AN ELECTRONIC DIGITAL POLYNOMIAL ROOT EXTRACTOR

Thesis by

Robert Royce Johnson

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1956

ACKNOWLEDGEMENT

I would like to express particular appreciation to Dr. Stanley P. Frankel for his ideas and suggestions, and for the stimulating encouragement he contributed during the development of this thesis. I would also like to thank Dr. G. D. McCann and the members of my thesis committee for their helpful interest in this work.

The material described covers a portion of the work done while I held a Howard Hughes Fellowship in Electrical Engineering and an International Business Machines Corporation Fellowship in Electrical Engineering. In addition, thanks are due Hughes Aircraft Company for providing both the magnetic drum heads and the magnetic oxide coating for the memory drum used in the computer.

To my wife, Mary, special thanks are due for her patient and omniscient comments, encouragement, and help in making the drawings and typing the rough draft of this thesis.

AN ELECTRONIC DIGITAL POLYNOMIAL ROOT EXTRACTOR

ABSTRACT

Many mathematical techniques exist for factoring algebraic polynomials. Most require much computation and programming and are practical methods only with large machine computers. A special purpose electronic digital computer designed to factor polynomials of high degree is described. The mathematical method is an adaptation of a Taylor serial approximation used to connect the problem and its formulation with a special machine implementation. The computer uses a small rotating magnetic drum, about 200 germanium diodes, and 20 logical flip-flops. Unique features of the system are the simple algebraic logical design techniques and the ease of programming. The result is a small, simple, and useful computer.

TABLE OF CONTENTS

		Page
I.	Introduction	1
II.	Principle of Operation	3
III.	Logical Design	10
IV.	Circuit Design	56
V.	Operational Instructions	89
VI.	Conclusions	93
	Appendix 1 - Error Analysis	97
	Appendix 2 - Polynomial Preparation	101
	Appendix 3 - Circuits	106
	Appendix 4 - Typical Problem	120
VII.	References	127

I. INTRODUCTION

The advent of the large scale digital computer has resulted in a general emphasis on numerical methods. This emphasis has led to many applications of digital techniques to problems having special characteristics; computers designed to capitalize on these characteristics can obtain solutions rapidly and with considerable savings in computing equipment. Reductions in preparation and programming time can result from machines designed to handle problems appearing repeatedly in practice.

In scientific computations, one universally belabored problem is that of factoring algebraic polynomials [1,2,3]. Many analog devices [4,5,6,7,8,9,10] and many mathematical methods [11,12,13,14,15,16,17,18] have been developed to solve this problem. A special purpose computer has been designed and constructed at the California Institute of Technology to reduce the programming and scheduling delays involved in placing polynomials in large scale machines. Other reasons for its construction were the development of new computer techniques and the desire for a machine having more versatility and accuracy than any of the devices now available.

This computer is designed to handle polynomials up to the 16'th degree. Operating in the binary number system, it requires 20 flip-flops and approximately 200 germanium diodes. The operating memory is one circulating register with one clock channel on a small magnetic drum. Input is bit by bit using a pair of switches, and output is visual on an oscilloscope. The only modifications necessary to extend this to higher degree polynomials would be a larger drum or a higher pulse

density. The computer obtains the complex roots of polynomials having real or complex coefficients with an accuracy of approximately six decimal digits. Solution times average about eight seconds per root.

II. PRINCIPLE OF OPERATION

The general method is that of evaluating a polynomial in the complex domain. The computer is designed to:

1) Evaluate the polynomial for successive increments in its complex argument, and
2) Select the complex increment that always decreases the absolute value of the polynomial.

A brief description is given of the problem preparation requirements and of the numerical accuracies attainable.

A. Mathematical Techniques

The computer generates values of the polynomial by repeated steps of Δ in its complex argument. Before each step the value: $\Delta = \pm \delta$ / $\Delta = \pm j\delta$ is chosen such that the step will diminish the absolute value of the polynomial. In this fashion the argument is modified as the computer assumes the direct path to the nearest root. When it reaches a minimum value for the polynomial, the normal Δ selection causes the computer to encircle the point of minimum absolute value.

The single memory channel contains 21 word positions. The real and imaginary components of the polynomial are in the first word; in each of the following 18 words is located the corresponding derivative of the polynomial. This notation is shown in Figure 1.

These derivatives, evaluated at some convenient point such as the origin, comprise the initial input data. The computational principle is to evaluate each derivative at an incremental distance Δ away from the initial coordinate. The computational principle is shown in Figure 2.

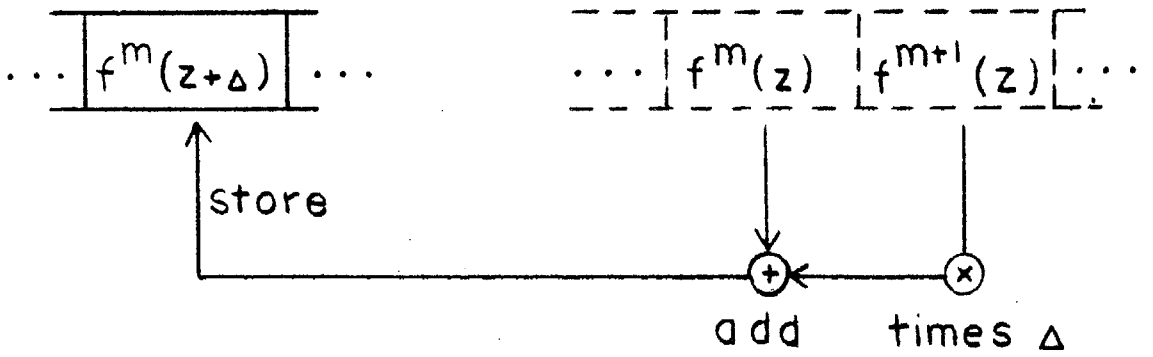
$$f(z) = a_n z^n + \cdots + a_1 z + a_0$$

$$f'(z) = n a_n z^{n-1} + \cdots + a_1$$

$$\vdots$$

$$f^n(z) = n! a_n$$

Figure 1



$$f^m(z+\Delta) = f^m(z) + \Delta f^{m+1}(z)$$

Figure 2

This approximation is the first term of a Taylor series expansion of the polynomial. All derivatives are recomputed for each step Δ using this principle.

For the initial root location an improved approximation formula is used with a larger value of δ. Higher order terms could be taken from the Taylor series, but a more elegant technique is to use part of each newly computed derivative:

$$f(z + \Delta) = f(z) + \frac{1}{2} \Delta f'(z) + \frac{1}{2} \Delta f'(z + \Delta) \qquad (3)$$

The $(m - 1)^{st}$ derivative is computed:

$$f^{(m-1)}(z + \Delta) = f^{(m-1)}(z) + \frac{1}{2} \Delta f^{(m)}(z) + \frac{1}{2} \Delta f^{(m)}(z + \Delta) \qquad (4)$$

Repeated approximations permit the computer to evaluate its polynomial for any complex argument. Coupled with the direction decision elements, the computer follows the shortest path to the nearest zero of the polynomial.

Several alternatives exist for finding all n roots. The method used is to continue the normal computation cycle but to force a desired Δ selection. The operator chooses the direction in which he wishes to look for roots and forces the computer to move in that direction. Released, the computer seeks the nearest root. Complex conjugate roots can be eliminated immediately.

B. Direction Decision

Considering the simplified approximation formula, the real and imaginary components for each step are found:

$$f(z) = u(z) + jv(z) \qquad f'(z) = u'(z) + jv'(z)$$

For $\Delta = \pm \delta$:

$$u(z+\Delta) = u(z) \pm \delta u'(z)$$
$$v(z+\Delta) = v(z) \pm \delta v'(z)$$

For $\Delta = \pm j\,\delta$:

$$u(z+\Delta) = u(z) \mp \delta v'(z)$$
$$v(z+\Delta) = v(z) \pm \delta u'(z) \qquad (5)$$

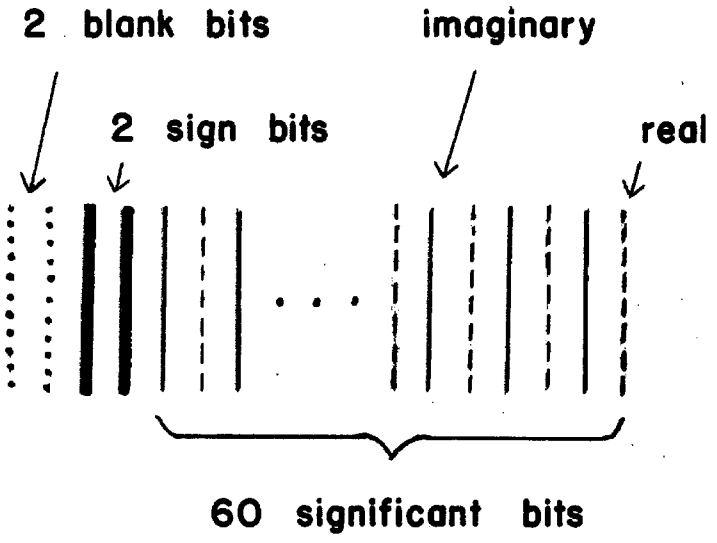
The coefficient of δ is selected in such a manner that the absolute value of both the real and the imaginary parts of $f(z+\Delta)$ would be diminished for an arbitrarily small δ. When the computer is encircling a root, each step Δ required by general rules may increase the absolute value of either coefficient. This overshoot is due to the fixed value of δ.

The direction control elements prescribe a sequence of steps which lead the computer to the vicinity of the dominant root. The computer encircles this region under the normal computational control of the direction decision elements until the operator forces the computer to move from the root.

C. Problem Preparation

There are two restrictions placed on the polynomial. It is necessary to convert the coefficients to binary numbers and to limit all numbers to absolute values less than $\frac{1}{2}$. There are 29 significant binary places available for the real and imaginary components of each derivative. Numbers are absolute value with sign when positive and zero's

complements with sign when negative. A zero in the sign position denotes a positive number, and a one denotes a negative number. The arrangement of the digits in each word is shown symbolically in Figure 3.



Biplexed Word Structure

Figure 3

To insure that $|z| < \frac{1}{2}$ for all roots, it is necessary to compute the radius of the contour of the z plane which encloses all n zeros of the polynomial.

$$R > \left| \frac{a_k}{a_n} \right| + 1 \tag{6}$$

where $|a_k|$ is the largest coefficient in the polynomial. A new argument is defined:

$$w = \frac{z}{2R} \tag{7}$$

which transforms the polynomial into:

$$f(w) = a_n (2R)^n w^n + \ldots + a_1 (2R) w + a_o . \tag{8}$$

The computer obtains the roots of (8). The roots of the original polynomial are found using (7). To insure that the absolute value of the n derivatives never exceeds $\frac{1}{2}$, it is necessary to divide f(w) by the largest coefficient in (8).

$$F(w) = \frac{f(w)}{2\left|(2R)^i a_i n!\right|} \tag{9}$$

$$= A_n w^n + \ldots + A_1 w + A_o \tag{10}$$

Here i is the index of the largest coefficient in (8).

These computations must be done manually before inserting the derivatives into the computer. This is done to retain the computational simplicity of the computer. No multiplications or divisions are provided internally.

D. Accuracy

Each step in the approximation of the function and its derivatives has a truncation error.

$$\epsilon = \sum_{i=m+3}^{n} \frac{\Delta^i}{2\ i!}\ (i-2)\ f^{(i)}\ (z) \qquad (11)$$

This is the error in the approximation to the m'th derivative. The error in the function itself is of the order of:

$$\frac{\Delta^3}{12} f^{(3)}\ (z)$$

Two values of δ are used: 2^{-10} and 2^{-20}. The computation sequence is to locate the root with the larger and refine it with the smaller δ. With the coarse δ, the truncation error at each step is of the order of 2^{-34}. The total error in 2^{10} steps is 2^{-24}. The root is refined using approximation (2) with $\delta = 2^{-20}$. In 2^{10} steps this produces an error of 2^{-32}.

Round-off errors may propagate to the 21'st binary position in 2^{10} steps so that the polynomial, its derivatives, and the value of z may be considered accurate to the 20'th binary place. The smallest increments in z are $\delta = 2^{-20}$.

To obtain full accuracy for all roots, it is necessary to normalize by (7) and locate the roots approximately. Full significance is obtained for the smaller roots by renormalizing the polynomial to an R just greater than the modulus of the desired root.

III. LOGICAL DESIGN

The computer has three modes of operation:

A. Input

B. Computation with $\delta = 2^{-10}$ or $\delta = 2^{-20}$

C. Direction Decision

The operational aspects of these modes have been described. This section presents the detailed logical design and the techniques by which these operational functions are achieved.

In order to understand these functions it is helpful to consider some of the internal features of the computer. The specific functional operations of the computer are described in terms of Boolean equations. There are many possible formulations for each of these operations. For brevity and clarity, only the final formulation is presented along with the reasoning associated with its derivation.

The design process for this computer is carried out in the following sequence. The initial study indicates the general mathematical approach and the probable engineering solution for the problem. In the present case it is desired to obtain the roots of algebraic polynomials. A rotating magnetic drum, electronic vacuum tubes, and germanium diodes constitute the engineering tools available for this high speed digital computer. The logical design of such a machine is based on the mathematical description of the properties of these components.

The active elements of most modern high speed digital computers have some form of power amplification and some memory properties. The

particular active device chosen for this computer is a bistable vacuum tube circuit generally called a flip-flop. In order to design computer circuits to operate with this basic element, it is convenient to describe the behavior of the flip-flop in mathematical terms.

Two types of these bistable elements are used. Each has essentially the same engineering features, but has distinct logical behavior. One type of flip-flop has a single input terminal, while the second type has two input terminals. Both types of flip-flop operate on the input signals by introducing a time delay. Time in this computer is measured in terms of a reference clock time generated from an auxiliary channel on the magnetic drum. Pulses are formed from the signals read from the clock channel; these clock pulses are used to actuate simultaneously all the flip-flops in the computer. All of the computer operations are synchronized by these clock pulses.

Information in the computer is handled in terms of the high and low voltage states of the flip-flops. A high voltage state is generally referred to as a 1, and a low voltage state is called a 0. During any one clock pulse interval a flip-flop may assume either state. Since either a binary digit 1 or 0 appears on the flip-flop during this clock interval, it is convenient to refer to a clock pulse interval as a bit time.

All numbers processed in the computer are handled in time sequence. In computer terminology, this machine is a serial, binary, special purpose, magnetic drum computer. This means that the binary numbers of each word appear sequentially on any specified flip-flop. These numbers, represented by voltage states, are set into one set of flip-flops which read the signals from the drum surface. Another set of flip-flops performs

arithmetic operations on these numbers and places the numbers back on the drum surface in the form of small regions magnetized in one direction or the other.

To facilitate both the description and the mathematical derivation of the computer operation, the two types of flip-flops are described in Table I.

TABLE I

Flip-Flop Truth Table

Type 1 - Two Input				Type 2 - One Input		
JQ_n	KQ_n	Q_{n+1}	$\overline{Q}_{n+1}$	JQ_n	Q_{n+1}	$\overline{Q}_{n+1}$
0	0	Q_n	$\overline{Q}_n$	0	0	1
1	0	1	0	1	1	0
0	1	0	1			
1	1	$\overline{Q}_n$	Q_n			

The inputs to flip-flop Q are designated as JQ or KQ. These inputs are formed with germanium diode gates, and the input signals may be complex functions derived from the output signals of other flip-flops. The subscripts n and n+1 denote the clock pulse intervals during which the input and its corresponding output occur. The bar above the letter Q denotes the complement of whatever signal the flip-flop Q contains. Each flip-flop has two outputs, Q and $\overline{Q}$. These signals are the opposite of each

other; Q high means $\overline{Q}$ is low, and Q low means $\overline{Q}$ is high.

In the design process it is the input equations to each flip-flop in the computer which must be specified. Each flip-flop is assigned a certain designation; the input equations for this flip-flop are specified by placing a J or a K in front of the letter and number identifying the particular flip-flop. The input equation itself is derived in terms of the two logical concepts, algebraic union and intersection. These two operations are respectively designated with a plus sign: + and a dot: · . In computer terminology, the germanium diode circuit performing the union operations is called an or gate. The diode circuit performing the intersection operation is called an and gate.

A. Input

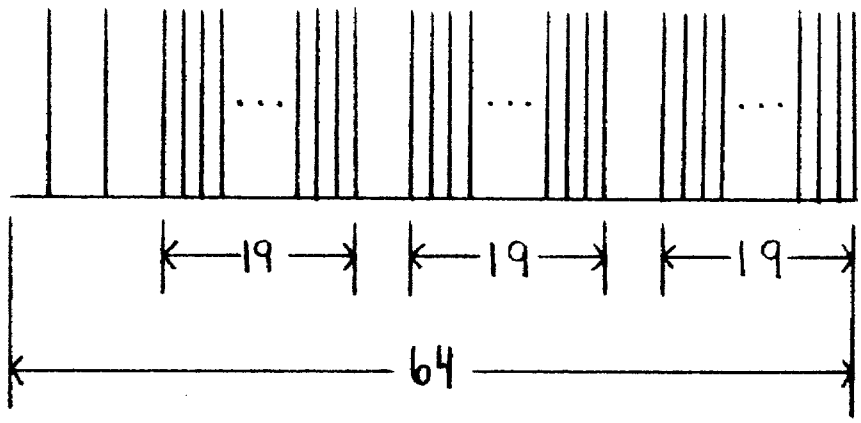
There are two timing operations which must be considered in order to understand the input operations. Phase and timing signals must be supplied to orient the computer contents into a recognizable sequence. These signals and the method of their derivation will be described before attempting the characterization of the input operations.

1. General Timing Signals:

The use of a single circulating register on the drum precludes the use of an origin pulse. Unless the register is a full drum circumference in length, it is necessary to provide signals which identify each word position and which do not depend on a particular drum location for their presence. The drum is divided into 21 word positions. Each word contains 62 bits and it is separated from the next word by two blank bits. Timing signals are necessary to identify the blank bits and the sign bits

of each word. In addition, the numerical operations of the computer require two other forms of timing signals. The word contents are biplexed so that it is necessary to provide a timing signal to distinguish the two biplexed components. Also, there must exist a timing signal during the word time. This timing signal is used in conjunction with the multiplication of the higher order derivatives by 1/2 δ. This multiplication is done by shifting the appropriate reading heads, and hence there must exist timing signals to identify portions of each shifted word.

This timing function is obtained from the clock channel. This channel is derived from the uniformly divided channel by occasionally omitting a single pulse. The clock channel thus has the appearance of Figure 4. Using the letter $\overline{P}$ to designate that voltage state existing when no pulse appears on the clock channel, one may derive a set of Boolean equations describing a counter which automatically orients itself to the proper configuration with respect to the clock channel. In Figure 5 the complete contents of a single word are shown in conjunction with the three timing signals generated by the flip-flops Q1, Q2, Q3. During all 21 word times required during a single drum revolution these identical signals appear.



Clock Pulses

Figure 4

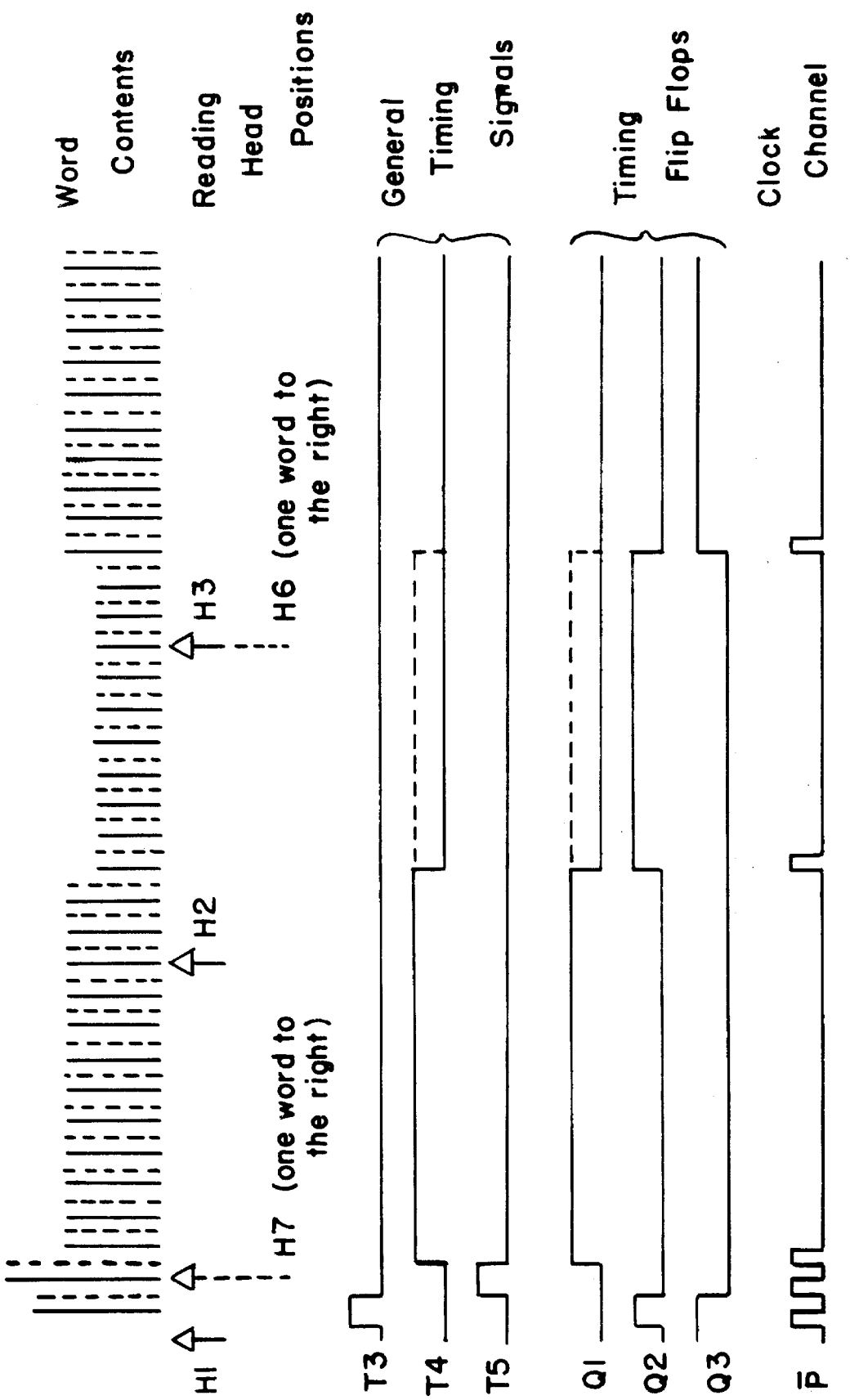


Figure 5 General Timing

These timing signals have the following significance:

T3 = Blank bit times.

T4 = δ - shifted blocking signal.

T5 = Sign bit times.

The use of T4 is simply that of preventing undesired bits from being read into the computer from the shifted reading heads H2 or H3 and H6 during that part of the word time when these heads read bits from the following word. These timing signals are derived from the flip-flops of the counter.

$$T3 = Q2\ Q3$$
$$T4 = Q1 \qquad (12)$$
$$T5 = \overline{Q1}\ \overline{Q2}\ \overline{Q3}$$

In deriving the input logic to Q1, Q2, Q3, it is necessary to consider the following effects. The odd-even timing within the word distinguishing the real and imaginary components is generated by the flip-flop B. During those times when a real bit is being read by H1, the signal B=1; when an imaginary bit is being read by H1, B=0. Thus B and P occur at corresponding times. B is synchronized by writing the input equations:

$$JB = C_p\ \overline{\overline{P}}$$
$$KB = C_p$$

Here C_p designates the clock pulse derived from the general clock pulse generator.

Two switches control the value of δ by which the computer is multiplying its higher order derivatives.

S1 = Fine delta: $\delta = 2^{-20}$

S3 = Coarse delta: $\delta = 2^{-10}$

It is necessary to change the timing signal T4 to accommodate this change in δ .

Two requirements are placed on the basic counter. First it must synchronize itself to give the T3, T4, T5 pattern regardless of the initial configuration. Second, it is desirable to have T4 consist of a single flip-flop signal. This is due to the heavy load of both and and or gate signals supplied by T4. A third requirement is that of changing from the fine and coarse δ sequences under the control of a switch. With these conditions, a set of logical equations may be derived.

$$JQ1 = \overline{P}\ \overline{Q2}\ Q3\ (S1 + \emptyset 1) + \overline{P}\ Q2\ \overline{Q3}\ S3 + B\ P\ \overline{Q2}\ \overline{Q3}$$

$$KQ1 = \overline{P}\ \overline{Q2}$$

$$JQ2 = \overline{P}\ \overline{Q1}$$

$$KQ2 = \overline{P}\ \overline{Q1} + \overline{P}\ \overline{Q3}$$

$$JQ3 = \overline{P}\ \overline{Q1}\ \overline{Q2}$$

$$KQ3 = \overline{P}\ \overline{Q2} + B\ P\ Q2$$

The terms on JQ1 and KQ3 involving B are used to synchronize the count sequence with that desired, if the initial conditions on Q1, Q2, Q3 are such that the counter would otherwise present the wrong timing signals during the various portions of each word.

In the equations above, $\emptyset 1$ is a phase signal which is 1 during the input phase or during any part of a computation phase when it is desired to circulate the memory contents without modification.

2. Phase Control:

Three flip-flops, $\phi 1$, $\phi 2$, $\phi 3$ are used to indicate particular word positions within the memory channel. The input information for these flip-flops consists of marker bits placed in the blank bit positions indicated by T3. These marker bits are inserted when the machine is placed in the Operate condition by switch S4.

Two fixed and one movable marker exist. The fixed markers appear in the first blank bit preceding the words containing f(z) and z. They appear on M1, the flip-flop reading H1, during the time interval T3 $\overline{B}$. The movable marker may be placed in the second bit position T3 B, preceding any of the words in the memory channel. This movable marker is used for input purposes where it is desired to operate on the word following the marker.

The fixed markers are referred to as m1 bits; the movable marker is called m2. m1 appears during time T3 $\overline{B}$ on M1 and during time T3 B on the writing flip-flop M7.

The functions of the three phase control flip-flops are simple:

$\phi 1$: This flip-flop distinguishes between recirculation of the memory and computation. During most of the input mode, $\phi 1 = 1$; during most of the computation mode, $\phi 1 = 0$.

$\phi 2$, $\phi 3$: These flip-flops generate phasing signals independent of $\phi 1$. This is necessary in order to synchronize circulation and computation during those intervals

when the computer is being switched from one mode to another.

The functional behavior of the phase control may be tabulated as follows:

$\phi 1$	$\phi 2$	$\phi 3$	Function:
0	0	0	Normal Computation
0	1	0	That word time during which f(z) is being read from M1. Refer to Figure 6.
1	0	1	Recirculation. This is the word holding Δ, and it is not necessary to add anything into this word.
0	1	1	Normal Computation. This is the word containing z.
1	0	1	Recirculation. This word contains the value of f"(z).
0	0	0	Normal Computation.

$\phi 2$ and $\phi 3$ constitute a counter which is continually counting. The sequence is independent of the mode of operation of the computer. The input equations may be written:

$$
\begin{aligned}
J\phi 2 &= T3 \; B \; M7 \\
K\phi 2 &= T3 \; B \; \overline{M7} \\
J\phi 3 &= T3 \; B \; \phi 2 \\
K\phi 3 &= T3 \; B \; \phi 1 \; \overline{M7}
\end{aligned}
\qquad (14A)
$$

The operation of $\phi 1$ is more complex. A five position shorting switch is used to control the mode changes.

S1 = Fine δ computational mode

S1' = Intermediate position

S2 = Input mode

S3' = Intermediate position

S3 = Coarse δ computational mode

The five position switch makes it possible to electrically separate the logical gates used during these separate modes. However, it is necessary to indicate to the computer that a mode change is about to occur. The two intermediate switch positions make this electrical separation possible.

By using all shorting connections, it is possible to isolate the gates having (S1 or S3) or S2 controlling their operation. When the switch is moved from a compute mode, however, contact is made with the intermediate position before the computing contact is broken. With this mutual contact, it is possible to set the phasing control $\phi 1$ to 1, which commands recirculation as soon as the origin is reached. Thus partial computations are prevented, and the logical gates may be considered separate entities.

$$J\phi 1 = T3 \; B \; \phi 2$$
$$K\phi 1 = \overline{(S1' + S2 + S3')} \; S4 \; + \; S8 \; \overline{S4} + \overline{\phi 2} \; \phi 3 \; T3 \; B \; \overline{S8} \qquad (14B)$$

During computation the three phase flip-flops operate as above. When it is desired to change modes, $\phi 1$ is set to 1 at the origin. Returning to a computational mode from a circulational one, $\phi 1$ is again synchronized

to the origin and set to 0 as soon as it is certain that all contacts on either S1 or S3 are made. The switch signal $\overline{S4}$ is used to preset $\phi 1$ to 0 as desired during marker insertion.

3. Marker Insertion:

There are two switches used to control the initial marker insertion in the recirculating memory channel. $\overline{S4}$ is used to erase the initial contents of the drum during warm-up of the computer. Switching S4 to the 1 state initiates the marker insertion and places the computer in an operational state. S8 is used to separate the gates used during marker insertion from the gates used on the same flip-flops during other modes of operation. S8 = 1 during marker insertion, and S8 = 0 for the remaining computer functions.

During marker insertion, S8 = 1. The markers are inserted via the writing flip-flop M7, which reads the contents of flip-flop G.

$$JM7 = S8(G\ C1\ T3 + M1\ S4) \tag{15A}$$

Since only 0's exist on the track initially, and since it is necessary to recirculate the markers after they are written, M1 is gated into JM7 as well. G and H are connected to form a counter, the contents of which is read into M7 as above.

$$\begin{aligned} JG &= S8\ S4(\overline{C1} + H)\ T3\ \overline{B} \\ KG &= S8\ (T3\ \overline{B}) \\ JH &= S8\ (G\ T3\ \overline{B}\ \phi 1) \\ KH &= S8\ (G\ T5\ B) \\ JC1 &= S8\ (G \cdot S4) \\ KC1 &= S8\ (\overline{S4}) \end{aligned} \tag{15B}$$

The time sequence in G and H generated by setting S4 = 1 is initiated by the single bit entered into JG when S4 = 1 and C1 remains 0. C1 is used here to form this one bit at the start of the marker insertion. The count sequence is tabulated below:

Time	G	H	∅1	M7
S4 = 0	0	0	0	-
S4 = 1, C1 = 0	1	0	0	-
S4 T3 $\overline{B}$ = 1	1	0	0	-
S4 T3 B = 1	0	1	0	1
(next bit) $\overline{B}$ = 1	0	1	0	0
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
S4 T3 $\overline{B}$ = 1	0	1	0	-
S4 T3 B = 1	1	1	0	0
(next bit) $\overline{B}$ = 1	1	1	1	1
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
S4 T5 B = 1	1	1	1	-
S4 T3 $\overline{B}$ = 1	1	0	1	-
S4 T3 B = 1	0	0	1	1
(next bit) $\overline{B}$ = 1	0	0	1	0

$\phi 1$ is used on JH to prevent its being reset to 1 after going through one cycle.

Setting switch S8 = 0 then places the computer in the Operate mode. However, to insure that the markers will recirculate during this switching operation, it is desirable to write the input to M7 as follows:

$$JM7 = S8(C1\ G\ T3 + M1\ S4) + (S1' + S3')\ \phi 1\ M1$$

The markers cannot be inserted with the control switch S2 = 1 because the memory channel cannot be recirculated under the control of $\phi 1$ alone during the input mode. It is necessary to insert information into the and f"(z) words otherwise marked by $\phi 1$ for recirculation only. Thus the computer is started with the mode switch in either of the two intermediate positions S1' or S3'.

4. Bit Insertion:

Switching the mode switch S2 = 1 and the marker switch S8 = 0 prepares the computer to accept input information. Input is bit by bit. Each new bit is entered most significant bit first into the word position marked by the marker m2. The input bits must be biplexed in the way that they are to appear in the word itself.

The four carry flip-flops of the arithmetic unit are used to enter the input data. They have the following functional operation:

C2: Accepts the input bit. Upon receipt of a new bit, the word marked by m2 is recirculated back to the memory via C2. This precesses the word contents by one bit position and inserts the new

bit in its proper place. Input is most significant bit first.

C1: Indicates when a new input bit has been stored in C2.

C3: Is set to 1 to precess the marker m2 by one word position.

C4: Controls the recirculation of M1 to M7 via C2 during the word marked by m2.

The marker delay function of C3 will be considered in the next section.

During input, the recirculation path is controlled by C4.

$$JM7 = S2\lceil M1\ \overline{C4} + C2\ C4 \rfloor + \ldots$$

This function is accomplished by setting C4 = 1 with m2:

$$JC4 = S2\lceil C1\ M1\ T3\ B \rfloor + \ldots \quad .$$

During T3 B = 1, the reading flip flop M1 contains a 1 if the marker m2 is present. C1 = 1 only when a new bit has been placed into C2. Using a single input type flip-flop necessitates holding the information until the end of the word:

$$JC4 = S2\lceil C1\ M1\ T3\ B + C4(\overline{T5} + \overline{B}) + \ldots \rfloor$$

In this fashion the 62 bits of each word are entered without affecting the marker locations. The presence of one new bit is indicated by setting G and C1 with the two input switches S5 and S6.

$$JC1 = S2\lceil G(S5 + S6) \rfloor$$

$$KC1 = S2\ C4$$

Similarly for G: $JG = S2\ C1\ (S5 + S6)$

$KG = S2\ \overline{S5}\ \overline{S6}$.

C1 remains in the 1 position until C4 resets it. G prevents resetting C1 until either S5 or S6 have been returned to their normal position.

The input bits are placed in C2 using switches S5 and S6.

$$JC2 = S2/\overline{C4}(S5 + \overline{S6}\ C2) + C4\ M1 + \ldots\ /$$

S5 = 1 when depressed and places a 1 in C2. S6 = 1 when depressed and places a 0 in C2. C4 is used to allow the other bits in the marked word to recirculate via C2.

5. Marker m2 Precession.

The two switches S5 and S6 are three position lever switches. Depressing either switch enters the indicated bit. The upper position of switch S6 sets H and C3 to initiate precession of marker m2.

$JC3 = S2\ \bar{H}\ S6^*$

$KC3 = S2\ C4$

$JH = S2\ C3\ S6^*$

$KH = S2\ \bar{S6}$

Note: S6* = 1 means raising the switch s6, while S6 = 1 means depressing it.

The rest state of C4 during S2 = 1 is 0. This holds C3 in the 1 state until C4 is activated. C4 is set to 1 at the end of the word following the old m2 for one bit time. This bit is entered into M7 as the new marker m2. C4 = 1 resets C3 to 0, and it also resets C2 to 0.

$$JC4 = S2/\ C3\ C2\ T3\ \bar{B} + C4\ \overline{C3}(\overline{T5} + \bar{B}) + \ldots\ /$$

C2 is set to 1 by the old marker m2.

$$JM7 = S2/\ M1\ \overline{C4}(\overline{C3} + \overline{T3} + \bar{B}) + C2\ C4\ /$$

$$JC2 = S2/\ M1\ C3\ T3\ B + C2\ \overline{C4}(H + \overline{S6}) + \ldots\ /$$

6. Summary of Logical Equations.

Input Mode:

$$JM7 = S8(C1\ G\ T3 + M1\ \overline{S4}) + S2/\ M1\ \overline{C4}(\overline{C3} + \overline{T3} + \overline{B}) + C2C4/$$
$$+ (S1' + S3')\ \emptyset 1\ M1$$

$$JC1 = S8\ G\ S4 + S2\ \overline{G}(S5 + S6)$$

$$KC1 = S8\ \overline{S4} + S2\ C4$$

$$JC2 = S2/\ \overline{C4}\ S5 + C2\ \overline{C4}(H + \overline{S6}) + M1\ C4 + M1\ C3\ T3\ B\ /$$

$$KC3 = S2\ C4$$

$$JC3 = S2\ \overline{H}\ S6^* + (S1' + S3')\ \overline{\emptyset 1}$$

$$JC4 = S2/\ C1\ M1\ T3\ B + C3\ C2\ T3\ \overline{B} + C4\ \overline{C3}(\overline{T5} + \overline{B})\ /$$

$$JG = S8/\ T3\ \overline{B}(H + \overline{C1})S4\ /\ ^* S2/\ C1(S5 + S6)\ /$$

$$KG = S8\ T3\ \overline{B} + S2\ \overline{S5}\ \overline{S6}$$

$$JH = S8\ \overline{\emptyset 1}\ G\ T3\ \overline{B} + S2\ C3\ S6^*$$

$$KH = S8\ G\ T5\ B + S2\ \overline{S6}$$

Phase Control:

$$J\emptyset 1 = T3\ B\ \emptyset 2$$

$$K\emptyset 1 = \overline{\emptyset 2}\ \emptyset 3\ T3\ B\ \overline{S8} + S4(\overline{S1' + S2 + S3'}) + S8\ \overline{S4}$$

$$J\emptyset 2 = T3\ B\ M7$$

$$K\emptyset 2 = T3\ B\ \overline{M7}$$

$$J\emptyset 3 = T3\ B\ \emptyset 2$$

$$K\emptyset 3 = T3\ B\ \emptyset 1\ \overline{M7}$$

Timing Mode:

$$JB = Cp\ \overline{P}$$

$$KB = Cp$$

$$JQ1 = \overline{P}\ \overline{q2}\ Q3\ S1 + \overline{P}\ Q2\ \overline{Q3} + B\ P\ \overline{Q2}\ \overline{Q3}$$

$$KQ1 = \overline{P}\ \overline{Q2}$$

$$JQ2 = \overline{P}\ \overline{Q1}$$

$$KQ2 = \overline{P}\ \overline{Q1} + \overline{P}\ \overline{Q3}$$

$$JQ3 = \overline{P}\ \overline{Q1}\ \overline{Q2}$$

$$KQ3 = \overline{P}\ \overline{Q2} + B\ P\ Q2$$

B. Arithmetic Computation.

The mathematical technique utilized to evaluate the polynomial is the repeated application of the truncated Taylor series, Equation (4). There are several reasons for using this formulation. The computational symmetry which enables each derivative to be approximated by the same formula is perhaps the strongest reason for using this technique. Another consideration is that the real and imaginary components are each treated in a nearly identical fashion. The initial input to the computer is the real and imaginary component of the polynomial and its n derivatives; these components are evaluated manually at z = 0 for the initial inputs.

This computation places a minimum burden on the memory facilities necessary for the internal computer operation. These considerations, the simple Δ selection rules, and the minimum requirements placed on the arithmetic unit-addition or subtraction, all combine to indicate that this is a practical formulation for the problem in terms of a special purpose machine implementation.

Finite difference methods could be used, and they would be exact insofar as polynomials are concerned. If real variables were being used, this exact method would be the way to solve the problem. However to handle the differences for complex numbers would place a much greater burden on the memory facilities of the computer. The selection of the direction and the corresponding difference would be more difficult, and the manual evaluation of the initial input quantities (the complex differences) would be much more complex. To handle the accuracy problem, the present computer uses long words and two values of the step δ. Thus any desired accuracy can be obtained with the Taylor series approximation used.

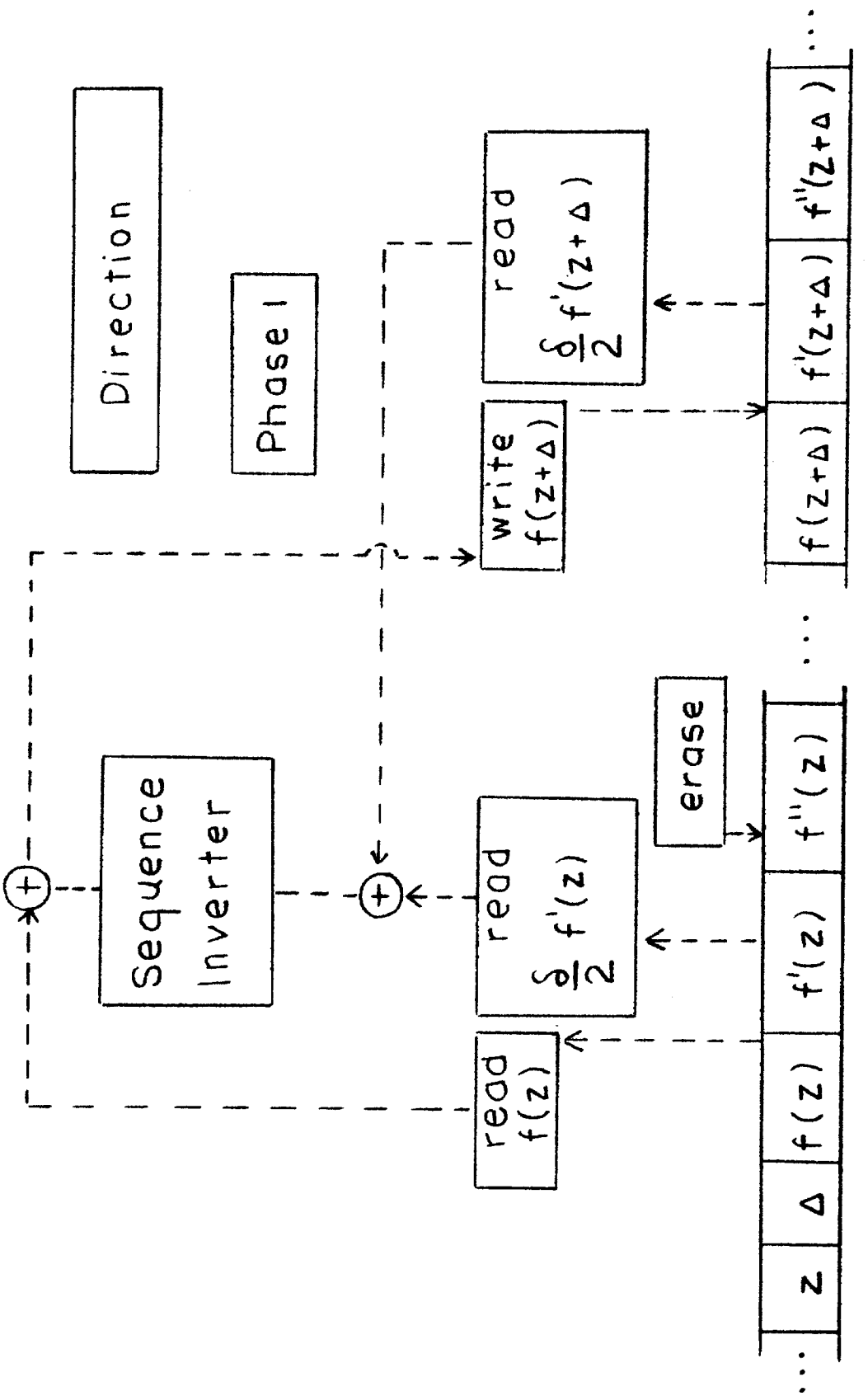


Figure 6

Computer Block Diagram

The general block diagram of the computer is shown in Figure 6. The contents of the computing channel on the drum are illustrated functionally. This channel occupies the entire circumferential length of the drum; it is essentially a recirculating register with several additional reading stations. Thus one reading station is set to read the complex value of $f(z)$ while its adjacent station is reading $\frac{\delta}{2} f'(z)$. Located one word position further around the drum is the writing station which is shown writing the value $f(z + \Delta)$ back onto the drum. Adjacent to this writing station is another reading position to obtain the value $\frac{\delta}{2} f'(z + \Delta)$. Figure 6 illustrates the memory orientation upon completion of one drum revolution. Starting with the highest derivative, reading and writing stations operate on a continuous basis; each lower order derivative is evaluated with this technique according to equation (4).

The output from the reading stations is always added:

$$S_1 = \frac{\delta}{2} f^1(z) + \frac{\delta}{2} f^1(z + \Delta)$$

or (16)

$$S_n = \frac{\delta}{2} f^n(z) + \frac{\delta}{2} f^n(z + \Delta)$$

This sum is placed into the Sequence Inverter, which does two things.

1) It places the proper biplexed bit sequence in synchronism with the output from the station reading $f(z)$. This sequence is determined from Equations (5).

2) It provides the 1's complements of the sum above when it is necessary to subtract according to Equations (5).

The outputs from the sequence inverter and the f(z) reading station are always added together and recorded back onto the drum.

The two special words, z and Δ, are used for the convenience of the operator. They are described in the input discussion, Part A of this section.

1. The Adders:

The derivative approximation (3) requires three numbers to be added together. From (5) it is shown that this must be done on both the real and the imaginary components. If the simple sum indicated by (16) is always generated, it may be added or subtracted from the value f(z). Thus two adders are used sequentially; the first generates the simple sum (16), and the second combines this output with f(z). Add-subtract control is governed by the direction flip-flops G and H.

The outputs from G and H are interpreted as follows.

$$\begin{aligned} H = 0 &\quad \text{corresponds to } \Delta = \pm \delta \\ H = 1 &\quad \text{corresponds to } \Delta = \pm j\delta \\ G = 0 &\quad \text{corresponds to } \Delta = -\delta \text{ or } -j\delta \\ G = 1 &\quad \text{corresponds to } \Delta = +\delta \text{ or } +j\delta \end{aligned} \tag{17}$$

A truth table shows the sign conditions and the value of Δ required to reduce the absolute value of $f(z + \Delta)$.

TABLE II

Sign of Components:				Flip-Flop States:		Operation on:		Sign of Components:				Flip-Flop States:		Operation on:	
u	u'	v	v'	Δ	H	G	R	I	u	u'	v	v'	Δ	H	G	R	I
+	+	+	+	$-\delta$	0	0	S	S	+	+	+	-	$-j\delta$	1	0	A	S
+	+	-	-	$-\delta$	0	0	S	S	+	-	-	-	$-j\delta$	1	0	A	S
-	-	+	+	$-\delta$	0	0	S	S	-	+	+	+	$-j\delta$	1	0	A	S
-	-	-	-	$-\delta$	0	0	S	S	-	-	-	+	$-j\delta$	1	0	A	S
+	-	+	-	$+\delta$	0	1	A	A	+	+	-	+	$+j\delta'$	1	1	S	A
+	-	-	+	$+\delta$	0	1	A	A	+	-	+	+	$+j\delta$	1	1	S	A
-	+	+	-	$+\delta$	0	1	A	A	-	+	-	-	$+j\delta$	1	1	S	A
-	+	-	+	$+\delta$	0	1	A	A	-	-	+	-	$+j\delta$	1	1	S	A

The operation column designates whether addition or subtraction is to occur on the real components (R) or on the imaginary components (I). Hence for the real components,

$$\begin{aligned} \overline{H}\cdot G + H\cdot\overline{G} &= 1 \text{ ------ Addition} \\ H\cdot G + \overline{H}\cdot\overline{G} &= 1 \text{ ------ Subtraction} \end{aligned} \tag{18A}$$

For the imaginary components:

$$\begin{aligned} G &= 1 \text{ ------ Addition} \\ \overline{G} &= 1 \text{ ------ Subtraction} \end{aligned} \tag{18B}$$

The sum (16) is always computed. The numbers read into the sequence inverter, however, are determined by Equations (18). Output from the

sequence inverter is added to the value of $f(z)$ and replaced on the drum. Hence the only arithmetic operation performed in the computer is addition.

From Figure 7, $\frac{\delta}{2} f'(z)$ is read by magnetic head H3 and placed in flip-flop M3. Similarly, $\frac{\delta}{2} f'(z + \Delta)$ is read by head H6 and placed in flip-flop M6. Two carry flip-flops are used, C1 and C2. This is convenient because there are two sets of numbers being added. The basic adder configuration is shown in Figure 7. The carry from each bit position and for both components is generated and placed in C2:

$$JC2 = M3 \cdot M6 + M3 \cdot C1 + M6 \cdot C1$$

$$JC1 = C2$$

Both C1 and C2 introduce delays of one bit time. Hence the corresponding carries appear in C1 for addition with the outputs from M3 and M6:

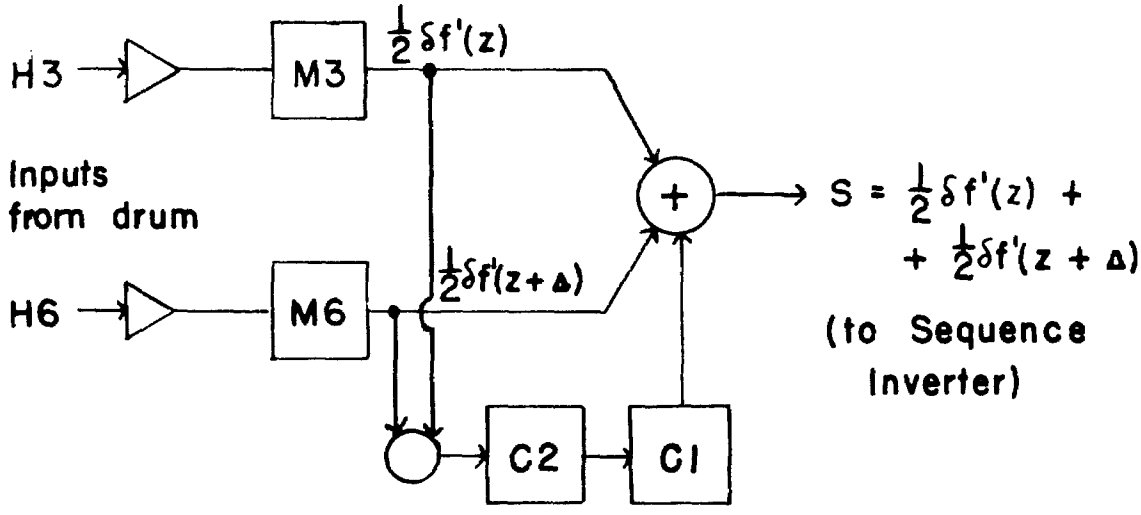
$$S = M3 \cdot M6 \cdot C1 + M3 \cdot \overline{M6} \cdot \overline{C1} + \overline{M3} \cdot M6 \cdot \overline{C1} + \overline{M3} \cdot \overline{M6} \cdot C1$$

A similar adder is used for $f(z)$ and the output from the sequence inverter. This sum is placed in the writing flip-flop M7, in Figure 8. The input to D2 is the modified information from the sequence inverter. The sum here is formed exactly as that in Figure 7.

$$JC4 = M1 \cdot D2 + M1 \cdot C3 + D2 \cdot C3$$

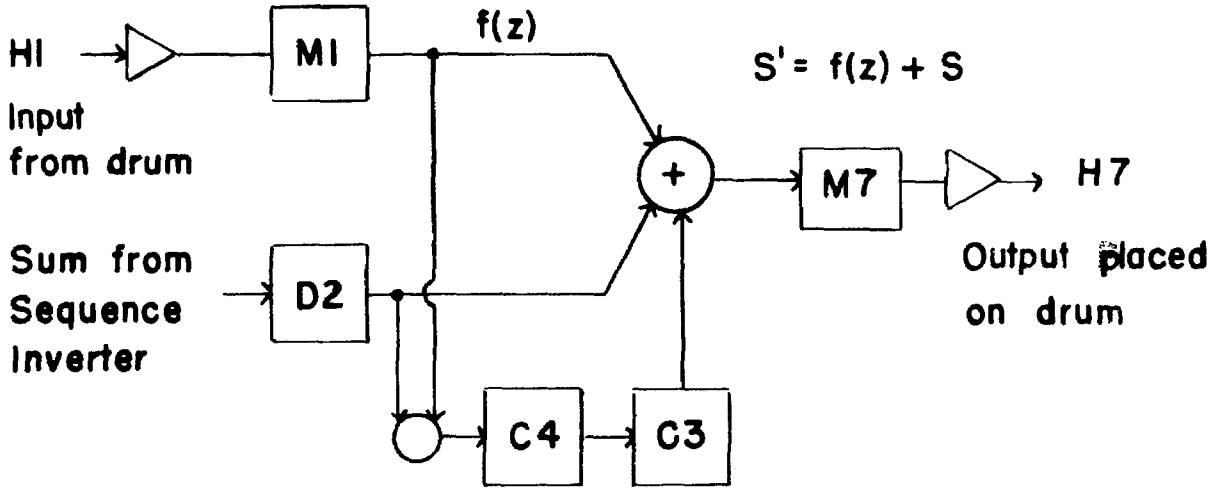
$$JC3 = C4$$

$$JM7 = M1 \cdot D2 \cdot C3 + M1 \cdot \overline{D2} \cdot \overline{C3} + \overline{M1} \cdot D2 \cdot \overline{C3} + \overline{M1} \cdot \overline{D2} \cdot C3$$



Basic Adder for Biplexed Numbers

Figure 7



Second Adder

Figure 8

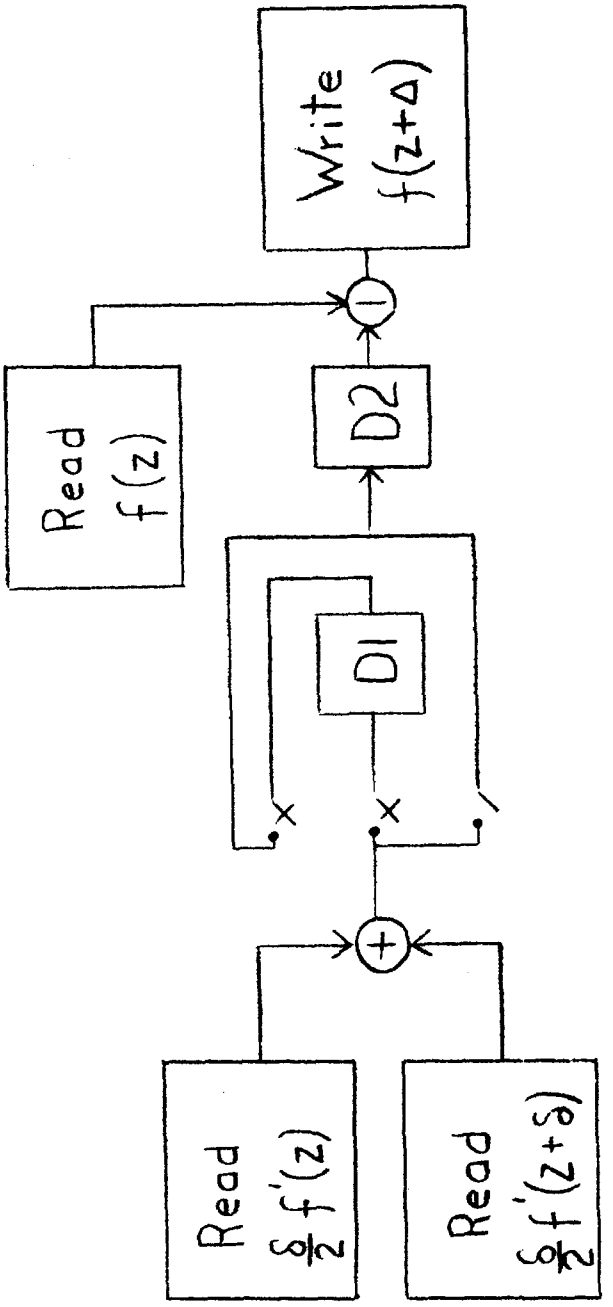
2. Sequence Inverter:

The output from the first adder is placed in delay flip-flop I1. This flip-flop is used to facilitate check-out and to save diodes. The output from I1 is the sum S. Two flip-flops D1 and D2 are used for the sequence inverter.

During computation with $\Delta = \pm\delta$, the flow of binary information proceeds sequentially through D1 and D2 as shown in Figure 9. However, during computation with $\Delta = \pm j\delta$, it is necessary to invert the biplexed positions of the binary information. This inversion is done with the inputs to D1 and D2 to provide the real and imaginary bits in their exchanged positions when read from D2 in accordance with Equations (5) and Figure 8.

The timing flip-flop B indicates which component, real or imaginary, is present at all times in M1. This is the flip-flop reading the value of f(z). B = 0 denotes the presence of a real bit in M1, and B = 1 denotes an imaginary bit. D1 and D2 introduce a delay of one bit time apiece. The contents of D1 are one bit time ahead of the corresponding contents of M1.

During $\Delta = \pm j\delta$ the role of D1 is that of storing each real bit for two clock pulse times. D2 reads the output from I1 during B = 1 time to obtain the imaginary bit from I1. This bit is to be added to the real bit from M1 in the following clock time. During B = 0 time D2 reads the stored contents of D1. Note that the real bit always precedes the corresponding imaginary bit in time sequence.



Sequence Inverter

Figure 9

The input equations to I1, D1, and D2 may now be written. I1 receives the sum S:

$$JI1 = S$$

D1 receives I1 during H = 0 time and accepts only the real bits during H = 1 time (from Equation 17).

$$JD1 = \overline{H}\cdot I1 + H\cdot I1\cdot\overline{B} + H\cdot D1\cdot B$$
$$= \overline{H}\cdot I1 + I1\cdot\overline{B} + H\cdot D1\cdot B$$

The third term in this equation is added because D1 is a single input flip-flop. This holding signal is used to retain the real bit read during $\overline{B}$ time. The second delay flip-flop D2 must read alternately I1 and D1 during H = 1 time:

$$JD2 = \overline{H}\cdot D1 + H\left[\overline{I1}\cdot B + D1\cdot\overline{B}\right]$$
$$= \overline{H}\cdot D1 + H\cdot I1\cdot B + D1\cdot\overline{B}$$

These equations represent the basic functional operations of the sequence inverter. However, there remain many subsidiary considerations.

One consideration is the effect of shifting the reading heads to obtain the multiplication by $\frac{1}{2}\delta$. This shift means that during portions of each word time it is necessary to block the output from I1. This prevents the shifted least significant part of the next word from being added into the most significant part of the present word. The timing signal T4 is used to indicate this blocking period. T4 also performs the control task of indicating recirculation from D2 to D1.

The problem arises of how to multiply negative numbers by shifting the reading heads. With the algebraic sign shifted also, it is necessary to insert additional 1's in the deleted bit positions. Thus to multiply $-\frac{1}{4}$ by 2^{-4} one finds that the shifting operation must perform the following steps:

$$-\frac{1}{4} \equiv 1.11000000$$

Shifting this by four binary places gives:

$$0.00011100$$

which is $+\frac{7}{64}$. The proper answer is:

$$(\tfrac{1}{4})(2^{-4}) \equiv 1.11111100$$

$$\equiv -\frac{1}{64}$$

Shifting positive numbers is permissible since 0's are normally entered in the deleted positions. The simplest technique to obtain the correctly shifted numbers is to enter the sign digit in each deleted bit position. This is done in the computer by catching the two sign bits in D1 and D2 during the last bit time preceding the blocking period T4. The sign bits are then circulated in D1 and D2 for the remainder of the word time.

$$JD1 = \overline{T4}\,\big/\overline{H}\cdot I1 + I1\cdot\overline{B} + H\cdot D1\cdot B\big\_/ + T4\cdot D2$$

$$JD2 = \overline{T4}\cdot H\cdot I1\cdot B + D1\,\big/T4 + \overline{B} + \overline{H}\big\_/$$

These equations would suffice if it were not necessary to subtract as well as add. A simple way of subtracting is to add the 0's complement of the subtrahend. In this computer an even simpler technique is possible. The 1's complement of the subtrahend may be added directly. This is due

to the shift of the subtrahend. The 0's complement is obtained from the 1's complement by adding a carry 1 into the least significant position. This least significant position is shifted by $\log_2 \delta$ bits to the right by the reading head shift. Hence the error in using the 1's complement instead of the 0's complement is negligible in comparison to truncation and round off errors.

Whenever a subtraction is indicated by G and H, $\overline{I1}$ is read by D1 and D2 instead of I1. Since the real and the imaginary components are handled differently, the control signals for addition and subtraction must be derived in terms of B and $\overline{B}$. For the input to D1:

$$\overline{B}\left[G\,H + \overline{G}\,\overline{H}\right] + B\,\overline{G} = 1 \qquad \text{-------- Subtraction}$$

$$\overline{B}\left[G\,\overline{H} + \overline{G}\,H\right] + B\,G = 1 \qquad \text{-------- Addition} \quad .$$

Inserting these control signals in the inputs to D1 gives:

$$JD1 = D2\ T4 + H\ B\ \overline{T4}\ D1 + \overline{T4}(\overline{H} + \overline{B})(I1\ G + \overline{I1}\ \overline{G}) \quad .$$

The control signals for use in the inputs to D2 are the same as those for D1 except that the roles of B and $\overline{B}$ are exchanged.

$$JD2 = D1(\overline{H} + \overline{B} + T4) + H\ B\ \overline{T4}(I1\ \overline{G} + \overline{I1}\ G)$$

To make the addition of Figure 8 a little more elegant, round off should be included to eliminate the propagation of error which occurs upon repetitive addition of (-0) to (-0). (-0) is represented by a word completely filled with 1's. Mathematically correct roundoff is obtained for both positive and negative numbers by presetting the carry flip-flop to the digit of the addend which just precedes the least significant digit of the augend. This should be done for both the real and the imaginary components. The term addend refers to the shifted number, and the term augend refers to the unshifted number.

In order to obtain the two correct round off bits and still retain the sign bits for the direction decision, one of several alternatives is available. One of the simpler techniques is to eliminate I1 and reuse the same flip-flop as D3. D3 would be a simple delay flip-flop reading D2. With this method, the sum logic from M3 and M6 is used directly on the inputs to D1 and D2. This necessitates a T4 signal that starts one bit sooner than the T4 now used. The revised T4 must also stop one bit later than the T4 shown in Figure 5. M7 reads D3 instead of D2 during normal computations. The correct round off bits can then be read into C4 during the normal T3 time:

$$JC4 = M1\ C3\ \overline{T3} + D2(M1 + C3 + T3)$$

By retaining the general functions of D1 and D2 and by advancing these operations one bit time, the direction decision equations are modified only by the new arrival times of the sign bits, and by the exchange of D3 and D2 for the D2 and D1 terms now written in the inputs to G and H.

Although roundoff was not included in the computer itself, it is felt that this is the first improvement to be made in the existing machine.

There are two timing sequences for T4 depending upon which value of δ is being used. Consider the bit times of each word to be numbered starting with t1 to denote the least significant bit.

With H1 reading the least significant bit during t1, H3 is reading the (10 +2) bit of the real component - if there were no timing delays inserted between D2 and H3. These numbers are for $\delta = 2^{-10}$ and the factor of $\frac{1}{2}$ from Equation (3). Since I1, D1, and D2 each insert a delay of one bit time, H3 must be shifted to read three bit times sooner. As each word is biplexed, H3 reads the 23'rd binary position if no timing

delays are inserted in the arithmetic unit. With a three bit delay, H3 is shifted to read the 26'th binary digit of the biplexed word when H1 is reading the least significant bit.

Since there are 62 bits of significance per word, T4 would be set to 1 after the sign bits had been read (if there were no other delays to be considered). T4 would be set immediately after t(1 + 62 - 26) = t(37). However, three bit times are required for the proper sign bits to appear in D1 and D2. Thus T4 is set to 1 after t(40).

From the general timing diagram of Figure 5, a $\overline{P}$ pulse appears during t(40). H1 and H3 are separated by (64 + 1 - 26) = 39 bit positions. For $\delta = 2^{-20}$, H2 must be placed 20 bit positions ahead of H3. Thus H1 and H2 are separated by 19 bit positions. This configuration requires that T4 is set to 1 after t(20) for $\delta = 2^{-20}$; hence $\overline{P}$ pulses appear during t(20), T(40), t(60), t(62), t(64).

At the end of each word time, T4 is reset to 0 after t(60). This allows the round off digits to enter D1 and D2 by the blank digit time T3.

While H1 is reading the least significant bit, M7 is writing the first blank bit between the words. Since the writing amplifier inserts one additional bit delay, H7 is set to write the last sign bit. Hence the following head spacings:

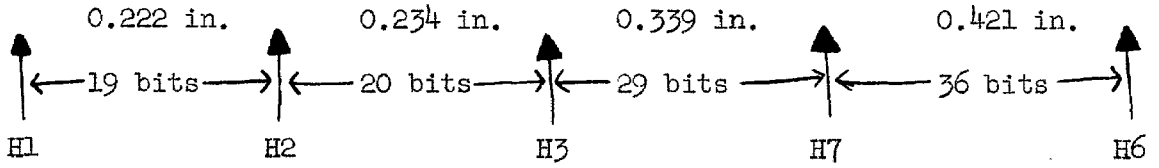


FIGURE 10

With a five inch diameter drum and 1344 pulses, each bit requires $\frac{5(3.14)}{1344} = 0.0117$ inches. Thus the head spacings at the drum surface are as shown above in Figure 10.

One remaining operation must be considered during the computational phase. When $\phi 1 = 1$, the contents of M1 circulate to M7. The blank bits occurring between words are also transferred directly from M1 to M7. These operations are described by modifying the inputs to M7:

$$\begin{aligned} JM7 &= (S1 + S3) \overline{\lfloor} (\phi 1 + T3) + (M1 \cdot D2 \cdot C3 + M1 \cdot \overline{D2} \cdot \overline{C3}) + \\ &\quad + \overline{\phi 1} \cdot \overline{T3}(\overline{M1} \cdot D2 \cdot \overline{C3} + \overline{M1} \cdot \overline{D2} \cdot C3) + (\phi 1 + T3)\, M1 \overline{\rfloor} \\ &= (S1 + S3) \overline{\lfloor} M1(D2 \cdot C3 + \overline{D2} \cdot \overline{C3} + T3 + \phi 1) + \\ &\quad + \overline{M1} \cdot \overline{\phi 1} \cdot \overline{T3}(D2 \cdot \overline{C3} + \overline{D2} \cdot C3) \overline{\rfloor} \end{aligned}$$

When switching the mode switches S1 or S3 to the input mode S2, it is necessary to insure that all information is circulated unchanged from M1 to M7. During S1' or S3' this circulation is automatic, but during S1' S2 = 1 or S3' S2 = 1, C3 and C4 must be set to 0. With C4 initially 0 and with C2 and C3 initially 1, one drum revolution could elapse before C4 is set to 1 and C2 and C3 set to 0. In one additional word time C4 is reset to 0. Hence about 16 milliseconds are required for

S1' or S3' and S2 to remain in mutual contact.

C. Direction Decision

This is the most complicated logical operation in the computer. Table II of section B-1 indicates the exact decision required. However, to implement this precise decision would require more equipment than is actually desired. To obtain a better picture of the information available to make the direction decision, it is advisable to consider the form in which the actual sign bits are available. This form is a function of the states of G and H during the preceding computation cycle.

Equations (5) and Table II may be used to restate the arithmetic operations performed under the control signals from G and H.

G	H	Arithmetic Operation Desired
0	0	$u(z) - \delta u'(z)$
0	0	$v(z) - \delta v'(z)$
1	0	$u(z) + \delta u'(z)$
1	0	$v(z) + \delta v'(z)$
0	1	$u(z) + \delta u'(z)$
0	1	$v(z) - \delta v'(z)$
1	1	$u(z) - \delta v'(z)$
1	1	$v(z) + \delta u'(z)$

(20)

Thus the state of G and H at the end of a computation cycle indicates the time sequence of the components present in the flip-flops D1 and D2.

TABLE III

Control flip-flops:			The signs of the modified components held during TO in the flip-flops:		
G	H	B	M1	D1	D2
0	0	0	u(z)	-v'(z)	-u'(z)
0	0	1	v(z)	-u'(z)	-v'(z)
1	0	0	u(z)	+v'(z)	+u'(z)
1	0	0	v(z)	+u'(z)	+v'(z)
0	1	0	u(z)	-u'(z)	+v'(z)
0	1	1	v(z)	+v'(z)	-u'(z)
1	1	0	u(z)	+u'(z)	-v'(z)
1	1	1	v(z)	-v'(z)	+u'(z)

The timing signal TO is derived from the phase control signal identifying the word holding the value of f(z).

$$TO = T5 \cdot \phi 2 \cdot \overline{\phi 3}$$

During the sign bit times of this word, the three flip-flops M1, D1, D2 contain the modified signs shown in Table III. B is the biplexing timing signal identifying real or imaginary components in M1.

TABLE IV
Direction Decision

Actual Sign of the Components:				Case 1 Initial Values of: G = 0 H = 0 Signs held in flip-flops designated during time TO:						Change State of:		Initial Values of G = 1 H = 0 Signs held during TO:						Change		Initial Values of G = D H = 1 Signs held during TO:						Change:		Initial Values of G = 1 H = 1 Signs held during TO:						Change:	
				$\overline{B}$			B					$\overline{B}$			B					$\overline{B}$			B					$\overline{B}$			B				
u	u'	v	v'	M1	D2	D1	M1	D2	D1	G	H	M1	D2	D1	M1	D2	D1	G	H	M1	D2	D1	M1	D2	D1	G	H	M1	D2	D1	M1	D2	D1	G	H
+	+	+	+	+	−	−	+	−	−	0	0	+	+	+	+	+	+	1	0^2	+	+	−	+	−	+	0^*	1	+	−b	+	+	+	−	1^*	1
+	+	−	−	+	−	+	−	+	−	0	0	+	+	−	−	−	+	1	0^2	+	−	−	−	−	−	0^*	1	+	+	+	−	+	+	1^*	1
−	−	+	+	−	+	−	+	+	+	0	0	−	−	+	+	+	−	1	0^2	−	+	+	+	+	+	0^*	1	−	−	−	+	−	−	1^*	1
−	−	−	−	−	+	+	−	+	+	0	0	−	−	−	−	−	−	1	0^2	−	−	+	−	+	−	0^*	1	−	+	−	−	−	+	1^*	1
+	−	+	−	+	+	+	+	+	+	1	0^2	+	−	−	+	−	−	0	0	+	−	+	+	+	−	1^*	1	+	+	−	+	−	+	0^*	1
+	−	−	+	+	+	−	−	−	+	1	0^2	+	−	+	−	+	−	0	0	+	+	+	−	+	+	1^*	1	+	−	−	−	−	−	0^*	1
−	+	+	−	−	−	+	+	+	−	1	0^2	−	+	−	+	−	+	0	0	−	−	−	+	−	−	1^*	1	−	+	+	+	+	+	0^*	1
−	+	−	+	−	−	−	−	−	−	1	0^2	−	+	+	−	+	+	0	0	−	+	−	−	−	+	1^*	1	−	−	+	−	+	−	0^*	1
+	+	+	−	+	−	+	+	−	+	0	1	+	+	−	+	−	+	1	1	+	−	−	+	−	−	0	0	+	+	+	+	+	+	1	0^2
+	−	−	−	+	+	+	−	+	+	0	1	+	−	−	−	−	−	1	1	+	−	+	−	+	−	0	0	+	+	−	−	−	+	1	0^2
−	+	+	+	−	−	−	+	−	−	0	1	−	+	+	+	+	+	1	1	−	+	−	+	−	+	0	0	−	−	+	+	+	−	1	0^2
−	−	−	+	−	+	−	−	+	−	0	1	−	−	−	−	−	−	1	1	−	−	+	−	+	−	0	0	−	−	−	−	−	−	1	0^2
+	+	−	+	+	−	−	−	−	−	1	1	+	+	−	−	−	−	1	0^2	+	−	+	−	+	−	1	0^2	+	−	+	−	+	−	0	0
+	−	+	+	+	+	−	+	−	+	1	1	+	−	−	+	−	+	1	0^2	+	+	−	+	−	−	1	0^2	+	−	−	+	−	−	0	0
−	+	−	−	−	−	+	−	+	−	1	1	−	+	−	−	−	+	1	0^2	−	−	+	−	+	+	1	0^2	−	+	+	−	+	+	0	0
−	−	+	−	−	+	+	+	+	+	1	1	−	−	−	+	−	−	0	1	−	−	+	+	+	−	1	0^2	−	+	−	+	−	+	0	0
				1			2			3		4						5		6						7		8						9	

The truth table of Table IV is used to derive the actual control signals needed to actuate G and H. Column 1 holds the actual signs of the real and imaginary components; these are all the possible sign cases which could arise during any computation cycle. Columns 2, 4, 6, 8 list the modified signs of each component. These signs are determined from Column 1 and Table III. Columns 3, 5, 7, 9 indicate the number of times to change the state of G or H. This information is derived from Table II which shows the final state required for both G and H as a function of the sign conditions in Column 1 of Table IV.

Note should be taken of several items. Two successive changes of state in a flip-flop result in the same final state as zero changes. The superscript $\underline{2}$ is used in Table IV to denote those cases where a flip-flop changes state during both the time intervals B = 0 and B = 1.

From Table II it is possible to obtain a set of general rules governing the changes in G and H.

1) Any odd number of sign changes among u, u', v, v' requires that the state of H be changed.

2) If H does not change, then G must change if:

   a) Initial H = 0 and the signs of u and u' change with respect to each other. This is true also for v and v' with respect to each other.

   b) Initial H = 1 and the signs of u and v' change with respect to each other. This is true for v and u' also.

3) If h does change, G must change if:

   a) Signs of the old u' and new v' are similar when the sign of u is unchanged.

   b) Signs of old u' and new v' are dissimilar when the sign of u changes.

   c) (a) and (b) are true for initial H = 0. For the case of H = 1, the roles of u' and v' are exchanged.

These rules are rather awkward. Hence it is simpler to derive the input equations for G and H from Table IV.

One set of signals sufficient to change H would be:

$$(M1 \cdot D2 + \overline{M1} \cdot \overline{D2})\ TO$$

or the signals:

$$(M1 \cdot \overline{D2} + \overline{M1} \cdot D2)\ TO$$

In either case, both of these signals operate during the two bit times: $(TO \cdot \overline{B} + TO \cdot B) = TO$. H may be changed zero, one, or two times during this interval. The numbers indicated below H in Table IV denote the number of state changes that the signals above provide during any TO period.

The signals required to change G are more complicated. From the general rules above, it appears impossible to obtain the final correct value for G without resorting to the use of another flip-flop for memory purposes. If such a flip-flop were available, it could be used to store the initial value of H, and the signals above would be used to modify the state of the flip-flop H. The desired setting for G could then be

found by changing the state of G whenever the following signal is 1:

$$(M1 \cdot D2 + \overline{M1} \cdot \overline{D2}) \text{ TO } \_\overline{/}(D1 \cdot \overline{D2} + \overline{D1} \cdot D2)\ \overline{B} + (D1 \cdot D2 + \overline{D1} \cdot \overline{D2})\ B\_/\ \overline{Q3} +$$
$$+ \_\overline{/}(D1 \cdot D2 + \overline{D1} \cdot \overline{D2})\ \overline{B} + (D1 \cdot \overline{D2} + \overline{D1} \cdot D2)\ B\_/\ Q3$$

An extra flip-flop Q3 would be used here to state the initial value of H.

A compromise decision can be made. Q3 may be eliminated and a considerable saving in diodes effected by allowing G and H to indicate a wrong decision in certain cases. These cases are marked with an asterisk in Table IV. This decision logic applied to the G inputs causes G to change state whenever the following signal is 1:

$$(M1 \cdot D2 + \overline{M1} \cdot \overline{D2}) \text{ TO } \_\overline{/}(D1 \cdot \overline{D2} + \overline{D1} \cdot D2)\ \overline{B} + (D1 \cdot D2 + \overline{D1} \cdot \overline{D2})\ B\_/$$

The decision error introduced by using this equation forces the computer to choose the opposite value of Δ from that required. Columns 7 and 9 show the decision error occurring when the computer is moving up or down in the imaginary direction in the complex plane. In both of these cases it is necessary to change to motion to the left or right in the real direction of the complex plane. This is accomplished by changing H to the 0 state. However, in the starred cases, G is set to the wrong value by the equation above. If the computer is moving up in the imaginary direction and if the sign conditions indicate motion to the right in the real direction, the computer actually takes one step to the left in the real direction. This opposite decision occurs for the other three cases indicated by an asterisk. However, after making one step in the wrong direction, the correct decisions of Columns 2 and 4 select the true value

of Δ , and the computer continues in the proper direction.

Summarizing, the input equations to the direction decision flip-flops may be written as follows:

$$JH = KH = (D2\ M1 + \overline{D2}\ \overline{M1})\ T0$$

$$JG = KG = (D2\ M1 + D2\ M1)\ T0 \left[(D1\ \overline{D2} + \overline{D1}\ D2)\overline{B} + (D1\ D2 + \overline{D1}\ \overline{D2})B \right]$$

D. Equation Summary

The logical design of the computer is summarized below by presenting the complete Boolean equations for each flip-flop along with brief comments on the flip-flop functions.

M7 - Writing Flip-flop.

$$JM7 = S8\left[C1\ G\ T3 + M1\ S4\right] + (S1' + S3')\left[\phi 1\ M1\right] + S2\left[M1\ \overline{C4}(\overline{C3} + \overline{T3} + \overline{B}) + C2\ C4\right] + (S1 + S3)\left[M1(D2\ C3 + \overline{D2}\ \overline{C3} + T3 + \phi 1) + \overline{M1}\ \overline{\phi 1}\ \overline{T3}(D2\ \overline{C3} + \overline{D2}\ C3)\right]$$

C1 - First Carry Flip-flop. This flip-flop:

a) Acts as a one bit time delay during computation.

b) Indicates when a new input bit has been inserted.

$$JC1 = S8\ G\ S4 + S2\ \overline{G}(S5 + S6) + (S1 + S3)C2$$

$$KC1 = S8\ \overline{S4} + S2\ C4 + (S1 + S3)\overline{C2}$$

C2 - Second carry flip-flop. This flip-flop:

a) Accepts the new input bits.

b) Writes the precessed marker m2.

c) Forms the carry input to the first adder during computation.

$$JC2 = S2\lfloor \overline{C4}\ S5 + C2\ \overline{C4}(H + \overline{S6}) + M1\ C4 + M1\ C3\ T3\ B \rfloor + (S1 + S3)\lfloor M3\ M6 + M3\ C1 + M6\ C1 \rfloor$$

C3 - Third Carry Flip-flop. This flip-flop:

a) Indicates when it is desired to precess the movable marker m2.

b) Presents the carry output to the second adder during computation.

$$JC3 = S2\ \overline{H}\ S6^{*} + (S1' + S3')\overline{\emptyset 1} + (S1 + S3)C4$$

$$KC3 = S2\ C4 + (S1 + S3)\overline{C4}$$

C4 - Fourth Carry Flip-Flop. This flip-flop:

a) Controls the circulation of M1 to M7 via C2 for insertion of the desired input bit.

b) Writes the precessed marker bit m2.

c) Generates roundoff during computation (if this feature is included).

d) Forms carry input to the second adder during computation.

$$JC4 = S2\lfloor M1\ C1\ T3\ B + C2\ C3\ T3\ \overline{B} + \overline{C3}\ C4(\overline{T5} + \overline{B}) \rfloor + (S1 + S3)\lfloor D2(M1 + C3) + M1\ C3 \rfloor$$

D1 - First Delay Flip-flop. This flip-flop is used only during computation. It acts to:

a) Invert the biplexed sequence of real and imaginary bits.

b) Generate the 1's complements for subtractions.

$$JD1 = D2\ T4 + D1\ B\ H\ \overline{T4} + \overline{T4}(\overline{H} + \overline{B})(I1\ G + \overline{I1}\ \overline{G})$$

D2 - Second Flip-Flop in the Sequence Inverter. It has the same general functions as D1.

$JD2 = D1(\overline{H} + \overline{B} + T4) + H\,\overline{T4}\,B\,I1\,\overline{G} + H\overline{T4}\,B\,\overline{I1}\,G$

I1 - Output Flip-Flop from the First Adder.

$JI1 = M3\,M6\,C1 + M3\,\overline{M6}\,\overline{C1} + \overline{M3}\,M6\,\overline{C1} + \overline{M3}\,\overline{M6}\,C1$

G - Direction Control. This flip-flop indicates whether Δ is plus or minus. During input, it is used to write the marker bits m1 and m2.

$JG = S4\,S8\,T3\,\overline{B}(H + \overline{C1}) + S2\,C1(S5 + S6) +$

$+ (S1 + S3)\underline{/}D1\,\overline{D2}\,\overline{B} + \overline{D1}\,D2\,\overline{B} + \overline{D1}\,\overline{D2}\,B + D1\,D2\,\underline{B}\overline{/}\underline{JH}\overline{/}$

$KG = S8\,T3\,\overline{B} + S2\,\overline{S5}\,S6^{*}$

H - Direction Control. This flip-flop indicates whether Δ is real or imaginary. During input it is used to write the marker bits m1 and m2.

$JH = S8\,\overline{\emptyset 1}\,T3\,\overline{B}\,G + S2\,C3\,S6^{*} + (S1 + S3)(D2\,M1 + \overline{D2}\,\overline{M1})TO$

$KH = S8\,T5\,B\,G + S2\,\overline{S6} + (S1 + S3)(\,JH\,)$

Q1, Q2, Q3 - General Timing. These outputs are grouped as follows:

$T3 = Q2\,Q3$

$T4 = Q1$

$T5 = \overline{Q1}\,\overline{Q2}\,\overline{Q3}$.

The inputs to these flip-flops are specified below.

$JQ1 = \overline{P}\,\overline{Q2}\,Q3\,S1 + \overline{P}\,Q2\,\overline{Q3} + B\,P\,\overline{Q2}\,\overline{Q3}$

$KQ1 = \overline{P}\,\overline{Q2}$

$JQ2 = \overline{P}\,\overline{Q1}$

$KQ2 = \overline{P}\,\overline{Q1} + \overline{P}\,\overline{Q3}$

$JQ3 = \overline{P}\ \overline{Q1}\ Q2$

$KQ3 = \overline{P}\ \overline{Q2} + B\ P\ Q2$

$\emptyset 1$ - Circulation Control. During computation the memory circulates directly from M1 to M7 with $\emptyset 1 = 1$.

$J\emptyset 1 = [\emptyset 2\ T3\ B]$

$K\emptyset 1 = \overline{\emptyset 2}\ \emptyset 3\ T3\ B\ \overline{S8} + S4(\overline{S1' + S2 + S3'}) + S8\ \overline{S4}$

$\emptyset 2$, $\emptyset 3$ - Phase Control. These signals are used to identify the words holding $f(z)$, Δ , z , $f^n(z)$.

$J\emptyset 2 = T3\ B\ M7$

$K\emptyset 2 = T3\ B\ \overline{M7}$

$J\emptyset 3 = T3\ B\ \emptyset 2$

$K\emptyset 3 = T3\ B\ \emptyset 1\ \overline{M7}$

B - Biplex Control. This flip-flop indicates the real or imaginary bits as they are read from M1.

$JB = C_p\ \overline{P}$

$KB = C_p$

The following timing signals are formed separately:

$T3\ B$

$T3\ \overline{B}$

The various switches used have the following significance:

S1 - Computation with fine delta: $\delta = 2^{-20}$.

S1' - Intermediate switch position separating S1 and S2.

S2 - Input mode.

S3' - Intermediate switch position separating S2 and S3.

S3 - Computation with coarse delta: $\delta = 2^{-10}$.

S4 - Channel erase. This switch is ordinarily set to 1 after the computer warm-up period.

S5 - Input bit. This switch inserts a 1 into the computer when depressed during the input mode.

S6 - Input bit. This switch inserts a 0 into the computer when depressed during the input mode.

S7 - Synchronization Selector for the output.

S8 - Marker Insert. The computer is started with S8 = 1 and S4 = 0. Setting S4 = 1 initiates marker insertion; setting S8 = 0 allows the computer to enter the input mode.

S9 - Output Selector.

S10 - Direction Control. Depressing this switch forces an imaginary delta selection; raising this switch forces a real delta selection. In the middle position, the switch allows the computer to make its own direction decision.

S11 - Direction Control. Depressing this switch sets G = 0, and raising this switch sets G = 1. In conjunction with S10 this switch determines the direction in which the operator can force the computer to move.

IV. CIRCUIT DESIGN

The equipment and engineering techniques used to implement the logical design of the computer will be described in this section. The major items used in constructing an operating computer are the flip-flops, the diode matrix, and the magnetic drum circuitry. The actual circuit details of the computer are presented in Appendix 3.

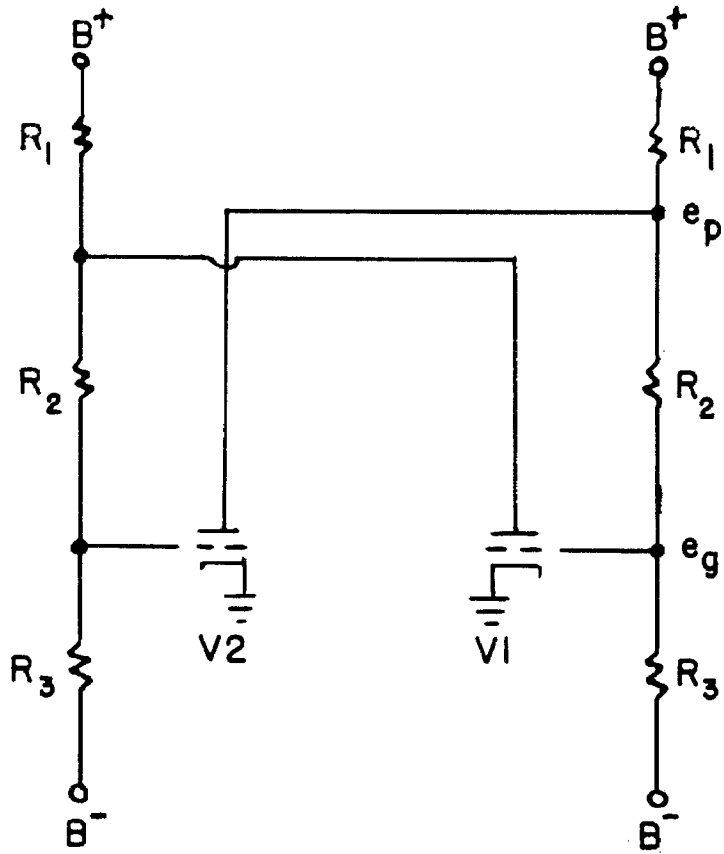
A. Flip-Flop Design

The circuit used to form the basic active component of the computer is a bistable device often called an Eccles Jordan, a counter, a toggle, or a flip-flop. Its essential mathematical properties are described in Part IV, Table I. To obtain the performance corresponding to that truth table, two triodes are connected as shown in Figure 11. This is a d-c coupled circuit which can be designed to have the two states of stable operation:

1) V1 conducting and V2 cut off.

2) V2 conducting and V1 cut off.

A circuit of this type is useful for several reasons. First, it can be designed to have moderate amounts of power amplification. Second, it will remain in either stable state once it has been set to that configuration. This behavior is usually referred to as memory. Third, since this circuit has d-c coupling between the two triodes, the flip-flop can reverse or change states at any rate below the limiting frequency. The limiting frequency is determined by the capacitance effects in the associated circuitry and tubes. This low frequency



Basic d.c. Coupled Flip Flop Circuit

Figure 11

characteristic is valuable in checking out and maintaining any computing circuit.

This section is included in order to emphasize the importance of the design of a reliable flip-flop circuit. Most of the ideas relating to the d-c stability of the flip-flop itself were developed by M. Phister[19]. The derivations of that discussion are summarized below. The design of the flip-flop circuit used for the Root Extractor utilizes those results.

In Figure 11, the voltage notation is defined as follows:

$e_g \equiv$ Grid voltage with respect to ground potential.

$e_p \equiv$ Plate voltage with respect to ground potential.

$e_{g1} \equiv$ Grid voltage of V1 when it is conducting.

$e_{g1}' \equiv$ Grid voltage of V1 when it is cut off.

$e_{p2} \equiv$ Plate voltage of V2 when it is conducting.

$e_{p2}' \equiv$ Plate voltage of V2 when it is cut off.

The design procedure is simply that of insuring that:

a) $e_{g1} \geq 0$

b) $e_{g1}' \leq E_c$

for all possible variations in resistor, tube, and power supply variations. E_c is the voltage below which the tube is always cut off under any possible operating conditions. Phister's thesis considers three cases; the Ideal, the Practical, and the Optimum Flip-Flop. The derivation for the Ideal case is summarized below. The results of the Practical case are tabulated, and the Optimum flip-flop for the computer is determined experimentally.

1. Ideal Flip-Flop:

Two equations can be written determining the two voltages e_{p2} and e_{g1}'. These involve the plate current i_{p2} drawn by V2 when it is conducting.

$$i_{p2} = \frac{B^+ - e_{p2}}{R_1} - \frac{e_{p2} - e_{g1}'}{R_2} \tag{21}$$

$$\frac{e_{p2} - e_{g1}'}{R_2} = \frac{e_{g1}' - B^-}{R_3} \tag{22}$$

Solving for e_{g1}' from (22):

$$e_{g1}' = \frac{e_{p2}R_3 + B^- R_2}{R_2 + R_3}$$

The first condition on the flip-flop is that this voltage be less than E_c:

$$\frac{e_{p2} R_3 + B^- R_2}{R_2 + R_3} \leq E_c \tag{23}$$

This insures that V1 is always cut off when V2 is conducting.

To guarantee that V1 is always conducting when V2 is cut off, the grid voltage e_{g1} can be set greater than 0.

$$e_{g1} = \frac{B^+R_3 + B^-(R_1 + R_2)}{R_1 + R_2 + R_3}$$

The second condition on the flip-flop is:

$$\frac{B^+R_3 + B^-(R_1 + R_2)}{R_1 + R_2 + R_3} \geq 0 \tag{24}$$

Solving for R_3:

$$R_3 \geq \frac{-B^-}{B+} \ (R_1 + R_2) \tag{25}$$

Solving for e_{g1}' from (21) and (22) in terms of i_{p2} gives:

$$e_{g1}' = \frac{B^+R_3 + B^-(R_1 + R_3) - i_{p2} R_1 R_3}{R_1 + R_2 + R_3} \tag{26}$$

Since $e_{g1}' \leq E_c$ for cut off, (26) can be written using (24):

$$i_{p2} R_1 R_3 \geq -E_c(R_1 + R_2 + R_3)$$

From (25) is derived:

$$i_{p2} R_1 \geq -E_c\left(1 - \frac{B^+}{B-}\right) \tag{27}$$

Solving (23) for the ratio R_2/R_3 yields:

$$\frac{R_2}{R_3} \geq \frac{e_{p2} - E_c}{E_c - B} \tag{28}$$

Substituting (25) into this result gives the ratio R_1/R_2:

$$\frac{R_1}{R_2} \leq \frac{-B^+}{B-} \left[\frac{E_c - B^-}{e_{p2} - E_c}\right] - 1 \tag{29}$$

From (27), (28), (29), one can compute the three resistors of the voltage divider for each tube of the flip-flop. Both tubes and both resistor dividers are similar.

An i_p and e_p are selected from the tube characteristics. This choice is limited by the power dissipation of the tube and the tolerance given to possible emission losses from the published tube characteristics.

It is usually desirable to have a flip-flop that switches between states as rapidly as possible. This implies a low plate impedance and a high transconductance. Generally speaking, it is advisable to have i_p and e_p as large as possible. The limits on e_p can be computed from (29) since the resistor ratio must be positive:

$$0 \leq \frac{-B^+}{B^-} \left\{ \frac{E_c - B^-}{e_p - E_c} \right\} - 1 \qquad (30)$$

$$e_p < B^+ + E_c\left(1 - \frac{B^+}{B^-}\right)$$

This voltage is a function only of the supply voltages and the tube cut off voltage. E_c is generally chosen as the cut off voltage corresponding to the supply voltage B^+ applied directly to the triode plate. The resistors specified by the inequalities (27), (28), and (29) are chosen from the standard values available commercially.

2. Practical and Optimum Flip-Flops:

In designing a circuit to operate reliably, tolerances are usually given to all the components. The tube characteristics are derated in terms of specified power dissipation and emission properties. The resistors are given a tolerance $\pm$ p%, and the supply voltages are allowed changes of $\pm$ q%. The combination of these tolerances which gives the worst circuit conditions is used to compute the required component values.

In that case when the grid voltage e_{g1} is supposed to be greater than 0, the combination of tolerances which would cause the lowest grid

voltage is:

a) The positive supply voltage B^+ is lowest:

$$B^+(1 - q)$$

b) The resistor R_3 is smallest: $R_3(1 - p)$

c) The negative supply voltage is highest:

$$B^-(1 + q)$$

d) The resistors R_1 and R_2 are largest: $R_1(1 + p)$, $R_2(1 + p)$

The equation for this case, which corresponds to equation (24) of the ideal flip-flop, is derived by the same general techniques.

$$\frac{B^+ R_3(1 - p)(1 - q) + B^-(R_1 + R_2)(1 + p)(1 + q)}{(R_1 + R_2)(1 + p) + R_3(1 - p)} \geq 0 \qquad (31)$$

For the cut off case, the grid voltage e_{g1}' is the highest when:

a) B^+ is highest: $B^+(1 + q)$

b) R_3 is largest: $R_3(1 + p)$

c) B^- is lowest: $B^-(1 - q)$

d) R_1 and R_2 are smallest: $R_1(1 - p)$, $R_2(1 - p)$

For $e_{g1}' \leq E_c$, these conditions give:

$$\frac{\left[B^+(1+q) - i_{p2}R_1(1-p)\right] R_3(1+p) + B^-(R_1+R_2)(1-p)(1-q)}{(R_1+R_2)(1-p) + R_3(1+p)} \leq E_c \qquad (32)$$

$$\frac{e_{p2} R_3(1+p) + B^- R_2(1-p)(1-q)}{R_2(1 - p) + R_3(1 +p)} \leq E_c \qquad (33)$$

From (31) one can compute the resistor ratio:

$$\frac{R_1 + R_2}{R_3} \leq \frac{-B^+}{B^-} \left\{\frac{(1-p)(1-q)}{(1+p)(1+q)}\right\} \tag{34}$$

From (32) the product $i_{p2}R_1$ is found:

$$i_{p2}R_1 \geq \frac{1}{1-p}\left\{B^+(1+q) - E_c - \frac{R_1 + R_2}{R_3}\ \frac{1-p}{1+p}\ \lfloor B^- E_c - B^-(1-q) \rfloor\right\} \tag{35}$$

Substituting (34) into (35) gives:

$$i_{p2}R_1 \geq \left(\frac{-E_c}{1-p}\right)\left\{1 - \frac{B^+(1-p)^2(1-q)}{B^-(1+p)^2(1+q)}\right\} + B^+\left\{\frac{4(p+q)(1+pq)}{(1+p)^2(1-p)(1+q)}\right\} \tag{36}$$

Solving (33) and (34) for the resistor ratio R_2/R_3 gives the result:

$$\frac{-\left(\frac{B^+}{B^-}\right)\left\{\frac{(1-p)(1-q)}{(1+p)(1+q)}\right\}}{1 + \frac{R_1}{R_2}} \geq \frac{R_2}{R_3} \geq \frac{(e_{p2} - E_c)(1+p)}{\lfloor B^- E_c - B^-(1-q) \rfloor (1-p)} \tag{37}$$

which yields:

$$1 + \frac{R_1}{R_2} \leq \frac{B^+(1-p)^2(1-q)\,E_c - B^-(1-q)}{B^-(1+p)^2(1+q)(E_c - e_{p2})} \tag{38}$$

A constant k is chosen to make (38) an equality. That is, the designer selects R_1 from (36) and R_2 from (38) basing the choice on the standard resistor values obtainable. This selection fixes k:

$$k = \left\{\frac{B^+(1-p)^2(1-q)\ \lfloor B^- E_c - B^-(1-q) \rfloor}{B^-(1+p)^2(1+q)\ \lfloor B^- E_c - e_{p2} \rfloor}\right\} \frac{R_2}{R_1 + R_2} \tag{39}$$

The limits on the ratio R_2/R_3 are determined from (37):

$$\frac{k(e_{p2} - E_c)(1 + p)}{E_c - B^-(1 - q)(1 - p)} \geq \frac{R_2}{R_3} \geq \frac{(e_{p2} - E_c)(1 + p)}{E_c - B^-(1-q)(1-p)} \qquad (40)$$

The selection of the plate voltage e_{p2} is based on (38) since $R_1/R_2 \geq 0$:

$$e_{p2} < \frac{B^+(1 - p)^2(1 - q)}{B^-(1 + p)^2(1 + q)} \left[-E_c - B^-(1 - q) \right] + E_c \qquad (41)$$

These formulae are applied to the design of a flip-flop using an RCA double triode type 5963. Several other types of tubes were studied, and many dynamic tests were run on the flip-flops designed for each tube. The 5963 was chosen because it gave a fast circuit and because it is a computer tube designed to retain its emission capabilities after long periods of cut off operation.

The supply voltage tolerances used are $\pm$ 10%, while the resistor tolerances are $\pm$ 5%. The cut off voltage E_c was chosen as -6 volts corresponding to a plate supply voltage of +100 volts.

The first step in the design procedure is determining the upper limit on the plate voltage. This maximum for e_{p2} is computed from (41) using the tolerances above.

$$e_{p2} < \frac{100(0.95)^2(0.9)}{100(1.05)^2(1.1)} \left[-6 + 100(0.9) \right] - 6$$

$$\therefore e_{p2} < 50.3 \text{ volts}$$

Equation (36) is used to compute the value for the product $i_{p2}R_1$:

$$i_{p2}R_1 \geq \frac{6}{0.95}\left[1 - \frac{100(0.95)^2(0.9)}{100(1.05)^2(1.1)}\right] + 100\left[\frac{4(.15)(1.005)}{(1.05)^2(.95)(1.1)}\right]$$

$\therefore\ i_{p2}R_1 \geq 63$ volts.

Derating the emission properties of the tube is one technique used to accommodate variations in the tube parameters. For commercial circuit design, most tubes are derated to 50% of their nominal performance. However, since the 5963 is a tube selected for computer applications where reliability and uniformity are at a premium, it is felt that derating the tube to 80% of its nominal emission is adequate for a laboratory machine. Figure 12 is a plot of the plate characteristics for the type 5963 double triode. The thirty tubes tested and used in the computer equalled or exceeded these nominal ratings when new.

The next step in the design procedure is selecting a working plate voltage and plate current. This selection is made from the actual tube characteristics; for practical circuits it is still an art to pick optimum values. Many circuits are designed and evaluated before the optimum is found. The object of the design is to obtain a flip-flop that will be set to the correct state when the input voltage is a small amount ϵ above or below the midpoint value of the signal swing. The trigger circuits used for the flip-flop are discussed in the next section. However, it was found that fairly large values for R_2 and R_3 were desirable using those triggering techniques. This is evident when it is recognized that the impedance seen at the grid determines the current necessary to trigger the tube.

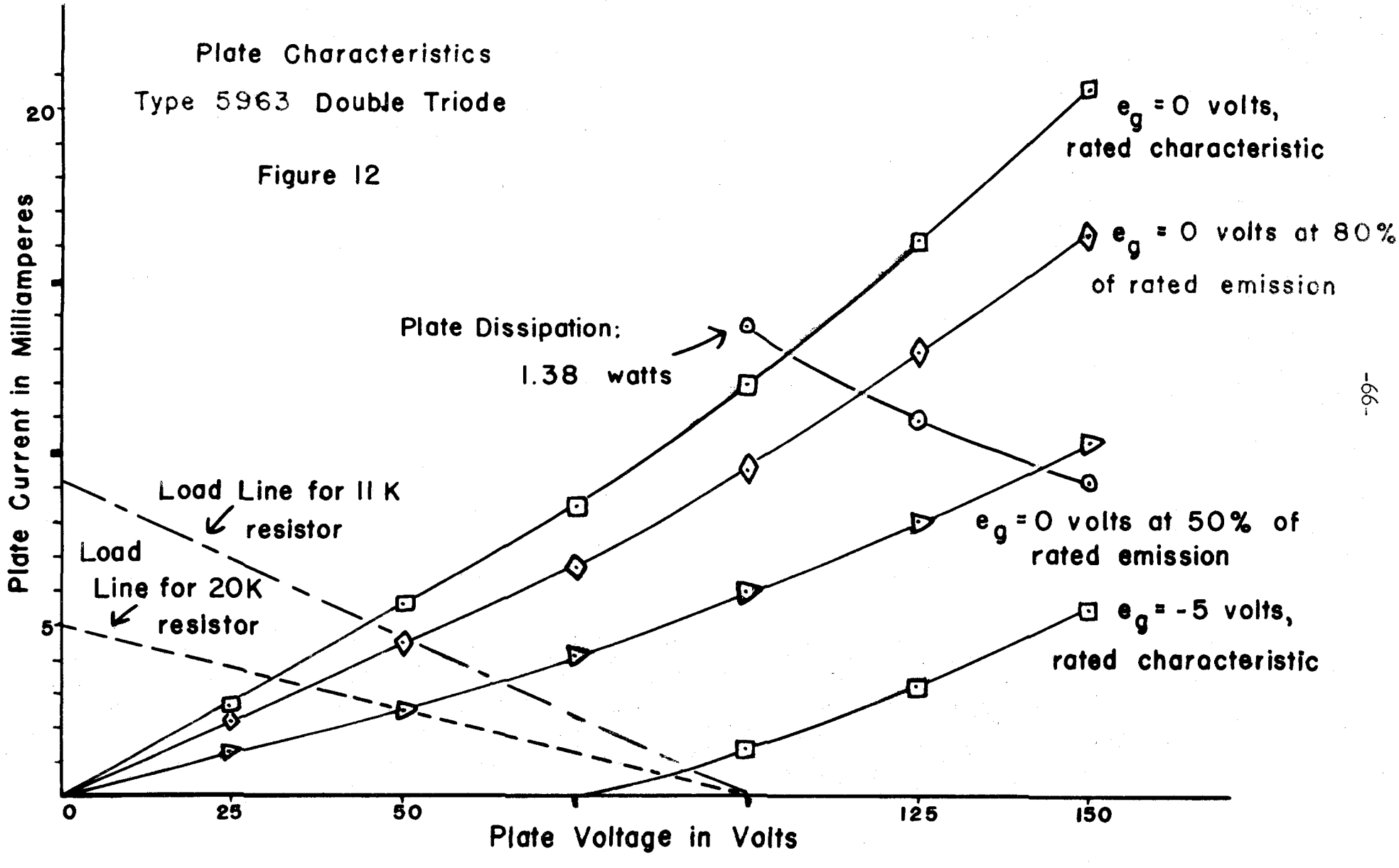


Plate Characteristics
Type 5963 Double Triode
Figure 12
Plate Current in Milliamperes
20
5
0
25
50
125
150
Plate Voltage in Volts
Plate Dissipation:
1.38 watts
Load Line for 11 K resistor
Load Line for 20K resistor
e_g = 0 volts, rated characteristic
e_g = 0 volts at 80% of rated emission
e_g = 0 volts at 50% of rated emission
e_g = -5 volts, rated characteristic

Another general feature of the designs based on Phister's derivations is that one cannot select the precise voltage given by (41) for the operating plate voltage. This would require $\frac{R_1}{R_2} = 0$. Furthermore, the closer one selects the operating plate voltage to the limiting value, the smaller the selection range for $\frac{R_2}{R_3}$ as given by (40). Using standard resistor values, it is necessary to have an R_2/R_3 ratio satisfying (40).

Within these general limitations, it is quite easy to select an operating plate current and voltage leading to standard resistor values. With a 20 kilohm plate resistor, R_1, a plate current of 3.2 milliamperes gives a plate voltage of 37 volts on the derated zero bias characteristic of the type 5963 tube. This value is found to be a satisfactory operating point in terms of tube tolerances, resistor values, and waveform rise times. This value for R_1 also allows the tube to degenerate to one-half its nominal emission properties before losing its two stable states. This is advantageous since the tube is the component most likely to change its characteristics with age.

The grid dropping resistor R_2 is determined from the inequality (38). Using the tolerances specified previously, the resistor ratio R_1/R_2 is computed as:

$$\frac{R_1}{R_2} \leq 0.31$$

or

$$R_2 \geq 3.2\ R_1 = 64 \text{ kilohms} \qquad (42)$$

The resistor ratio R_2/R_3 is found from (40).

$$0.56\ K \geq \frac{R_2}{R_3} \geq 0.56 \tag{43}$$

The resistor values chosen for the computer are tabulated below.

Resistor	Value	Tolerance
R_1	20,000 ohms	5%
R_2	220,000 ohms	5%
R_3	330,000 ohms	5%

This provides a k = 0.67 satisfying (43).

These resistors used with the single input trigger circuit of the next section provide a very stable flip-flop. Two speed-up capacitors are used shunting R_2; their function is to hasten the transition from one state of the flip-flop to the other. Twenty micromicrofarad capacitors are used to provide equally fast rise and fall times for the plate voltages of about 0.7 microseconds. In most of the flip-flops built and tested the plate voltage swings from +92 to +30 volts approximately. This causes the grids to swing from 0 to -25 volts in general.

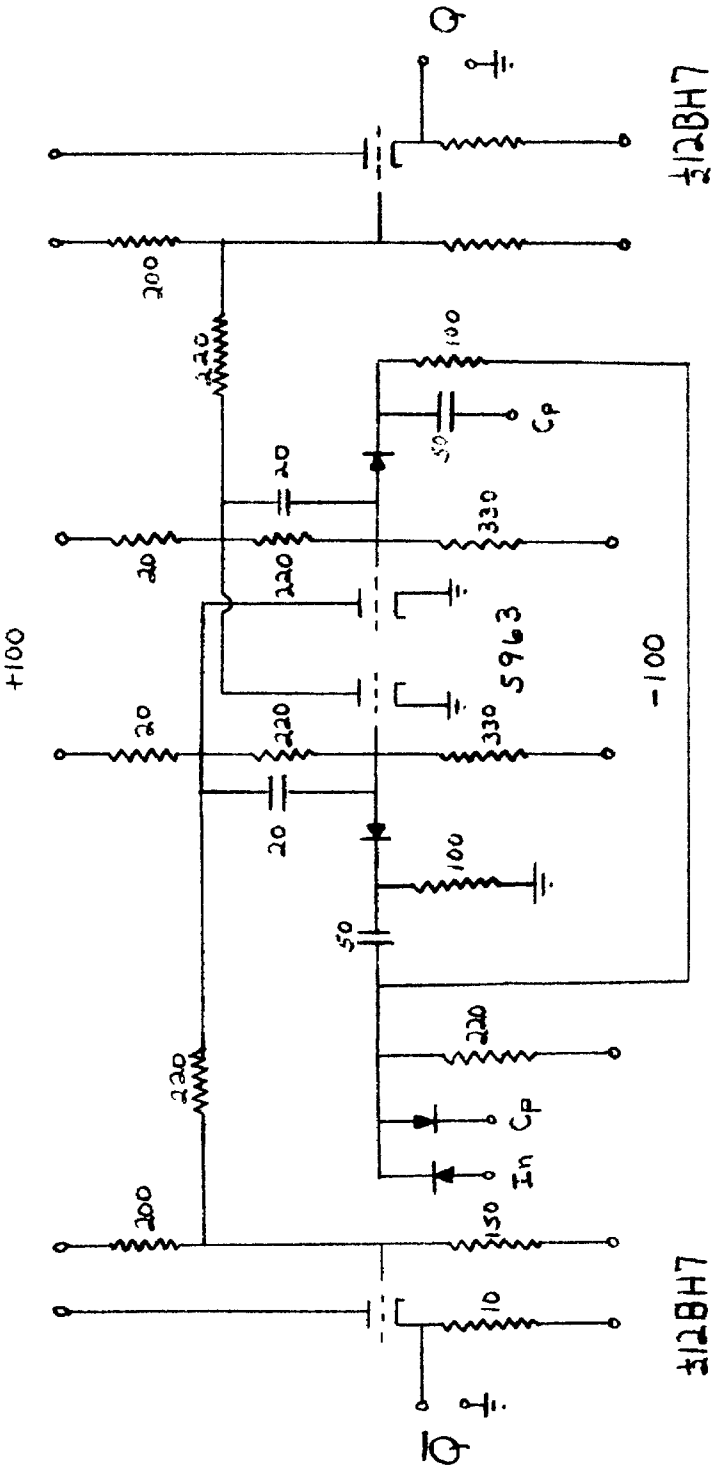
One of the difficulties encountered in designing circuits involving tolerances is selecting the ideal component values and considering all possible causes for failure. The technique described does not consider several factors. Some of these omissions are the triggering method, the dynamic and nonlinear change of state, and the factors affecting reliability over long periods of time. For example, the method used designing the resistor dividers to provide reasonable tolerances in the

resistors, in the supply voltages, and in the tube current emission capabilities accomplishes this result by demanding fairly large plate voltage swings. The usual result of a large plate swing is a large grid swing. The difficulty with driving the grid far below cut off is that the tube then is hard to trigger back to conduction. One device used to remedy this effect is to clamp the grid to a certain cut off voltage. That is, a diode is connected to the grid to prevent the grid from swinging below a certain value. This costs extra diodes per flip-flop. Since the resistor divider is designed to hold the grid below the clamp voltage, current flows from the resistor divider through the clamp diode. The triggering action in the flip-flop must supply this current before the circuit can change state.

Most of the undetermined tolerances are factors affecting the state changing process. A very thorough treatment of these general problems for transistor flip-flops appears in Scarborough[20]. The optimum design for any circuit should weight all tolerance assignments with reliable data concerning their probable occurrence and influence in affecting the circuit performance. The ideal design could be considered represented by a n dimensional surface, the interior of which designates satisfactory operation of the circuit. The operating point would be located within this surface in such a way that the variation in each component contributes an equal probability to the circuit failure.

3. Output Circuit:

A cathode follower output stage is used to provide isolation and gain for the basic flip-flop. The circuit for the entire



Basic Flip Flop

Figure 13

flip-flop is shown in Figure 13. The output from the cathode follower swings from +3 to +18 volts. The single input flip-flop is reliably set by any signal above or below +10.5 volts. The reasons for using a cathode follower output are outlined below.

a) The extra stage of current gain provided by the cathode follower allows each flip-flop to supply current to a large number of diode gates.

b) The isolation provided by the cathode follower between the flip-flop and the diode gate load means that loading effects will not affect the logical operation of the flip-flop. The resistor divider driving the cathode follower grid is an excellent buffer isolating the grid current effects that are generated if the cathode follower approaches zero bias.

c) The use of clamping diodes is eliminated with the cathode follower and resistor divider. With the large plate swing available at the flip-flop, the resistor divider attenuates the voltage swing applied to the cathode follower grids. The cathode follower output is made considerably less dependent on the flip-flop plate swing with this circuit. In general it is found that the cathode follower is a cheaper item to construct than clamp diodes with their associated power supplies.

d) High and low power output stages can be substituted for each other by exchanging the tubes. However, for

this laboratory model of the computer two different sets of cathode resistors are used for the two different output stages. For the supply voltages used in this computer this is a more effective utilization of the tube capabilities.

The two tube types used for the cathode follower stages are the 5687 and 12BH7 double triodes. The supply voltages of $\pm$100 volts were chosen to enable construction of a simple power supply having a very low output impedance; but the power supply is unregulated. The new diffused junction germanium power rectifiers were used for these power supplies. Each power supply has an output impedance of 10 ohms at one ampere of rectified current, with a ripple of one volt, peak to peak. The heaviest current drain appears from the cathode followers which do not reflect the ripple voltage on their outputs. A maximum load of 500 milliamperes is placed on the power supplies in the heaviest operating condition.

The 5687 cathode follower is designed with a 5,000 ohm cathode resistor. For the 15 volt output swing desired at the cathode, this tube will deliver the following performance.

Plate Current through Cathode Follower	Cathode Follower Grid Voltage	Cathode Voltage	Current Load
20 ma.	+ $\frac{1}{2}$ volts	+3 volts	0 ma.
23 ma.	+17 volts	+18 volts	0 ma.
5 ma.	+ $\frac{1}{2}$ volts	+ $5\frac{1}{2}$ volts	-15 ma.
28 ma.	+17 volts	+$17\frac{1}{2}$ volts	+5 ma.

This is a conservative utilization of a powerful tube. The maximum plate dissipation is one-half the rated value. The current loads are given as positive or negative. Negative current is used to designate the current absorbed by the cathode follower from an and gate when the signal is low. When the signal is high the cathode follower supplies positive output current to or gates. The gating techniques are described in the next section. Maximum current loads of +4 milliamperes and -13 milliamperes can exist on one or two flip-flops in the computer.

The 12BH7 cathode follower is designed with a 10,000 ohm cathode resistor to supply smaller current loads. The performance characteristics for this output circuit are summarized below.

Plate Current	Grid Voltage	Cathode Voltage	Current Load
10 ma.	$+\frac{1}{2}$ volts	+ 3 volts	0 ma.
12 ma.	+17 volts	+18 volts	0 ma.
2 ma.	$+\frac{1}{2}$ volts	$+5\frac{1}{2}$ volts	-8 ma.
14 ma.	+17 volts	$+17\frac{1}{2}$ volts	+2 ma.

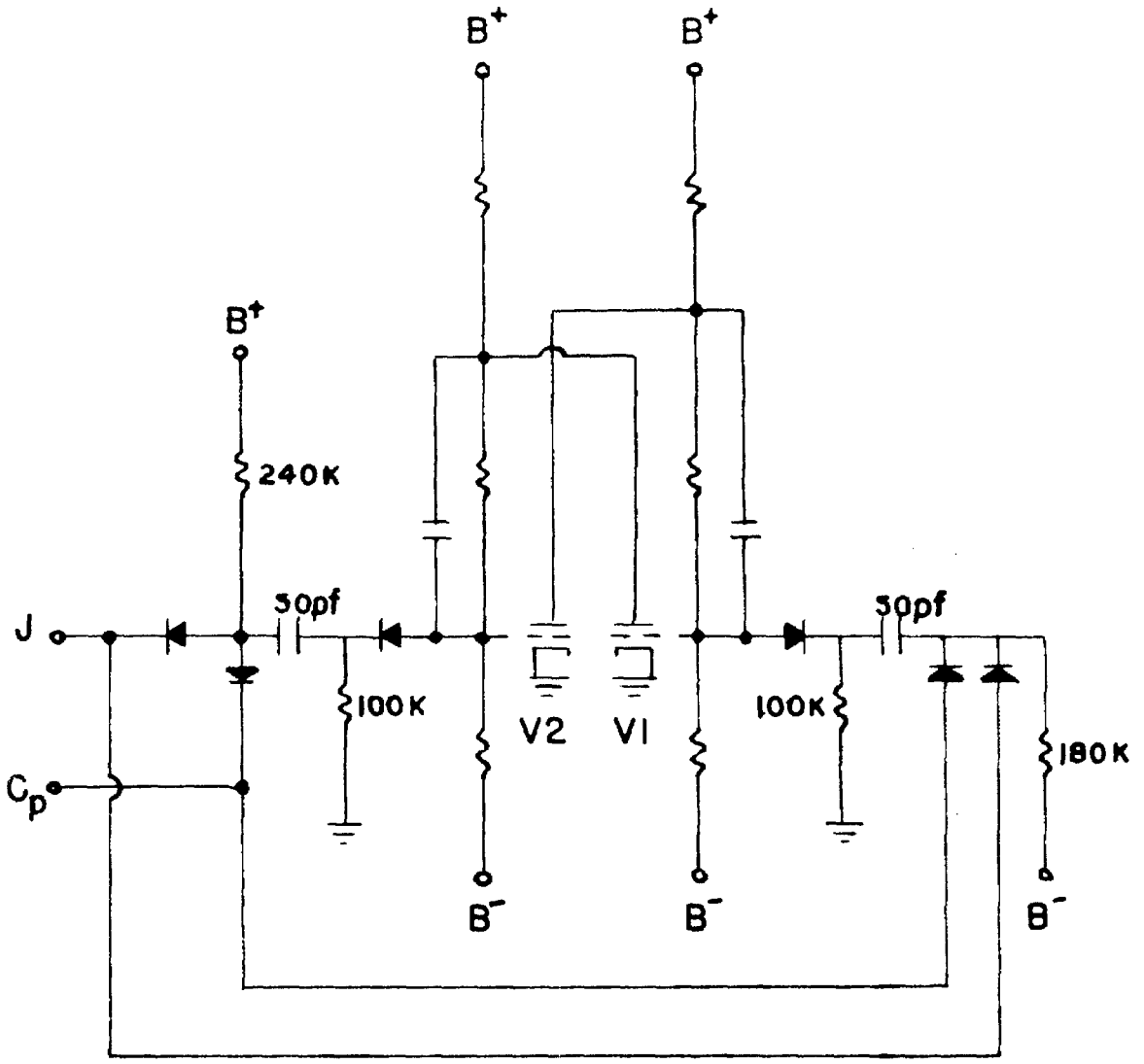
With this conservative use of the tube, the maximum plate dissipation is less than one-third the rated value. Maximum cathode current required in the worst case is 70% of the rated value.

More efficient use could be made of these cathode followers if the cathode swing had been set between 0 and -15 volts. However, setting the swing above 0 was convenient in designing the single input trigger circuit. Pulse pedestal gating is used on one grid; Nelson gating[21] used on the other grid provides a simple flip-flop, the

output of which follows the input in value but is delayed in time by one clock pulse period.

Subsequent to the construction of all the operating circuitry a new input gate was developed which has many practical advantages for the single input type flip-flop. This circuit is explained briefly, and it is presented in Figure 14. This flip-flop requires only the single input J. The trigger circuits on both grids are a-c coupled. The principle of operation of the trigger circuit on V2 is explained in detail in the next section. This circuit shown on V2 in Figure 14 is one of two trigger circuits used in the computer. In general, the input signal J charges the capacitor. The negative clock pulse is referenced to the high signal voltage, and it disconnects the input if J is high, thereby pumping a negative pulse into the grid of V2 and shutting it off. With J low, no pulse is fed through the input capacitor.

The trigger circuit on V1 in Figure 14 is unique in that exactly the opposite effect occurs. This circuit is not used in the computer because it was developed after all the equipment had been built. The circuit is mentioned because of its many advantages. In this case the clock pulse is again negative, and it is referenced to the high voltage state of the input J. Hence the clock pulse source holds the input gate high between clock pulses. The application of a negative clock pulse disconnects the clock source from the gate. A high input signal J holds the gate up and no pulse is fed into the grid of V1. With J low, however, a current pulse is fed into the grid of V1 driving it



Modified Single Input Flip Flop

Figure 14

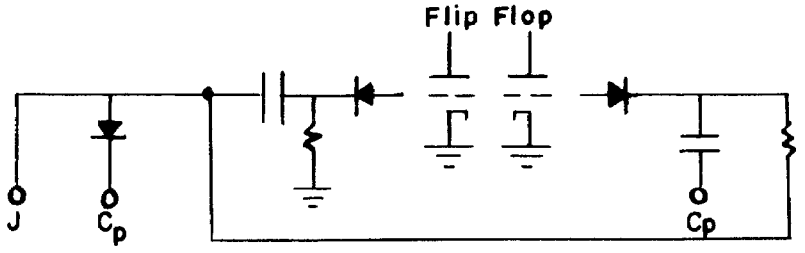
negative. This circuit has three advantages over the circuit actually used. First it is a-c coupled which eliminates much of the difficulty experienced with pulse pedestal gates; this factor alone would permit operating the cathode followers and the flip-flops at the optimum d-c voltage levels. Second, the clock pulse is applied to both grids through diodes. This allows symmetrical current pulses and provides a much more reliable circuit. One additional advantage of the circuit driving V1 is that all transient effects within the network are blocked out by the holding action of the clock pulse diode during all time intervals between clock pulses. This is a significant advantage where signal-to-noise problems exist. If time permitted, the flip-flops would all be converted to this type of trigger circuit. To make this possible, it is necessary to rederive most of the logical equations of the computer.

B. Diode Matrix Design

1. Trigger Circuits:

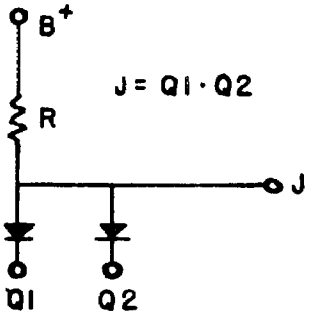
The basic flip-flop in the computer requires only one input. It is necessary to provide current from the diode matrix to this input; the current must be adequate to support a 15 volt swing in approximately ten microseconds. The gating circuit used for the one input flip-flop is drawn in Figure 15. The input J may be supplied either from an and gate or an or gate. Each case will be considered separately.

For the and gate input, J is generated as shown in Figure 16a; an or gate input is shown in Figure 16b. The flip-flop output swing is from +3 to +18 volts. Hence the input swing is the same voltage less the drop across the diodes. For practical purposes the



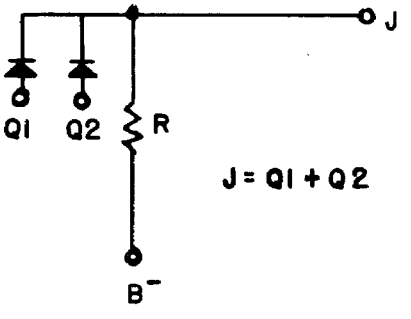
Trigger Circuits for
Basic Single Input Flip Flop

Figure 15



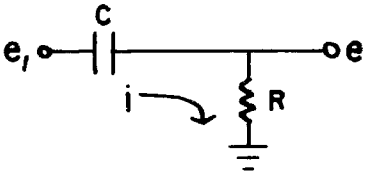
And Gate Input Circuit

Figure 16a



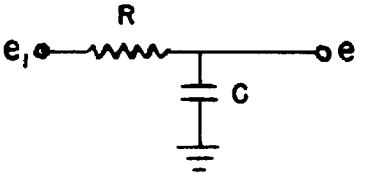
Or Gate Input Circuit

Figure 16b



Trigger Circuit

Figure 17a



Complementing Trigger Circuit

Figure 17b

forward drop across these gating diodes may be neglected.

In general, the flip-flops driving the gate must supply the charging current for the input capacitor. The time required to charge the input capacitor determines the charging current. If it is desired to charge the capacitor in Figure 17a to 13 volts in ten microseconds, the RC time constant must be 5×10^{-6} sec. as shown above. This charging time is readily computed:

$$e_1 - e = \frac{i}{CS}$$

$$i = \frac{e1}{R + \frac{1}{CS}} = \frac{E_1}{S(R + \frac{1}{CS})} \qquad \text{if } e_1 = \frac{E_1}{S}$$

$$\therefore (e_1 - e) = \frac{E_1}{RCS(S + \frac{1}{RC}}$$

$$(e_1 - e) = E_1(1 - e^{-t/RC})$$

$$\frac{13}{15} = 1 - e^{-t/RC}$$

$$e^{-t/RC} = 0.135 \qquad \text{or } \frac{t}{RC} = 2$$

$$RC = \frac{t}{2} = \frac{10^{-5}}{2} = 5 \times 10^{-6} \text{ sec.}$$

The component values selected for the computer are:

$$R = 100{,}000 \text{ ohms}$$

$$C = 50 \text{ uuf}$$

The charging current is simply $\frac{E_1}{R}$. Hence:

$$1 = \frac{15}{10^5} = 0.150 \text{ milliamperes.}$$

This same current is required to charge the capacitor in Figure 17b.

When the input diodes are back biased, the gating resistors R in Figure 16 must supply the charging current. If a single input flip-flop is used, as in Figure 15, 0.3 milliamperes are required to charge both capacitors. This determines the size of R in Figure 16 since the supply voltage is fixed at 100 volts:

$$1 = \frac{E}{R}$$

$$3 \times 10^{-4} = \frac{100}{R}$$

$$R = 330{,}000 \text{ ohms}$$

If a two input flip-flop is used, the gating resistor can be twice this size.

In order to determine the exact value for the input resistor, the voltage supply tolerances, the resistor tolerances, and the input voltage swing must be considered.

In the case of the and gate input, the supply voltage may drop to 95 volts, the resistor has a 5% tolerance, and the input voltage swing is from +18 to +3 volts. The input resistor R is calculated when the input signal is at +18 volts and when the resistor has the largest value within its tolerance. Five percent resistors are used throughout the gating matrices. Hence the value for the and gate input resistor is

computed as follows:

$$0.3 = \frac{95 - 18}{(1 + 0.05)R_1}$$

$$\therefore R_1 = \frac{77}{1.05(0.3)} = 240 \text{ kilohms}$$

For a two input flip-flop, 0.15 milliamperes are required. The input gate resistor is found:

$$0.15 = \frac{95 - 18}{(1 + 0.05)R_2}$$

$$R_2 = 470 \text{ kilohms.}$$

The conditions for an or gate input are somewhat different from those of the and gate. The voltage supply may be -95 volts, but the input swings down to +3 volts. Hence for the single input flip-flop:

$$0.3 = \frac{95 + 3}{1.05\ R_1}$$

$$R_1 = 300 \text{ kilohms.}$$

The two input flip-flop uses:

$$R_2 = 620 \text{ kilohms.}$$

These resistor sizes are specified in terms of the RMA standards.

Table V summarizes the resistor sizes and current requirements on the gating flip-flops as a function of the input gate.

TABLE V

Input Gate	Flip-Flop Input	Resistor	Current Requirements on the Driving Flip-Flops
And	Single	240 K	0.3 + 0.42 + 0.05(i-1)ma.
Or	Single	300 K	0.3 + 0.4 + 0.05(i-1)ma.
And	Two	470 K	0.15 + 0.16 + 0.05(i-1)ma.
Or	Two	620 K	0.15 + 0.16 + 0.05(i-1)ma.

The currents specified are the sum of the charging current required, the d-c current drawn through the gating resistor, and the back currents drawn through the back biased gating diodes. The number i denotes the number of gating diodes on the input gate.

The large tolerances included above are assumed to encompass the effects of capacitance and the variations in the input capacitors or resistors.

2. Resistor Computation:

Since either an and gate or an or gate can be used with the trigger circuit, two sets of computations are required to determine the resistor sizes in the remaining gates. These derivations will be given in terms of an aribtrary current requirement at the trigger circuit.

a) And gate input

The gate of Figure 16a may be driven from a second diode gate. Ordinarily the second gate is an or gate similar to

Figure 16b. The output of this or gate is connected to the input of the and gate as shown in Figure 18. There may be i diodes connected to the first level gate and j diodes connected to the second level gate. $Q2_1$ is shown as being supplied by the second level gate.

The most unfavorable load on the second level gate is when e_1 is held at +3 volts by $Q2_1$ and all other diodes are back biased. All the second level inputs must be at +3 volts for this condition to exist. R_2 must then absorb all the current indicated in rows 1 or 3 of Table V. It is computed as follows:

$$1.05\ R_2 = \frac{95 + 3}{I_1}$$

$$R_2 = \frac{93.3}{I_1}$$

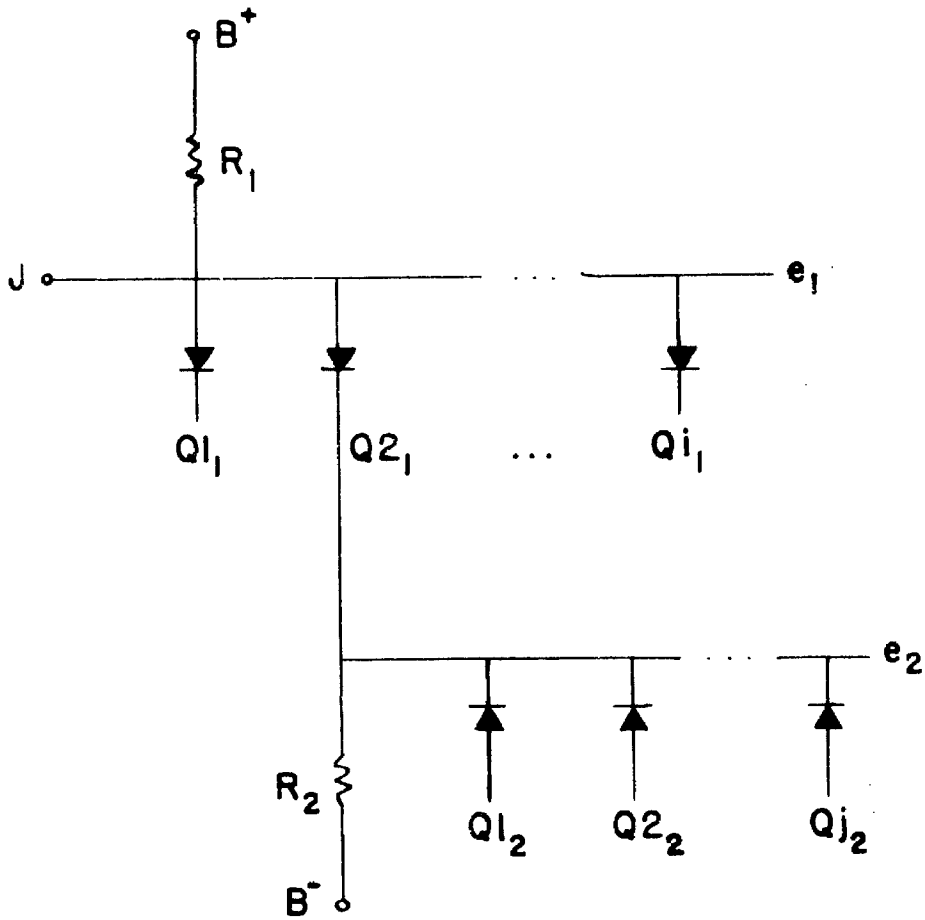
The most unfavorable load for a flip-flop Q_{j2} is when it is at +18 volts and all other second level inputs are at +3 volts.

$$I_{2j} = \frac{0.95\ B^- + 18}{0.95\ R_2} + 0.05(j - 1)$$

$$= \frac{119}{R_2} + 0.05(j - 1)$$

The gating resistor for the third level is returned to B^+. Here the worst condition for the gating resistor is when all third level inputs are at +18 volts; the resistor must then supply I_{2j} ;

$$1.05R_3 = \frac{95 - 18}{I_2}$$



Cascaded Gating Circuit:

Or Gate Output Supplying And Gate Input.

$$J = Q1_1 \cdot \left[Q1_2 + Q2_2 + \cdots + Qj_2 \right] \cdot Q3_1 \cdot \cdots \cdot Qi_1$$

Figure 18

$$R_3 = \frac{73}{I_{2j}}$$

$$I_{3K} = \frac{95 - 3}{0.95\ R_3} + 0.05(K - 1)$$

$$I_{3K} = \frac{97}{R_3} + 0.05(K - 1)$$

The fourth level gate is not generally used, but the requirements are:

$$R_4 = \frac{93}{I_{3K}}$$

$$I_4 = \frac{119}{R_4} + 0.05(\quad - 1)$$

b) Or gate input

The gating matrix is developed in the same fashion as Figure 18 except that the first level is an or gate. The worst load for the gating resistor of the second level exists when all inputs to the second level are at +18 volts. The second level resistors and current are found as follows:

$$1.05R_2 = \frac{95 - 18}{I_1}$$

$$R_2 = \frac{73}{I_1}$$

$$I_{2j} = \frac{95 - 3}{0.95R_2} + 0.05(j - 1)$$

$$= \frac{97}{R_3} + 0.05(j - 1)$$

The computations here correspond to those of part 1.

Tabulating the results:

$$R_3 = \frac{93}{I_2}$$

$$I_{3K} = \frac{119}{R_3} + 0.05(K - 1)$$

$$R_4 = \frac{73}{I_{3K}}$$

$$I_4 = \frac{97}{R_3} + 0.05(\quad - 1)$$

Table VI summarizes these results.

TABLE VI

First Level Gate	Resistor	Second Level Gate Current
And And	$R_2 = \frac{93}{I_1}$	$I_2 = \frac{119}{R_2} + 0.05(j - 1)$
Or	$R_2 = \frac{73}{I_1}$	$I_2 = \frac{97}{R_2} + 0.05(j-1)$

- - - - - - -

Third Level Gate		Fourth Level Gate	
Resistor	Current	Resistor	Current
$R_3 = \frac{73}{I_2}$	$I_3 = \frac{97}{R_3} + 0.05(K-1)$	$R_4 = \frac{93}{I_3}$	$I_4 = \frac{119}{R_3} + 0.05(\quad -1)$
$R_3 = \frac{93}{I_2}$	$I_3 = \frac{119}{R_3} + 0.05(K-1)$	$R_4 = \frac{73}{I_3}$	$I_4 = \frac{97}{R_4} + 0.05(\quad -1)$

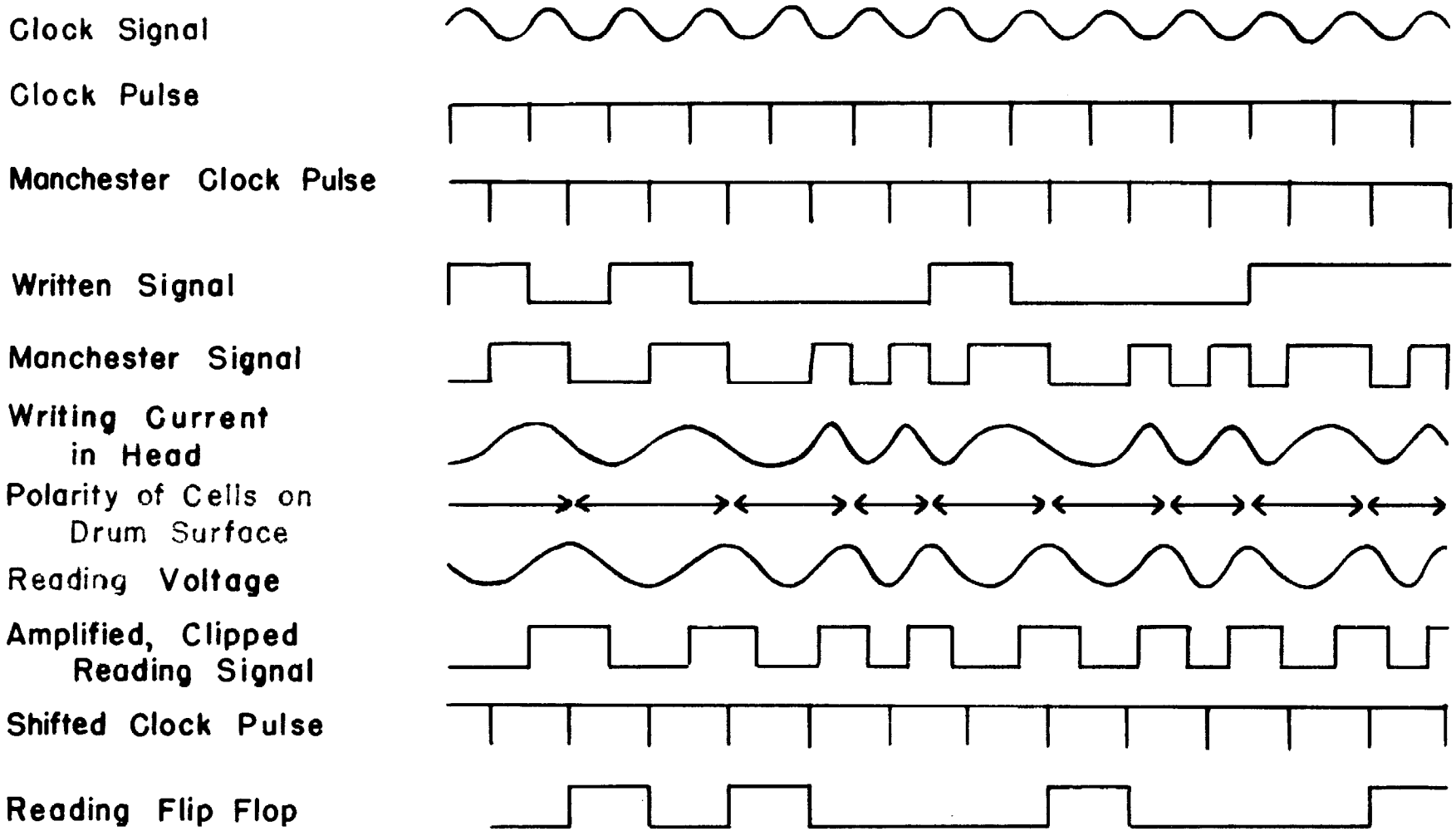
C. Memory Techniques

The memory provided for the computer is a rotating magnetic drum. One circulating register and one timing track are placed on the surface of the five inch diameter drum. The motor driving the drum is a hysteresis synchronous motor, rotating at 3600 rpm. With 1344 pulses spaced evenly around the drum circumference, the computer operating speed is 80,640 pulses per second.

The clock pulse track is used to generate pulses for synchronizing all of the computer operations. The clock pulse generator is a blocking oscillator triggered by the leading edge of the amplified signal read from the clock pulse channel. A second clock pulse is derived from the falling edge of the amplified signal read by the clock channel. This is accomplished by inverting the triggering signal and applying it to another blocking oscillator.

The recording system used is sometimes referred to as Manchester recording because the technique was originally developed for the drums used at Manchester University.(22) The second clock pulse described above is used in this recording process. In many ways the Manchester process resembles conventional Return-to-Zero recording. The major difference is in the way that the signal is obtained. Figure 19 illustrates the general principles involved in the drum writing and reading processes. It is interesting to observe that the signal held by the reading flip-flop is delayed $1\frac{1}{2}$ bit times with respect to the signal to be written. In a circulating register this is taken care of by advancing the reading head around the drum circumference. The signal

as read and amplified from the drum is used directly in diode gating circuits, if the flip-flops set by the diode gates use the trigger circuit shown on tube V1 of Figure 18. The output from the Manchester writing flip-flop is also used in diode gates under a similar restriction.



Magnetic Drum Reading and Recording Techniques Figure 19

V. OPERATIONAL INSTRUCTIONS

The polynomial is prepared for insertion in the computer by the general techniques of section III-c. The radius which includes all roots of the polynomial is computed by (6); the new variable w is found by (7); and the coefficients of the transformed polynomial are computed from (9). The n derivatives of the polynomial are computed at w = 0 and converted to binary numbers. These numbers are the quantities to be entered into the computer.

The control panel for the computer is shown in Figure 20. The steps taken in placing the computer in operation are given in numbered sequence below. The initial switch configuration is depicted in the figure.

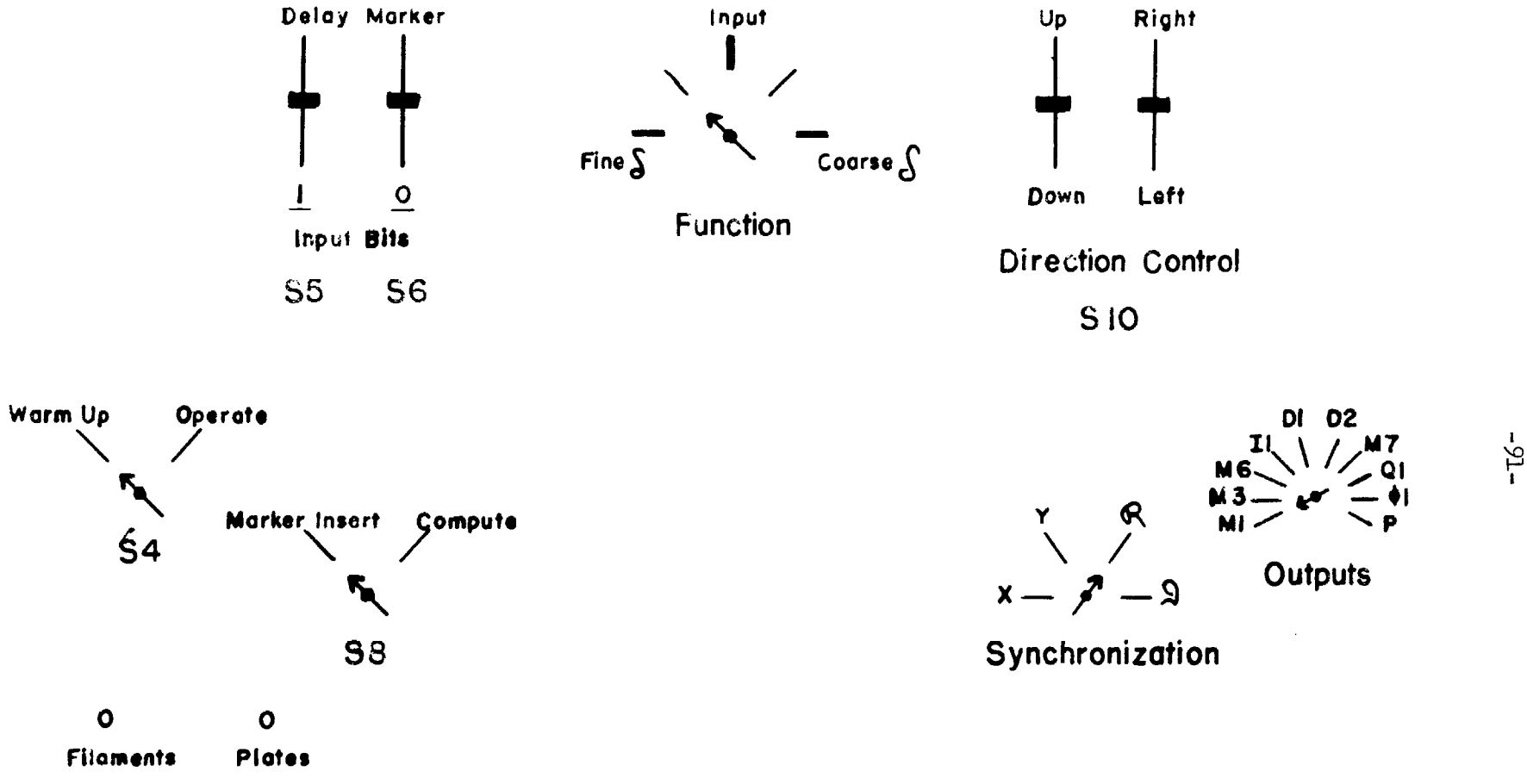
1) Set all switches to the initial positions shown in Figure 20.
2) Filaments are turned on and allowed to warm up for one minute.
3) Plate power supply is turned on.
4) Switch S4 is set to Operate.
5) Switch S8 is set to Compute.
6) Function switch is set to Input.
7) Input bits are inserted via S5 and S6. Depressing S5 places a 1 in the computer; depressing S6 places a 0 in the computer word position marked by the movable marker m2. Marker m2 is automatically set to identify the word position holding Δ. Only three digits are entered into this first word. These are a 0 placed in both sign positions and a 1 placed in the most

significant real bit position. Input is always bit by bit, most significant bit first. The real and imaginary bits are entered sequentially with the real bit preceding the corresponding imaginary bit.

8) Marker is delayed by raising either S5 or S6. After filling one word, the next word is prepared for accepting inputs by marking this next word with the marker m2. The marker is moved around the drum by delaying it one word at a time. The number of word times that the marker is delayed after filling the word is determined by the degree of the polynomial. For a 16'th degree polynomial either S5 or S6 is raised twice. This places the movable marker to identify the word which is to hold the 16'th derivative. The word preceding the 16'th derivative contains the value of z; initially z is zero and requires no input data. Upon filling each complete word position with the binary values for the real and imaginary components for the designated derivative, S5 or S6 is raised to delay the marker one word position. This prepares the computer to receive input data for the next lower derivative.

9) Function switch is set to Coarse δ.

After all input information is entered, the computer is placed in the coarse computational mode. When values of both x and y cease changing as viewed on the oscilloscope, the computer has found the root located closest to the origin.



Control Panel

Figure 20

10) Function switch is set to Fine δ.

The root location is refined in this computational mode. When a stable configuration for x and y is reached, the value of the root is read out visually from the oscilloscope.

11) Function switch is reset to Coarse δ.

12) Direction Control is set to the direction in which it is desired to look for the next root. The computer seeks the root nearest the complex point at which the computer was located when the Direction Control switch was released. This sequence of operations is repeated until all roots are located.

The numbers read from the oscilloscope are least significant digit first. Output provides either the real or imaginary component of the word marked by the movable marker. The bits are read as a high voltage representing a 1 and a low voltage representing a 0.

At present no indication is provided if the value of any component exceeds unity. This would be a useful signal, since it is possible to force the computer to generate such numbers with the Direction Control switch. Care must be exercised in searching for roots near the value $z = 1$. If any number exceeds unity, the computer must be reprogrammed. It would be best to utilize a larger value for R in (7) if a difficulty of this type occurs. In general, when the operator is forcing the search for a new root in the regions where any component is near unity, the safe procedure is to use the fine δ instead of the coarse δ . This allows the operator to exercise more control over the proximity of the computer to unit value for the component in question.

VI. CONCLUSIONS

Several observations result from the design, construction, and test of the computer. The machine requires about one half the equipment necessary for a medium speed general purpose computer of limited memory capacity. The ease and speed of coding problems for the computer and the relatively fast solution times provided by the Root Extractor are distinct advantages of a special purpose machine. The binary coding, bit by bit input and output, and the lack of any systematic method for finding successive roots are severe limitations on the extensive use of the computer by other than scientific personnel.

One of the unexpected results of the experimental test routines was the discovery that the computer locates the saddle points as well as the zeros of the complex polynomial.[25]

The reason that the computer homes on these saddle points is that the direction decision technique does not recognize a zero value for any number as being different from any other positive, non-zero number. When the computer makes a direction decision based on a zero value for a derivative, the step designated by this decision can be in the wrong direction. This is because the increment to be added to $u(z_0)$ is the sum:

$$\tfrac{1}{2}\delta\left[u'(z_0) + u'(z_0 + \delta)\right]$$

Since the direction decision for the step was based on the sign of $u'(z_0)$, and since the value of $u'(z_0 + \delta)$ is not known until the step has been taken, it is possible for the computer to move back and forth

between the two values of z: (z_0) and $(z_0 + \Delta)$. The points for which the signs of $u'(z)$ and $v'(z)$ can differ from the signs of $u'(z + \Delta)$ and $v'(z + \Delta)$ are those where the first derivatives vanish; that is, saddle points.

One solution to this problem is to recognize zero values for the derivatives as numbers differing from all other numbers. The direction decision logic must then be based on the signs of the non-zero components u, u', v, v'. One difficulty with this approach is that the saddle points can occur inside the finest mesh that the computer can recognize. Hence "zero" must be defined as an absolute value less than or equal to the smallest step.

Another solution to the problem of eliminating saddle points is to compute the values of u,u',v,v' for each of the four possible values of Δ. The direction decision in this case has several stages.

(1). After each of the four steps, the normal direction decision computation should be made based on the signs of $u(z_0 + \Delta)$, $u'(z_0 + \Delta)$, $v(z_0 + \Delta)$, $v'(z_0 + \Delta)$.

(2). If the direction indicated by (1) is the same as the direction of the step, this is the correct direction for the computer to move.

(3). If the direction indicated by (1) for each step does not coincide with the direction of that step, then the correct direction to move is that indicated by the signs of $u(z_0)$, $u'(z_0)$, $v(z_0)$, $v'(z_0)$.

Following these three rules, the computer would home only on zeros.

There is always one direction in which the computer can move away from a saddle point and still diminish the values of $u(z_o + \Delta)$ and or $v(z_o + \Delta)$.

The two striking disadvantages of this technique are the complicated decision process and the extra time required to move in any direction. The time requirement can be minimized by always choosing as the first trial step the value of Δ as indicated by the signs of the components at the point z_o. One advantage of this technique is that it would provide a signal indicating that a root had been found. Two consecutive failures to satisfy condition (2) at z_i and then $z_i + \Delta$ indicate a root in the immediate vicinity.

Either of the two automatic saddle point elimination methods could be built into the computer. In order to maintain the simplicity of the experimental machine, neither technique has been implemented. The operator must inspect the values of the first derivatives at any homing point selected by the computer in order to distinguish zeros from saddles. Similarly, the operator must force the computer away from a saddle point using the same method described to find successive zeros.

A total of twenty flip-flops, 260 logical diodes, and 15 drum amplifier and clock pulse tubes constitute the basic equipment of the computer. This equipment could be reduced by eliminating the reading flip-flops and by constraining the computer to the use of one value of δ. Another simplifying technique is to recognize that the value of z at each root is the value:

$$z = \frac{f^{n-1}}{n!\, a_n} - \frac{a_{n-1}}{n\, a_n}$$

Many of the logical design and circuit techniques developed for the root extractor are directly useful in other types of digital devices. In addition to this, the computer has several other possible functions.

The machine can be considered as a function generator of a complex variable; it evaluates polynomials of high degree. Quite complicated functions can be approximated by such polynomials. With the computer operating in this sense, the inputs could be any desired x and y. The internal programming would be set to approach the values of the x and y inputs rather than to diminish the real and imaginary components of the polynomial. Inputs in this case could also be applied to the word position corresponding to any of the derivatives.

In the field of servomechanisms, the computer can be used to obtain root locus plots. A curve plotter would be attached to read the z word position and some form of internal gating would be used to indicate that a root had been located.

Many of the analysis and synthesis techniques of modern network and servomechanism practice require the zeros of complex polynomials. Where there is need for its special purpose of polynomial factoring, the readily available and easily programmed Root Extractor is a valuable scientific tool.

APPENDIX 1

Error Analysis

Two basic types of errors occur in the repeated evaluation of the polynomial by means of a truncated Taylor series. The maximum error due to the truncation of the Taylor series can be evaluated in terms of the expansions used in the computer. The maximum error due to representing each derivative of the polynomial by a digital number is a direct function of the number of binary digits in that number. One object of an error investigation is the determination of the system configuration that makes all errors the same order of magnitude. Another object of an error study is the development of rapid techniques for arriving at solutions of known accuracy.

The error due to the expansion used in the computer can be determined by expanding all functions in their Taylor series form. The basic approximation formula used in the computer to evaluate all derivatives is written:

$$f(z + \Delta) \doteq f(z) + \frac{\Delta}{2} \frac{f(z)}{z} + \frac{\Delta}{2} \frac{df(z+\Delta)}{dz} \tag{1}$$

The Taylor expansions for $f(z+\Delta)$ and $\frac{df(z+\Delta)}{dz}$ are written:

$$f(z+\Delta) = f(z) + \Delta \frac{df(z)}{dz} + \frac{\Delta^2}{2!} \frac{d^2f(z)}{dz^2} + \frac{\Delta^3}{3!} \frac{d^3f(z)}{dz^3} + \cdots$$

$$+ \cdots + \frac{\Delta^{m-1}}{(m-1)!} \frac{d^{m-1}f(z)}{dz^{m-1}} + R_m \tag{2}$$

$$R_m = \frac{\Delta^m}{2\pi i} \oint \frac{f(z_1)dz_1}{(z_1-z-\Delta)(z_1-z)^m}$$

$$\frac{df(z+\Delta)}{dz} = \frac{df(z)}{dz} + \Delta\frac{d^2f(z)}{dz^2} + \frac{\Delta^2}{2!}\frac{d^3f(z)}{dz^3} + \frac{\Delta^{m-1}}{(m-1)!}\frac{d^mf(z)}{dz^m} + R_m' \qquad (3)$$

$$R_m' = \frac{\Delta^m}{2\pi i}(m-1)\oint \frac{f(z_1)dz_1}{(z_1-z-\Delta)(z_1-z)^{m+1}}$$

Hence the approximation (1) can be written:

$$f(z+\Delta) \doteq f(z) + \frac{\Delta}{2}\frac{df(z)}{dz} + \frac{\Delta}{2}\left\{\frac{df(z)}{dz} + \Delta\frac{d^2f(z)}{dz^2} + \frac{\Delta^2}{2!}\frac{d^3f(z)}{dz^3} + \cdots\right\} \qquad (4)$$

The error between (2) and (4) is:

$$-\epsilon = \frac{\Delta^3}{12}\frac{d^3f(z)}{dz^3} + \frac{\Delta^4}{24}\frac{d^4f(z)}{dz^4} + \cdots + \frac{\Delta^m}{2m!}(m-2)\frac{d^mf(z)}{dz^m} + \cdots \qquad (5)$$

For the values of δ used in the computer, and from the fact that the absolute value of all derivatives is less than unity, it is reasonable to state that the error in (1) is of the order of magnitude:

$$o(\epsilon) = \frac{\Delta^3}{12}\frac{d^3f(z)}{dz^3} \qquad (6)$$

Thus for $\delta = 2^{-10}$, $o(\epsilon) = 2^{-33}$.

The error due to digital representation is of the order of 2^{-31} since thirty binary digits are used for each component of all derivatives.

After a large number of applications of (1), the maximum error in the polynomial is simply the product:

$$\S\, \epsilon$$

where ϵ is the maximum error at each step and $\S$ is the number of steps (23). Hence, after the 2^{10} steps necessary to cross the normalized region in the complex plane, the maximum error in the polynomial is found in the 21'st binary digit:

$$2^{10}(2^{-31}) = 2^{-21}$$

The probability that this maximum error will occur in the polynomial is $2^{-(2^{10})}$, even if it is assumed that the error at each step is a maximum. This is a probability of one in 10^{301}.

To refine the location of the root after it has been located with the coarse $\S$ of 2^{-10}, a second value of $\S$ equal to 2^{-20} is used with the simpler expansion formula:

$$f(z + \Delta) \doteq f(z) + \Delta \frac{df(z)}{dz} \qquad (7)$$

By changing to a smaller value of it is possible to stop the propagation of the possible error at the 20'th binary position. With $\S = 2^{-20}$ the error due to (7) is of the order of 2^{-41}, and the error due to digitalization of the numbers is of the order of 2^{-31}. Thus in arriving at

the root location represented by the twenty most significant binary numbers, it is necessary to take $2(2^{10}) = 2^{11}$ steps. This places the maximum possible error in each of the derivatives in the 20'th binary position.

The advantages of using two values of δ may be summarized as follows. First, for any given fixed magnitude of maximum possible error, shorter digital words (fewer digits) can be used to obtain this final error by providing two values of δ . This effect in itself usually means faster solutions and smaller memory requirements. Second, solutions of the same accuracy are obtained much more rapidly by using two values of δ since far fewer steps need be taken. In the Root Extractor, the use of two values for δ makes the computer 1000 times faster than a corresponding computer using a $\delta = 2^{-20}$. This is because approximately 1000 fewer steps are taken in finding a given root. Since a single machine using $\delta = 2^{-20}$ in equation (7) would require forty digit numbers instead of thirty as utilized in the Root Extractor, one could say that the two δ computer is about 1300 times faster than a single δ computer. In terms of solution times, the Root Extractor obtains one root every 16 seconds on the average. A single δ computer would obtain one root every $5\frac{1}{2}$ hours, on the average.

This comparison is particularly valuable when it is realized that an ordinary digital differential analyser can be programmed to find the real roots of real polynomials. For twenty binary digit accuracy, the digital differential analyser would require about six hours average time to obtain one root. It would seem possible to program the digital differential analyser to find complex roots.

APPENDIX 2

Polynomial Preparation

The computer places one major restriction on all numbers: their absolute values must be less than unity. The direction control apparatus operates without reference to the value of each number and can force numbers to exceed unit value. Since there is no signal indicating overflow if any number exceeds one, it is convenient to normalize the polynomial to include all roots within a circular contour of unit radius in the complex plane. The second operation performed on the polynomial is insuring that the polynomial and its derivatives are themselves all smaller than unity.

A simple equation is derived to give the radius of the circular contour in the complex plane enclosing all n roots of an n'th degree polynomial. This derivation uses the theorem[24] stating that if at all points of a contour the following inequality is satisfied:

$$\left|a_n z^n\right| > \left|a_0 + a_1 z + \cdots + a_{n-1} z^{n-1}\right| \qquad (1)$$

then all n roots lie within the contour. Dividing this inequality by the absolute value of the largest coefficient a_K of the polynomial and modifying the inequality provides a simple relationship indicating the radius of the contour enclosing all the roots.

$$\left|\frac{a_n}{a_K} z^n\right| > \left|\frac{a_0}{a_K}\right| + \left|\frac{a_1}{a_K} z\right| + \left|\frac{a_2}{a_K} z^2\right| + \cdots$$

$$+\left|\frac{a_{n-1}}{a_K} z^{n-1}\right|$$

$$\left|\frac{a_n}{a_K} r^n\right| > \left|\frac{a_0}{a_K}\right| + \left|\frac{a_1}{a_K} r\right| + \left|\frac{a_2}{a_K} r^2\right| + \cdots$$

$$+\left|\frac{a_{n-1}}{a_K} r^{n-1}\right|$$

where: $z = re^{j\theta}$. Since the coefficients of r are all less than one, the inequality can be simplified to a geometric series:

$$\left|\frac{a_n}{a_K} r^n\right| > 1 + |r| + |r^2| + \cdots + |r^{n-1}|$$

This series is summed:

$$\left|\frac{a_n}{a_K} r^n\right| > \frac{|r^n| - 1}{|r| - 1}$$

Dividing through by r^n gives the result:

$$\left|\frac{a_n}{a_K}\right| > \frac{1 - \frac{1}{|r|^n}}{|r|}$$

For polynomials in general, the radius of that circular contour enclosing all roots of the polynomial is computed as:

$$r > \left| \frac{a_K}{a_n} \right| \tag{2}$$

Stronger conditions can be derived, but in programming the Root Extractor it is convenient to employ the simplest possible manual computations for preparing the polynomial coefficients.

Transforming the polynomial to the new variable:

$$w = \frac{z}{2^r} \tag{3}$$

contracts all the roots of the polynomial in w to within the unit circle. For this transformation it is convenient to use r as that power of either 10 or 2 which satisfies (2).

The second modification applied to the polynomial is insuring that all derivatives are less than one. It is convenient to keep this calculation simple.

$$f(z) = a_n z^n + \cdots + a_1 z + a_0$$

$$f(w) = a_n 2^{n r} w^n + a_{n-1} 2^{(n-1) r} w^{n-1} + \cdots + 2 a_1 r w + a_0 \tag{4}$$

In dividing through by the largest coefficient, $2^K a_K r^K$, the following statement can be made:

$$\left| \frac{1}{2^K} \cdot \frac{f(w)}{a_K r^K} \right| \leq n + 1$$

This is because $|w| < 1$. Similarly for the first derivative:

$$\left| \frac{1}{2^K a_K r^K} \quad \frac{df(w)}{dw} \right| \leq n^2$$

The largest number appearing as this upper limit for any derivative is n! for the n'th derivative:

$$\left| \frac{1}{2^K a_K r^K} \quad \frac{d^n f(w)}{dw^n} \right| \leq n! \tag{5}$$

Dividing f(w) by the quantity $n! a_K r^K 2^{K+1}$ thus insures that all derivatives will be less than unity. The polynomial and its derivatives are finally evaluated at w = 0 . These binary values are entered into the computer.

$$F(w) = \frac{f(w)}{a_K r^K n! 2^{K+1}} \tag{6}$$

$$= \frac{a_n}{a_K} \frac{r^n}{2^K r^K} \frac{w^n}{2 n!} + \frac{a_{n-1}}{a_K} \frac{r^{n-1}}{2^K r^K} \frac{w^{n-1}}{2 n!} + \cdots$$

$$+ \frac{a_1}{a_K} \frac{r}{2^K r^K} \frac{w}{2 n!} + \frac{a_0}{2 a_K r^K n! 2^K}$$

Evaluated at w = 0, the transformed polynomial and its derivatives are summarized as follows:

$$F(0) = \frac{a_0}{2 a_K r^K n! 2^K}$$

$$\left. \frac{dF(w)}{dw} \right|_{w=0} = \frac{a_1}{2 a_K r^K n! 2^K}$$

$$\left.\frac{d^2F(w)}{dw^2}\right|_{w=0} = \frac{2a_2}{2a_K r^K n! 2^K}$$

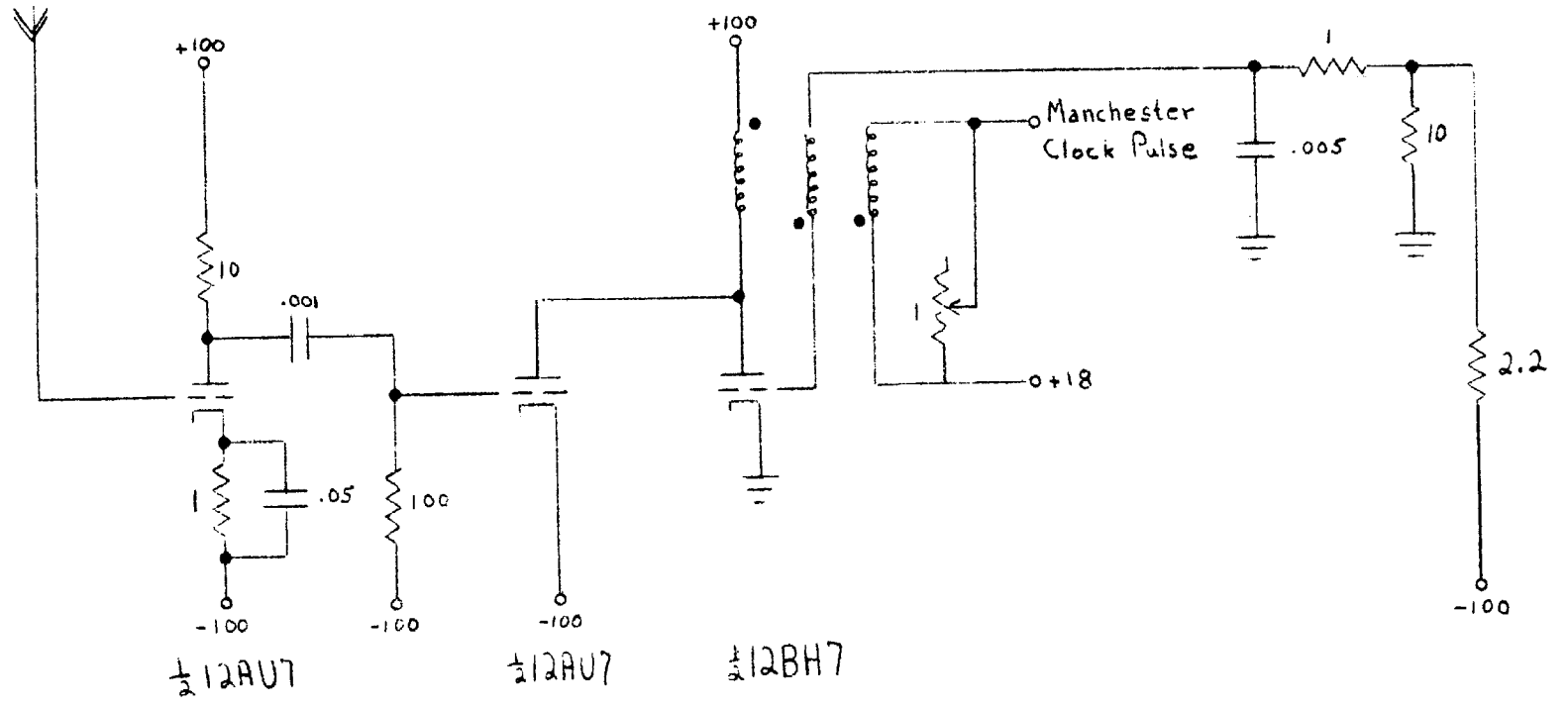
$$\left.\frac{d^3F(w)}{dw^3}\right|_{w=0} = \frac{3!a_3}{2a_K r^K n! 2^K}$$

$$\left.\frac{d^mF(w)}{dw^m}\right|_{w=0} = \frac{m!a_m}{2a_K r^K n! 2^K}$$

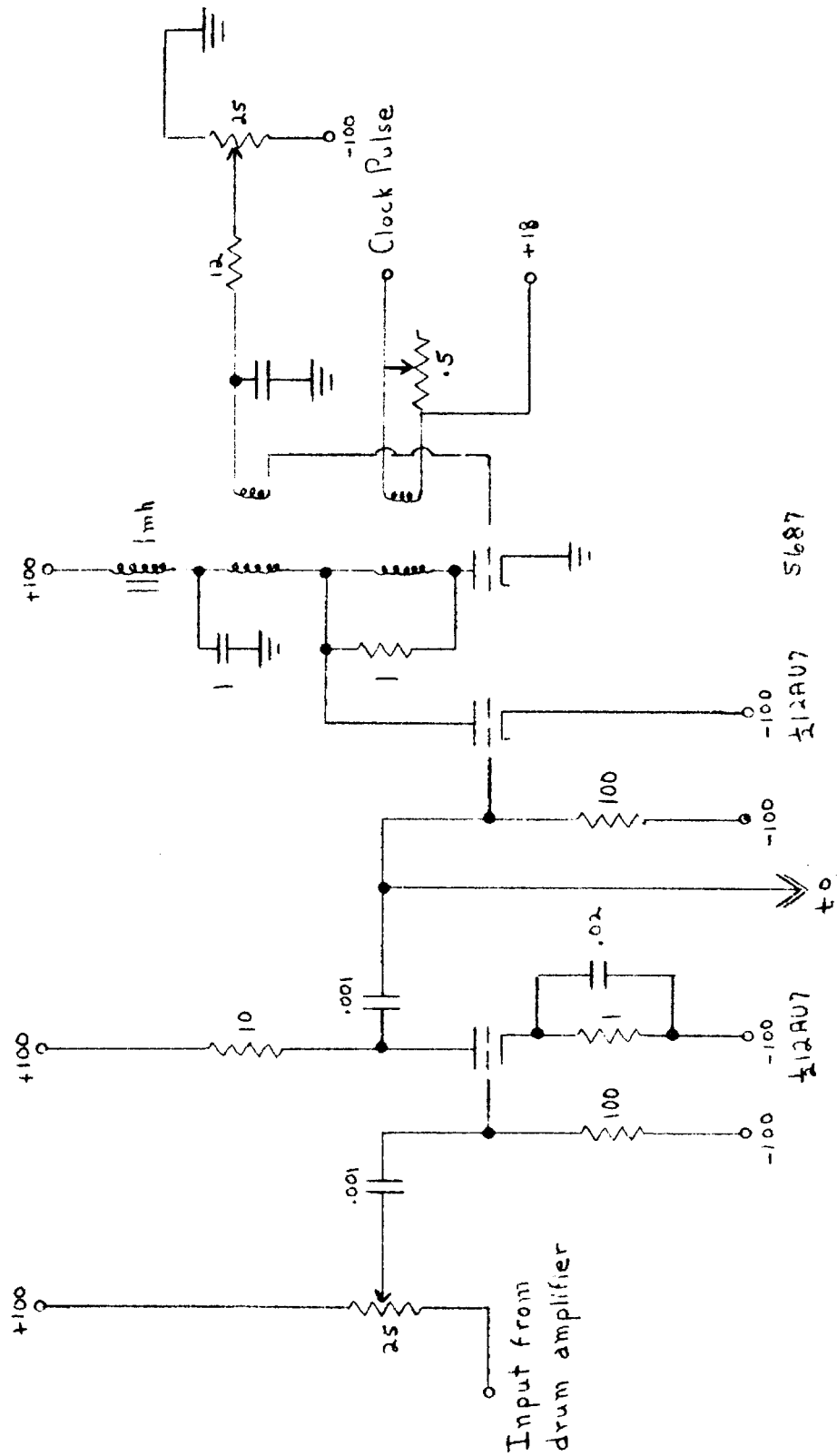
The quantity $a_K r^K 2^{k+1}$ is the largest coefficient of the modified polynomial (4).

The conditions (5) and (2) are mathematically weak criteria for placing the bounds on z and the derivatives. It is possible that the use of these formulae on high degree polynomials will result in a loss of significance for certain root loci. In such situations the best solution is to use (2) and (5) and obtain the approximate location of all roots. Renormalizing the polynomial to various values of r which are only a bit larger than each root will permit solutions yielding the full significance of twenty binary places.

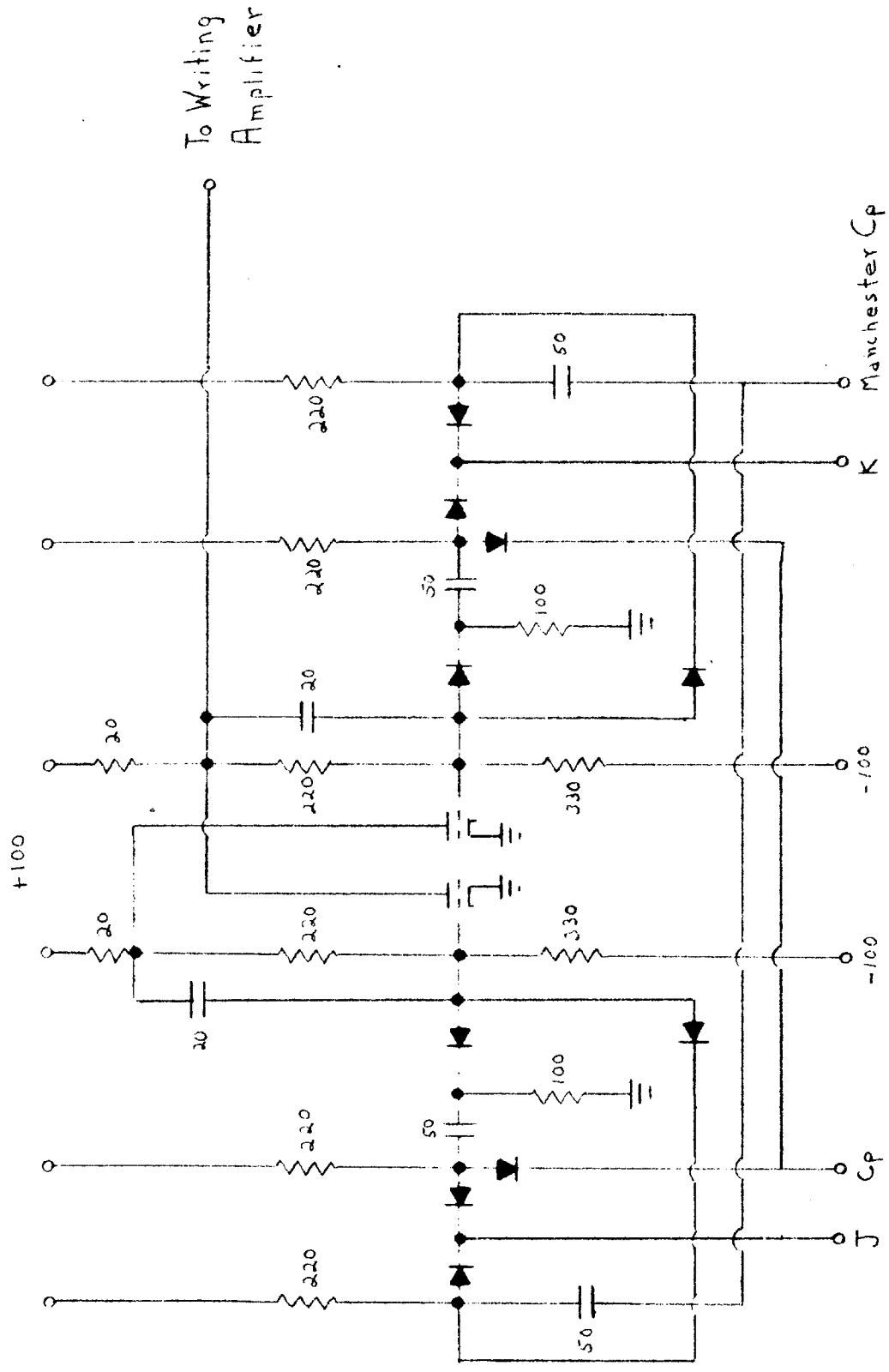
Appendix 3.



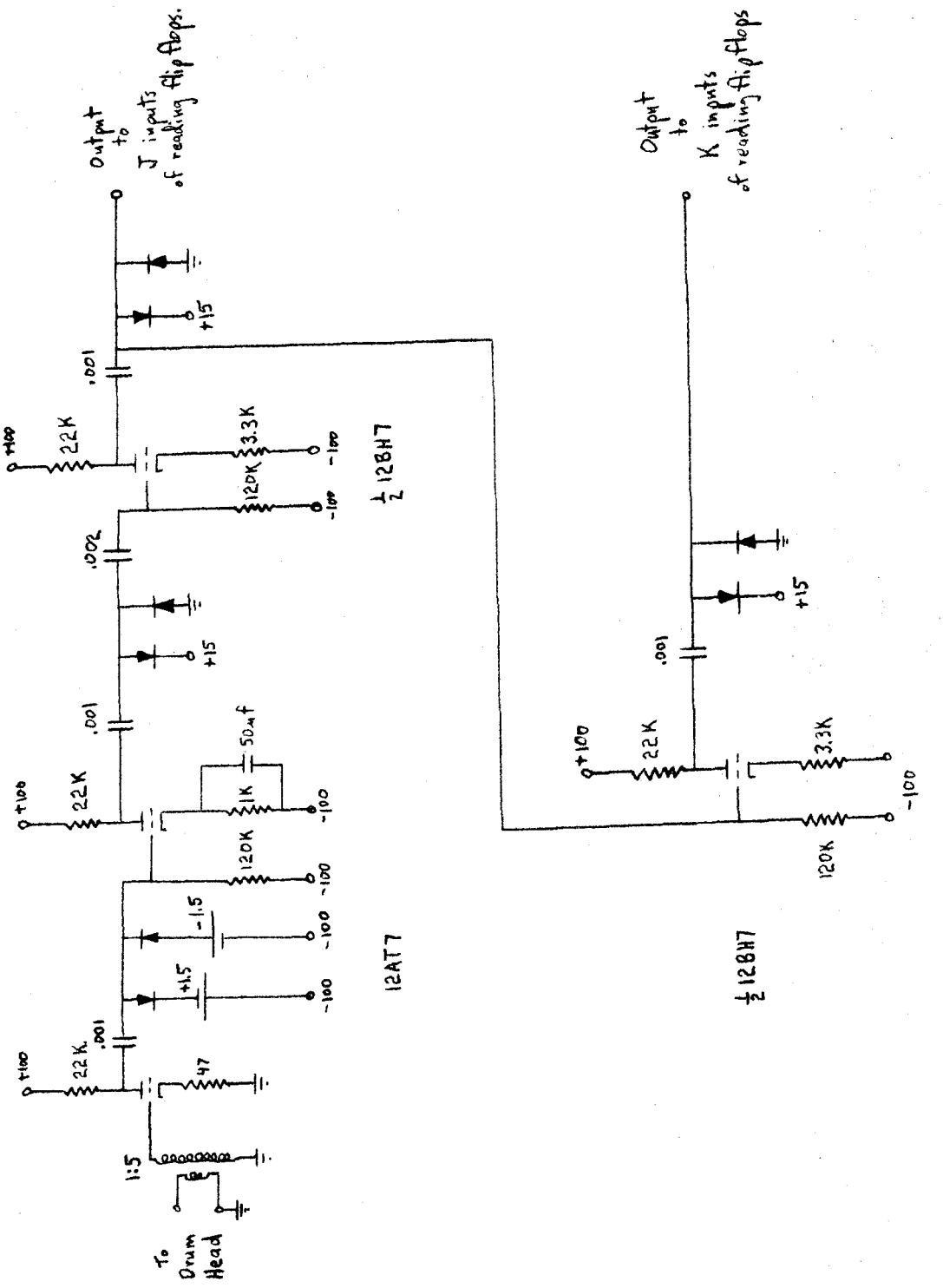
Manchester Clock Pulse Generator



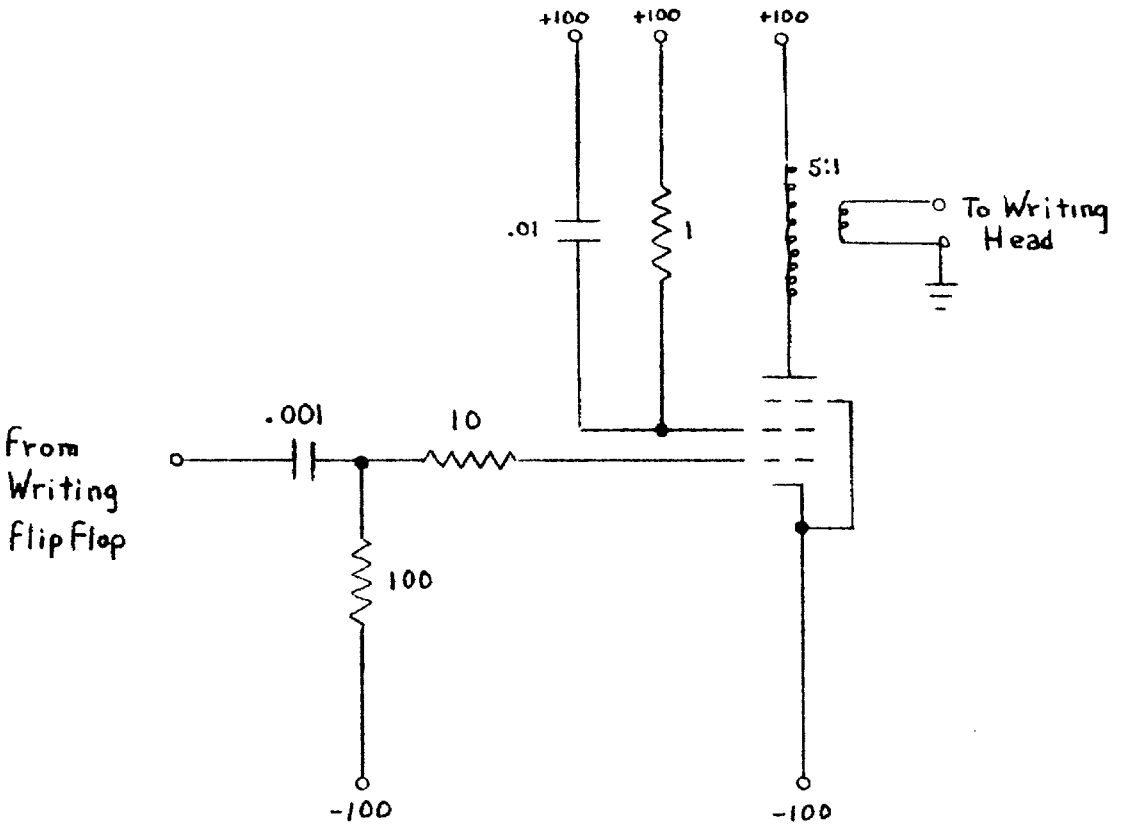
Clock Pulse Generator



Manchester Writing Flip Flop

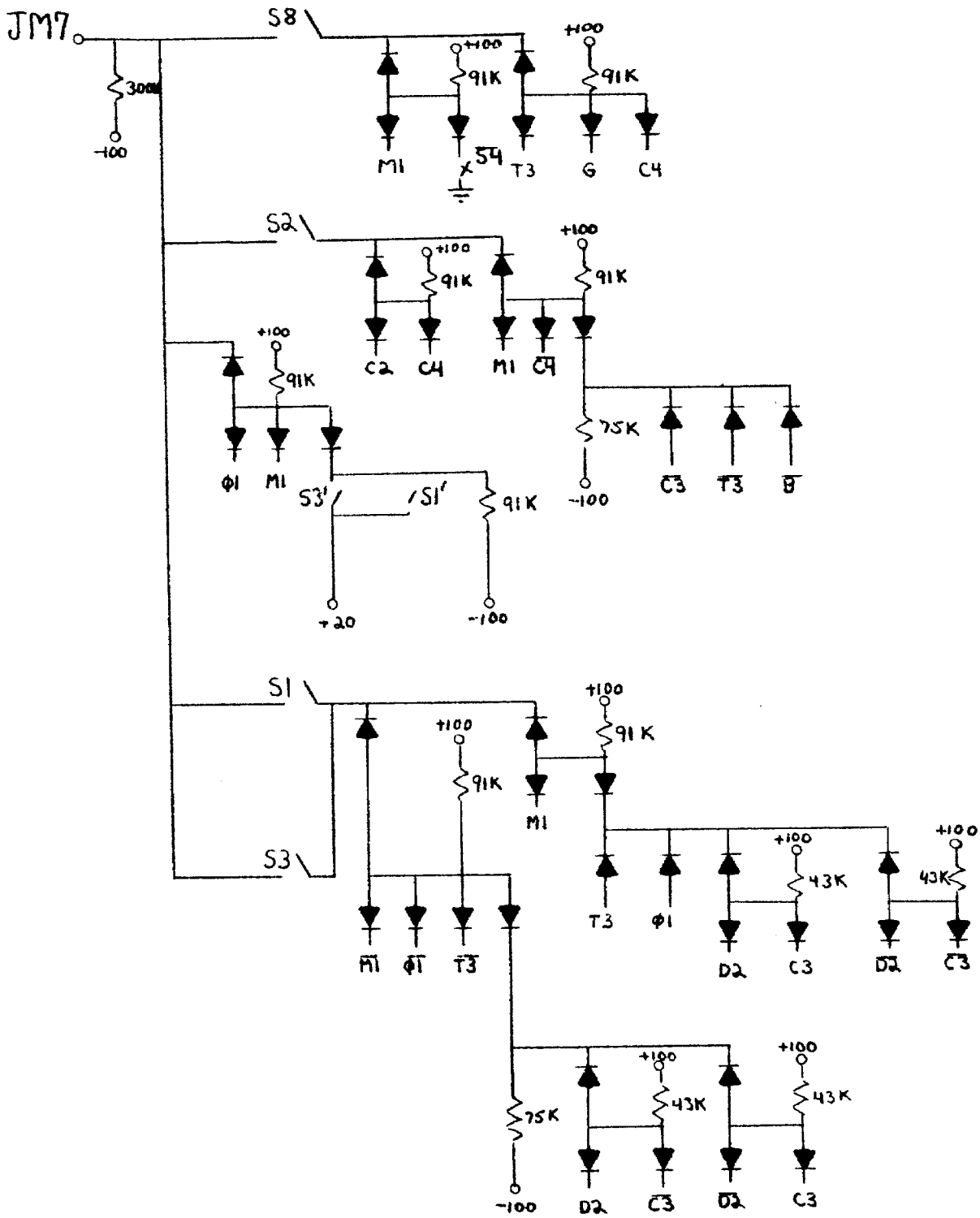


Drum Reading Amplifier

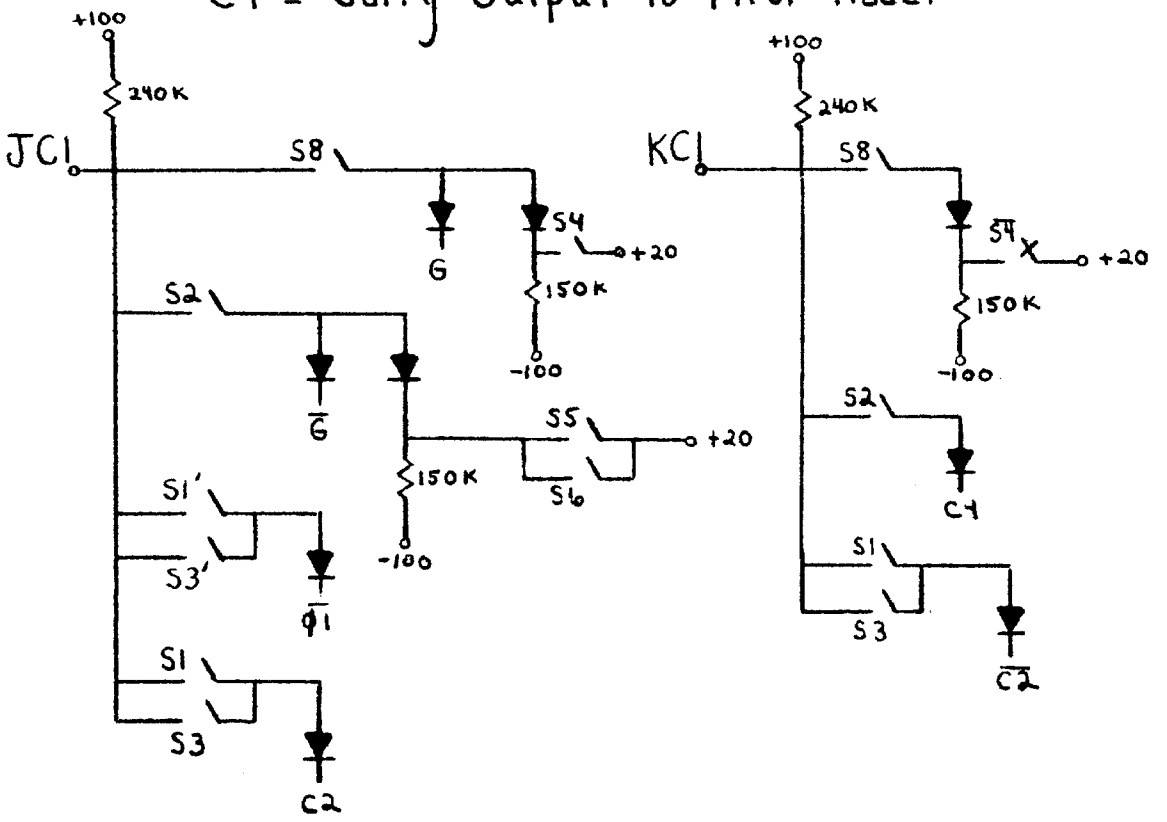


Manchester Writing Amplifier

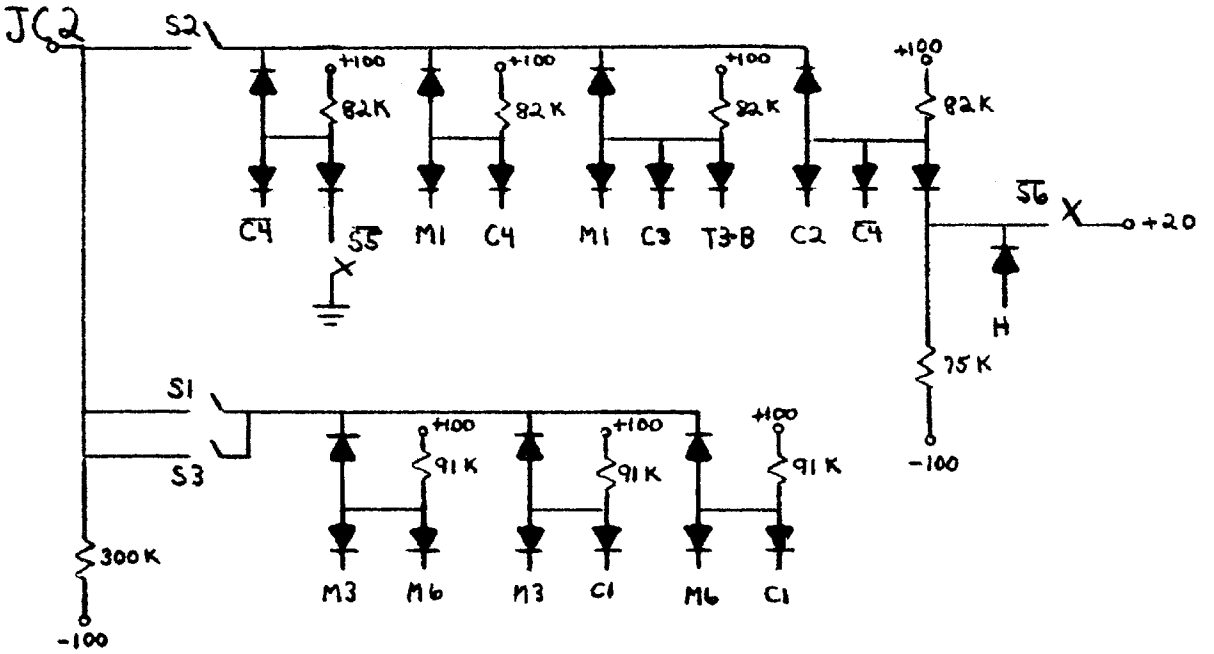
M7 = Writing Flip Flop



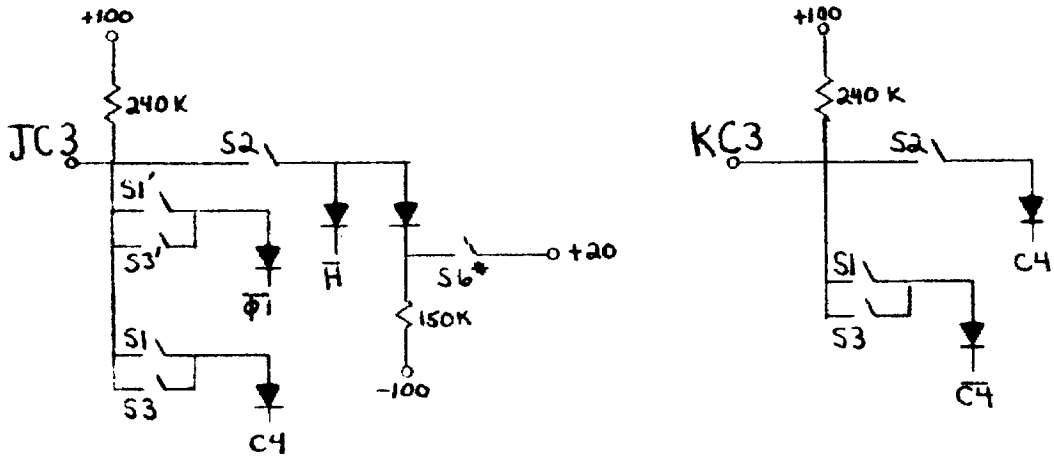
C1 ≡ Carry Output to First Adder



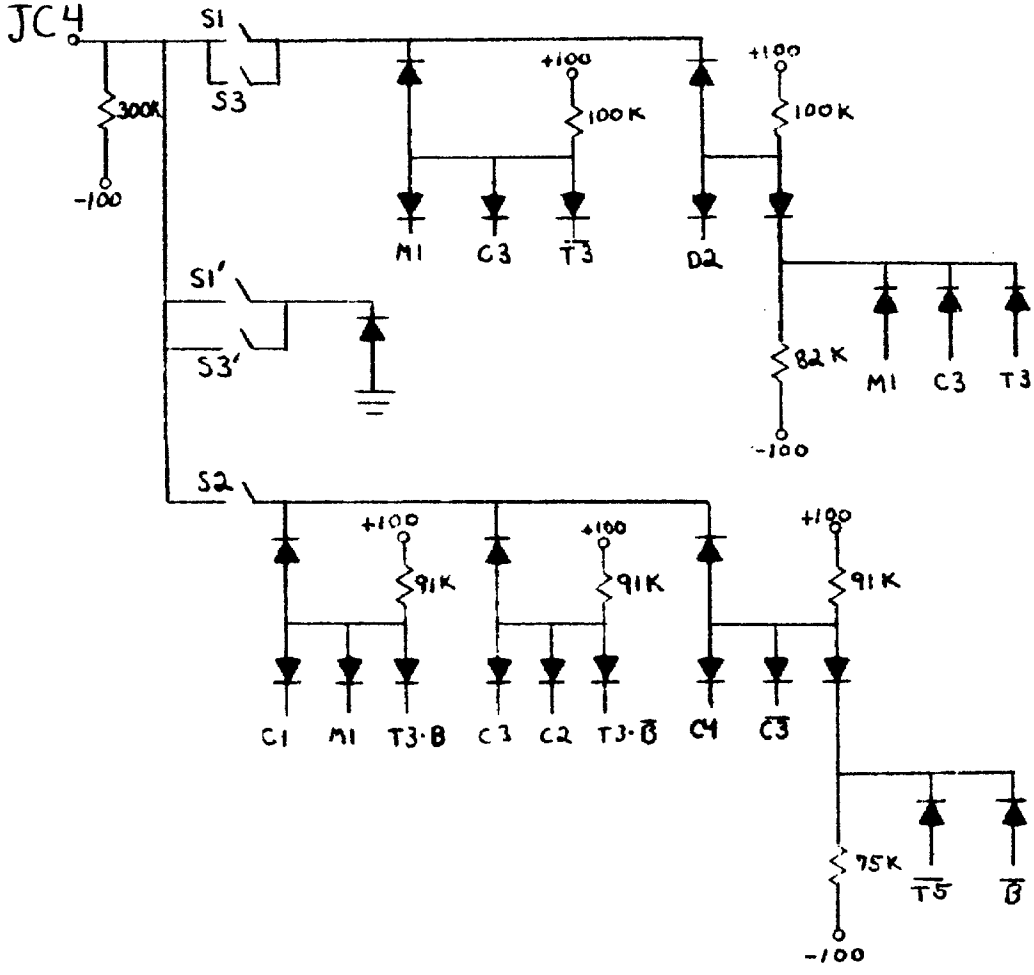
C2 ≡ Carry Input to First Adder



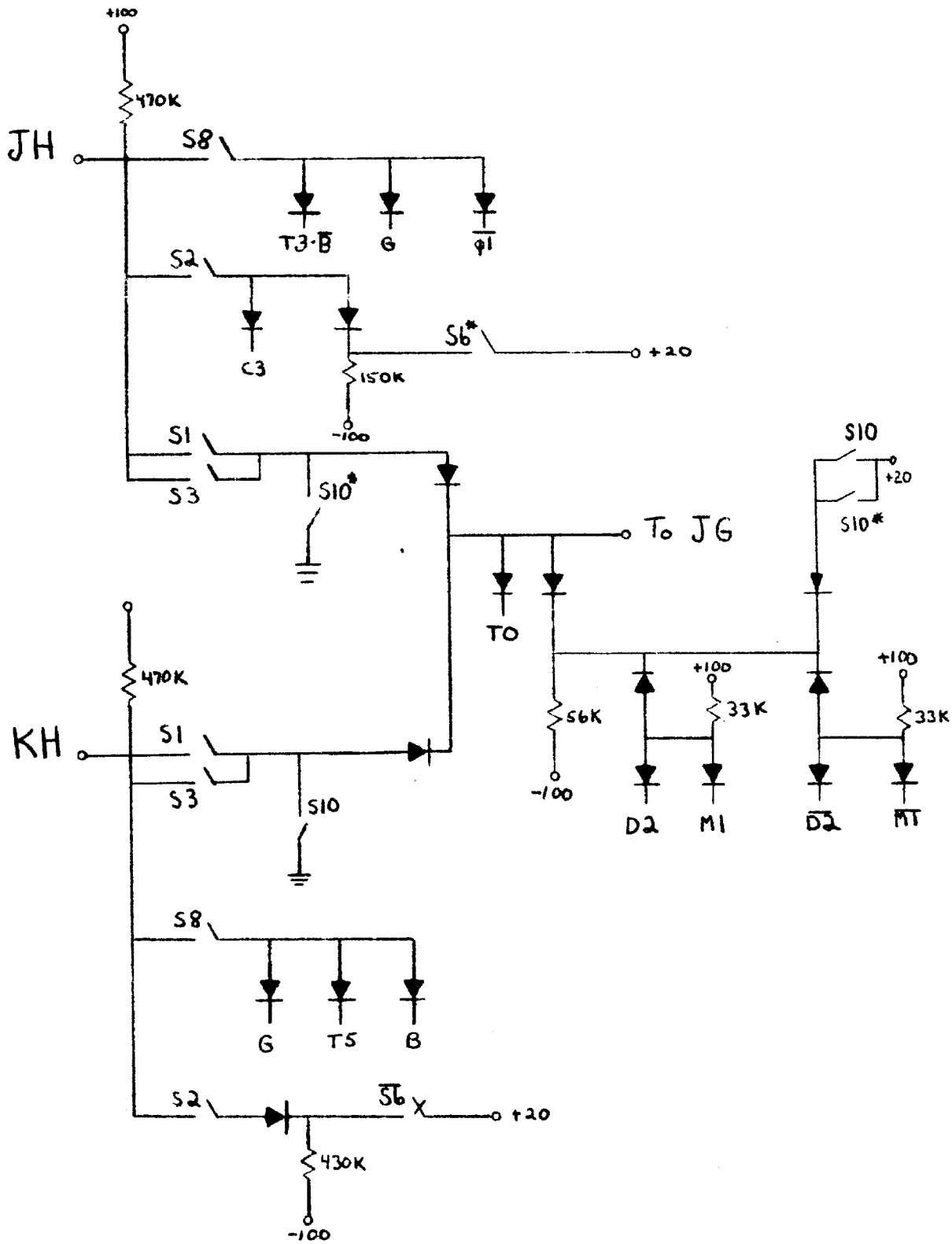
C3 ≡ Carry Output from Second Adder



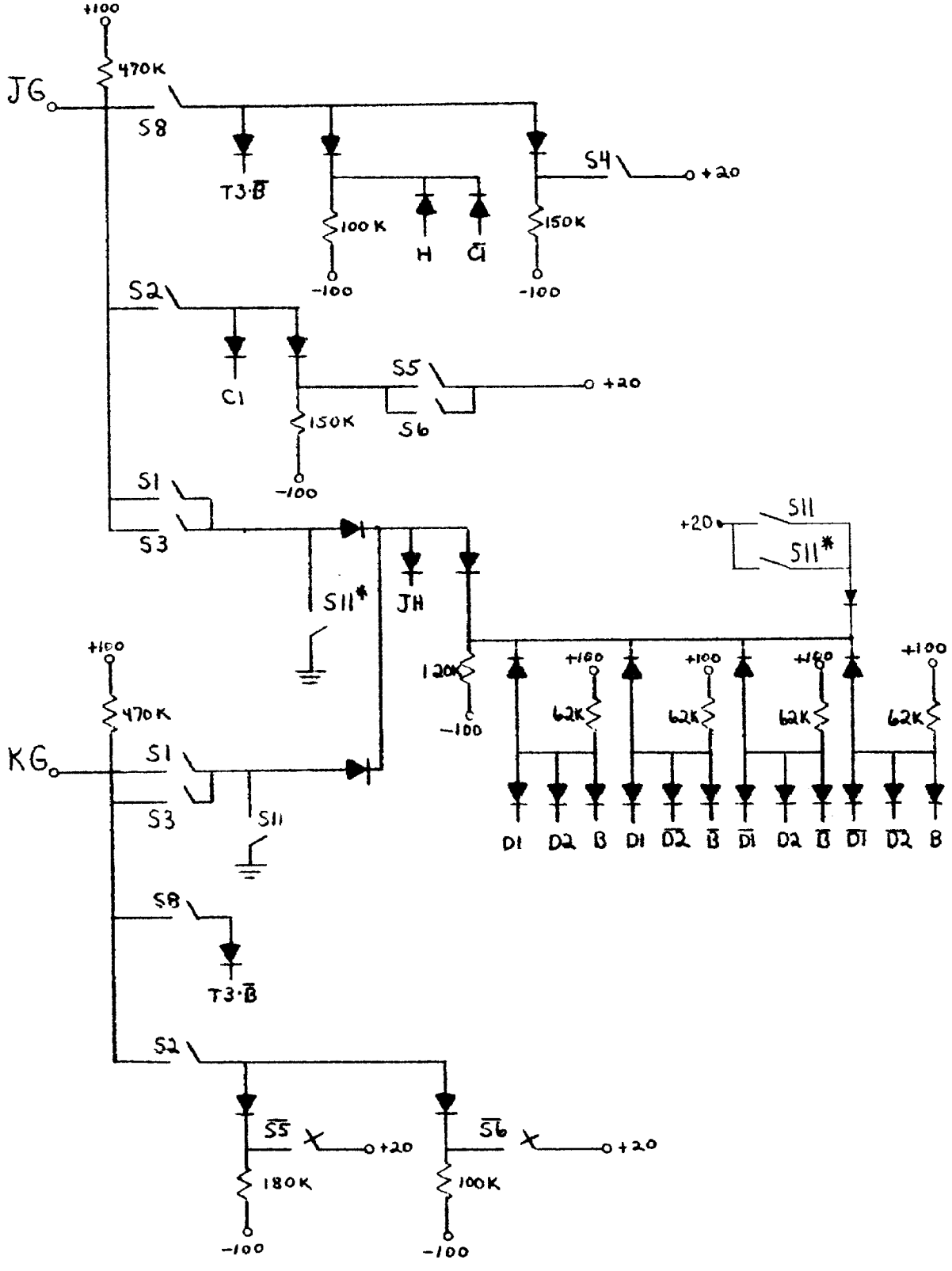
C4 ≡ Carry Input to Second Adder



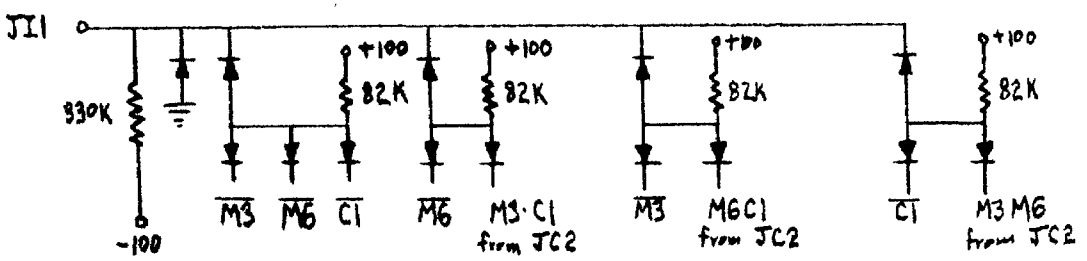
H ≡ Direction Control (Real or Imaginary)



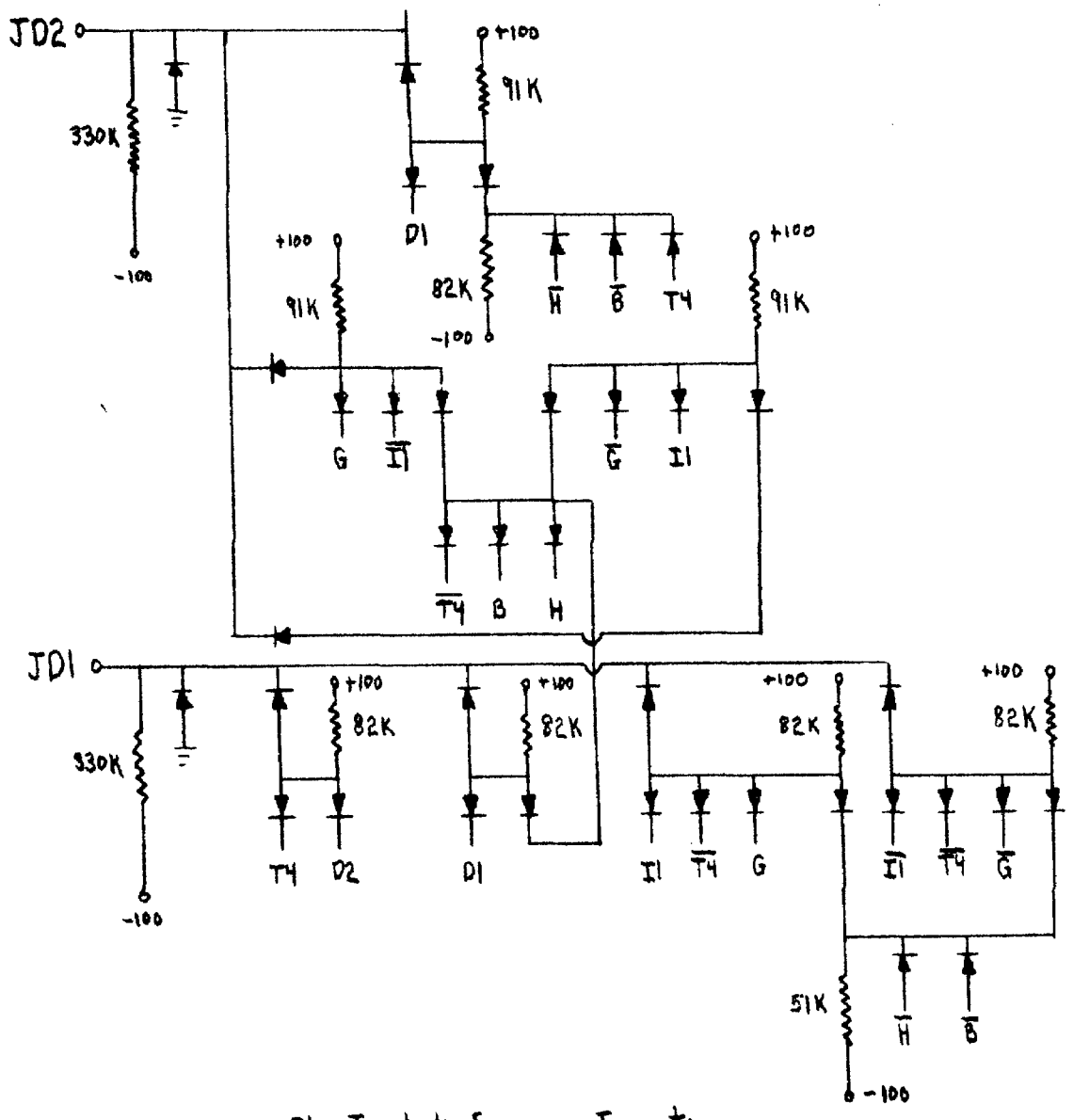
G ≡ Direction Control (plus or minus)



I1 ≡ Adder Output

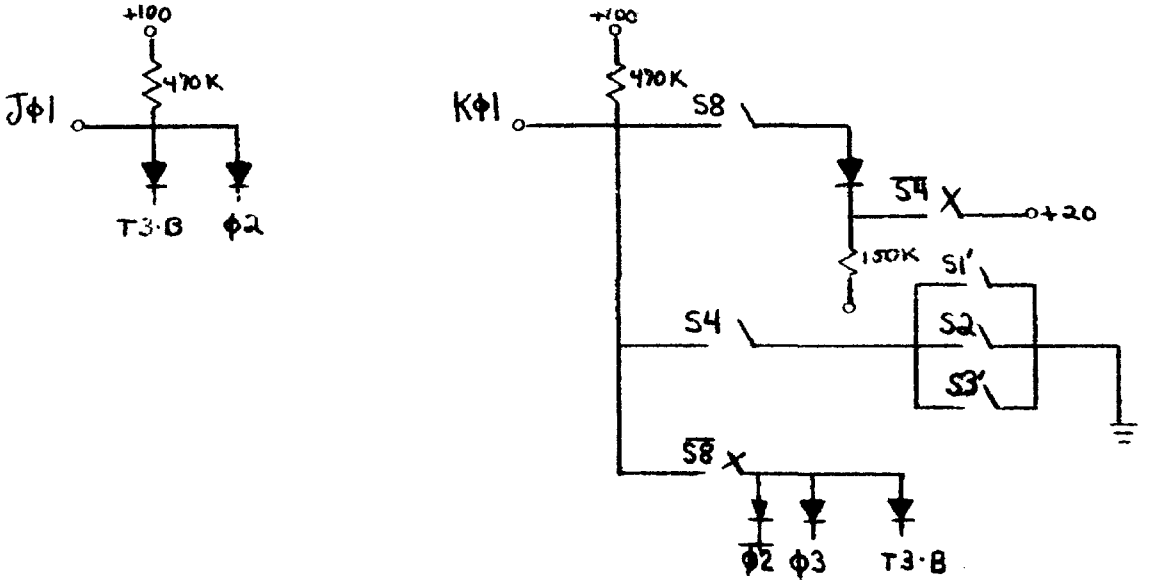


D2 ≡ Output from Sequence Inverter

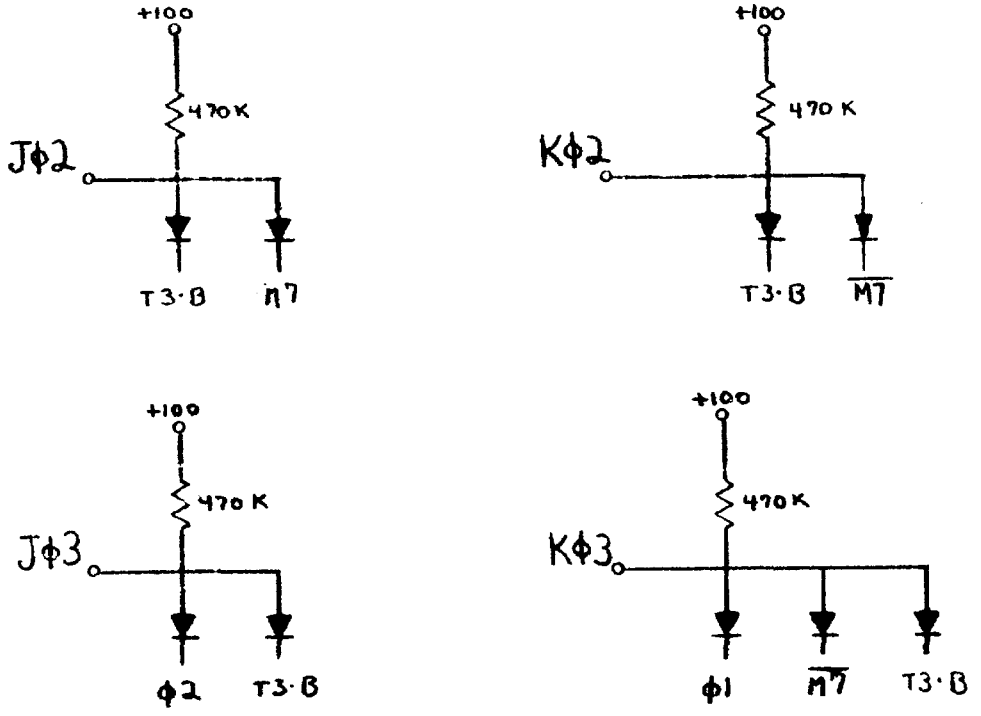


D1 ≡ Input to Sequence Inverter

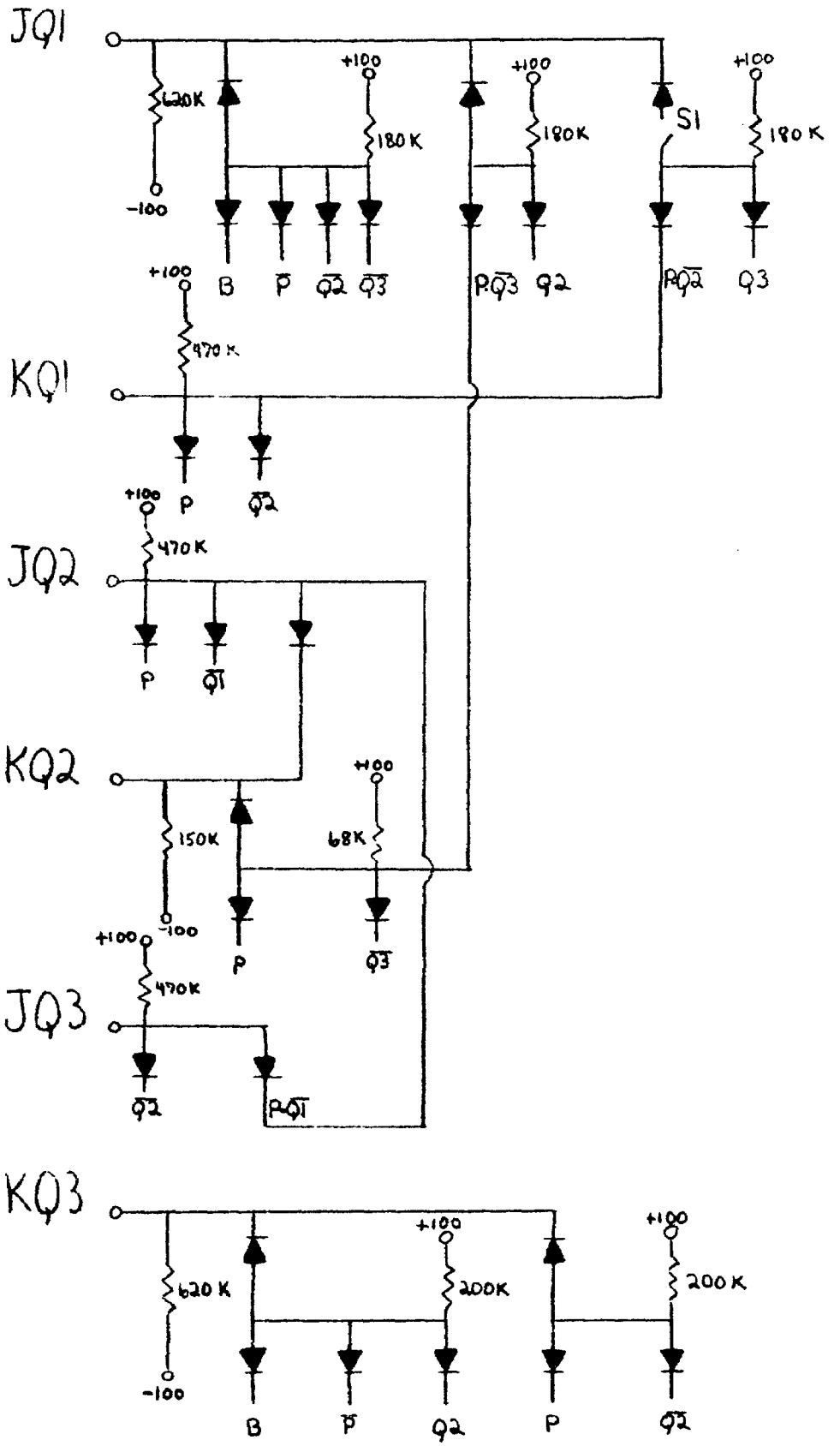
$\phi 1 \equiv$ Circulation Control



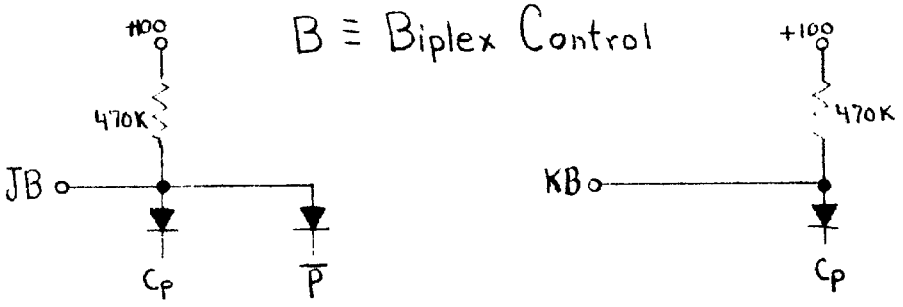
$\phi 2, \phi 3 \equiv$ General Phase Control



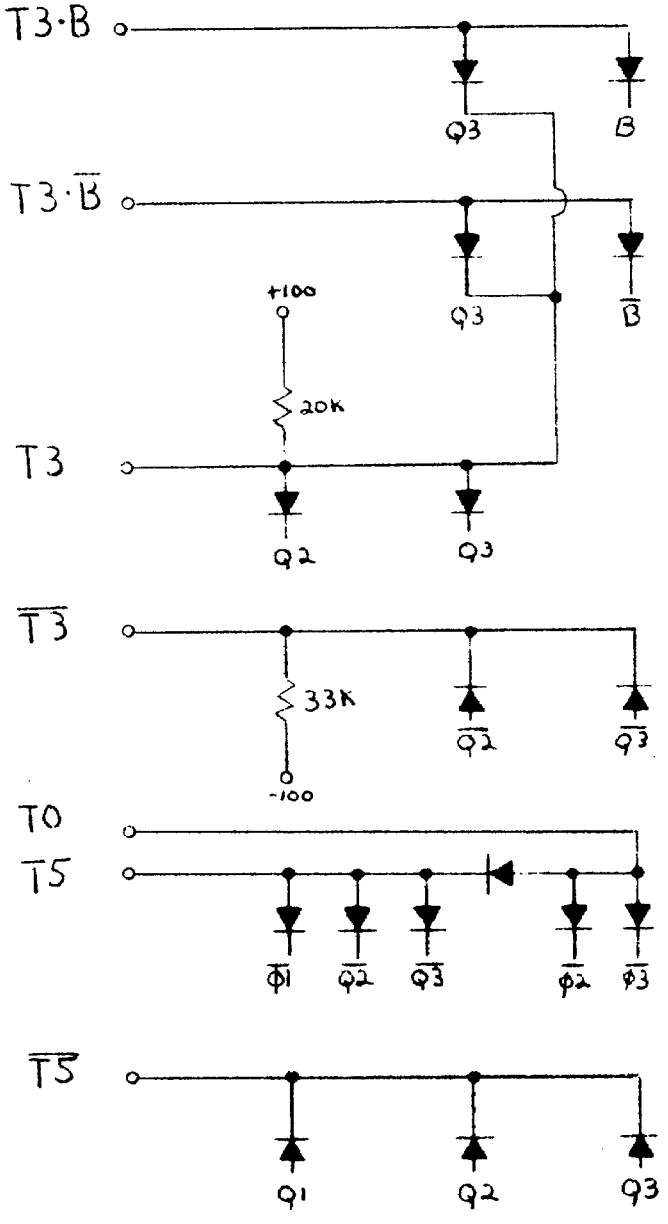
Q1, Q2, Q3 ≡ General Timing Flip Flops



B ≡ Biplex Control



Separate Timing Signals



APPENDIX 4

Typical Problem Solution

To illustrate the problem preparation and the computer operation, the following polynomial was prepared for insertion into the root extractor.

$$f(z) = (z - 8)(z - j64)(z + 1 + j1) \qquad (1)$$

$$= z^3 + z^2(-7 - j63) + z(56 + j440) - 512 + j512 \qquad (2)$$

The circular radius enclosing all roots of (2) is found from (2) of Appendix 2:

$$r > \left| \frac{a_k}{a_n} \right| = \frac{512}{1} = 2^9 . \qquad (3)$$

In polynomials having complex coefficients, it is adequate to use the larger of either the real or the imaginary part of the largest coefficient as a_k in the formula. Transforming to the new variable defined by (3) of Appendix 2:

$$w = \frac{z}{2r} = \frac{z}{2^{10}} \qquad (4)$$

$$f(w) = w^3 - w^2(7 + j\,63)2^{-10} + w(7 + j55)2^{-17} + - 2^{-21} + j2^{-21} .$$

The largest coefficient of $f(w)$ is that of w^3 so that by (6) of Appendix 2:

$$F(w) = \frac{f(w)}{2(1)3!} = \frac{f(w)}{12} . \qquad (5)$$

For convenience in programming, $f(w)$ is divided by the power of 2 just greater than 12. Since this is $2^4 = 16$, the polynomial to be entered into the computer is written:

$$F(w) = 2^{-4}w^3 - w^2(2^{-14})(7 + j63) + w(7 + j55)2^{-21} - 2^{-25} + j2^{-25} . \qquad (6)$$

The three derivatives of F(w) are:

$$F'(w) = 3\cdot 2^{-4} w^2 - w(2^{-13})(7 + j63) + (7 + j55)2^{-21}$$

$$F''(w) = 3(2^{-3})w - (7 + j63)2^{-13} \qquad (7)$$

$$F'''(w) = 3(2^{-3}) .$$

Evaluating these derivatives at w = 0 gives:

$$\begin{aligned} F(0) &= -2^{-25} + j2^{-25} \\ F'(0) &= 7(2^{-21}) + j(55)2^{-21} \\ F''(0) &= -7(2^{-13}) - j(63)2^{-13} \\ F'''(0) &= 3(2^{-3}) . \end{aligned} \qquad (8)$$

Converting these values to binary numbersgives:

$$\begin{aligned} F(0) &= 1.11111111111111111111111100000 + \\ &\quad + j0.00000000000000000000000100000 \\ F'(0) &= 0.00000000000000000111000000000 + \\ &\quad + j0.00000000000000011011100000000 \qquad (9) \\ F''(0) &= 1.11111111110010000000000000000 + \\ &\quad + j1.11111110000010000000000000000 \\ F'''(0) &= 0.01100000000000000000000000000 + \\ &\quad + j0.00000000000000000000000000000 . \end{aligned}$$

These numbers biplexed for entry into the computer read as follows.

$$\begin{aligned} F &= 10.10101010101010101010101010101010101010101010101010110000000000 \\ F' &= 00.00000000000000000000000000000010100111111000000000000000000 \qquad (10) \\ F'' &= 11.11111111111111101010000011000000000000000000000000000000000000 \\ F''' &= 00.00101000000000000000000000000000000000000000000000000000000000 \\ &= 00.00100000000000000000000000000000000000000000000000000000000000 \end{aligned}$$

The bits from equations (10) are entered into the proper word positions starting from the left or most significant end. Using the oscilloscope sweep to inspect these values, each word appears on the sweep in the inverted order. The most significant bits are on the right hand end of the oscilloscope display.

In order to observe the path taken by the computer, it is instructive to use the single step feature. Depressing the single step switch causes the computer to take one step Δ at a time. This mode of operation permits the operator to inspect the values of the function and its derivatives at each step in the root searching cycle.

With the computer as now set up, the value of δ is 2^{-9} rather than 2^{-10}. Also, the second value of $\delta = 2^{-20}$ has not been connected into the switching assembly so that only one value, $\delta = 2^{-9}$, is now available. Hence in single step operation, the computer is seen to move between the two values of z:

$$\begin{aligned} z_0 &= 0 + j0 \\ z_1 &= 0 + j1.11111111111100000000 \ldots \end{aligned} \tag{11}$$

Multiplying these values of z by 2^2 due to the marker convention gives the two numbers:

$$\begin{aligned} w_1 &= 0 + j0 \\ w_1' &= 0 - j2^{-9} . \end{aligned}$$

The values of the function oscillate between the value at the origin and:

$$\begin{aligned} F(-j\delta) = {} & 0.00000000000000000000000010111 + \\ & + j\ 0.00000000000000000000000101001 . \end{aligned} \tag{12}$$

On the oscilloscope display, these bits appear in reversed order and are biplexed. The reason that the computer oscillates between these values of z is that the root nearest the origin is at $z = -2^{-10} + j(-2^{-10})$.

Since $\delta = 2^{-9}$, the values (11) are the closest indication of the root location that is possible with the coarse δ. Using a finer δ would enable one to actually locate this root to much greater precision.

The second root can be found by forcing the computer to move in any specified direction. If one forces several step in the $\pm x$ or + direction, the computer will then move under its own accord to the values:

$$
\begin{aligned}
z_2 &= 0.00000000100000000000000000000 + \\
&\quad + j\ 0.0000000\ \ldots \\
z_3 &= 0.00000000100000000000000000000 + \\
&\quad + j\ 0.000000000010000000000000000000 \qquad (14) \\
F_2 &= 1.11111111111111111111111111110 + \\
&\quad + j\ 1.11111111111111111111111110010 \\
F_3 &= 0.00000000000000000000001000000 + \\
&\quad + j\ 0.00000000000000000000000010001
\end{aligned}
$$

Because of the marker convention, the root is located between:

$$
\begin{aligned}
&w_2 = 2^{-7} + j\ 0 \\
\text{and} \qquad &w_2' = 2^{-7} + j\ 2^{-11}\ . \qquad (15)
\end{aligned}
$$

The third root is found by forcing the computer to move at least sixteen steps in the $(\pm j\ \delta)$ direction and then releasing it. Forcing the computer to move farther than this is all right as long as an overflow does not occur (ie., fewer than 256 steps). The computer will then converge to the following set of values.

$z_4 = 0.000000000010000.... + j\ 0.0000010000000...$

$z_5 = 0.000000000000000.... + j\ 0.0000010000000...$

$z_6 = 0.000000000000000.... + j\ 0.0000010000100...$

$z_7 = 0.000000000010000.... + j\ 0.0000010000100...$

Hence the third set of values for z show that the third root is located between:

$$w_3 = 2^{-9} + j\,2^{-4}$$

$$w_3' = 0 + j\,2^{-4}$$

$$w_3'' = 0 + j(2^{-4} + 2^{-9})$$

$$w_3''' = 2^{-9} + j(2^{-4} + 2^{-9})$$

Due to the lack of round off in the computer, it is not advisable to allow continued circulation about the root. The round off errors propagate in direct proportion to the number of steps the computer takes.

If a finer value of δ were available, two other points could be found about which the computer would oscillate. These two points are the saddle points where the first derivative vanishes:

$$w_4 = (1.2 + j85)2^{-10}$$

$$w_5 = (3.4 - j43)2^{-10}$$

Saddle points are readily recognized in the computer since the first derivatives vanish while the value of the polynomial does not vanish. In the solution of polynomials, the saddle points must be identified and distinguished from the zeros. Successive solutions are still found by forcing the computer away from the values toward which it homes. Around saddle points however, it is necessary to force the computer to take just one step in the right direction. After this one step the computer homes on the nearest root.

For general reference, the rules of converting binary information to decimal and back again are summarized below.

A. Conversion of Decimal to Binary.

1) Decimal numbers greater than unity.

Divide the decimal number by 2 and record the remainder. Repeating this operation, record the binary digits obtained in this fashion as most significant digit first. For example:

$137 = 10010001$ as obtained by -

$\frac{1}{2}(137)$	$= 68 + \frac{1}{2}$	thus	$R_1 = 1$
$\frac{1}{2}(68)$	$= 34 + 0$		$R_2 = 0$
$\frac{1}{2}(34)$	$= 17 + 0$		$R_3 = 0$
$\frac{1}{2}(17)$	$= 8 + \frac{1}{2}$		$R_4 = 1$
$\frac{1}{2}(8)$	$= 4 + 0$		$R_5 = 0$
$\frac{1}{2}(4)$	$= 2 + 0$		$R_6 = 0$
$\frac{1}{2}(2)$	$= 1 + 0$		$R_7 = 0$
$\frac{1}{2}(1)$	$= 0 + \frac{1}{2}$		$R_8 = 1$

2) Decimal numbers less than unity.

Multiply by 2 repeatedly;

Record the carry beyond the decimal

point as most significant binary digit first. For example:

$0.137 \doteq 0.00100011$

$.137 \times 2 = .274 \qquad C_1 = 0$

$.274 \times 2 = .548 \qquad C_2 = 0$

$.548 \times 2 = .096 \qquad C_3 = 1$

$.096 \times 2 = .192 \qquad C_4 = 0$

$.192 \times 2 = .384 \qquad C_5 = 0$

$.384 \times 2 = .768 \qquad C_6 = 0$

$.768 \times 2 = .536 \qquad C_7 = 1$

$.536 \times 2 = .072 \qquad C_8 = 1 \qquad$ etc.

B. Conversion of Binary to Decimal.

The simplest technique is to weight each binary coefficient with the power of two that corresponds to its binary position. Hence for binary numbers greater than unity the conversion is straightforward. For binary numbers less than unity, the following example illustrates the general technique.

0.00100011 corresponds to: $\frac{2^5 + 2^1 + 2^0}{2^8} = \frac{35}{256}$ which is approximately 0.137.

REFERENCES

1) Whittaker and Robinson, The Calculus of Observations, (1940), Blackie and Son Limited, London, Chapter VI.

2) Willers, Fr. A., Practical Analysis, Dover, (1948), Chapter IV.

3) Householder, A.S., Principles of Numerical Analysis, McGraw Hill, (1953), Chapter 3.

4) Brown, S. L., and Wheeler, L. L., "Mechanical Method for Graphical Solutions of Polynomials", Franklin Institute, Journal, (1941), v. 231 n.3, Pg. 223.

5) Marshall, B. O., "Electronic Isograph for Roots of Polynomials", J. Applied Physics, (1950), v. 21 n.4, Pg. 307.

6) Willoghby, E. O., Rose, G. A., Forte, W. G., "Analog Computers to Solve Polynomial Equations with Real Coefficients", Proceedings of the Conference on Automatic Computing Machines, Commonwealth Scientific and Industrial Research Organization, August, 1951.

7) Herr, D., "Two New Economical Computers for Design and Analysis of Servomechanisms, Networks, Amplifiers, and Analogous Dynamical Systems", American Society of Naval Engineers, Journal, v. 63 n.4, Pg. 950.

8) Bubb, F. W., "Circuit for Generating Polynomials and Finding Their Zeros", Institute of Radio Engineers, Proceedings, v. 39 n.12, Pg. 1556.

9) Loefgren, L., "Analog Computer for Roots of Algebraic Equations", Institute of Radio Engineers, Proceedings, (1953), v. 41, Pg. 907.

10) Morgan, M. L., "A Computer for Algebraic Functions of a Complex Variable, Ph.D. Thesis, (1954), California Institute of Technology.

11) Lin, S. N., "A Method of Successive Approximations of Evaluating the Real and Complex Roots of Cubic and Higher Order Equations", Journal of Mathematics and Physics, (1941), v. 20, pp 231.

12) Samuelson, P. A., "Iterative Computation of Complex Roots", Journal of Mathematics and Physics, (1949), v. 28, pp 259.

13) Luke, Y. L , and Ufford, D., "On the Roots of Algebraic Equations", Journal of Mathematics and Physics, (1951), v. 30, pp. 94.

14) Steinman, D. B., "Simple Formula Solves all Higher Degree Equations", Civil Engineering (N.Y.), (1951), v. 21, n.2, pp.44.

15) Oliver, F. W. J., "Evaluation of Zeros of High Degree Polynomials", Royal Society of London , Philosophical Transactions, (1952), v. 244 n.885, pp. 385.

16) Salzer, H. E., "Calculating Zeros of Polynomials by Method of Lucas", U. S. Bureau of Standards, Journal of Research, (1952), v. 49, n.2, pp. 133. (RP 2348).

17) Koenig, J. F., "On Zeros of Polynomials and Degree of Stability of Linear Systems", Journal of Applied Physics, (1953), v. 24, n.4, pp. 476.

18) Tasny-Tschiassny, L., "Location of Roots of Polynomial Equations by Repeated Evaluation of Linear Forms", (1953), Quarterly of Applied Mathematics, v.11, n.3, pp. 319.

19) Phister, M.,"Prevention, Detection, and Correction of Errors in Automatic Digital Computers,"(1953), Ph.D. Thesis, University of Cambridge, England.

20) Scarborough, A. D.,"The Use of Junction Transistors in Switching Circuits", (1955), Ph.D. Thesis, California Institute of Technology, Pasadena, California.

21) Nelson, E. C., "An Algebraic Theory for Use in Digital Computer Design", (1954), Transactions of the Institute of Radio Engineers, (1954), v. E.C.-3 n.3, pp. 12.

22) Williams, F. C., et al, "Universal High Speed Digital Computers, A Magnetic Store", Proceedings of the Institute of Electrical Engineers, (1952), Part 2, v. 99, pp. 94.

23) Householder, A. S., Principles of Numerical Analysis, McGraw-Hill, (1953), pp. 14.

24) Whittaker, E. T., and Watson, G. N., Modern Analysis, Cambridge University Press, (1950), pp. 120.

25) Guillemin, E.A., The Mathematics of Circuit Analysis, Wiley, (1949), pp. 298-302.