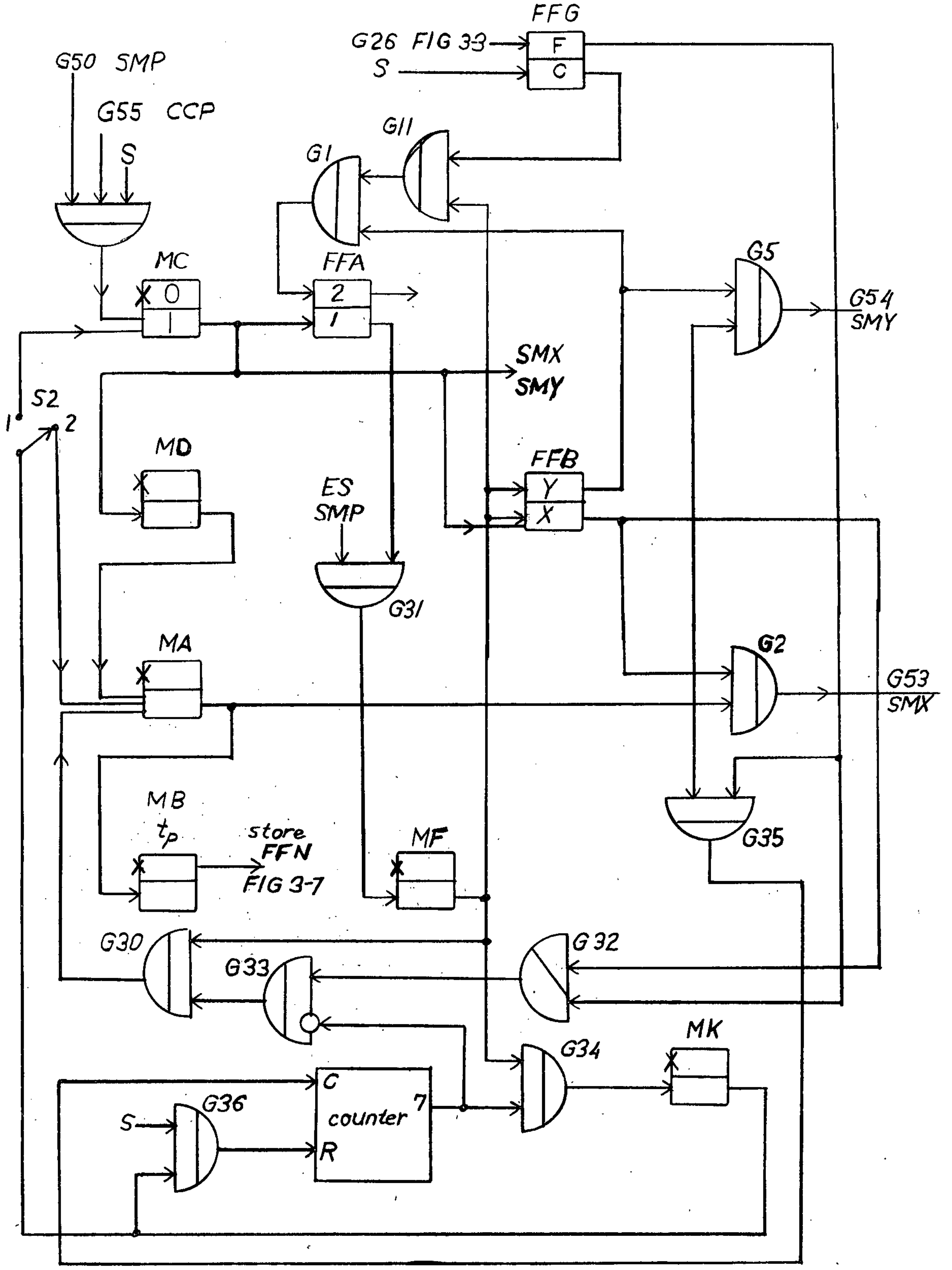


Figure 3-2 Control Block - Modes 1 and 4

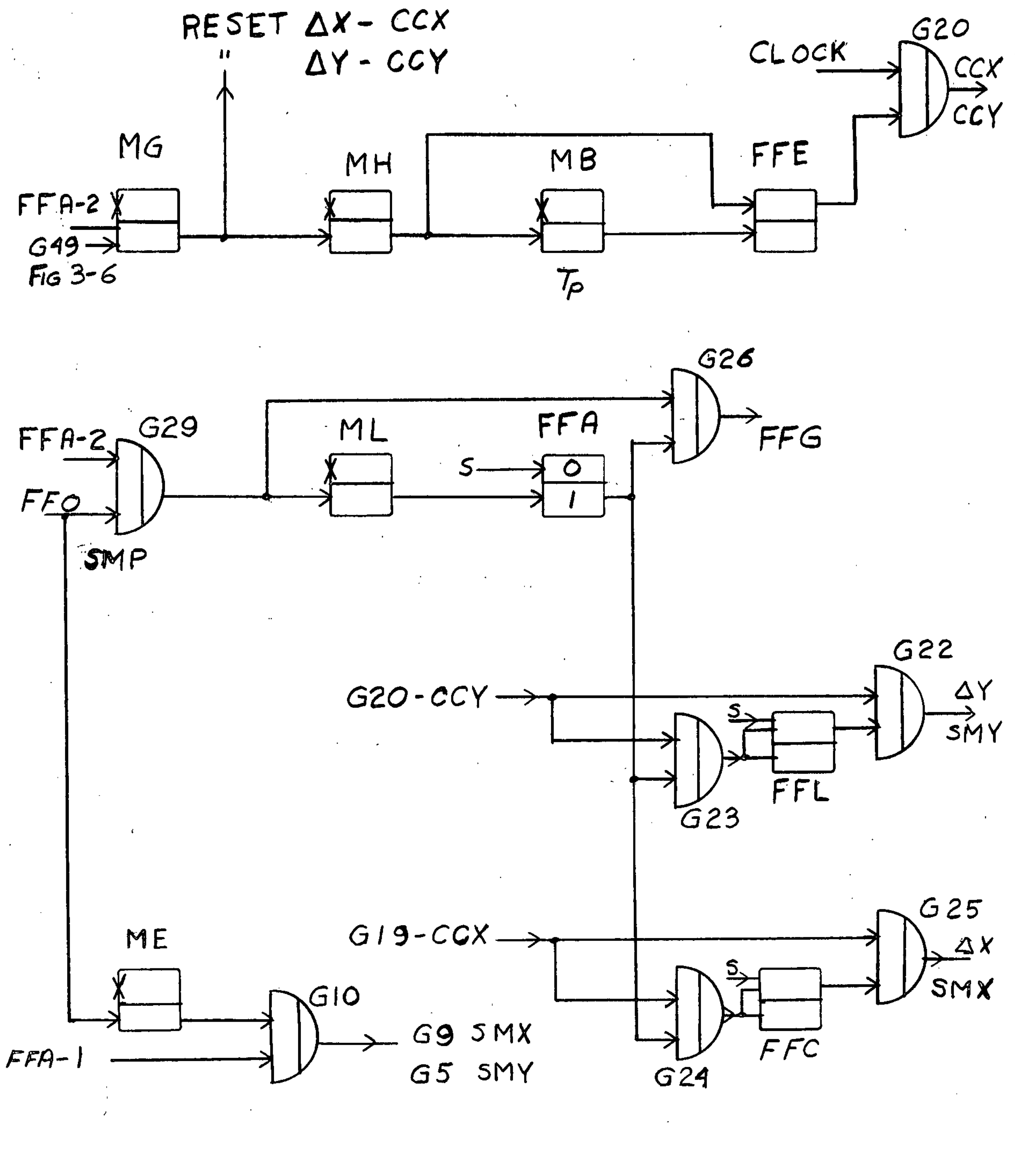


Figure 3-3 Control Block - Mode 2

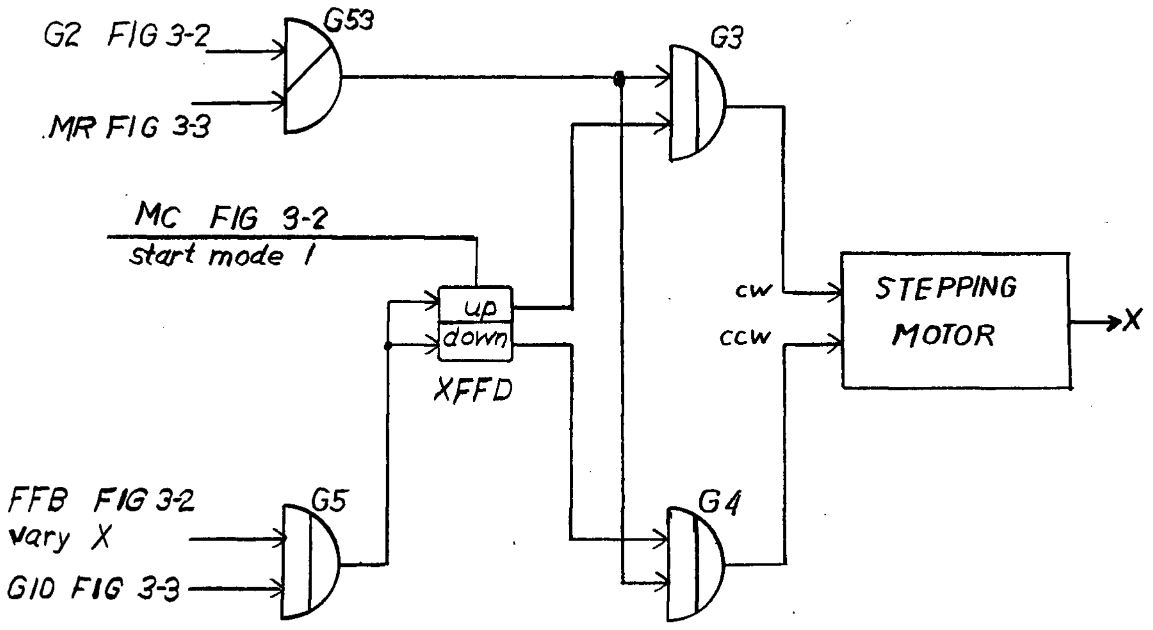


Figure 3-4 Block Diagram of SMX

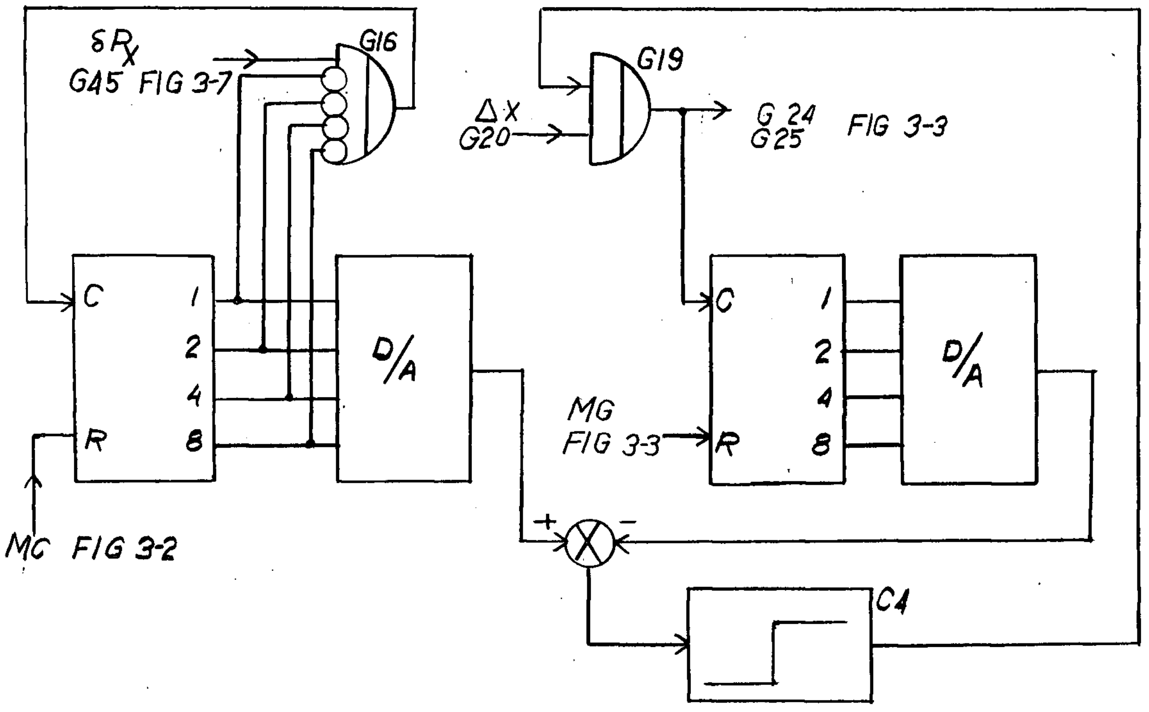


Figure 3-5 Block Diagram of CCX

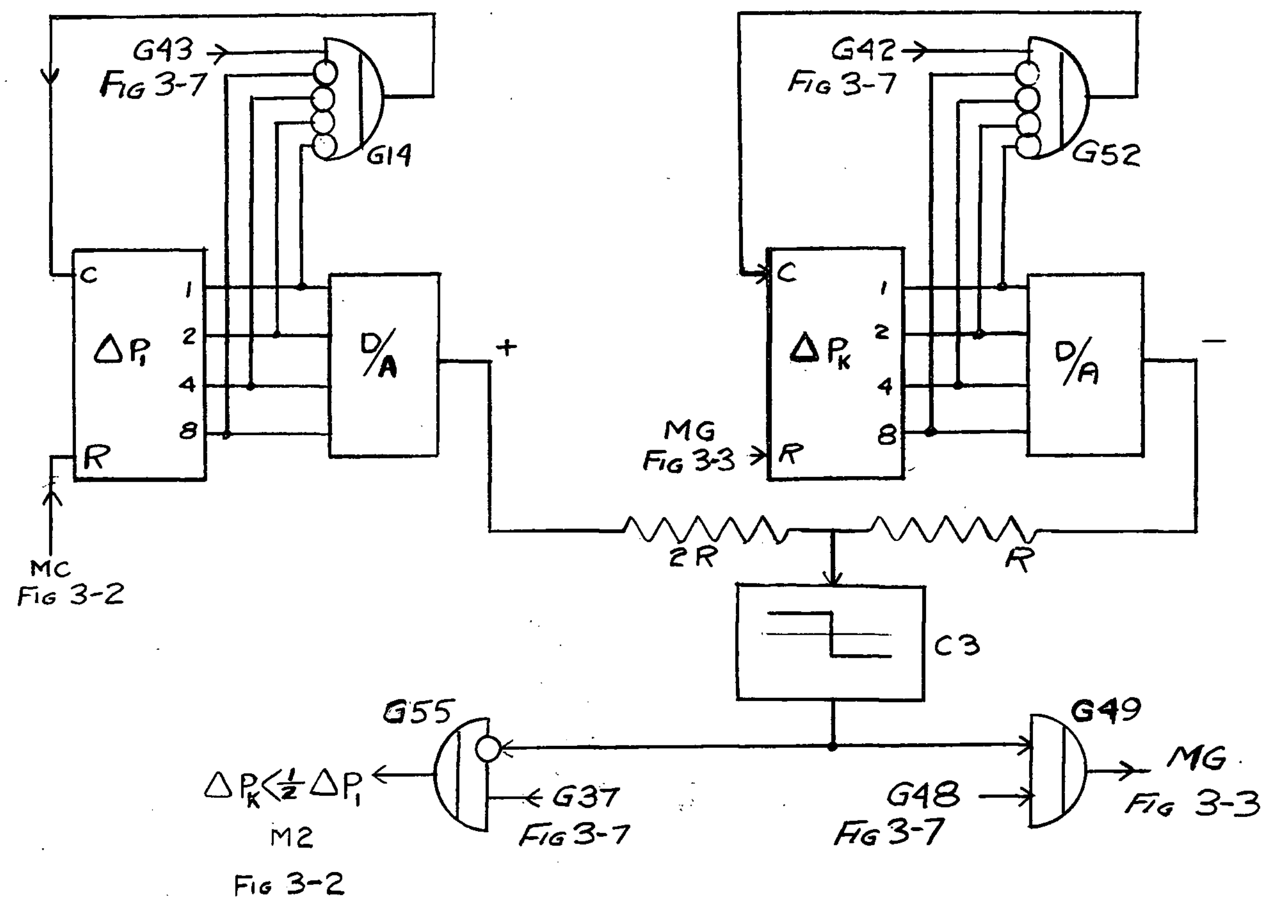


Figure 3-6 Block Diagram of CCP

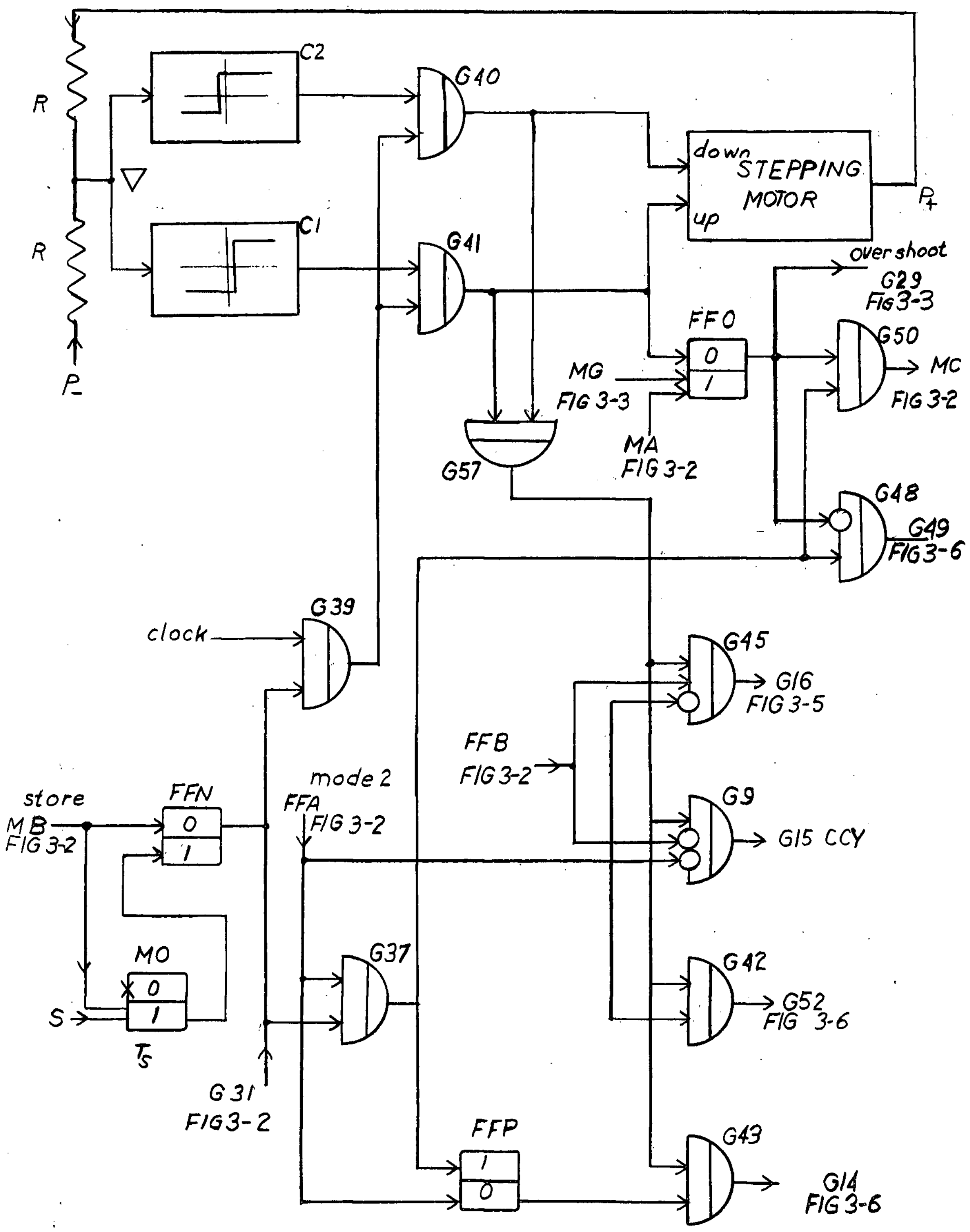


Figure 3-7 Block Diagram of SMP

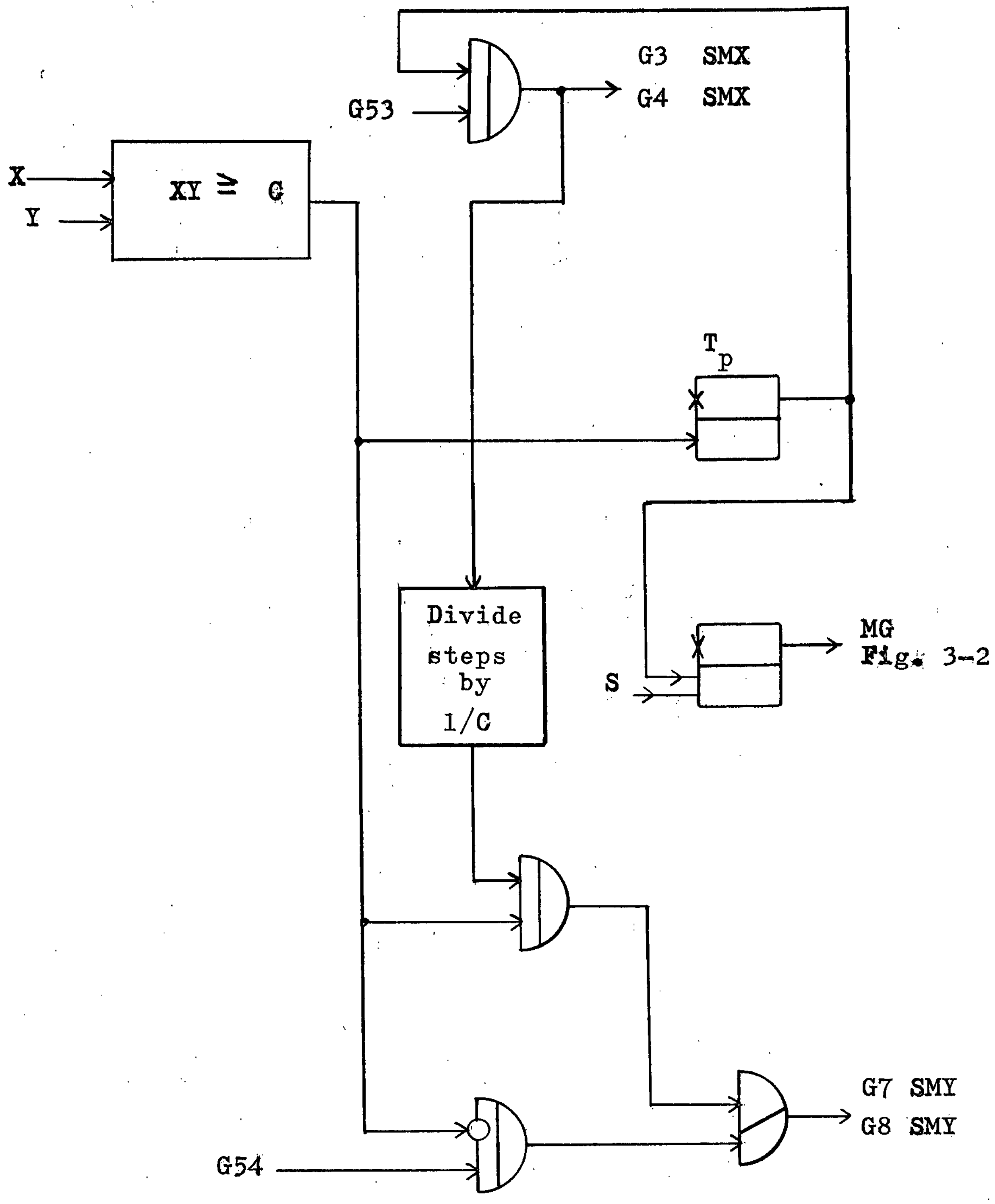


Figure 3-8 Block Diagram of Constraint Circuit (XY $\geq$ C)

4. TEST OF COMPUTER

A computer as described in Chapter 3 was built with the exception of the constraint handling mode. To obtain the logic blocks required, transistor monostables and flip-flops on printed circuit boards were used along with resistor transistor logic gates, but any logic system would do just as well.

The computer was set up as in Figure 2-1 with a Pace Analog computer simulating the controlled system. Various simple P surfaces were obtained using the diode function generator of the Pace. The surfaces were kept simple as it was a test of the SPDC to carry out a specific search strategy, not a test of the search to find the minimum of any particular P surface. Tests on the search strategy are well documented.[16,23] P surfaces such as $P = XY$ or $P = X/Y$ could have been produced from extra potentiometers on the shafts of the X and Y stepping motors; this technique could be used on a SPDC optimizing a real process where P is not directly available but is a known function of the variables. Chapter 5 contains details of the search over a particular surface.

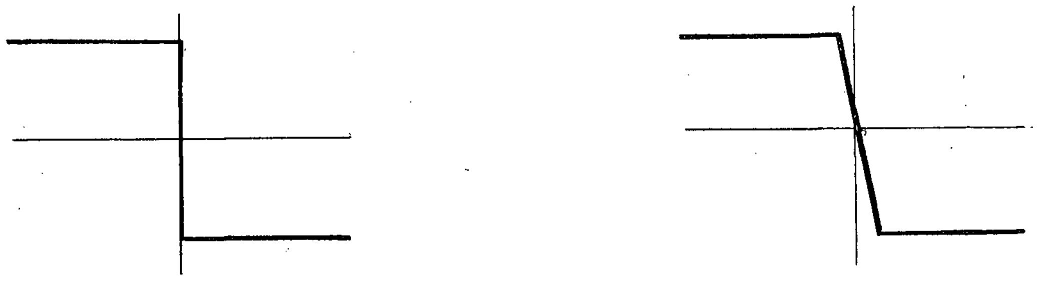
The first change necessary in the computer was the addition of a manual control (push button) to replace the clock pulses during circuit tests. Perhaps the manual control should be installed permanently on a SPDC as it is needed not only for initial testing but for any required service and maintenance.

4.1 Comparators

The two comparators labelled C1 and C2 in Figure 3-7 are to give one of two possible voltage levels depending on the sign of the analog input $\bigtriangledown$. The possible outputs are adjusted to the 0 and 1 levels of the digital equipment for direct use. The units built in the computer were D.C. difference amplifiers (gain ≈ 120) with one input grounded and the other input $\bigtriangledown$. This particular choice of comparator caused amplification of noise present on the computer ground level allowing spurious pulses through G40 and G41 which completely upset the computer operation. The optimum would eventually be found but many unnecessary steps would be taken. The exact reaction to one of these steps in P_+ depends upon when it happens during the search procedure and through which gate it is routed.

This ground noise problem was overcome on the computer by using an isolating transformer and carefully rerouting the ground lines so that only one ground level existed for the entire computer. The grounds for the various parts of the computer were run in parallel, using heavy gauge wire, to this common ground level. This ground arrangement and the integration of the output of C2 and C3, so that very short pulses on the output of C2 and C3 are not fed into G40 and G41, managed to make the output of C2 and C3 relatively undisturbed by ground noise. Before building any computer of this type, literature on noise and ground systems should be read carefully.[21]

The output of the comparator units, C2 and C3, was not the desired perfect two levels as shown in Figure 4-1(a) but rather as shown in Figure 4-1(b).



(a) Theoretical Comparator Output (b) Actual Comparator Output

Figure 4-1

Because of this sloping zone in the output of the comparators, G40 and G41 could output pulses which were unable to step the P_+ stepping motor since the motor requires pulses of 5 volts amplitude (minimum) with a rise time of .1 to 2.5 µs. and a minimum width of 10 µs. As the pulses of G41 have an exponential rise time, it might take a pulse with an overall amplitude of 8 volts to obtain 5 volts with a rise time of 2.5 µs. A clock pulse with faster rise times and faster transistor gates could have made the entire edge of the pulse faster than 5 volts/2.5 µs. so that the amplitude of the G40 and G41 pulses is the only deciding variable in the stepping of the stepping motor. But with the components used, if SMP were to be built according to Figure 3-7, G57 would have to pass only pulses which have a specified rise time, width, and amplitude to step the P_+ stepping motor. Rather than build a special gate, the 3 phases of the stepping motor were capacitively coupled to the inputs of a standard AND gate, resulting in the correct pulses from G57 (Figure 4-2).

The stepping of P_+ should depend only on the value of

∇. However, with the particular clock and flip-flop circuits used, the G39 pulses have two possible rise times depending upon the level of the clock output when FFN is set to state 0. (Figure 4-3 (a)). The slope in the comparator output allows only the faster of these rise times to step the P_+ motor for a particular range of ∇. This is because the stepping motor requires not only a minimum amplitude but also a minimum rise time for an input pulse to step the motor. This could be overcome in various ways, the most obvious being to increase the gain of the comparator amplifiers and thus narrow the sloping region in their output. Unfortunately any increase in gain will increase the dependence of the amplifier output on the ground level. One modification which will overcome both the dependence on ground level and the sloping region of the D.C. amplifier output is to replace the D.C. amplifier with a very high gain A.C. amplifier and make ∇ an A.C. signal by chopping P_+ and P_-. Figure 4-2 shows the block diagram of this suggested SMP block. Because the A.C. amplifier does not produce an output proportional to the difference between an input voltage and ground as did the D.C. differential amplifier, the gain may be very high and ground noise will not cause the serious problems it did with the D.C. comparator. This much larger gain will produce a comparator output which will overcome the problem as shown in Figure 4-3(b) because C2 can be adjusted to allow no output of G40 when a step is not required in the "down" direction of the stepping motor.

Just as G57 (Figure 3-7) would have required delicate biasing to pass only pulses with rise times and amplitudes to step P_+, so the input of FFO (Figure 3-7) required sensitive "trimming" so that FFO was set to the 0 state only by a pulse which would step the motor. It is possible to obtain the overshoot pulse by a circuit which would require no adjusting and be independent of rise times and voltage levels. The motor phases, A, B, C, (Figure 4-2) have one sequence for the "down" direction and another for "up". If the "up" sequence is A, B, C, then any circuit that can indicate this sequence can be used as an overshoot signal. Such a circuit, using standard digital components, is shown in Figure 4-4. The motor phase going from the 1 state to the 0 state sets off the monostable and only the phase sequence A, B, C would produce an overshoot pulse.

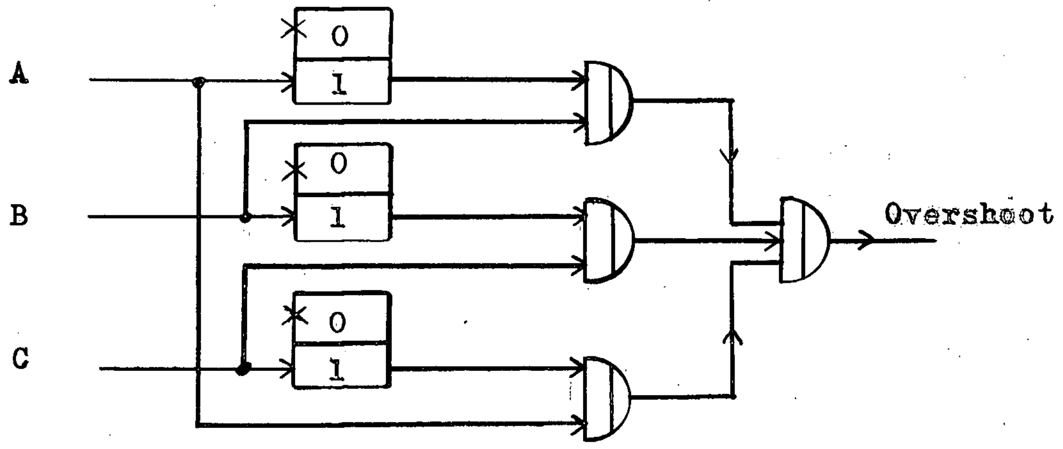


Figure 4-4 Digital Overshoot Circuit

Since the comparators are the only units of the computer not made up of only standard digital logic components, and, in general, gave more trouble than any other unit, alternatives to them should be considered. There are two in the SMP block and one in each of CCX, CCY, and CCP blocks. The two in SMP can be replaced by A.C. amplifiers as suggested but other possibilities exist for CCX, CCY, and CCP.

The function of CCP is to compare ΔP_1 and ΔP_K. This comparison is required by the quite arbitrary decision in section 2.3 to return to mode 1 from mode 2 if $2\Delta P_K < \Delta P_1$. Since for an unknown P surface this decision has no inherent value over any of the other possible variations, the comparator of CCP could be eliminated. If the computer returns to mode 1 from mode 2 at the local minimum or at the N^{th} mode 2 move, CCP can be replaced by a N counter (counting MG (Figure 3-3) pulses) and an arrangement of standard gates to route G49 (Figure 3-6) pulses to MA (Figure 3-2) when the count is N. This replacing of the amplifier comparators by a combination of standard digital logic circuits is desirable if consideration is being given to building the computer using commercial integrated circuits. The small voltage difference between the logic states of integrated circuits would require very high comparator gain and careful choice of resistors in any simple D/A units (Appendix A).

This N step variation would also make the computer more flexible as the number of steps which cause return to mode 1 could easily be changed at a patch panel. While this patch panel would add very little to the cost of the computer, it would add much to its search efficiency as the search could be varied from steepest ascent ($N = 1$) to various relaxation strategies. This flexibility would be useful if the irregularity of the P surface is a function of an uncontrolled variable.

The amplifier comparators of CCX and CCY can be replaced by pure digital comparators with a set of gates for each possible state of the counters; however, this would increase the number of gates and flip-flops used by the computer. The use of ring counters would make this type of comparator easier to design.

Another solution would be to allow ΔX to be one of a few possible sizes depending on the size of δP_X. For example:

$\Delta X = \delta X$	$0 < P_X < L$	$L = \text{Constant}$
$\Delta X = 5\,\delta X$	$L < P_X < 2L$	
$\Delta X = 10\,\delta X$	$2L < P_X$	

This step size arrangement could be instrumented by having the most significant flip-flop in state 1 of the δP_X counter gate MG (Figure 3-3) pulses to one of three pulse train generators.[22] This pulse train generator would step X. The size of ΔX would depend on which pulse generator was triggered.

The same type of arrangement could be used for step size choice in a direct search strategy and would involve only digital logic blocks.

4.2 Dead Band

A problem developed with SMP which would be present regardless of the comparator used. On the "store" signal, MB (Figure 3-2), the stepping motor is driven until $P_+ = P_-$. The motor is driven through G40 or G41 depending upon the sign of $\nabla = P_+ - P_-$. If at $\nabla = 0$ G40 drives the motor one step in the "down" direction (P_+ becomes less positive) so that $\nabla = -A$ volts, and if this $-A$ volts is enough to drive the motor through G41 back one step to $\nabla = 0$, there will be a continuous "chatter" of SMP for the entire MO pulse which was originated by the "store" signal. With the circuit as in Figure 3-7, this "chatter" would result in the δP_X (δP_Y) counters of CCX (CCY) to be at maximum for any step δX (δY) regardless of the performance surface. It would also cause FFO to be set to the 0 state, the condition that δP is negative, even if the δP resulting from a parameter step is positive. So with this circuitry a dead band must be designed in the comparator circuit with a width of at least A, the change in ∇ caused by one step of the motor setting P_+.

This dead band in the comparators results in a hysteresis effect in the ability of SMP to track P_-. If $\nabla = +\frac{1}{2}A$ and X is stepped δX resulting in a change in P_- such that $\nabla = 2\frac{1}{2}A$ upon the arrival of the MB pulse at FFN, two sequential pulses will leave G57 resulting in $\delta P_X = 2$. If $\nabla = -\frac{1}{2}A$ when X is stepped δX, the same change in P_- will be recorded as $\delta P_X = 1$ since $\nabla = 1\frac{1}{2}A$ when the MB pulse arrives at FFN. The value of ∇ when any parameter is stepped could be any value within the comparator dead band, $\pm A/2$. This comparator dead band decreases the sensitivity of the computer since a δP_X, or δP_Y, less than A may or may not be recognized depending upon the starting value of ∇. The computer built using the circuitry of Chapter 4 also could drift away from the optimum if $\delta P_X \approx \delta P_Y < A$. As soon as δP_X or $\delta P_Y > A$ the computer would function properly but, instead of staying within a few steps of a varying optimum once it was found, the computer could drift until a $\delta P > A$ was recognizable. If, during mode 1, the performance surface is such that $\delta P_X \approx \delta P_Y < A$ the control parameters will not be moved during the next mode 2 as the counters have values $\delta P_X = \delta P_Y = 0$. The computer would then stay in mode 2 indefinitely unless an uncontrolled variable forced ∇ to become negative and greater than A. As this is an unacceptable condition, a gate should be added to CCP so that, if the ΔP_K counter (Figure 3-6) is zero when G37 gives a pulse, the computer should be returned to mode 1.

In general, the wider the dead band in the SMP comparator the poorer is the performance of the computer; therefore, A, the width, should be made as narrow as possible by adjusting the gearing between the stepping motor and the P_+ potentiometer so that one step of the motor produces a small change in P_+, thus a narrow A. Reducing the voltage across the P_+ potentiometer would also reduce A but this should not be done since the maximum range of P_- is the voltage drop across the P_+ potentiometer. As the range of P_- is made smaller, any percentage change in P_- is also made smaller and more difficult to measure.

The direct search strategies do not use the size of δP and so could use a comparator with no dead band by finding the sign of δP from the direction of the first motor step after the arrival of the "store" pulse and ignoring any later steps.

A majority of these suggestions may make the search strategy less refined than that of section 2.3, but the overall performance, cost, and reliability of the control computer are often of greater importance than the mathematical sophistication of the search strategy.

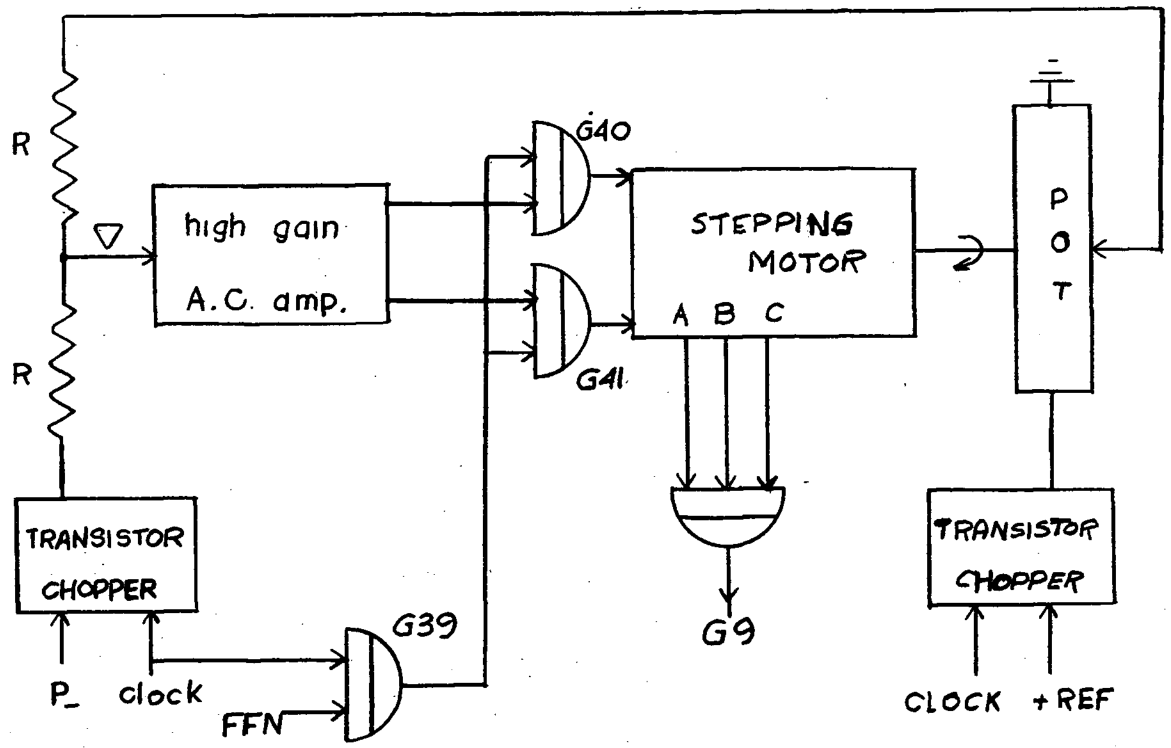
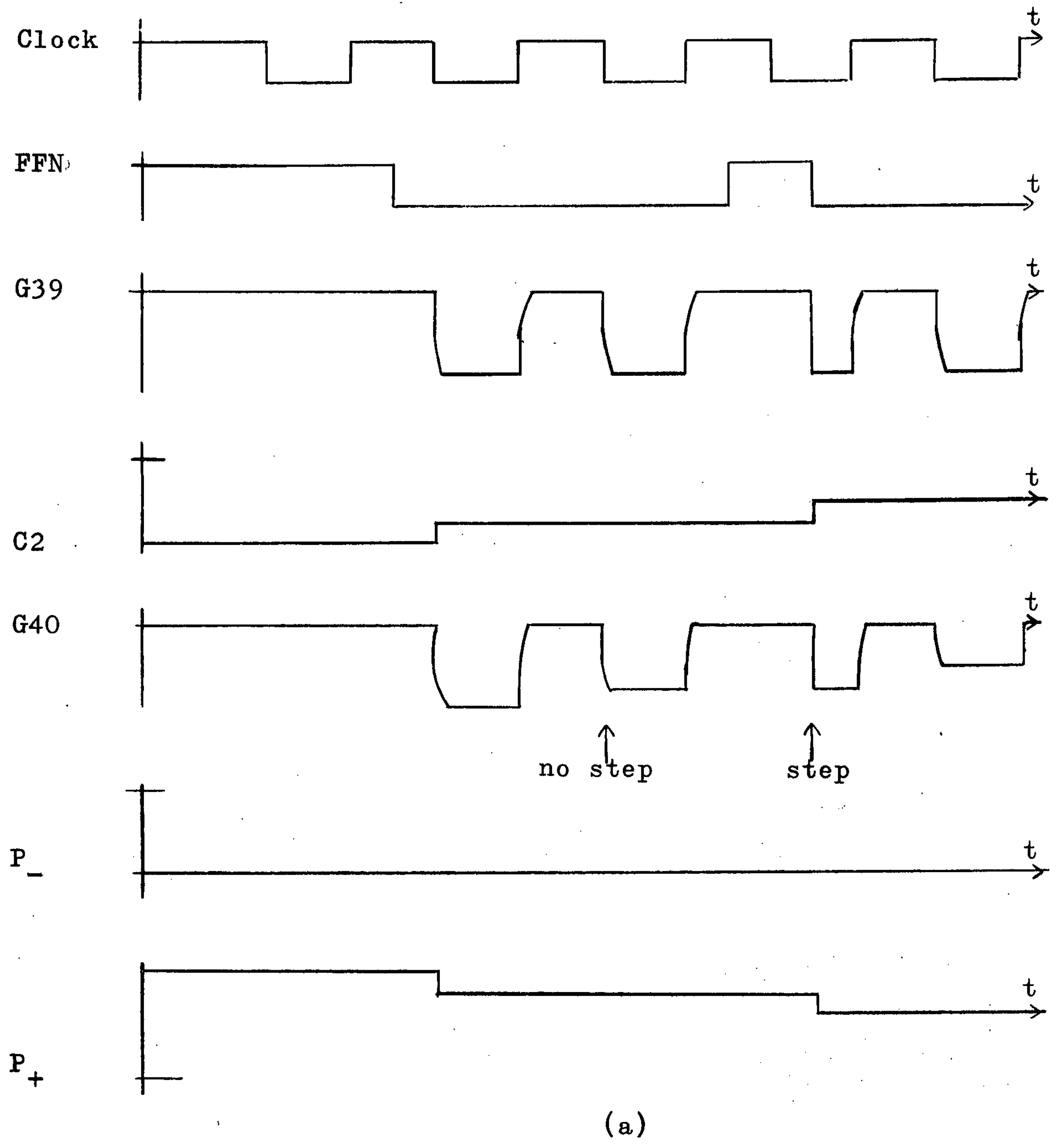
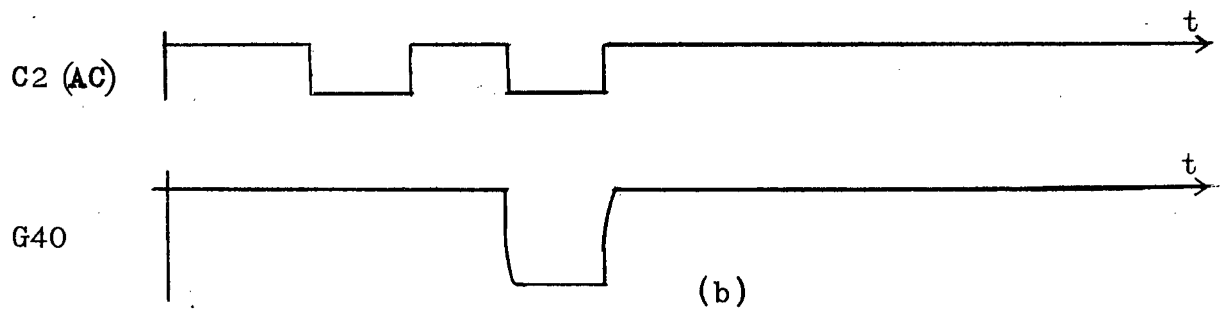


Figure 4-2 Changes to SMP



(a)



(b)

Figure 4-3 SMP Waveforms

5. EXAMPLE OF USE OF SPDC

As an example of a process which could be optimized by a low cost SPDC, consider the flotation process of a complex ore. Flotation is the separation of minerals from pulverized ore by methods which cause the valuable particles to rise to the surface of a liquid. Various chemicals, known as collectors, are used to cause selective attachment of particles of one mineral to air bubbles which float to the surface of the liquid while the other minerals remain in suspension in the liquid. There is an optimum concentration of the collector in the liquid as too high a concentration will float too much of an unwanted mineral and too small a concentration will not recover all that is possible of the desired mineral.

The control problem is finding this optimum concentration. The amount of collector can be increased if the tailing losses are high and decreased if the percentage of the desired mineral in the floated concentrate is too low. The difficulty of this problem depends on the variability of the ore. In ore of continually shifting composition, the flotation results will change with the composition causing fluctuations in the grade of the concentrate. Economic incentive for tracking the optimum collector concentration increases with the amount of ore processed. In a large mill operation a small improvement in percentage recovery would pay for any added control equipment.[24]

Unfortunately, no simple formula is known linking the collector added to an ore and the percentage of a mineral floated.

Laboratory flotation tests have shown that, for at least one ore, flotation results are a function of $m[H^+]$ where

m = residual concentration of xanthate

$[H^+]$ = hydrogen ion concentration

Curves of the type in Figure 5-1 show the relationship between $m[H^+]$ and %, the percentage of mineral recovered.[25] As the product $m[H^+]$ is increased, the recovery of lead increases until a plateau of maximum recovery is reached. Further increases in $m[H^+]$ will decrease the grade of lead concentrate because of increasing flotation of zinc. Therefore, the control problem is not to minimize or maximize a performance function, but to stabilize $m[H^+]$ at the point at which maximum recovery is first reached. If % is considered the performance function, the optimum operating point is at a specific value of the slope $\delta P/\delta \log m[H^+]$. A SPDC would therefore have to be designed to track not the point at which $\delta P = 0$ but rather $\delta P = D$ where D is a constant determined by the operating personnel of the mill.

Both m and $[H^+]$ can be controlled with simple analog feedback control loops and % can be considered to be continuously available. In this process the knowledge of the performance surface would dictate a very simple search strategy as the P surface is very smooth and unimodal. A single mode search like the mode 4 described in section 2.3 would probably suffice as the approximate optimum point is known from past operating experience and the purpose of the SPDC would be to track the optimum operating point as it changes with the characteristics of the ore being floated.

The SPDC described in Chapter 3 was set up as shown in Figure 5-2 to simulate control of a flotation process. The SPDC was designed to track a minimum of P. So P = -% + D $(m[H^+])$, which has a minimum at the required operating point, was set up on the Pace's diode function generator. The actual process has a T_p in the order of many minutes, but a more convenient T_p of .4 seconds was used in the test of the SPDC.

With $v_1 = v_2$ = constant, the computer would greatly overshoot the optimum if the initial values of m and $[H^+]$ were low but this is to be expected from the gradient search of the SPDC and would not happen in a SPDC designed to optimize this particular process.

For v_1 = constant, and the peak to peak value of v_2 equal to one tenth the range of P_-, the SPDC in mode 4 could track the optimum for sinusoidal, triangular or square wave as long as the period of v_2 was greater than 10 T_p.

For v_2 with a period of 10 T_p and an amplitude of v_1 that varied $P_=$ one tenth of its maximum value, the SPDC could track the optimum for sinusoidal and triangular wave form v_1 with period greater than 10 T_p. For a square wave the period of v_1 had to be greater than 30 T_p for the SPDC to track the optimum.

The T_p or phase lag of a process will limit the frequency of change of the input signal that can be tracked for any direct method of optimization.[26]

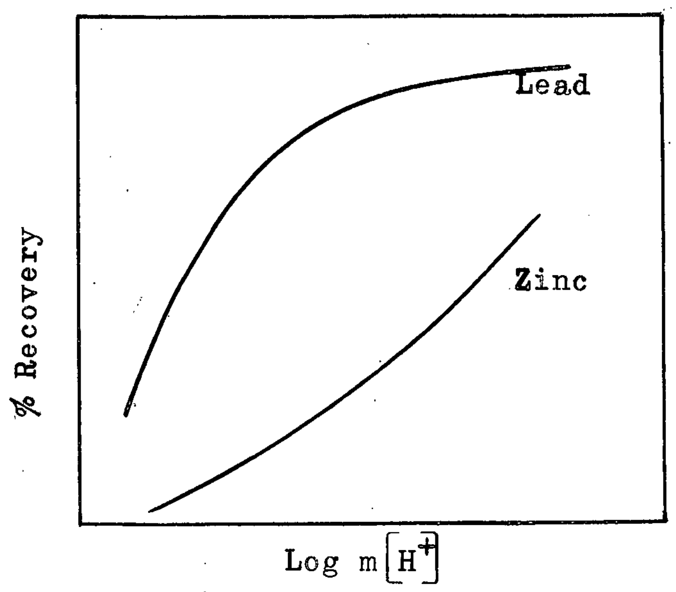


Figure 5-1 Percentage Mineral Recovery vs $m[H^+]$

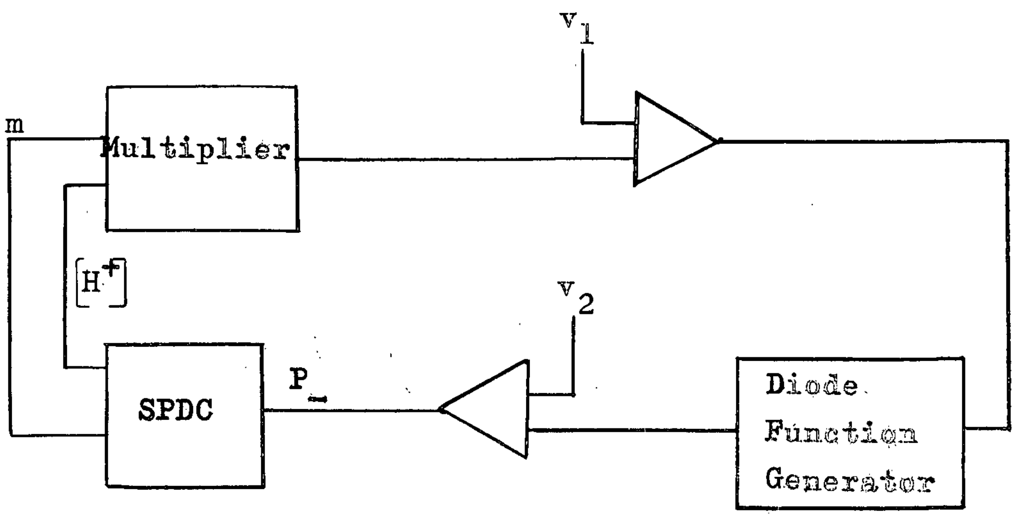


Figure 5-2 Test Set Up

6. CONCLUSIONS

It is possible to build special purpose digital computers for process control purposes at much below the minimum cost of large scale general purpose computers. The SPDC can be designed using only "off the shelf" components to handle many control problems such as single loop optimization.

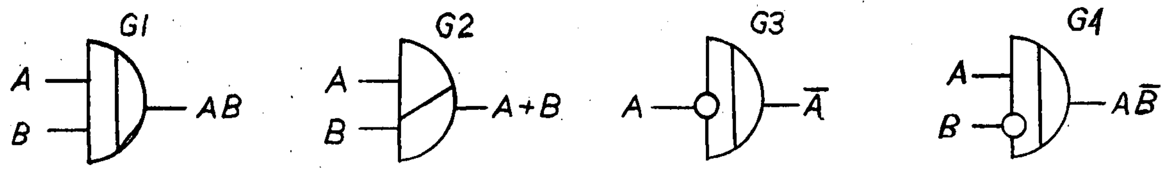
The simple logic of direct search makes it very well suited for SPDC's, and for an unknown performance surface it should be as capable of finding the optimum as any of the classical search strategies.

A possibility for further study is an investigation of how the choice of search strategy affects the size of the computer as the number of control parameters increases.

APPENDIX A

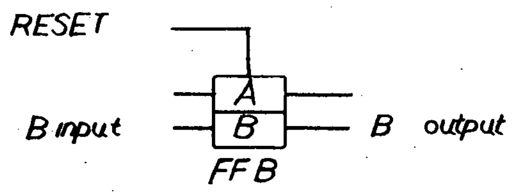
LOGIC SYMBOLS

Gates



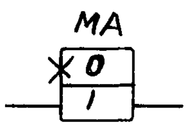
Gates are given a number and in the text the signal from a gate is referred to by this number, e.g. G1 is " 1 " when both A and B are "1", G3 is " 1 " when A is "0".

Flip-flop



The "1" level appears on the B output line if the device is set to the B state by a positive going signal on the "1" input. All flip-flops are labelled with letters, i.e. FFB.

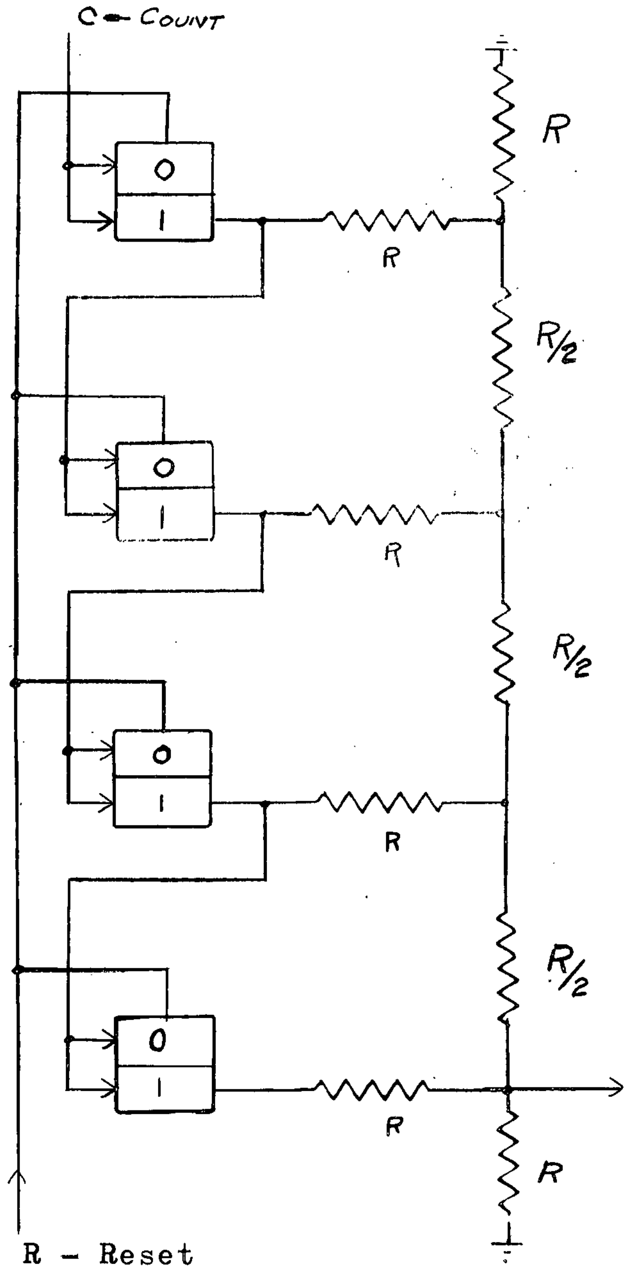
Monostables

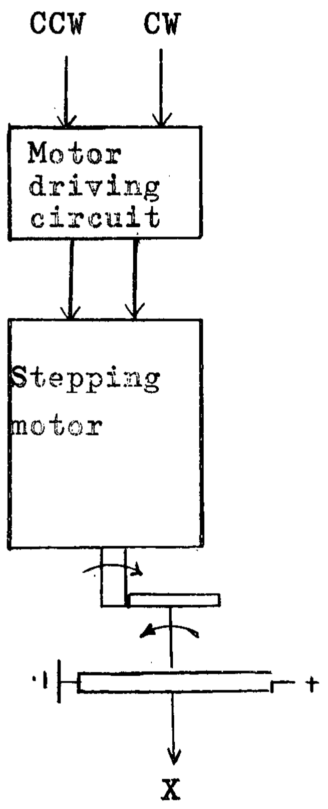


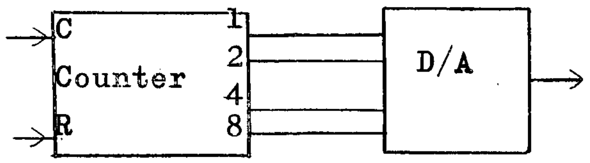
The monostable is normally in state 0 until set to 1 by a pulse and remains so set for an adjustable time period.

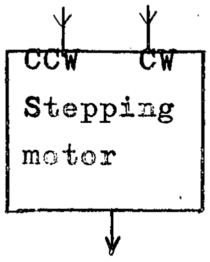
Very often combinations of logical elements will reoccur. These units will be given a unique symbol--a rectangle with a designated letter. (see page 55)

D/A	Digital to Analog converter
C	Comparator


C ← Count
0
1
R
R/2
R
R/2
R
R/2
R
R
R - Reset


CCW
CW
Motor driving circuit
Stepping motor
X


C
Counter
R
1
2
4
8
D/A


CCW
CW
Stepping motor

REFERENCES

1. Process Computer Scorecard Updated, Control Engineering, March, 1965, p. 57.

2. Grabbe, E. M., "Digital Computer Control Systems; an Annotated Bibliography", Proceedings of the First Congress of International Federation of Automatic Control, Butterworth, London, 1960.

3. Bohn, E. V., "The Practical Realization of Optimal Control of Multivariable Dynamic Processes", Proceedings of the Conference on Canadian Industrial Research, Carleton University, July, 1964.

4. Bohn, E. V., "The Synthesis of Real-Time Optimal Controllers by Hybrid Computer Techniques", Proceedings of the National Conference on Automatic Control, Carleton University, September, 1965.

5. Gibson, J. E., "Mechanizing the Adaptive Principle", Control Engineering, October 1960, p. 109.

6. White, B., "The Quarie Optimal Controller", Instruments and Automation, November, 1956, p. 2213.

7. Progress Report on Opcon: Dow Evaluates Optimizing Control, Control Engineering, November, 1959, p. 124.

8. Opcon Fills a Gap, Control Engineering, February, 1959, p. 27.

9. Feldbaum, A. A., "Automatic Optimalizer", Automat. Telemech., July, 1958, (English Translation April, 1959, p. 718.)

10. Stakhovski, R. I., "Twin-Channel Automatic Optimizer", Automat. Telemech., July, 1958, (English Translation April, 1959, p. 729.)

11. Boas, A. H., "Optimizing Multivariable Functions", Chemical Engineering, Vol. 70, No. 5, March 4, 1963.

12. Kushner, H. J., "Hill Climbing Methods for Optimization of Multi-Parameter Noise Disturbed Systems", J.A.C.C., 1963.

13. Brooks, S. H., "A Comparison of Maximum-Seeking Methods", Operations Research, Vol. 7, 1959.

14. Levine, L., "Methods for Solving Engineering Problems Using Analog Computers", Optimization Techniques, McGraw-Hill Inc., Toronto, 1964, Chapter 8.

15. Turnblade, R. C., "Random Optimization & Multiple Adaptive Control", Adaptive Control Systems, Pergamon Press, New York, 1963.

16. Wilde, D. J., "Optimization Methods", Advances in Chemical Engineering, Vol. 3, Academic Press, New York, 1962.

17. Idelsohn, J. M., "Ten Ways to Find the Optimum", Control Engineering, June, 1964, p. 97.

18. Hooke, R. and Jeeves, T. A., "'Direct Search' Solution of Numerical and Statistical Problems", Assn. for Computing Machinery Journal, The Association for Computing Machinery, Vol. 8, 1961, p. 212.

19. Mitchell, B. A., "A Hybrid Analog-Digital Parameter Optimizer for ASTRAC II", Simulation, Vol. 4, No. 6, June, 1965.

20. Chestnut, H., "Automatic Optimization of a Poorly Defined Process", J.A.C.C., 1963.

21. Korn, G. A., and Korn, T. M., Electronic Analog and Hybrid Computers, McGraw-Hill Inc., Toronto, 1964.

22. Flores, I., Computer Logic, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1960.

23. Roberts, S. M., and Lyvers, H. I., "The Gradient Method in Process Control", Industrial and Engineering Chemistry, Vol. 53, No. 11, November, 1961, p. 877.

24. Bushell, C., and Malnarich, M., "Reagent Control in Flotation", Mining Engineering, July, 1956.

25. Bushell, C., "Behaviour of Xanthate in Flotation", The Canadian Mining and Metallurgical Bulletin, March, 1958.

26. Lefkowitz, I., "Computer Control", Handbook of Automation, Computation, and Control, Vol. 3, John Wiley & Sons, Inc., New York, 1961.