R-358

A DIGITAL CONTROL COMPUTER DEVELOPMENTAL MODEL 1B

by

R. L. Alonso, A. I. Green, H. E. Maurer,
and R. E. Oleksiak

April 1962

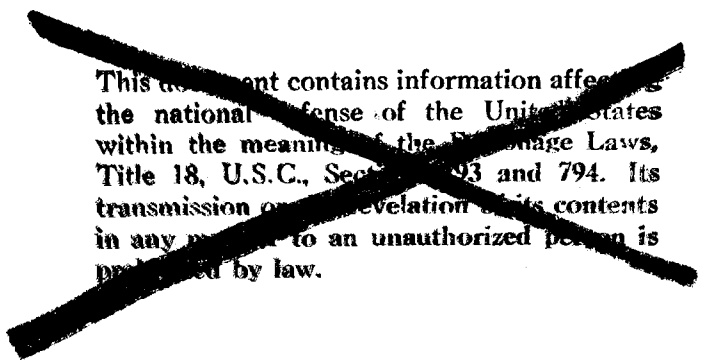
INSTRUMENTATION LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE 39, MASSACHUSETTS

Approved: Eldon C Hall
Assistant Director

Approved: R. Woodbury
Associate Director

ACKNOWLEDGMENT

This report was prepared under the auspices of DSR Project 52-156, sponsored 'by the Ballistic Systems Division of the Air Force Systems Command through USAF contract AF 04(647)-303.



The publication of this report does not constitute approval by the Air Force of the findings or the conclusions contained therein. It is published only for the exchange and stimulation of ideas.

R-358

A DIGITAL CONTROL COMPUTER DEVELOPMENTAL MODEL 1B

ABSTRACT

This report describes in detail a pilot model of a parallel, programable digital control computer. Magnetic cores are used as storage devices and extensively in the logic, thus reducing power consumption. Transistors are the active elements.

The computer has capabilities for automatic incrementing of counters and automatic program interruption, or mode changing. These are used to advantage in the control of a stepping-motor.

by Ramon L. Alonso
Alan I. Green
Harold E. Maurer
Robert E. Oleksiak

April 1962

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Page

LIST OF TABLES

INTRODUCTION

by R. L. Alonso

INTRODUCTION

Report R-276 [1] proposed a computer which the authors believed to have several advantages over others available at that time. The design arose from the exploitation of certain circuit techniques, namely the wired-in "core rope" memory and various core-transistor circuits such as the Priority Circuit, and was primarily logical in character.

Some months prior to the publication of R-276 it was thought desirable to build a small demonstration computer based on those principles, as opposed to continuing to build isolated circuits and subsystems. The resulting program list called for the construction of Mod 1A and later of Mod 1B.

Mod 1A (Fig. I-1) was to be small, as simple as possible, and devoid of any sophisticated input and output. At the same time it had to demonstrate the essential features of the R-276 design. The programs written for Mod 1A, although not trivial, were of a "make work" kind, The most ambitious of these programs was a set of interpretive subroutines by means of which programs could be written in pseudo-instructions more powerful than the basic four Mod 1A instructions. Various programs were then written in pseudo instructions; these programs were essentially memory checks, including sum checks, displays and score keeping of the number of successful executions of a basic loop.

The Mod 1A design called for 256 words'of storage, each word consisting of eleven bits plus a parity bit. Of these words, 208 were wired in programs of constants; of the remaining 48 words, 42 were erasable storage, 3 were input registers (connected to manual switches), and 3 were output registers, connected to lights. The order code set was composed of four instructions. Algebraic addition was a subroutine.

The specific goals for Mod 1A were: to perform the functions expected of it; to show that the design in which power consumption is proportional to speed is both sound and possible; to show that a 256 core rope is practical; to show that the sequence generator is a practical subsystem. Furthermore, Mod 1A was meant to bring forth any hidden problems, and to serve as a test for the many different circuits necessary for such a computer.

Mod 1B (Fig. I-2, I-3, 1-4) was based on Mod 1A, but with a much more complicated sequence generator capable of Automatic Counter Incrementing and Automatic Program Interruption. The basis for these features was the Priority Circuit, which was not used in Mod 1A. The Mod 1B memory was expanded to include 32 more erasable registers, and 224 more fixed ones. The additional tasks set for Mod 1B were the control of a stepping motor and shaft encoder system. Either angular rate or angular position were to be controlled while concurrently executing any one of the original Mod 1A programs,

The goals for Mod 1B were the same as those for Mod 1A, plus the additional one of demonstrating the Priority Circuit and the sequence generator system based on it. Moreover, Mod 1B was to be in a real time loop and in actual control of a stepper-motor, while receiving inputs from a shaft encoder. This fact made the whole system much more complicated than was the case for Mod 1A. Furthermore, at the time of conception of the construction program it was hoped that Mod 1B would also serve as a vehicle for testing various packaging schemes.

The preceeding paragraphs describe the planned program. Table I-1 shows the approximate dates on which several events actually took place. Mod 1A was abandoned before completion for a number of reasons having to do primarily with difficulties of construction and the inexperience of the constructors. The point at which Mod 1A was abandoned was just short of being able to execute programs. Construction had been essentially completed

and the major task of debugging had begun. Unfortunately, the assembly methods chosen were such that attempts to modify a circuit often resulted in more troubles than were present originally. Cold solder joints were a particularly frequent and frustrating source of troubles. At the time of abandonment, all of the various major subsystems had worked individually but not together. Mod 1A was abandoned because of a gradual shift in interest towards Mod 1B. This replacement was no doubt accelerated by the desire of the builders to start with a clean slate.

Mod 1B was a more successful enterprise, attaining all goals except the packaging one. Mod 1B was able to do everthing Mod 1A was meant to do some six months after construction was started; the construction and debugging of the additional parts of Mod 1B required an additional 6 months. The peripheral equipment, i.e., the stepper-motor, encoder, and auxiliary driving electronics had been designed and built some time prior to January 1961.

Mod 1B is composed of eight trays, two and a half of which contain circuitry additional to what was needed for Mod 1A. The entire computer required about one thousand transistors, and consumed about 25w when operating at full speed. This latter figure includes about 10w used for input, output, and display purposes.

Those registers which are common to both Mod 1A and Mod 1B are marked by the letter A following their octal address, which in this case is from *000* to *377*. All addresses from 400 to *777* have the letter B following them, and represent registers added to Mod 1A to form Mod 1B.

The rest of this report contains a detailed description of Mod 1B, the various programs, and the peripheral equipment. Some effort has been made to evaluate as well as describe the points which might have been done differently.

The labor of doing and building various tasks and devices described here was shared approximately as follows: the original

Mod 1A and Mod 1B concepts were proposed by J.H. Laning Jr. and R.L, Alonso*. These programs were later modified by H, Blair-Smith. Electrical design and construction of Mod 1A was done by R.E. Oleksiak and R.L. Alonso. The detailed planning of the Mod 1B system of computer and motor was done primarily by A.I. Green and H.E. Maurer. A.I. Green wrote the sequences and the detailed control program; both he and H.E. Maurer designed a first version of the Automatic Interrupt and Automatic Increment circuitry and shared the system analysis. The stepper-motor,encoder and auxiliary circuitry were the responsibility of H.E. Maurer. The exacting task of,writing the listings for the harness wiring was done by A.I. Green and T.E. Burke. As was the case for Mod 1A, the electrical design, construction, and debugging of Mod 1B was carried out by R.E. Oleksiak and R.L. Alonso. R.E. Oleksiak designed the Parity, Z, and S registers.

H. Blair-Smith wrote the IBM 650 programs which generated the rope wiring lists, and also wrote some further programs for Mod 1B.

*The first Mod 1A instruction sequences and test programs, including the interpretive program, were written by J.H. Laning Jr.

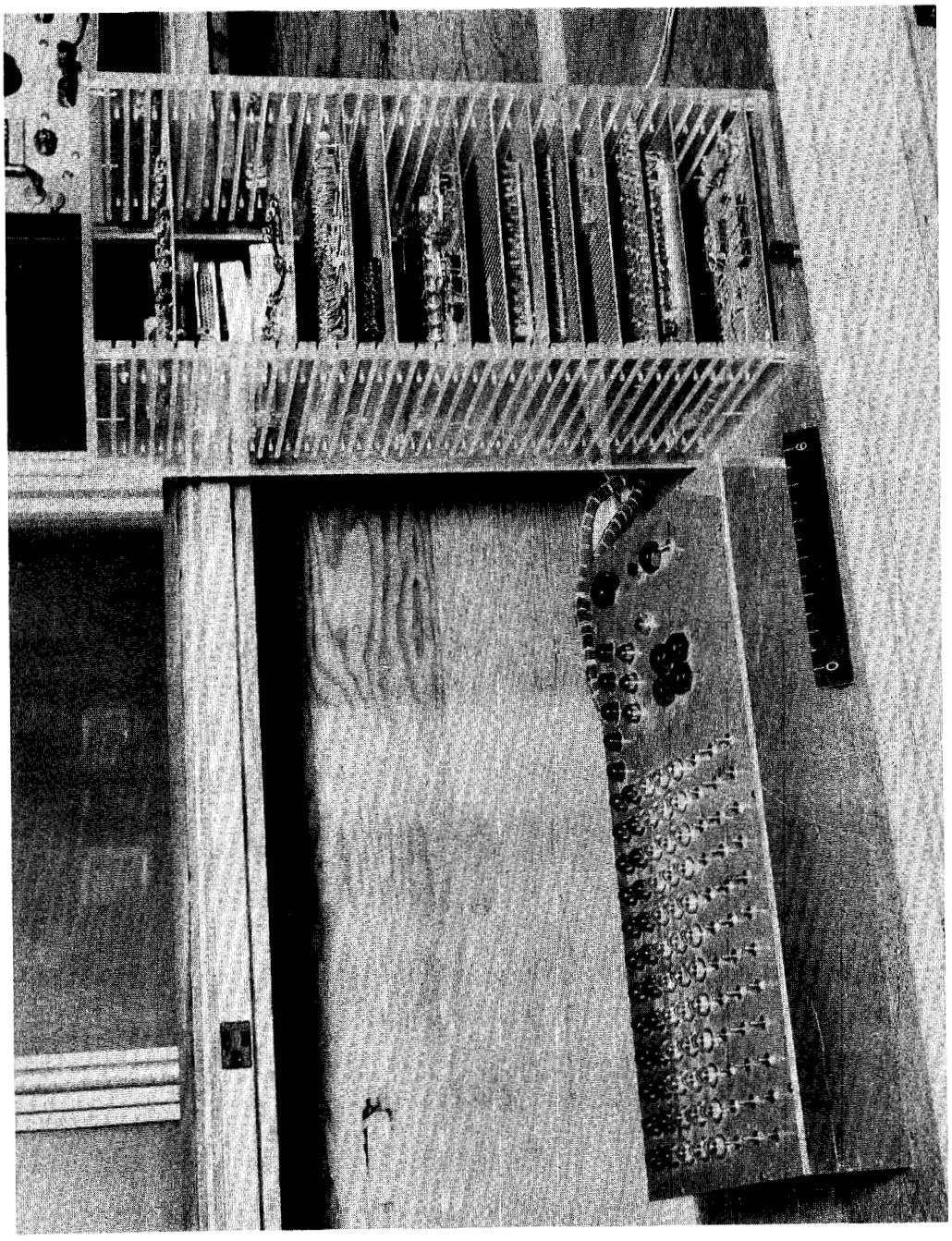


Fig. I-1 Mod 1A computer and control panel

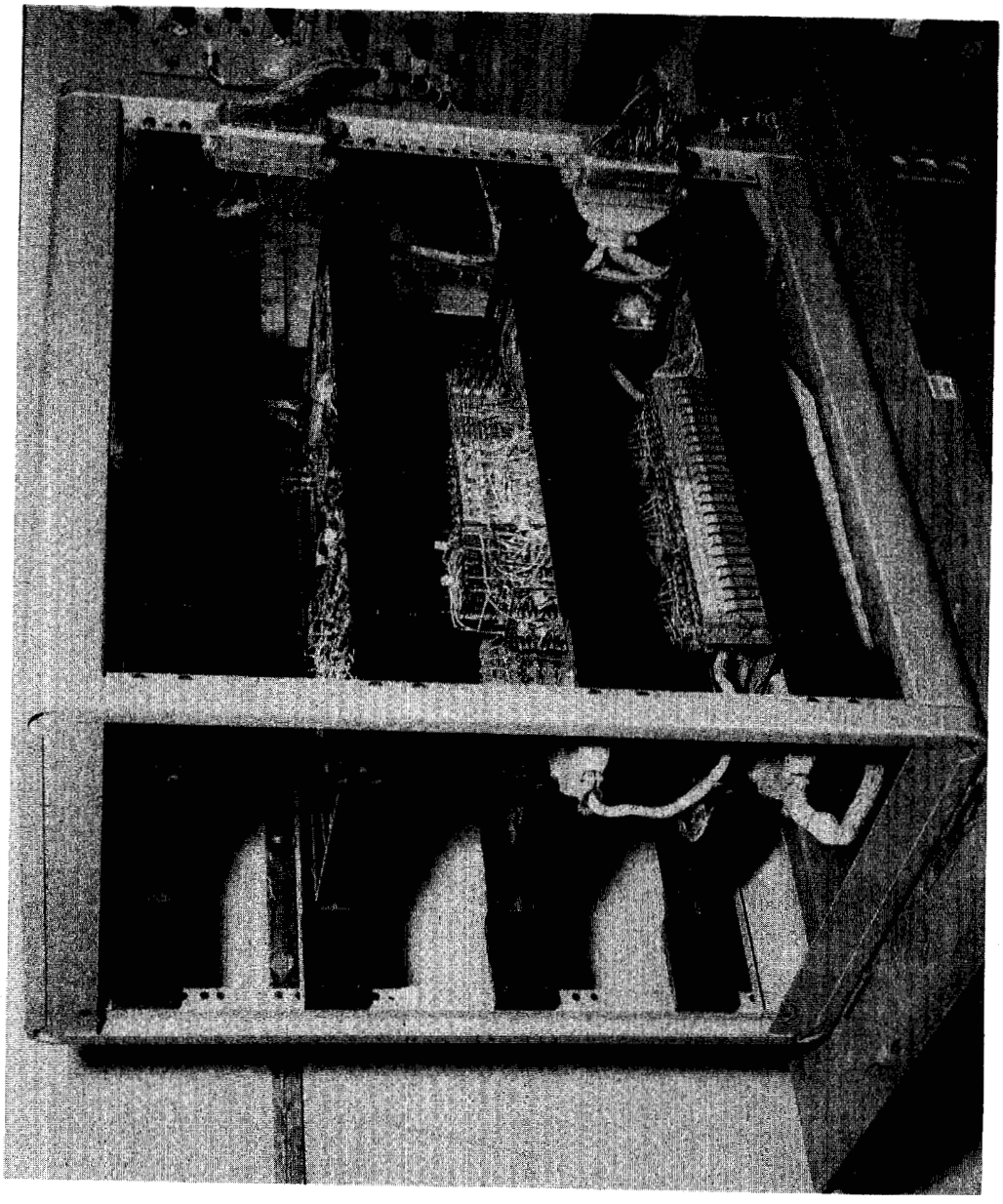


Fig. I-2 Mod 1B computer

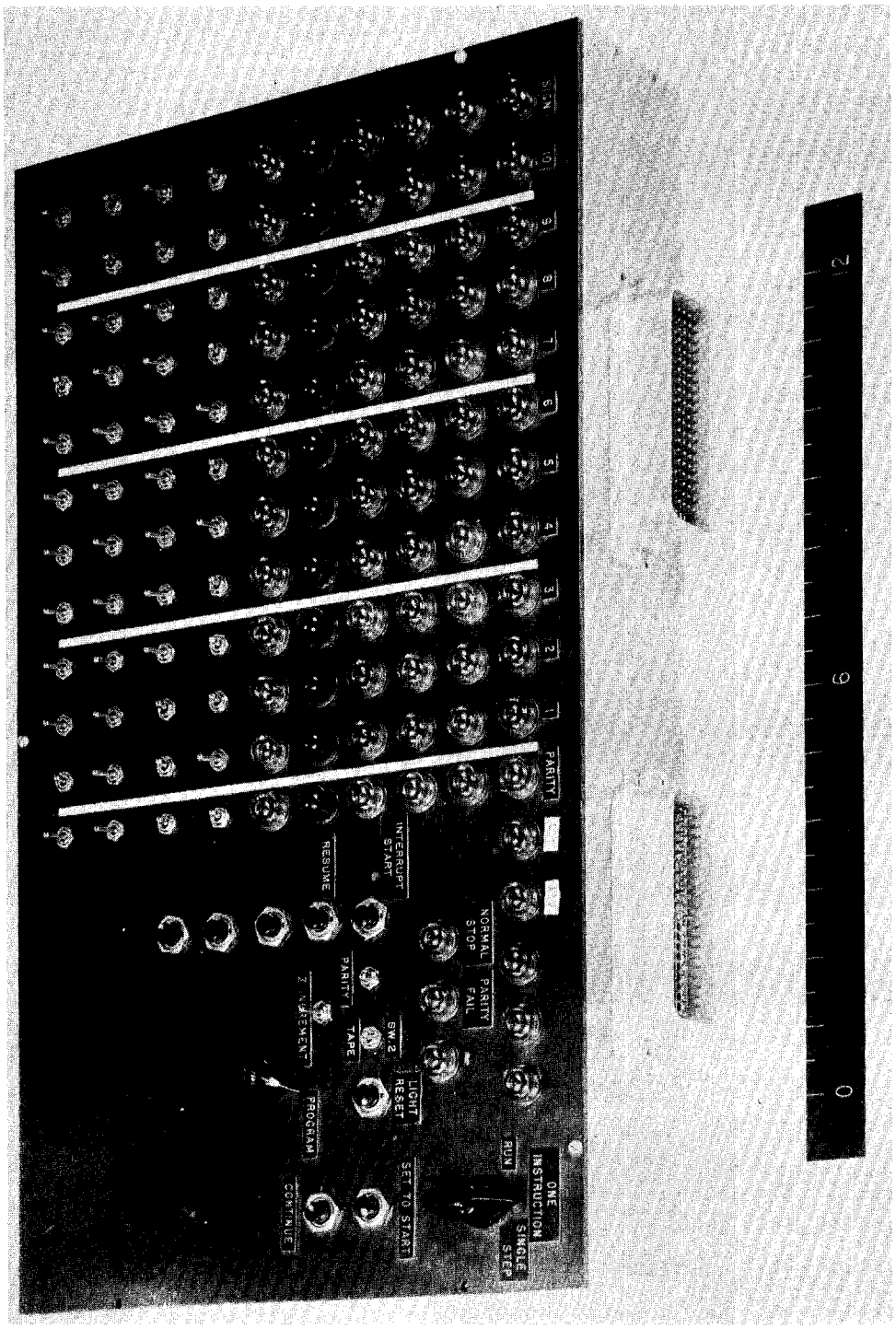


Fig. I-3 Mod 1B control panel

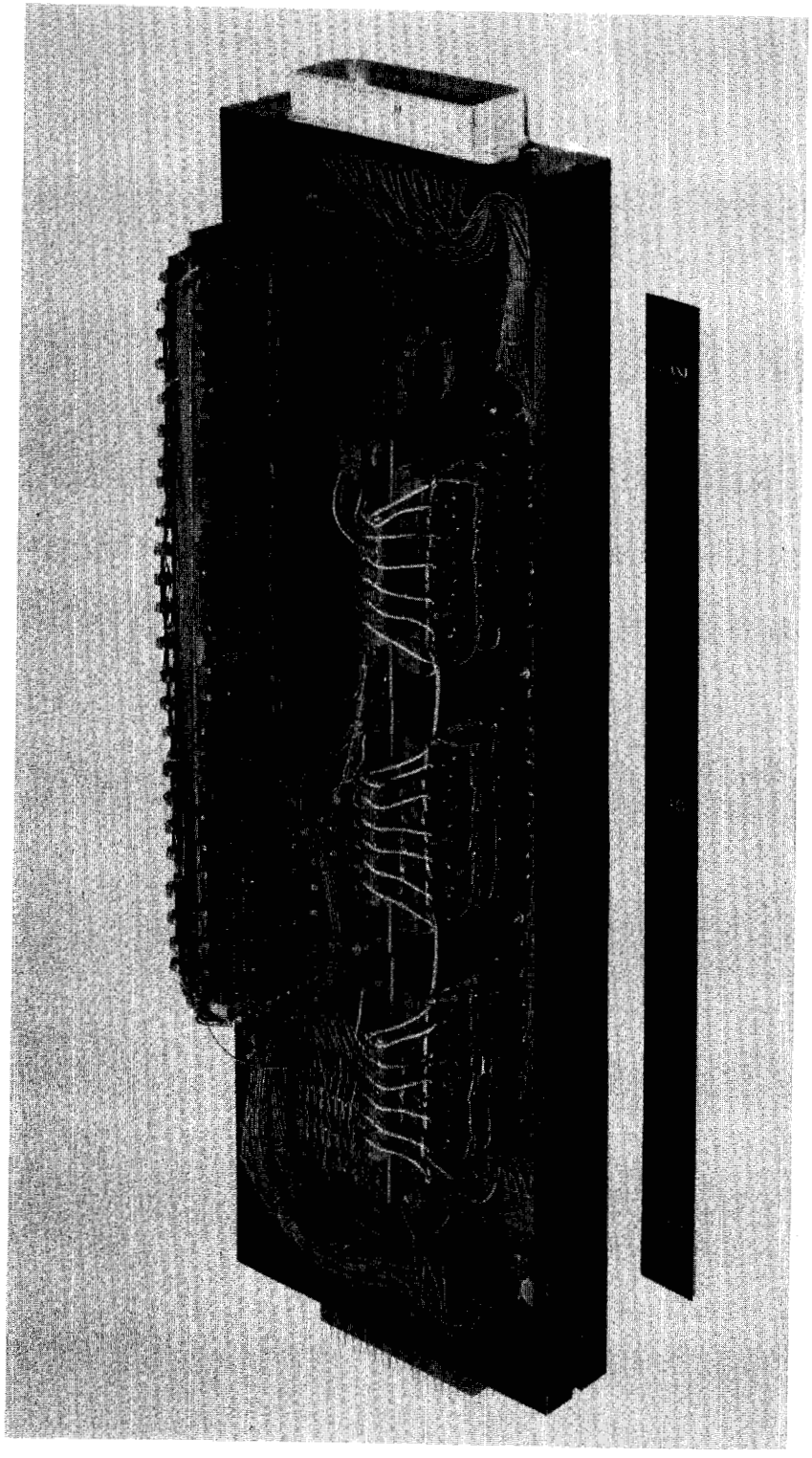


Fig. I-4 Tray 5 from Mod 1B

Table I-1 Approximate Dates of Events

Approximate Date	Event
July 1959	Mod 1A construction started.
December 1959	Planning for Mod 1B started.
July 1960	Construction of Mod 1B started.
August 1960	Major portion of Mod 1B planning completed.
September 1960	Mod 1A abandoned,
November 1960	Stepper auxiliary equipment complete.
February 1961	Part of Mod 1B common to Mod 1A completed and debugged (Mod 1B able to do everything Mod 1A was supposed to do).
July 1961	Construction of Mod 1B completed.
August 1961	Debugging of Mod 1B completed, Mod 1B performs all intended functions of Program Interruption, Automatic Counter Incrementing, and control of the stepper-motor.

CHAPTER 1
COMPUTER STRUCTURE

by R. L. Alonso

CHAPTER 1

COMPUTER STRUCTURE

1.1 Order Code

A convenient way to start is to consider the organization of the computer from the point of view of the programer (Fig. 1-1). All transactions are parallel word transfers to addressable storage from register A, or to register A from addressable storage. The four instructions which comprise the 1B order code are listed, and their net effects described, in Table 1-1.

The instruction NOR derives its name from the following: if address x contains X and address y contains Y (ie, $c(x) = X$,, and $c(y) = Y$) then assuming A contains all 1's, the execution of the two instructions

NOR x

NOR y

results in $c(A_i) = \overline{X_i \cdot Y_i}$,

where the subscript refers to the ith bit position or the ith bit. That is, register A now contains the bit by bit NOR of X and Y. The names of the other instructions are self explanatory.

The Table 1-1 order code is about as small and as weak as a useful order code can be. In the present case, the order code does not reveal certain special properties of some of the erasable registers (e.g., cycling right or left), which normally would be achieved by special instructions.

1.2 Word Structure

The word size of Mod 1B is 12 bits, named

S, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, P.

Bit P is the parity bit for both data and instruction words, and it is ONE if all other bits are ZERO, or if there is an even number of ONEs among the other eleven bits. Bit S is the sign bit in the case of numerical data. The number system is "ONE's complement", in which $-x = \overline{x}$.

The instruction word format is:

Bits S, 10:	Order code.
Bits 9-1:	Data address.
Bit P:	Parity,

1,3 Special Function Registers

There are four special function registers which are listed in Table 1-2. The cycling registers are *so* wired that a word written into them (e.g., by TSN CYCL) is shifted, with end-around-carry, by the act of writing into that register. A subsequent instruction NOR CYCL will read the cycled word out of CYCL. Thus the pair of instructions

```
TSN  CYCL,
NOR  CYCL,
```

has the net effect of causing the contents of A to be cycled left by one bit position as well as complemented.

The Shift R Register operates in much the same way, but with an important exception. The sign bit is written not only into bit position 10, as in the case of CYCR, but into the sign position as well; and bit 1, which in CYCR is written into the sign position, is lost. The shifting operation is equivalent to multiplication by 2^{-1}, whereas the cycling operation is not.

The parity bit is not disturbed in the case of cycling, and parity is preserved. This is not the case in the shifting operation, and special provisions must be made for this.

1.4 Input Registers

The input registers are listed in Table 1-3. The Switch

Registers are ordinary erasable registers which have the core windings, normally connected to the Write Busses, connected to external toggle switches, The electrical details are shown elsewhere in this report.

The toggle switches take the place of the A register in the execution of TSN x, if x is a Switch Register. Hence the pair of instructions

TSN SW1,

NOR SW1,

results in $c(A) = -c(SW1) = \overline{c(SW1)}$.

The three Counter Registers are again ordinary erasable registers which have additional means for being addressed by external circuitry. These registers act as accummulators of trains of pulses where the incrementing is done in such a way that there is no logical interference with the normal execution of instructions, even if these involve the active counter register. Each pulse counted requires 30 psec, *so* that counting at a rate of 1 kc slows down the normal program execution by $\frac{30 \times 10^3}{10^6} = .03$ of full speed. The details of operation of these Counter registers are given in other sections of this report. As far as the programer is concerned, counters are erasable registers which may be interrogated or written into as any other registers, and which have the further property of accummulating pulses from an external source. Writing into a counter is a way of presetting it *so* that it will overflow after *so* many pulses. The overflows, underflows, or end-around carries of these counters can be considered to be outputs.

1,5 Output Registers

Output registers are listed in Table 1-4, Each bit of an output register has associated with it a latch, which is a bistable device analogous to a flip-flop. The instruction TSN LT1 for example, turns to ON all those latches of register LT1 which correspond to positions in which A had a ONE. If c(A) = 0,

then all the latches of LT1 will be left in the OFF state, regardless of their previous states.

Latch outputs are d-c levels, suitable for lighting small incandescent bulbs or turning on relays. This is true for all output registers except the STOP and RESUME registers. In these two cases the output is not a level but a short pulse, lasting about 3 μsec. The functions of these registers will be described later.

It is not possible to interrogate an output register. The result of NOR **x,** where *x* is an output register, is,to read a word of all ZEROs from storage, where the parity bit is also ZERO. This will fail to pass the parity check.

1.6 Central Registers

Figure **1-2** shows the Central Registers of Mod 1B. The computer is divided into two halves, an α side and β side, "α Time" is defined as the time of clearing information out of the α side, sensing it, turning ON the appropriate Write Busses, and writing it somewhere in the β side; "βTime" is the converse. Times α and β alternate.

The various registers are operated by means of Control Pulses. These pulses are generated by a subsystem called "Sequence Generator", which is discussed later. Listed below are all the central registers, together with their pertinent control pulses and a pulses which clear the named (Z or S here) register to all ZEROs. Pulses WZ, WS are such that they cause writing into the named register. What is written into a register is whatever is present in the Write Busses.

The following is a list of central registers and their properties. The parenthesized (α) or (β) refers to which side of the machine they belong.

Z(α) Capable of adding 0, ± 1 to whatever is written into it. Control Pulses CLZ; WZ; ZA (for Z Add); ZS (for Z subtract).

WZ alone causes the number on the Write Busses to be written into Z unchanged, i. e., cores of Z are driven to ONE if they correspond to Write Busses at ONE, and left at ZERO otherwise. WZ and ZA together cause the number to be incremented by one. WZ and ZS together cause the number to be decremented by one. The Z register has circuits for detection of overflow, underflow and, end around carry and borrow. Of these, only overflow and end around carry are used in 1B.

A(α) Complementing register. Control pulses CLA, WA, W1A.

Pulse W1A sets A to all ONEs. WA results in cores of A being driven to ZERO if they correspond to Write Busses which are at ONE, and left undisturbed otherwise; This is the opposite action from the normal, in that WA results in clearing cores to ZERO rather than setting them to ONE.

S(α) Address register. Control Pulses CLS, WS.

WS results in writing into cores of S the information present on the first nine Write Busses. CLS clears the cores of S, and in *so* doing a single core of the 512 cores Rope is driven from ZERO to ONE. Every core in the Rope corresponds to a unique word cleared out of S. Both Fixed and Erasable are addressed by means of the cores in the core Rope.

T2(α) Storage for interrupt. Control Pulses CLT2, WT2.

When a program interruption is called for, the contents of Central Register C are preserved in T2. As will be repeated below, c(Z) and c(A) are preserved in T1 and T3 respectively. When the Resume Sequence

is called for, i.e., when the interrupted program is resumed, c(T1), c(T2), and c(T3) are returned to Z, C, and A, and the interrupted program is resumed.

B(β) Buffer register. Control Pulses CLB, WB.
Used for temporary storage of information coming from Erasable and Fixed Storage.

C(β) Auxiliary Buffer. Control Pulses CLC, WC

P(β) Parity Register. Control Pulses CLP, WP, TP (Test Parity).
P is a one bit register. WP results in writing ONE into P if the number of ONEs in the incoming word, which includes any previous parity bit, is even; ZERO otherwise. CLP clears the P register. If Control Pulse TP is also present, the computer will stop if the bit in P was not ZERO. An 11 bit word with its correct parity added to it always has an odd number of ONEs; e.g. the nurnbcr zero is represented by a word of 11 ZEROs, with the parity bit being ONE.

O.C.(β) Overflow register. Control Pulses CLOC.
O.C. is a one bit register. It is written into if and only if Z overflows as result of adding one to a number, a fact which is electrically detected at CLZ time. If O.C. has a ONE in it, then CLOC results in turning ON the Overflow Buss. This Buss is analogous to a Write Buss.

E.C.(β) End around carry register. Control Pulse CL E.C.
Identical to O.C., except that a ONE is written into E.C. if there is end around carry instead of overflow in Z.

T1(β) Storage for interrupt. Control Pulses CLT1, WT1.
Stores C(Z) during an interrupt.

T3(β) Storage for interrupt. Control Pulses CLT3, WT3.
Stores c(A) during an interrupt.

SQ(β) Sequence Decoder. Control Pulses WSQ, CLSQ.
WSQ results in writing into SQ the two highest order bits (bits *S* and 10) of the Write Busses. CLSQ select:; one of four sequences, in this case one of the four possible instructions. This action is similar to the selection of one out of 512 cores in a rope.

I. 7 Sequence Generator

The device for generating the various necessary sequences of control pulses is the Sequence Generator. A general description of this device and how it operates is given in R-276.

Mod 1B has a total of 10 sequences. Of these, two are called Engineering Sequences E1 and E2, and play no part in the normal machine operation. 'The other 8 are listed in Table 1-6. The leftmost column lists the 14 Time Pulses, TP1 through TP14. These time pulses are generated by a shift register (abbreviated SR) in which a single ONE circulates. The columns of Table 1-6 list the various sequences by showing which control pulses occur at which times.

Each of the four instructions has a sequence associated with it. The .instruction BNZN actually has two subsequences within it, as will be explained later. Sequences INT and RES refer to program interruption and resumption of interrupted program respectively. Sequences INC + and INC - refer to counter incrementing and decrementing. These last four sequences will be discussed in detail in Chapter 2.

To understand the instruction sequences it is necessary to know the initial conditions of several registers. Actually, it is easiest to start with the events of TP13 and TP14, which are the same for all instructions. At TP13 the Control Pulse CLE results in reading out of memory (Fixed or Erasable) the next instruction to be

executed. The next instruction to be executed is defined as being the word OPx, located at address L, The symbol OP refers to the two highest order bits of the word, and denotes any of the instruction codes; the symbol x refers to the address part of the instruction, The word OPx is written into registers B, C,, and **P,** and the OP part of the word is written into SQ. At the same time register **T2** is cleared to all ZEROs. Since T2 is on the β side its contents are not sensed. Clearing T2 is done for reasons having to do with a possible program interruption, and not for reasons having to do with normal program operation.

At TP14, pulses CLB, **WE** write back into memory the word read out at TP13, if it came from Erasable. If the word came from Fixed, the **WE** pulse has no effect.' Pulses CLP, TP test the parity of the word, and the computer will be stopped with an alarm if the parity check fails. Pulse WS results in writing the lower order 9 bits, i. e., the address **x,** of the word present on the Write Busses into the S register (the address register), The pulse CLSQ clears the **SQ** register, and in so doing generates an output pulse which selects the next sequence to be executed. This selection is accomplished by setting to a **ONE** a subset of cores in the Sequence Generator, where upon they are cleared to ZERO one by one by the TP's. As these cores are cleared to ZERO they generate outputs which are sensed and amplified, forming Control Pulses.

Simultaneously with the selection of the next sequence to **be** executed, a ONE is placed somewhere in the Shift Register, as shown in the bottom row of Table 1-6. Sequences are not all of the same length; in selecting OP=NOR or OP=TSN a ONE is placed in position 5 of the Shift Register, whereby the first Time Pulse of those sequences is TP5. Notice that all sequences end at TP14, but because of different lengths, begin at different Time **Pulses.**

After TP14, the initial conditions are: the next instruction: **has** been selected and its sequence set up. The address part *x* of

the instruction is in S, and the entire instruction OPx in C. The address L + 1 of the next instruction following the one about to be executed is in register Z.

Suppose now that the selected instruction is NOR, where x is the address part already in S. At TP5 S is cleared, thereby switching a selected rope core from ZERO to ONE. At TP6 register C is cleared. The instruction was originally placed in C in case an interrupt occurred, At TP7 the Control Pulse CLE resets the selected rope core to ZERO. If this core was part of Fixed Storage, then the information wired into it is sensed by the Rope sense amplifiers, and the proper Write Busses turned on, If the selected core was part of Erasable, then a transistor amplifier (one per register) generates a current which clears the selected Erasable register, The information being cleared out is likewise used for turning on the appropriate Write Busses. There is no possibility of conflict between the two kinds of storage because only one core in the selection rope will switch at this time. The information corning from storage is written into C and P. At TP8, C is cleared and the word written back into Erasable, if it came from there, and the previous transfer is checked by the pulses CLP and TP, which check parity, The word being cleared out of C is also written into register A, where each Write Buss at ONE drives to ZERO the appropriate core of A, If A had all ONEs In it, then c(A) will be the complement of what was read out of storage. The action of writing into A in this special way results in the desired operation $c(A) = c(\overline{X}) \bullet b(A)$, where b(A) is the "before" contents of A, and c(A) the contents of A after TP8.

The next six time pulses, starting from TP9, are concerned with fetching the next instruction to be executed. Just before this time c(Z) = L + 1 = address of the next instruction. TP9 results in clearing Z, and writing L + 1 into C. At TP10 the address L + 1 is written into address register S, and into the Z register. Since control pulse ZA is present at TP10, the net result of this

pulse time is to cause L + 2 to be written into Z, and L + 1 into S. After TPlO follows TP11, in which S is cleared, and which results in setting to ONE the core in the rope which corresponds to L + 1; the contents of L + 1 are the next instruction to be executed, and they are sensed at TP13. This, it will be recalled, is the Time Pulse with which this exposition was started; the cycle of operations has been completed.

The point at which an instruction starts or stops is a matter of somewhat arbitrary definition. In the present case the dividing time is defined by the time at which the next sequence is selected.

The sequences for instructions TSN and AOTSN follow the same pattern as the one for NOR. The sequence for BNZN, the branch instruction, has certain characteristics worth mentioning. As is shown in Table 1-6, there are really three sequences involved in this instruction. The first of these generates the control pulses for TPI and TP2. During these times the contents of A are examined to see if they are identically positive zero or not. Then, depending on the outcome of this examination, one of two further sequences is selected. If c(A) ≠ + 0, the next instruction is taken from L + 1, where c(Z) = L + 1 at the start of the Branch instruction. If c(A) ≡ + 0, the next instruction is taken from x, where x is the address part of BNZN x. Both subsequences generate the same control pulses for TP11, TP12, TP13, and TP14.

There are two additional sequences, shown in Table 1-7, which are not used in normal computer operation. These are "Engineering Sequences" the main purpose of which is to provide some meaningful repetitive activity of various parts of the computer. Sequence E1 allows the interrogation of any addresss, as specified by switch register (input register) SW1. After interrogation, c(SW2) is written into the selected register. The selected address is displayed in light register 1 (LT1, an output register), and the word read from memory is displayed in LT2.

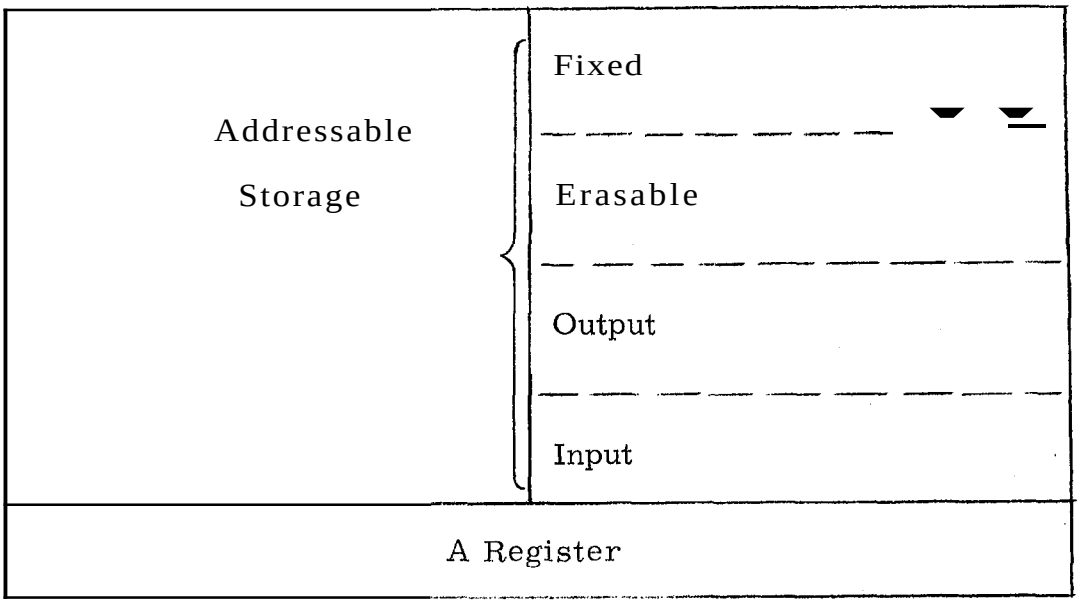
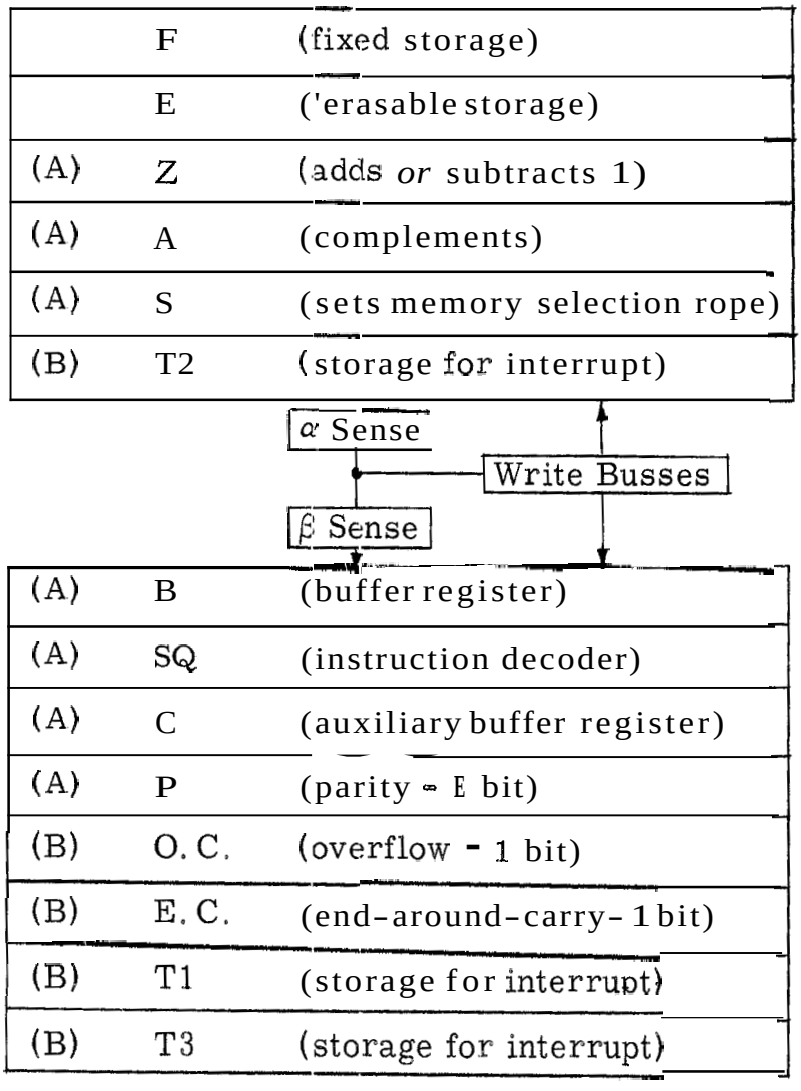


Fig. 1-1 Organization of the 1A part of 1B from the programer's view point.



Note: (A's refer to register of Mod 1A, and B's to those additional registers necessary to make up Mod 1B)

Fig. 1-2 Detailed structure of Mod 1B

Table 1-1 Mod 1B Order Code

00. NOR x:	$\overline{c(x)} \cap c(A) \rightarrow A$.. c(x) are preserved.
01. TSN **x:**	(Transfer c(A) to storage; normalize A) c(A) → x; 1 → A, i.e., leave A. with all ONES in it.
10. AOTSN x:	(Add ONE to c(A), transfer to storage; normalize A) c(A) ⊂ 1—x; 1— A.
11. BNZN x:	(Branch non-zero to x; normalize A) if c(A) ≠ + 0, take next instruction from x; 1→A. if c(A) ≡ + 0, go to next instruction; 1—A.

NOTES:

a. Symbol ∩ means the bit by bit AND of the two quantities.

b. The number representation is ONE's complement; hence it is necessary to distinguish between + 0 and -0.

c. The instruction word structure is:

Order code, bits S, 10;

Data Address, bits, 9, **8,** 7, 6, **5,** 4, **3, 2,** 1;

Parity, bit 0.

Table 1-2 Special Function Registers

SPECIAL FUNCTION REGISTERS

Octal Address	Name	Function	Bits
006 A	CYCL	Cycle word left one bit	ALL
007 A	CYCR	Cycle word right one bit	ALL
761 B	CYCLB	Same as CYCL	ALL
762 B	SHIFT R	Shift Right by one bit	ALL

Table 1-3 Input Registers

INPUT REGISTERS

Octal Address	Name	Function	Bits
000 A	SW 1	Switch register 1	ALL
001 A	SW 2	Tape input	ALL
010 A	SW 3	Switch register	ALL
760 B	SW 4	Switch register	ALL
765 B	PRESET	Counter	ALL
766 B	4PPS	Counter	ALL
767 B	PAC	Counter	ALL

Table 1-4 Output Registers

OUTPUT REGISTERS

Octal Address	Name	Function	Bits
003 A	LT 1	Light register	ALL
004 A	LT 2	Light register	ALL
005 A	LT 3	Light register	ALL
377 A	STOP	Stops computer	1
763 B	LT 4		ALL
764 B	RESUME	Note (a)	1
774 B	RATE MAGNITUDE	Sums outputs from rate generator	
775 B	4PPS ON/OFF		1
776 B	RATESIGN		2
777 B	5:1/1:1		2

Note (a) Addressing RESUME selects the Resume sequence which resumes the interrupted program.

Table 1-5 Mod 1B Storage Assignments

Octal Address	Function
000 A	IN Switch #1
001 A	IN Switch #2 or Tape Reader IN
002 A	IN Switch '#3
003 A	OUT Light #1
004 A	OUT Light #2
005 A	OUT Light #3
006 A	Cycle left
007 A	Cycle right
010 A	Normal erasable
011 A	Normal erasable
012 A	Normal erasable
013 A	Normal erasable
014 A	Normal erasable
015 A	Normal erasable
016 A	Normal erasable
017 A	Normal erasable
020 A	Fixed
to	
037 A	Fixed
040 A	Normal erasable
to	
077 A	Normal erasable

Table 1-5 (cont'd)

Octal Address	Function
100 A	Fixed
to	
367 A	Fixed
370 A	No registers (i.e., No sense lines through these cores.)
376 A	No registers (i.e., No sense lines through these cores.)
377 A	OUT Normal stop
400 B	Fixed
624 B	Fixed
625 B	Not used (fixed)
655 B	Not used (fixed)
656 B	Fixed
726 B	Fixed
727 B	Not used (fixed)
737 B	Not used (fixed)
740 B	Normal erasable
741. B	Normal erasable
742 B	Normal erasable
743 B	Normal erasable
744 B	Normal erasable
745 B	Normal erasable
746 B	Normal erasable
747 B	Normal erasable

Table 1-5 (cont'd)

Octal Address	Function
750 B	Normal erasable
751 B	Normal erasable
752 B	Normal erasable
753 B	Normal erasable
754 B	Normal erasable
755 B	Normal erasable
756 B	Normal erasable
757 B	Normal erasable
760 B	IN Switch #4
761 B	Cycle left
762 B	Shift Right
763 B	OUT Light #4
764 B	OUT Resume
765 B	CTR IN (Preset) Counter
766 B	CTR IN (4 PPS) Counter
767 B	CTR IN (PAC) Counter
770 B	Automatic Interrupt Address Registers
771 B	Automatic Interrupt Address Registers
772 B	Automatic Interrupt Address Registers
773 B	Automatic Interrupt Address Registers

Table 1-5 (cont'd)

Octal Address	Function	
774 B	OUT Output Latch Bits 6-10	RATE MAG Control
775 B	OUT Output Latch Bits S	4PPS ON/OFF Contol
776 B	OUT Output Latch Bits S, 10	RATE SIGN Control
777 B	OUT Output Latch Bits S, 10	5:1/ 1:L Control

CONFIDENTIAL

Table 1-6 Sequences Involved

TIME PULSES	NOR	TSN	AOTSN	BNZN		INT	RES	INC +	INC -	
TP1 α			CLA, WB, WP	CLA, WB, ZT						1 α
TP2 β			CLC	CLB, Test for ZERO		CLT1, CLT3, BI				2 β
TP3 α			CLZ, WC			CLT2				3 α
TP4 β			CLB, CLP WZ, ZA			CLB, CLP, TP WE	CLB, CLP, TP, WE	CLOC, CLEC	CLOC, CLEC	4 β
TP5 α				Zero	Non-Zero					5 α
	CLS	CLA, CLS, WB, WP	CLZ, WB, WP, CLS	CLS	CLZ	CLZ, WT1	CLZ	CLF/E, WB, WP	CLF/E, WB, WP	
TP6 β	CLC	CLC		CLC		CLC, WT2	CLC	CLB, WZ, ZA, CLP	CLB, WZ, ZS, CLP	6 β
TP7 α	CLF, WC WP	CLE	CLE	CLE WC, WP		CLF, WC WP, WB		CLZ, WB, WP	CLZ, WB, WP	7 α
TP8 β	CLC, CLP, TP, WE, WA	CLB, CLP, WE	CLB, CLP, WE	CLC, CLP, TP, WE		CLB, CLP, TP, WE	CLT3, WA	CLB, CLP, WE, CLEC, CLOC	CLB, CLP, WE, CLEC, CLOC	8 β
TP9 α	CLZ, WC	CLZ, WC	CLZ	CLZ, WC	CLZ	CLA, WT3, CLZ	CLA, WT3, CLZ	CLZ	CLZ	9 α
INCREMENT, if any, (Set Core 4 of Shift Register)										
TP10 β	CLC, WS, WZ, ZA	CLC, WS, WZ, ZA	CLC, WS, WZ, ZA	CLC, WS, WZ, ZA	CLC, WZ ZA	CLC, WS, WZ, ZA	CLT1, WZ, W1A			10 β
TP11 α	CLS	CLS	CLS	CLS		CLS				11 α
TP12 β		W1A	W1A	W1A		W1A	CLT3, WA			12 β
TP13 α	CLE, CLT2, WB, WC, WP, WSQ	SAME	SAME	SAME		CLE, WB, WC, WP	SAME			13 α
INTERRUPT, if any, (Set Core 2 of Shift Register)										
TP14 β	CLB, CLP, TP, WS, WE, CLSQ	SAME	SAME	SAME		SAME	SAME			14 β
TO Enter:	**Set Shift Register Core 5**	**Set Shift Register Core 5**	**Set Shift Register Core 1**	**Set Shift Register Core 1**		**Set Shift Register Core 2**	**Set Shift Register Core 4**	**Set Shift Register Core 4**	**Set Shift Register Core 4**	

Table 1-7 Test Sequences

	E_1	E_2
1 α	CL LT 1, LT2	CL LT1, LT2
2 β	W SW1, SW2	CLC, WZ, ZA
3cY	CL SW1, WB	CLZ, wc, WB
4 P	CLB, WS, WLTl	CLB, WS, WLT1
5cY	CLS, CLSW2, WC	CLS
6 β	-	-
7 α	CLF/E, WB, WP	CLF/E, WB, WP
8 β	CLC, WE	-
9 α	-	-
10 P	CLB, WE, CLP, TP, WLT2	CLB, WE, CLP, TP WLT2
11 α	-	-
12 P	Test Recycle. Set Shift Register Core 1	Test Recycle. Set Shift Register Core 1

Table 1-8 List of Control Pulses

(Those Control Pulses not mentioned previously have a note number next to their names.)

BI (Note 1)	CLZ
CLA	SI (Note 4)
CLB	Test for ZERO (Note 5)
CLC	WLT1 (Note 2)
CLEC	WLT2 (Note 2)
CLE	
CL LT1, LT2 (Note 2)	WSW1 (Note 3)
CLOC	WSW1, SW2 (Note 2)
CLP	
CLS	
CLSW1 (Note 3)	
CLSW2 (Note 3)	
CLT1	
CLT2	
CLT3	

Note 1 BI is used to inhi'bit further interrupts until the RESUME sequence is actuated.

Note 2. LT1 and LT2 are two output registers (octal addresses 003 and 004) with lights attached to the output latches. The single contrcl pulse CLLT1, LT2 clears both these registers, turing off all lights. WLTl allows whatever is on the Write Busses to be written into LT1, and similarly for LT2.

Note 3. SW1 and SW2 are two input registers (octal addresser; 000 and 001), with toggle switch connected to each core. The single control pulse WSW1, SW2 allows the information

Table 1-8 (continued)

	of the toggle switches to be written into SW1 and SW2. CLSW1 and CLSW2 are control pulses for clearing these registers.
Note 4.	SI inhibits the flow of setting and inhibiting currents for the core rope, *so* that register S may be cleared without altering the state of the rope.
Note 5.	Test for Zero and ZT. These control pulses are used for determining whether c(A) = 0 or not.

Table 1-9 Mod 1B Tray Contents

Tray 1	Tray 2	Tray 3	Tray 4	Tray 5	Tray 6	Tray 7	Tray 8
Clock	Main Sequence Generator	β to α Amplifiers	α to β Amplifiers	1A Erasable Registers	1B Erasable Registers (Control Registers, Counters)	1A Fixed Storage	Interrupt Circuit
Main Shift Register Time Pulse Amplifiers		Sequence Selection Logic	A Register *Z* T2 DA	1A Switch Registers (3)	Rate Generator	1B Fixed Storage	Encoder Logic
Increment Circuit	Test Sequence Generator	B Register *OC* EC *C* T1 T3 DB Parity Latches	Parity Timing Circuit	1A Light Registers (3) Memory Selection Rope	1B Light Register 1B Switch Register	Rope Sense Amplifiers S Register	
Engineering Sequence Electronics					Memory Selection Rope		
Panel Mode Electronics					1B Erasable Amplifiers		

CHAPTER 2
MOD 1B CONTROL OF STEPPER-MOTOR

by A.I. Green

CHAPTER 2

MOD 1B CONTROL OF STEPPER-MOTOR

2. 1 General Discussion of Systems Application of Mod 1B

The systems application of Mod 1B computer was devised to demonstrate the ability to solve several problems simultaneously by time-sharing the central computing equipment of a general purpose machine rather than by permanently committing specific physical sections of a special purpose computer to each problem. The computer was given the task of controlling a rotating device (a common requirement in space vehicles) while, at the same time, performing the details of a main program, such as the solution of guidance equations.

The rotating device, in this case, was a commercially available stepper-motor, a device which rotates a fixed, finite number of degrees for each input pulse, The choice was based upon considerations of availability, familiarity, and the existence of already proven input and output circuitry. Experiments with a continuously rotating, higher inertia motor are planned for a later stage in development.

Mod 1B has several. versatile and efficient facilities available which make it particularly well suited for performing several different tasks simultaneously. There is an "Automatic Increment" facility which can be used to accumulate input data increments without interfering with the main problem the computer is solving. This equipment can also be used internally whenever counting techniques are required,

An important capability of Mod 1B is that of mode changing at the program level, This "Automatic Interrupt" accomplishes a transfer of control to a specified sub-routine upon the occur-

rence of a pre-specified signal, When this Interrupt Sub-routine is completed, control is returned to the point in the main program at which the interruption took place and the solution of the main program continues, These Automatic Interrupt capabilities furnish the ability to solve several different classes of problems at essentially the same time, without consuming large amounts of time in the usual scanning procedures, Different problem modes may be assigned a predetermined priority of solution; thus, less pressing solutions may be deferred until the computer is relatively idle and can easily perform them.

The output equipment required for a task such as controlling a stepper-motor consists of a pulse rate source and several gating switches for applying the required rates to the proper equipment. These gating switches are operated by regular erasable registers.

In order to furnish the Increment and Interrupt capabilities, it was necessary to make certain modifications and additions to the Sequence Generator. An INTERRUPT and a RESUME Sequence are provided, The INTERRUPT Sequence stores the location of the instruction being performed when the Interrupt occurred; and performs the steps necessary to transfer control to the desired Interrupt Sub-routine. The RESUME Sequence uses the information stored during the INTERRUPT Sequence to return control to the point in the main program where the interruption occurred.

In order to perform the increment function, an INCREMENT Sequence is included in the Sequence Generator. This sequence selects the contents of a predetermined erasable register, adds one to it (or subtracts one from it), and returns the new result to that same erasable register.

2.2 The Automatic Increment Circuit

The heart of the Automatic Increment mechanism is the Priority Circuit. See Fig, 2-1. This is a chain of cores and transistors inserted between two successive cores of the Shift

Register in the Sequence Generator. In our case, it is placed between cores 9 and 10. (Core 9 furnishes Time Pulse 9; core 10, Time Pulse 10). If no cores in the Priority Chain are set to ONE, the Priority Chain acts like a short-circuit, and Time Pulse 10 simply follows Time Pulse 9, as if there were no Priority Chain present, However, if any of the Priority Chain cores is set to ONE, the advance current is diverted away from core 10 and Time Pulse P0 does not follow Time Pulse 9. Instead we perforrn an INCREMENT Sequence between Time Pulses 9 and 10 of the instruction that was being performed. In. particular, the advance current is diverted in such a way that it selects the INCREMENT Sequence; sets to ONE the fourth core of the Sequence Generator shift register; sets to ONE the clear core of the memory selection rope corresponding to a predetermined counter, (The Priority Chain core is also cleared to ZERO.) Thus a Time Pulse 4 follows a Time Pulse 9, and the INCREMENT Sequence is performed on Time Pulses 4 through 9. If there are no further Increments to be performed, the Sequence Generator next performs Time Pulse 10 of the original instruction, and continues until the instruction is completed, The counter is merely an erasable storage register the contents of which will be incremented (or decremented) by one whenever its particular core in the Increment Circuit Priority Chain is set to ONE.

The physical sequence of the cores in the Priority Chain determines the order in which the Increments are processed. The Priority Chain core closest to Shift Register Core 10 has the lowest priority; it will be processed only after all higher priority cores have been processed. By processing a core in the Increment Priority Chain, we mean incrementing (or decrementing) by one the counter register which is associated with that core in the Priority Chain, Notice that a Priority Chain core can not be set to ONE at Time Pulse 9, when the advance current passes through the Increment Priority Chain. Also, that the order in

which the Priority Chain cores were set to ONE does not determine the order in which they are processed. That is determined by the priority position of the core. When counters are used as internal timing devices, the timing pulse that is to be counted down is applied to the input set line of the Increment Priority core.

The counter registers have associated with them two special bits in addition to the regular 12 bits. These are called EC and OC and are used to indicate the fact that the last time the given counter was incremented, either an end-around carry or an overflow occurred. Very often these EC or OC signals are used to initiate an Automatic Interrupt, or change of program mode.

Since an Automatic Increment may be performed during any instruction, it is necessary to arrange the sequences for all instructions *so* that at the Time Pulse when an Increment may *occur*, all sequences leave certain pertinent central equipment in the same state. In particular, the Z register must be cleared to ZERO since it will be used to perform the increment; the B register must be cleared to ZERO since it will be used in transferring the counter contents to the Z register; both the clear and write cores of the memory selection rope must be cleared to ZERO since the counter register contents will be fetched and later returned to memory.

The INCREMENT Sequence is selected upon the action of the Increment Priority Chain. It is not decoded from the highest order two bits of an instruction word, as are normal programmable sequences. It fetches the contents of the counter associated with the Priority Chain core which set up the INCREMENT Sequence and reads this into the *Z* register, which performs the addition (or subtraction) of one and sets to ONE either the EC core or the OC core, if appropriate. It then reads the new result back into the counter register along with the EC and OC bits. The In-

crement Priority Chain cores for negative increments select a Negative Increment Sequence, which differs from the Positive Increment Sequence only in that it causes the Z register to subtract one from the number read into it. In this case, the EC records the fact that an end-around borrow has occurred; the OC, that an underflow has occurred.

2.3 The Automatic Interrupt Circuit

Many of the ideas developed in the section on the Automatic Increment Circuit are pertinent to the Automatic Interrupt Circuit. Once again, a Priority Chain is inserted between two cores of the shift register in the Sequence Generator. The performance of the Interrupt Circuit is slightly more complex than the Increment,

The Interrupt Priority Chain is inserted between cores 13 and 14 of the Sequence Generator Shift Register. See Fig. 2-2. If no cores in the Priority Chain are set to ONE, Time Pulse 14 follows Time Pulse 13. However, if any of the Priority Chain cores are set to ONE, the advance current is diverted away from core 14 in such a way that it selects the INTERRUPT Sequence; sets to ONE the first core of the Sequence Generator shift register; sets to ONE the clear core of the memory selection rope corresponding to a predetermined Interrupt Address Register. (The Priority Chain core is also cleared to ZERO,) Thus a Time Pulse 1 follows a Time Pulse 13, and the INTERRUPT Sequence is performed on Time Pulses 1 through 14. An interlock is also activated *so* that no further Interrupts are processed until a RESUME indication is received. (This is described below,) The Interrupt Address Register is simply an erasable register the contents of which will be interpreted as the location of the first instruction of a sub-routine which we wish to enter when the Interrupt is processed.

The Priority Chain cores cannot be set to ONE at Time Pulse 13, when the advance current passes through the Interrupt Priority Chain. The order in which the Interrupts are processed is determined by the priority position of the cores, not by the order

in which they are set.

Since an Automatic Interrupt may be performed during any programable instruction, it is necessary to arrange the sequences :for all instructions sothat, at the Time Pulse when an Interrupt may occur, all sequences leave certain pertinent central equipment in the same state. In particular, the instruction that would have been performed next, if there had been no Interrupt, must be in the B and C registers *so* that it can be stored in a special return register and be available after the Interrupt Subroutine is completed, It also must be read back into the memory, in case it came from an erasable register. To accomplish this read back, the write core of the memory selection rope must be set for the locatian from which the next instruction was obtained. The 1+ (location of the next instruction) must be in the Z register so that this can be stored for use after the Interrupt Sub-routine is completed, In order to fetch the contents of the Interrupt Address Register that will be selected by the Interrupt Circuit, the memory selection rope clear cores must be cleared to 'ZERO. The S register must be cleared to ZERO *so* that no subsequent "clear S" control pulses select unwanted memory locations.

The INTERRUPT Sequence is selected upon the action of the Interrupt Priority Chain. It is not decoded from the highest order two bits of an instruction word, as is the case with normal, programable instructions. It stores the location of the instruction that would have been performed next, if the Interrupt had not occurred; it stores this instruction; it also stores the contents of the A register before the interrupt occurred. It then fetches the contents of the Address Register associated with the Interrupt Priority Chain core which set up the INTERRUPT Sequence. It fetches the instruction in the location specified by the contents of the Interrupt Address Register and performs it. Thus we have transferred control to a sub-routine, and the Interrupt action is complete.

Each Interrupt Address Register is permanently associated

with a specific core in the Interrupt Priority Chain. However, these Address Registers are regular erasable registers, the contents of which can be modified by program action. Thus, we have the ability to change which sub-routine is entered when a given event sets an Interrupt Priority Chain core. This proves to be very useful for changing modes of solution of a problem.

In order to return to the instruction that was about to be performed when the Interrupt occurred, it is necessary to indicate the completion of the sub-routine to which we interrupted. Since all four possibilities arising from the two bit; operation code are already assigned, a RESUME indication can not be decoded from the highest order two bits of a word, Instead, we must assign a special address for RESUME and select this special instruction whenever reference is made to this address. This reference to the RESUME Address is placed at the end of all Interrupt Sub-routines. Whenever the clear core of the memory selection rope corresponding to this special Resume core is cleared to ZERO, the first core in the Interrupt Priority Chain is set to ONE. See Fig. 2-3. This core has the highest priority in the Interrupt Priority Chain, and processing it selects the RESUME Sequence. It is necessary to perform a RESUME Sequence after each Interrupt Sub-routine, so that there is no loss of information corresponding to the point in the main program from which the Interrupt caused departure. However, if another core in the Interrupt Priority Chain is waiting to be processed while the first Interrupt Sub-routine is being performed, placing the Resume core at the end of the Priority Chain would result in processing the second Interrupt before the Resume. Thus the information stored at the time of the first Interrupt would be lost, To prevent this, the Resume core is assigned the first position in the Priority Chain.

The RESUME Sequence places in the A register the contents it had before the Interrupt occurred. It fetches the instruction that would have been. performed next, if the Interrupt had not occurred, and arranges for it to be performed immediately after

the RESUME Sequence. It also restores the contents of *Z* register to correspond to this instruction. Thus, we have transferred control back to the point in the main program from which the Interrupt caused departure.

From the above discussion it should be obvious that the Interrupt facility is quite different from a programed branch instruction in that it can occur at any point in the main program. In programed branches, the programer specifies the point in the main program at which the transfer of control occurs. In the case of the Interrupt transfer of control, the programer specifies the point to which the transfer of control occurs, but can not specify the point in the main program from which control is transferred, .Another important point is that an Automatic Interrupt can not be processed while an Interrupt Sub-routine is being performed. .Any Interrupt Priority cores that, are set while another Interrupt Sub-routine is being performed hold their information and are processed when the sub-routine is completed. Thus no Interrupt information is lost.

2.4 Output Registers

There are several types of outputs required for Mod 1B, but they all employ a latch type of switch (memory type switch), 'The latch is a switch which maintains an output corresponding to the last input applied. This is useful for the two main classes of outputs. One is signalling an event or representing a bit of data by turning a light off or on. Another is applying a pulse rate to some external equipment (such as the stepper-motor drive logic), In the first case, the light is simply made to follow the state of the latch. In the second case, the latch *is* used to apply a ground (or open circuit) to the emitters of transistors in a pulse rate generator circuit, When the latch is left on (short circuit), the pulse rate is gated on and continues to appear until the latch, is turned off. Thus the latch is used to "key in" rates from free running rate generators.

The "key in" latch can be controlled by the action of a core. The switching action of the core persists for only a few microseconds, but the latch is used to provide an output long after the controlling core has switched. In particular, the core which controls the latch can be a member of a regular erasable storage register, with only minor modifications. See Fig. 2-4. In this way, the output latches are controlled by the simple, programable action of placing a ONE or ZERO into a bit of an addressable erasable register.

2.5 Stepper-Motor Control System

The system for the control of position or rate of rotation of a stepper-motor is the Mod 1B computer together with the necessary speed reducer, stepper-motor drive logic, and shaft encoder.

The equipment external to Mod 1B is shown in Fig. 2-5. The stepper-motor rotates 45^{O} for each pulse applied to the stepper-motor drive logic. It can be pulsed to rotate in either a positive or negative direction. The heart of the Drive Logic consists of an eight state counter. The position of the stepper-motor rotor is made to correspond to the state of this counter through necessary logic and power amplifiers.

The 20:1 speed reducer is a simple, commercially available gear reduction device. The optical encoder is a device which furnishes a pulse rate proportional to the speed of rotation of the stepper-motor. It does *so* by observing the number of alternating opaque and transparent sectors on the encoder disk which passes before the photo-electric sensor, The output of the encoder is actually two different square wave patterns, observed by separate sensors located 90 mechanical degrees apart. After suitable phase sensing of the two square waveforms, it is possible to have 2048 pulses per revolution of the encoder shaft, plus direction information. The stepper-motor, drive logic, speed reducer, and optical encoder are discussed more fully in Chapter 4.

.An important quantity is the gain of the equipment shown in

Fig. 2-5, that is the gain of the drive logic, stepper-motor, speed reducer, and encoder. For each pulse applied to the drive logic, the stepper-motor rotates 45". The speed reducer output rotates 2.25^{o}. The optical encoder puts out $(\frac{2.25}{360})$ (2048)pulses. Thus, the overall "pulse gain" is 12.8 pulses out of the encoder for each pulse applied to the Drive Logic,

An output register of the kind described in Section 2.4 and a counter register are necessary for computer control of the equipment of Fig, 2-5. The output register controls latches which apply pulse rates from an internal pulse rate source to the Stepper-motor Drive Logic. Thus, commanding the rotation of the stepper-motor is accomplished by reading the proper word into the output register. This is a program action. See Fig. 2-6.

For computer monitoring of the rotation of the stepper-motor, it is necessary to accumulate the pulses which appear at the encoder output. This is clone by feeding these pulses to a core in the Increment Priority Chain which is associated with a counter which will then keep the computer informed of the net stepper-motor rotation. Since the motor can rotate in either direction, we have both a positive Increment core and a negative Increment core. Both are associated with the same counter. See Fig. 2-6.

2.6 Rate Control of Stepper-Motor

The configuration shown in Fig. 2-7 is used to control the rate of rotation of the stepper-motor. Since the stepper-motor actually rotates a finite increment (45^{o}) for each pulse applied to it, it is not possible to obtain a smooth rate of rotation. However, this is closely approximated.

Pulse rates available for driving the stepper-motor are: 12 pps, 24 pps, 48 pps, 96 pps, and 192 pps, or any combinations of these rates. They are derived by counting the basic clock frequency of 100,*000* pps down successively by two. The output register which selects combinations of these pulse rates is called the Rate Magnitude Control Register. Bit 10 turns on 192 pps;

bit 6, 12 pps. The Rate Sign Control Register applies the pulse rate selected by the Rate Magnitude Register to either the positive or negative input of the Stepper-Motor Drive Logic. Bit S turns on positive; bit 10, negative.

To measure the rate at which the motor is actually rotating, the output of the encoder is accumulated in the P_{AC} Counter, by means of the Automatic Increment Circuit. The contents of the P_{AC} Counter is examined approximately 5 times per second, and compared with the theoretically correct value, The difference is displayed in a light register on the console. The "5 pps" Counter is used to count 24 pulses per second down by 5. Each time this counter overflows, the main program is interrupted and the 5 pps Interrupt Sub-routine is entered, This subroutine clears the P_{AC} Counter, writes an initial condition into the 5 pps Counter, computes the difference between the command and actual number of pulses out of the encoder, and displays this difference is the light register. (The maximum uncertainty in the 5/24 second timing is 125 psec).

The difference between the command and the actual number of pulses out of the encoder is computed for display purposes only. The difference is not used to re-adjust the pulse rate applied to the stepper-motor. Such fine control of rate seemed unworthy of the effort in view sf the inherent lack of smooth rotation in a device such as a stepper-motor. Rate feedback will be used in later experiments with a smoothly rotating device.

The coding of the command number in the Rate Magnitude Control Register was chosen *so* that the difference could be formed by direct subtraction. For example, the command number.for applying 12 pulses per second to the Drive Logic is a one in bit 6 of the Rate Magnitude Control Register. This should cause (12) (12. *8)* pulses per second to be accumulated in the P_{AC} Counter. But the P_{AC} Counter is examined 5 times per second (actually once every 5/24 sec). Thus there should be (12) (12. 8) (5/24) or 32 pulses accumulated in the P_{AC} Counter. This is exactly the

binary number that a one in bit 6 of the Rate Magnitude Register represents. Thus, if the stepper-motor is rotating at exactly the proper rate, the computed difference between the command and actual number of pulses accumulated in the P_{AC} Counter should be zero.

In order to set up the rate problem, or to set up the position problem, or merely to change the command number for either problem, a START INTERRUPT button is supplied on the console. As is shown in Fig. 2-7, the START INT has the highest priority of the Interrupts in the Interrupt Priority Chain, This Interrupt enters a sub-routine which transfers a new commandnumber frorn an input Switch Register into the Rate Magnitude Control Register (if the rate problem is selected). The START Interrupt Sub-routine also examines bit 1 of the input register. If it is a ONE, the rate problem is selected; if a ZERO, position. If the rate problem is selected, the 5 pps Control Register is turned on, *so* that the 5 pps Counter is activated.

2.7 Position Control of Stepper-Motor

The configuration shown in Fig. 2-8 is used to control the position of the stepper-motor. As described previously, the START Interrupt Sub-routine examines the least significant digit of the input Switch Register. If this is a ZERO, the position problem is selected.

To rotate the stepper-motor to the desired position, 96 pps will be used in all cases. The actual number of pulses out of the encoder will be accumulated in the P_{AC} Counter. We desire to code the command number to be placed in the input Switch Register *so* as to minimize computations within the computer, There is still a pulse gain of 12.**8** from the drive logic input to the encoder output. If we scale the pulses out of the encoder by a factor of five, . we will have an effective pulse gain of 64. This is convenient, for it is both a whole number and also a multiple of two. If the command number in the input Switch Register is code? to be 64 times

the binary number of pulses desired, the Switch Register contents and the P_{AC} Counter contents are directly comparable.

Thus Fig. 2-8 shows a gain of five from the output of the encoder to the P_{AC} Counter. This is accomplished by using an output pulse from the encoder to set five cores in the Increment Priority Chain. This increments the P_{AC} Counter five times for each encoder pulse, The 5:1 gain is turned on at the beginning of a position problem by means of the 5:1, 1:1 Control Register, The overflow from the P_{AC} Counter selects an interrupt sub-routine which stops the computer, for such an overflow indicates that the capacity of the P_{AC} Counter has been exceeded.

When the position problem is selected, 96 pps is applied to the Drive Logic and is not removed until the actual rotation corresponds to the command number. The criterion is as follows:

if P_D is + form $N = \dfrac{64\,\overline{P}_D + \overline{32} + P_{AC}}{64}$

if P_D is - form $N = \dfrac{64\,\overline{P}_D + 32 + P_{AC}}{64}$

$64\,\overline{P}_D$ is 64 times the complement of the binary number of command pulses desired, The 32 or $\overline{32}$ is a bias number to minimize round-off. P_{AC} is the contents of the P_{AC} Counter (which has the 12.8 and 5 factors already included,) If N is +, place $1 + \overline{N}$ into the Preset Counter and resume to the main program. If N is -, place 1+ N into the Preset Counter and resume. Thus the position problem is set up and the main program solution proceeds while the Preset Counter keeps track of the number of pulses sent to the Motor Drive Logic. When the Preset Counter generates an end-araund carry, an interrupt is processed which recomputes N and completes the fine positioning of the stepper-motor. Thus the computer sets up a rough positioning situation and returns its attention to the solution of the main program. When the rough positioning is completed, the computer enters the

final fine positioning mode. This last step is important to correct for overshoot, sticking of the shaft, or the addition of an outside disturbance to the shaft system. Only when N is found to be zero does the position problem end. Of course, repeated closures of the START Interrupt switch will result in additional positioning, since the P_{AC} Counter is cleared at the beginning of a START Interrupt.

Fig. 2-9 shows a flow diagram of both the rate and position programs.

2.8 Main Programs

There is included in the Mod 1B program rope a main exercise routine which continually adds one to a chain of three registers. The overflows from the first register are the inputs to the second, etc. The third register overflows once in several weeks. This routine is found very useful in trouble-shooting, for the contents of these three registers furnishes a good record of when a failure occurred.

2.9 Results - Rate Problem

Table 2-1 shows the results of data taken on the rate control of the stepper-motor. The computer was set up to control the stepper-motor at each of the five rates, and 25 readings of the displayed difference were taken. The average of the absolute value of the displayed difference is computed. This is tabulated with the calculated number of pulses expected in the P_{AC} Counter, if there were no errors.

There are two sources of error that must be considered. First, previous measurements have shown that backlash in the shaft coupling can account for an uncertainty, of 2 pulses out of the encoder. Second, the stepper-motor does not rotate exactly 45^{o} for each pulse applied to it. It does, however, rotate 360^{o} for 8 input pulses, but the motor available did not divide the 360^{o}

into 8 equal segments.

At an applied pulse rate of 12 pps, the rotor accelerates to align with the new position of the flux vector, overshoots this position, and performs a damped oscillation about this position. Before this damped oscillation has ended, the flux vector advances to the next position, and the rotor accelerates to align with it. During the acceleration, only positive pulses are observed out of the encoder. During the damped oscillation, both positive and negative outputs are present.

From Fig. 2-10 we see that during observation period 1, there are 3 pulses applied to the stepper-motor; but during period 2, only 2. The end of observation period 1 (the " 5 pps" pulse) occurs somewhere near the end of the burst of positive pulses out of the encoder during the acceleration of the rotor. Thus the count of encoder pulses during period 1 will be somewhat high; and during period 2, somewhat low, Ideally, we expect 12.8 pulses out for each command pulse. The command number with which the contents of the P_{AC} Computer are compared is based on an average of 2.5 command pulses per observation period. The worst case would occur when all the output pulses from the 3 command pulses are included in observation period 1, and output pulses from only 2 command pulses are included in observation period 2. In this case, the magnitude of the error would be 6,4 pulses. Thus the error of 5.2 pulses is quite reasonable.

At an applied pulse rate of 48 pps, almost no negative pulses are observed out of the encoder, indicating that the damped oscillation following an overshoot no longer occurs. However, resonance phenomena contribute to the error. Fig. 2-11 shows the cycles of high and low pulse densities that appear at the output of the encoder. During one period of observation, there are 2.5 cycles of the maximum and minimum pulse densities. A period of maximum density corresponds to too large a rotation of the rotor; one of minimum density, too small a rotation, The period of observation begins at

roughly 45^{o} after the maximum pulse density; it ends at roughly 45^{o} after a minimum. Because of this phasing of the period of observation, the maximum and minimum periods do not balance each other very closely. Thus one period of observation will have too many counts; another, too few.

At applied pulse rates above 100 pps, a nearly uniform pulse density is observed out of the encoder, indicating a nearly uniform rate of rotation of the rotor. Thus, the error is seen to be quite small at these higher driving frequencies. At 192 pps, the error is *so* small that it is insignificant.

An important result obtained from the rate control of the stepper-motor is the amount of slowdown of the main program. To determine this, the counting routine described in Section 2.8 was run up to a predetermined value with no rate control operating. This took 220 sec. The procedure was repeated with the computer controlling the motor at 196 pps. The main program is slowed up due to incrementing the P_{AC} Counter to count the pulses out of the encoder, and also due to interrupting 5 times per second to compute and display the difference. It now took 245 sec for the main program to reach the same value as before. Thus there is a slowdown of 11.4% (An estimate based on a knowledge of Interrupt and Increment times predicted a slowdown of 11.1%).

2.10 Results - Position Problem

A slight modification of the procedure outlined in Section 2.7 was necessary to carry out correct position control of the stepper-motor. It was necessary to insert a time delay between the end-around carry of the Preset Counter and the computation of N, in order to allow the stepper-motor to respond to the last command pulse applied to it before examining the contents of the P_{AC} Counter. The 5 pps Counter was used since it was convenient. Thus, a 1/5 sec delay was inserted.

The results are tabulated in Table 2-2, Four different ex.-

periments are shown. First, the position problem was run eight successive times with a command number corresponding to a single negative increment. The contents of the P_{AC} Counter were noted after each positioning, The same procedure was repeated with a command of a single positive increment. The shaft pointer had returned to the same position from which it had started, as close as the eye could determine.

Next, the command number was changed to 5 negative increments, and eight positionings were performed, This was repeated with a command number corresponding to 5 positive increments. The pointer returned to about within 2^{o} of its original position. With 10 increments command for each positioning, the pointer returned very nearly to its original position. Similarly close results were obtained for 15 increments. Of course, the pointer dial was scaled at only 10^{o} intervals, but the results were still quite good in view of the fact that backlash in the coupling causes an uncertainty of 2 pulses, and unequal field strengths in the stator cause unequal displacements.

2, 11 Tape Input

As part of the input equipment, an eight channel punched paper tape reader is included. Mod 1B has a program under the control of which data is read from the paper tape, assembled into computer words, and placed in the designated erasable storage. Erasable storage is loaded from the tape in blocks of 32 words,

2. 12 Future Applications

At present there are plans for modifying Mod 1B to control both the rate and position of a high inertia, continuously rotating device, Methods are being investigated for controlling a star tracker or similar optical device. Details are found in Chapter 4.

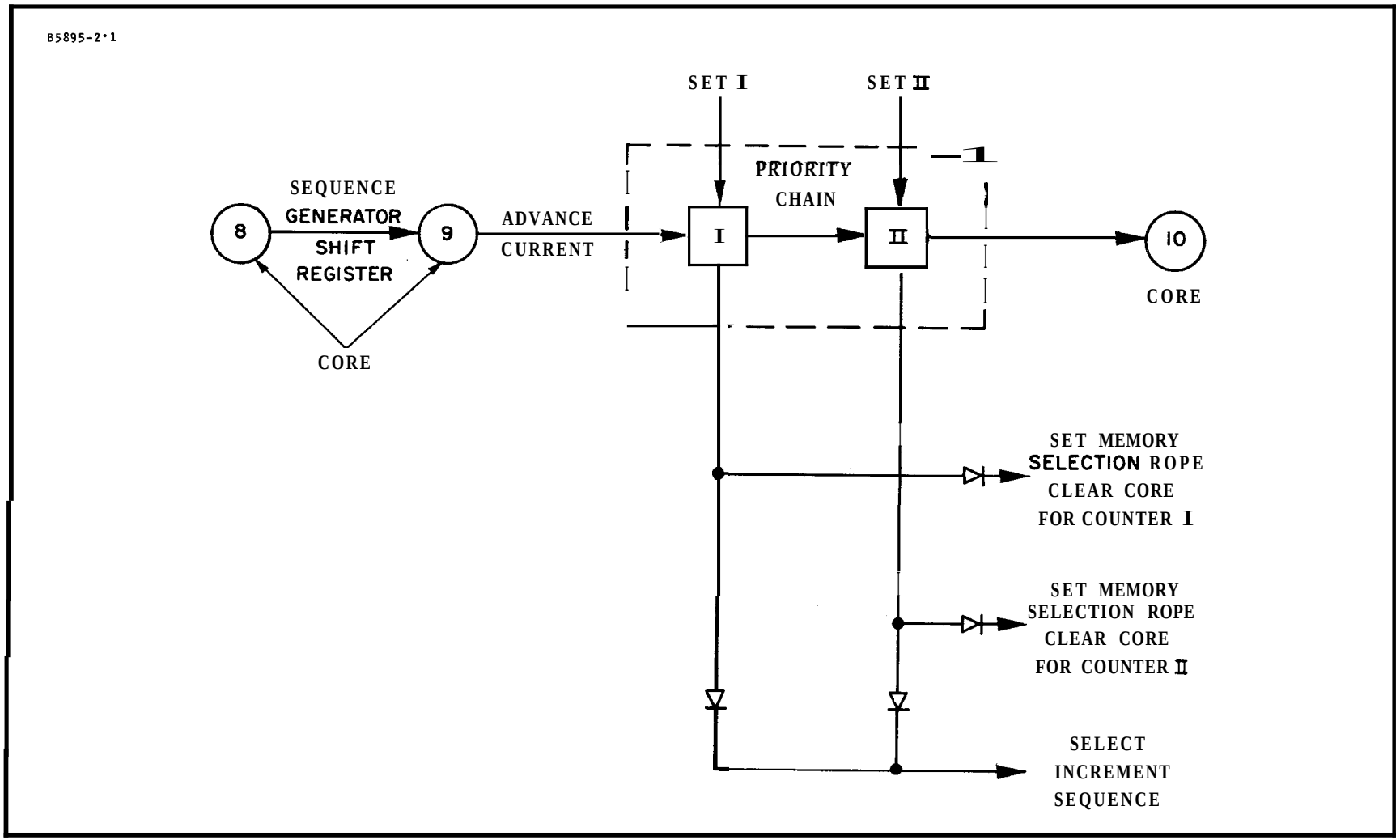


Fig. 2–1 Automatic increment priority chain

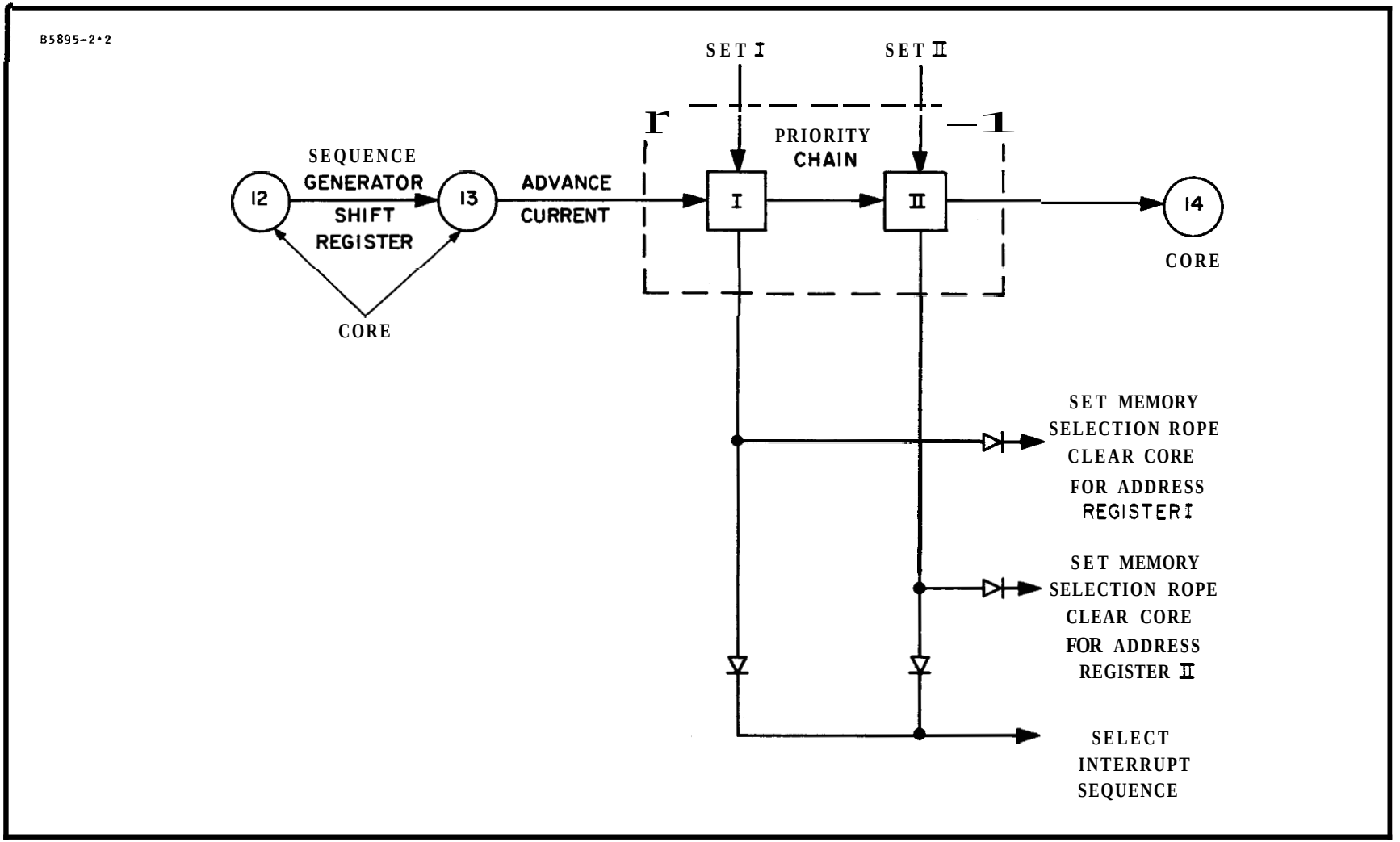


Fig. 2-2 Automatic interrupt priority chain

CONFIDENTIAL

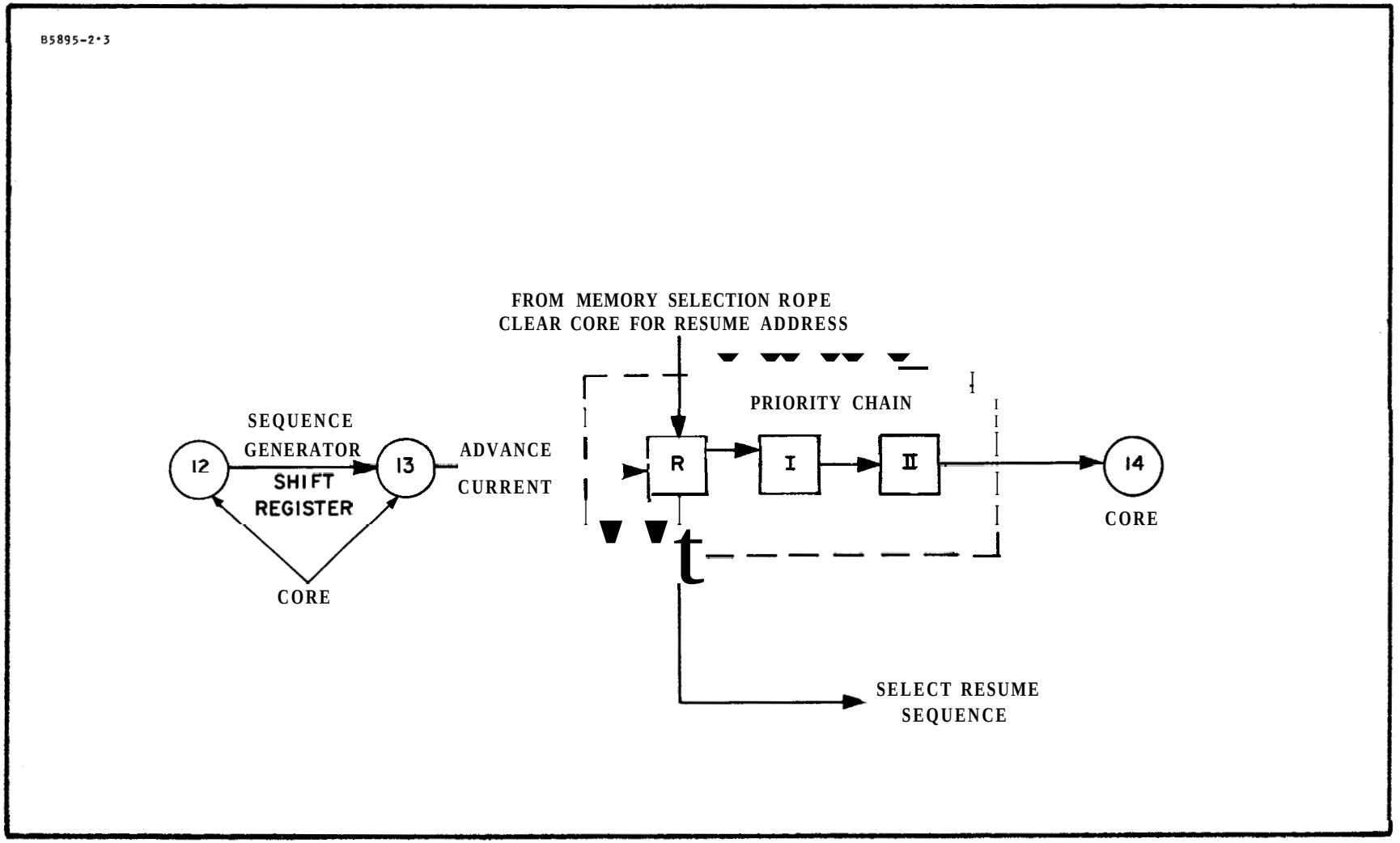


Fig. 2-3 Automatic interrupt priority chain showing RESUME core

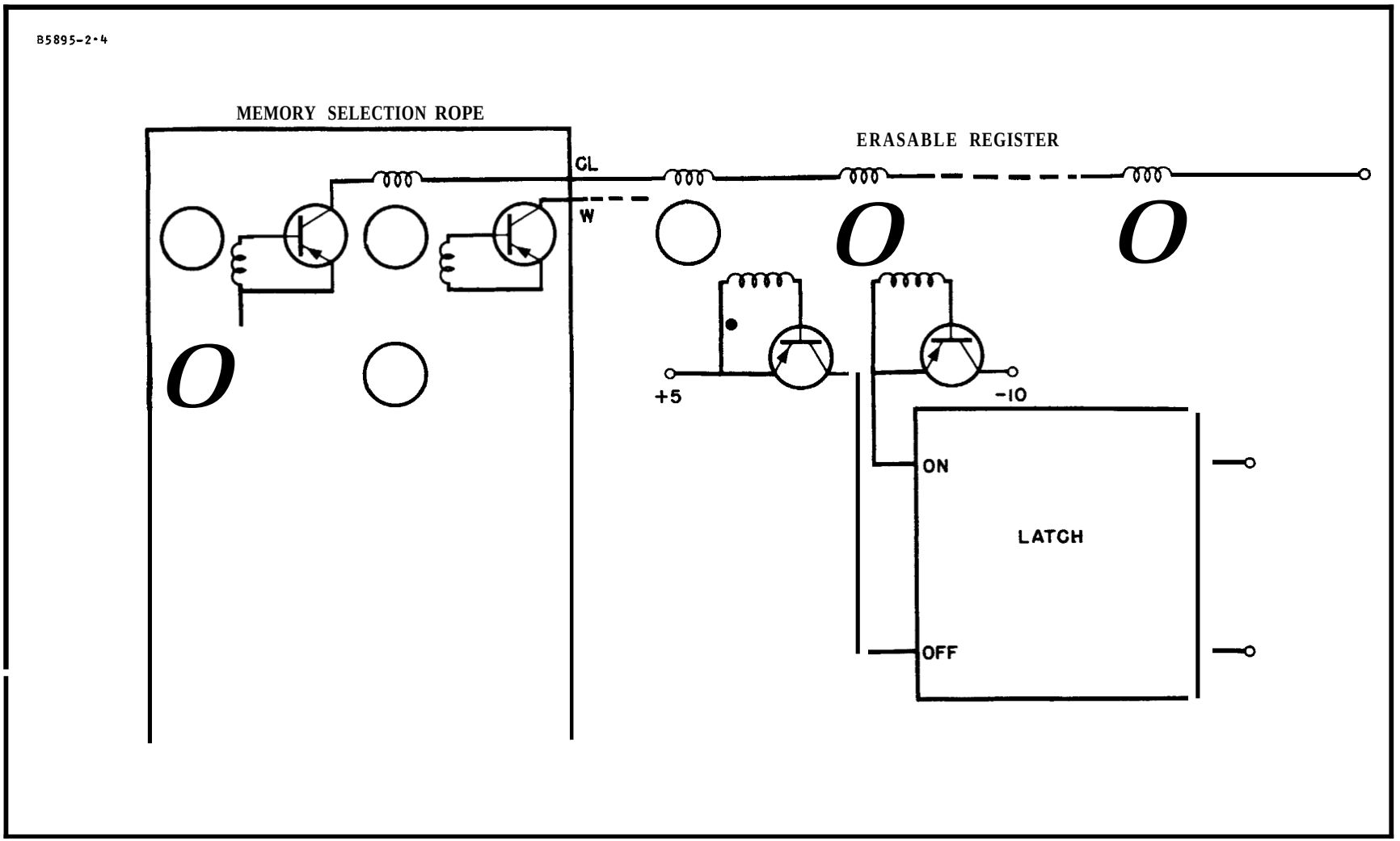


Fig. 2-4 Latch control register

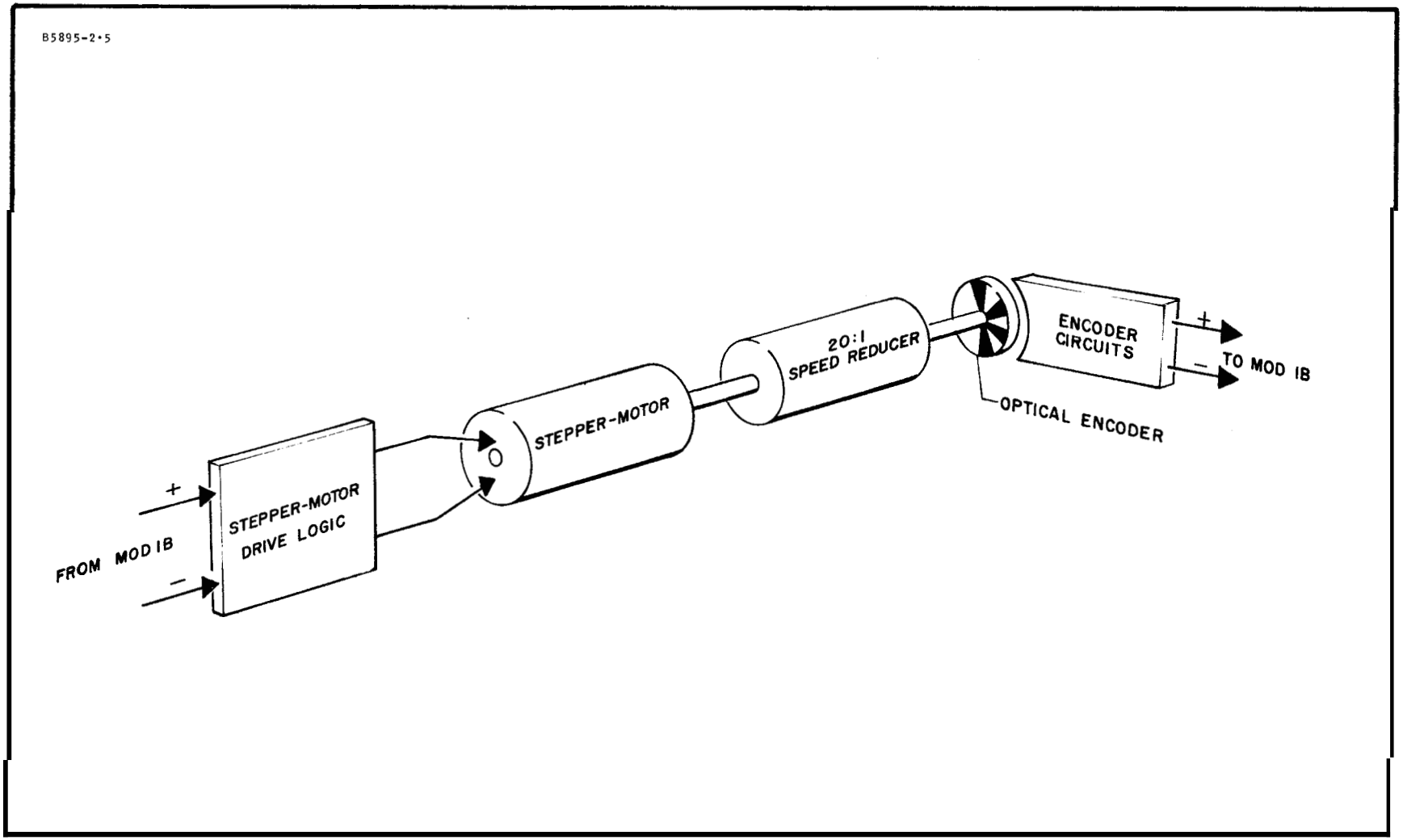


Fig. 2-5 Stepper-motor drive and encoder

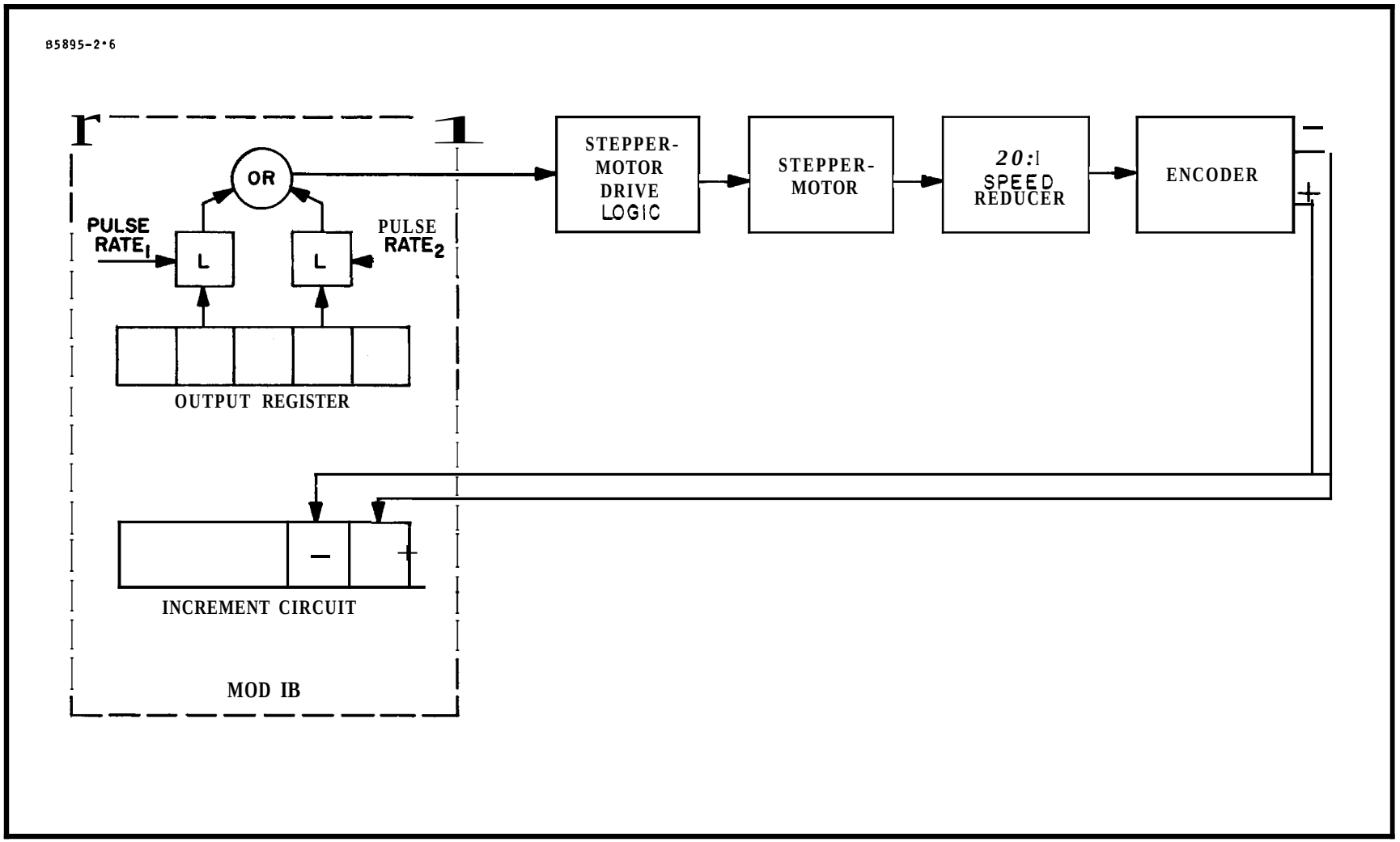


Fig. 2-6 Stepper-motor control system

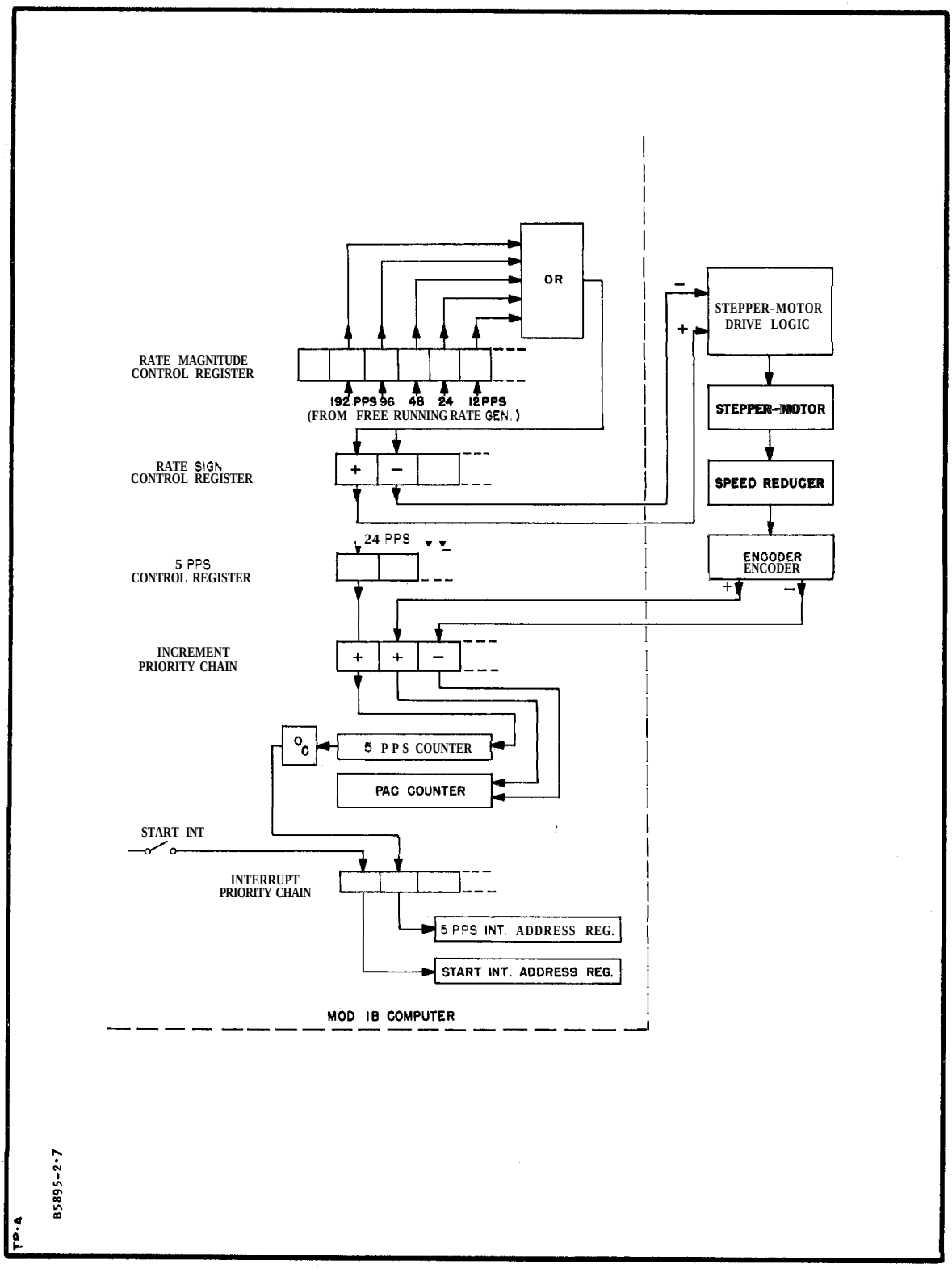


Fig. 2-7 Detail of rate control

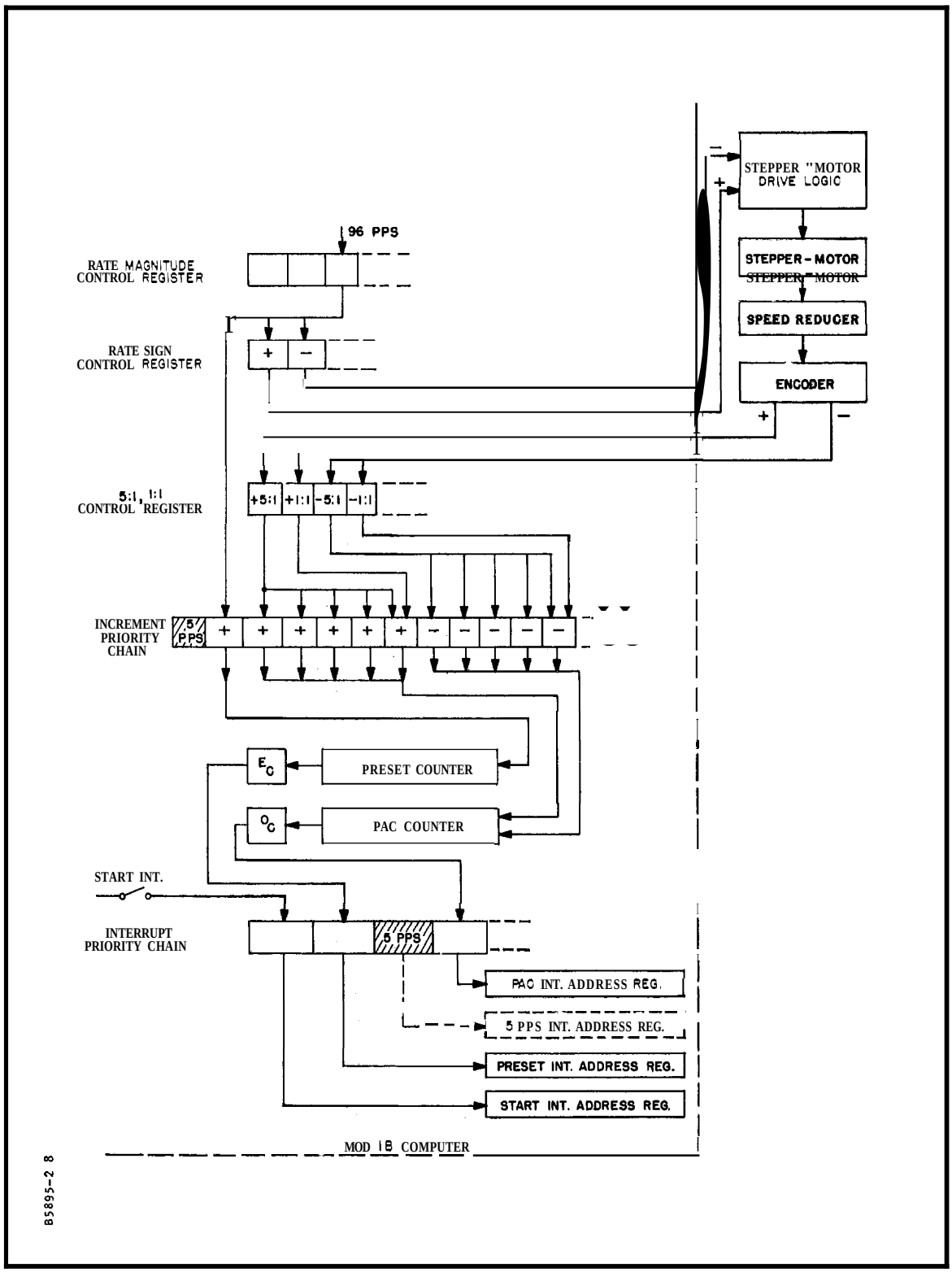


Fig. 2-8 Detail of position control

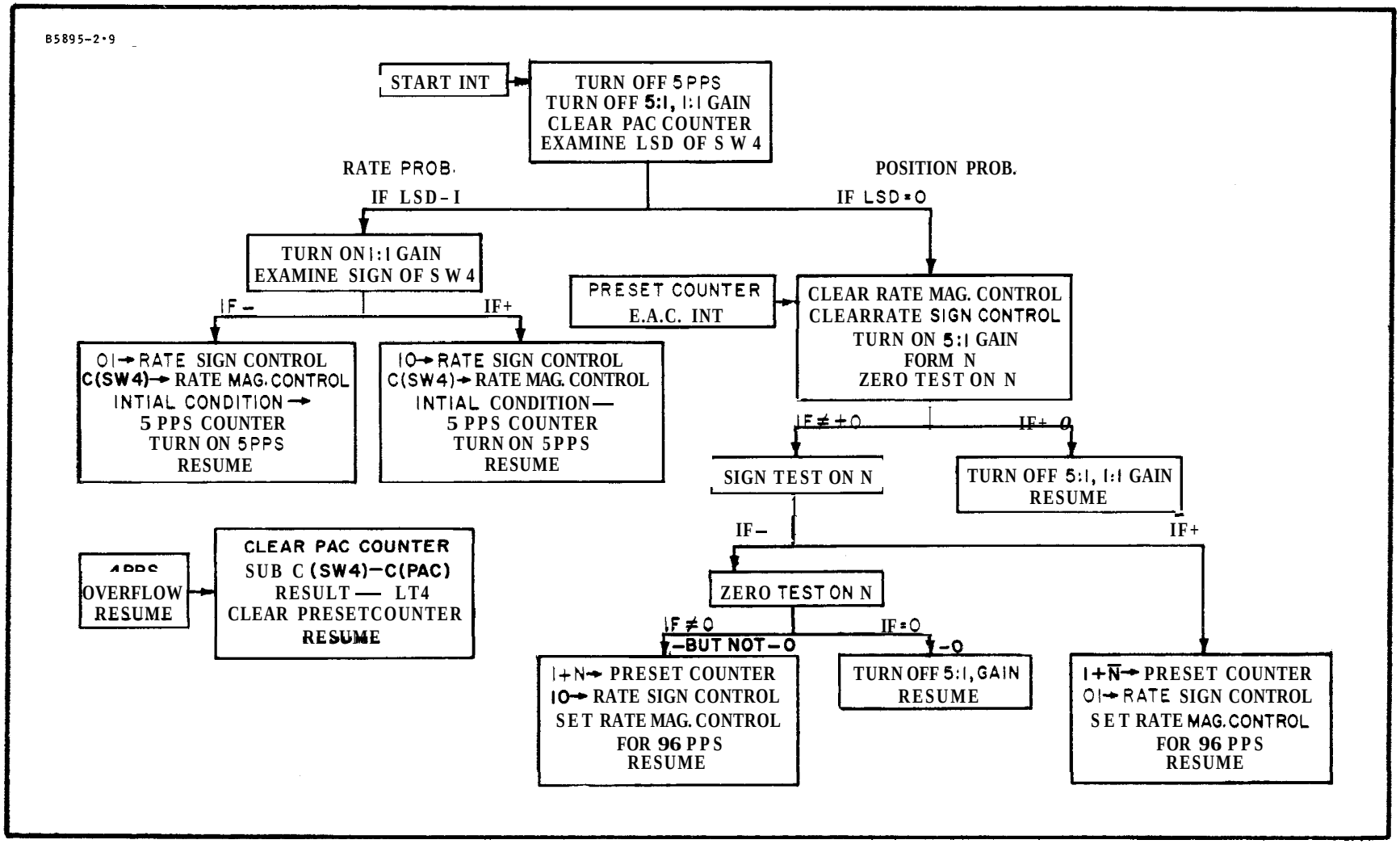


Fig. 2-9 Flow diagram of control program

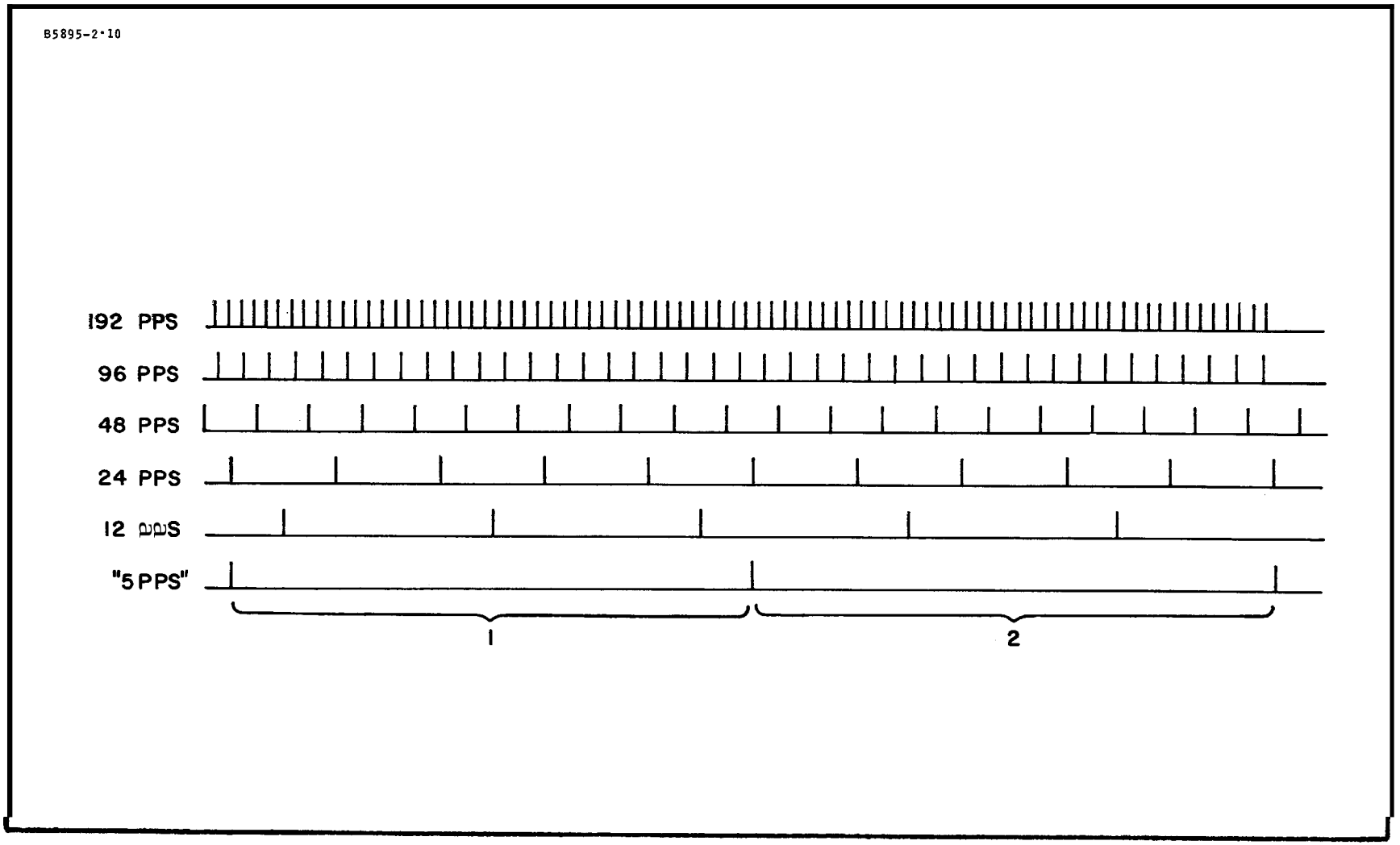


Fig. 2-10 Applied pulse rates

CONFIDENTIAL

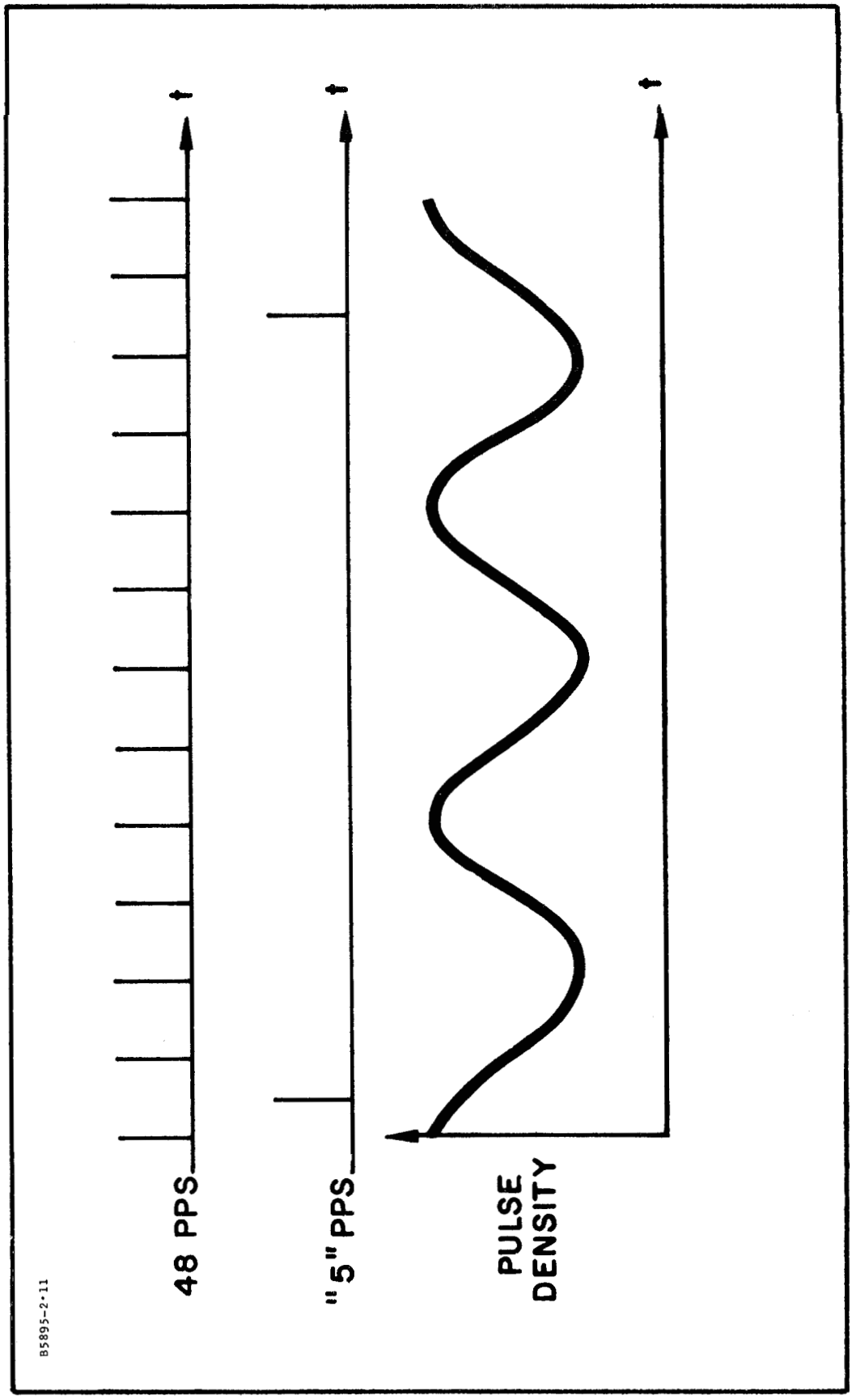


Fig. 2-11 Pulse density of encoder output

Table 2-1 Rate Problem Results

Pulse rate applied	12	24	48	96	192
Average of |error| (25 readings)	5.2	5.5	15.3	1.9	.64
Theoretical number of pulses in P_{AC}	32	64	128	256	512
Error accumulated over 5/24 sec.					

Table 2-2 Position Problem Results

Number of Increments Commanded	-1	-5	-10	-15
Contents of P_{AC} Counter after each Position Problem	65	310	625	1024
	55	325	615	940
	60	335	645	961
	75	320	630	971
	60	319	645	971
	60	320	630	951
	45	325	650	961
	85	325	635	956
Actual Total Number of P_{AC} Pulses	505	2570	5075	7735
Correct Number of Pulses	505	2560	5120	7680
% Error	0	.3970	.87%	.72%

Number of Increments Commanded	+1	+5	+10	+15
Contents of P_{AC} Counter after each Position. Problem	65	320	620	949
	60	325	635	998
	55	315	630	959
	65	325	635	945
	65	335	650	964
	70	320	640	969
	60	335	635	964
	65	320	635	965
Actual Total Number of P_{AC} Pulses	505	2595	5080	771.3
Correct Number of Pulses	1505	2560	5120	7680
% Error	0	1.3%	.78%	.43%

CHAPTER 3
ELECTRICAL DESIGN
by R. E. Oleksiak

CHAPTER 3

ELECTRICAL DESIGN

3.1 General Discussion

The objective of the electrical design of the Mod 1B computer was to develop reliable, low power circuits which could efficiently implement the desired logic,operations. The circuit design was never fixed, in order to allow all possible circuit advances to be used in the computer. Various circuit improvements occurred during construction and were incorporated in the computer. Different circuits have been used to perform similar functions and the advantages and disadvantages of the circuits will be noted.

The general circuit configuration which best suited the logical requirements of the computer is shown in Fig. 3-1. This circuit is not used as a basic building block, but the majority of the logical circuits are variations and combinations of this circuit configuration. The design criteria for this type of circuit may be found in the MIT Instrumentation Laboratory Report, E-987, "Design and Analysis of Pulsed Magnetic Core Circuits" by Wilbert H. Aldrich.

The main advantage of this type of core-transistor circuit. is that it does not require stand-by power. Power is dissipated only when the core is being set to ONE or cleared to ZERO, and because of the low duty factors of many sections of the computer, the effective power dissipation is very low. This circuit is also very insensitive to transistor and core parameters. Various germanium PNP transistors are used in the circuits. The only parameter which is important for circuit operation is the base-emitter voltage required to saturate the transistor. The clamping action

of the base-emitter junction is used to determine the switching time of the core. A transistor current gain of E0 is adequate for circuit operation. The circuits will operate effectively with voltage rise times as slow as 0.5 μsec. The only constraint on storage and fall time of the transistor is that turn-off be completed before the next pulse time. With the 1.5 psec switching time and 200 kc clock rate, the combined storage and fall time must not exceed 3 psec.

The use of silicon transistors in the computer has been limited because their much higher base-emitter voltage requirement would have necessitated cores with turn ratios which were not immediately available. These transistors have been used in applications where they are not driven directly by cores. In future computers, the use of silicon transistors in the core circuits would be desireable because they will operate over a wider temperature range than germanium transistors.

Both tape wound cores and ferrite cores were considered for use in the computer. Tape wound cores were selected because their parameters are stable over a wider temperature range and they also have lower power requirements than ferrite cores. The cores used in the logic circuits are 11 maxwell, 1/8 mil 4-79 Mo-Permalloy tape wound cores, with a minimum switching current of 0.13 turns.

In order to facilitate construction, the cores were purchased with the windings applied and packaged in epoxy modules. Appendix *C* contains a list of the core packages and the number of turns in the various windings.. The circuits were designed for these "standard" windings. New sets of windings were added to the list only if significant component savings or greatly improved. circuit operation would be realized.

The circuits selected for discussion in this chapter represent about 95% of the computer circuitry. The circuits will be

discussed in logical groupings, so that their relation to the computer can be seen. An attempt will 'be made to evaluate honestly the particular logical and electrical merits of the various circuits. In cases where the logical and electrical characteristics of the circuits are self evident, the subject will be treated with little detail. In some instances, circuits which seem to offer better solutions to the problem will be suggested.

The circuit diagrams have been simplified In the interest of clarity. Some of the minor modifications which were required because of oversights in the computer logical design have been omitted. In all cases, however, the circuit diagram represents the particular circuit under discussion as it is used in the computer. Changes in the diagrams have been limited to the methods of communication with other sections of the computer.

3.2 Control Circuits

Clock

Clock pulses are generated by circulating a single ONE in a variable length shift register. The shift register is driven by both phases of a 100 kc crystal oscillator for an effective 200 kc pulse rate. The block diagram of the shift register, with the various loops, is shown in Fig. 3-2.

The Increment and Interrupt elements represent major sections of circuitry and are discussed separately in Chapter 2. In relation to the shift register, they are simply decision circuits; which determine if the ONE shall proceed along the shift register or shall be looped back. The SQ circuit decodes the instruction being read out of memory at Time Pulse 14 and places the ONE in the appropriate stage of the shift register. In the case of a BNZN or a AOTSN instruction, the ONE is placed in the 1st stage. A NOR or a 'TSN instruction causes the ONE to be placed in the 5th stage.

The output of the RUN latch is used to gate the advance pulses from the oscillator to the shift register. During normal computer operation, the RUN latch is always on and the shift register receives advance pulses. When the RUN latch is turned off by a programed Normal Stop or by a parity failure, all computer operations cease. It is also possible to turn the RUN latch off at every Time Pulse 14. This feature is very useful in debugging both the computer and computer programs because the instruction next to be executed is displayed at Time Pulse 14.

The circuit for a single stage of the shift register is shown in Fig. 3-3. Core 1 is set to ONE when transistor T1 of the preceeding stage is turned on, causing current to flow into the 50 turn input winding. In stages 2, 4 and 5, Core 1 may also be set to ONE by a ground applied to the diode at the input winding. The ground is supplied by one of the possible loops in the shift register. The current which sets Core 1 to state ONE also sets Core 2 to state ONE, but the advance line which clears Core 1 to state ZERO does not pass through Core 2. Core 2 always remains in state ONE, and is used to cancel the noise induced by the fall of the set current, With 10 turns on the base-emitter winding, the noise is sufficient to turn on the transistors momentarily. The 10 turns on the base-emitter winding are necessary to insure that the clock pulses have sufficient duration and power to clear cores throughout the computer.

When Core 1 is cleared to ZERO, the bases of transistors T1 and T2 are driven negative with respect to the emitter, causing the transistors to be turned on. The output at the collector of T1 is used to set a ONE in the next stage. The output at the collector of T2 is amplified by transistors T3 and T4 to provide the clock pulse for general use in the computer. This amplifier was added when it was found that T2 did not have adequate power to meet the needs of the computer. Diagrams of this and other power amplifiers used in the computer are found in Appendix D.

There are two advance lines in the shift register, one being driven by each phase of the oscillator. The advance windings of the odd numbered stages are connected in series as are the advance winding of the even numbered stages. In this way, when an odd numbered stage is being cleared to a zero, there is no clear current in the even numbered stages to prevent the core from being set to ONE. Because of this, it is necessary that the loops contain an even number of stages.

It should also be noted that the rise of clear current induces noise on the base-emitter winding, which tends to turn the transistors on. Since there are seven clear windings connected in series, the amount of noise coupled by each core is much less, therefore noise cancelling is not required.

Sequence Generator

The Sequence Generator is the source of control pulses for use in the computer. An array of cores, such as shown in Fig. 3-4, is used to generate the control pulses. The operation of the Sequence Generator is as follows: When an instruction is decoded, the appropriate group of cores is set to ONE by current in the proper set line (A, B, C, or D). Now, as the time pulses cause current to flow in the clear lines, the cores, one at a time, are cleared to ZERO. When a core is being cleared to ZERO, the voltage induced on the sense windings linking that core turns on the transistors connected to the sense windings. The outputs of the transistors are then used as control pulses.

Note that more than one set line may thread a given core. This allows several control pulses, which may be required by two or more instructions at the same time pulse, to be generated by the same core.

This Sequence Generator, simple as it may be in concept, has innumerable problems associated with it. With as many as 10 multi-turn windings on a core, it was not feasible to use the

potted modules because of the number of interconnections which would be required. The core array was mounted on a phenolic board, and the cores were hand wound. Because of this construction technique, changes and corrections could not easily be made. The maze of wires that resulted was also subject to short-circuits. When the Sequence Generator was finally operating, it was sprayed with insulation and left undisturbed. All future changes which became necessary were made in some other section of the computer.

Two major electrical problems associated with the Sequence Generator are caused by noise. Noise cancelling cores, with the clear and sense windings, were required at each point in the array, thereby doubling the construction and wiring problems. Also, current in the sense lines under normal conditions would be sufficient to set partial ONE's into cleared cores. If these cores set to partial ONE's were cleared, they would cause noise on the sense lines, producing unwanted control pulses. In order to minimize this effect, 51 ohm resistors were placed in series with the sense lines. Although the control pulses suffered from increased rise time and decreased width, because of the reduced driving current, they were still usable, and needed reshaping in only a few instances,

This sequence generator is used in the Mod 1B because lack of time prohibited a major redesign. Since most of the electrical problems in Mod 1 B are caused by variation in delays, and widths, of the control pulses, designs for future sequence generators should include provisions for adjustment of these control pulse parameters. This would simplify many of the design problems in other sections of the computer.

Priority Circuit

The Increment and Interrupt Priority Chains have the same logical function. They provide an "instantaneous" scanning of a number of inputs, consumming no system time when the inputs do not require service. These chains are located in the Clock Shift Register, (Fig. 3-2).

The first basic design of the priority circuit is used in the Increment Priority Chain. (See Fig. 3-5) The operation of the Priority Chain can be most easily understood if it is considered to be a lumped parameter delay line. The winding on the core is the inductor and the transistor is the capacitor. The effective capacitance of the transistor is much larger than its intrinsic capacitance because, as the current rises in the 50 turn winding, noise induced in the base-emitter winding causes the shunting transistor to conduct momentarily. When the cores are at state ZERO, so that they will not switch because of current on the clear line, the net effect is a 0.05 psec delay per stage of the leading edge of the clear waveform, as it propagates along the priority chain. When a core is at state ONE, the shunting transistor, after it is turned on by induced noise, remains on as long as the core is switching, This diverts the clear current away from the remaining cores. The 50 turn winding of the switching core causes a voltage drop along the cleared line at that stage. The net result is that there is not enough current or voltage on the clear line of the following stages to disturb any core which is also at state ONE. Due to storage effects, the shunting transistor will stay on for a short time after the core has finished switching. If the clear pulse ends after the core has switched and before the end of the storage time of the shunting transistor, the first core along the chain set to ONE, and only that core, will be cleared to ZERO.

In this type of priority chain it is vitally necessary for the clear pulse to be the proper width. If the clear pulse is too short, the core will not finish switching, and at the next clear pulse will cause an output from that stage before going on to the next stage. If the clear pulse is too long, the first core will finish switching and the next core will switch partially, giving an output. In both cases, the unwanted outputs will cause a system malfunction.

In order to obtain reasonable working margins, the clear pulse is generated by a core-transistor combination similar to

that used in the priority element. Because of this, the variation of the duration of the clear pulse with voltage is the same as the voltage variation of the core switching time and transistor storage time in the priority element.

Another effect which must be considered is that the effective capacitance of the transistor, in the delay line analogy, is different for the leading and trailing edges of the clear pulse. As previously mentioned, the capacitance at the leading edge of the clear current is much greater than the intrinsic capacitance of the transistor; but at the trailing edge, the capacitance is only the intrinsic capacitance of the transistor. Because of this, the trailing edge is delayed much less than the leading edge, The duration of the clear pulse is thereby shortened approximately 0.05 ysec per stage, the amount that the leading edge is delayed.

The reason for placing the diode in series with the collectors of the shunting transistor deserves explanation. When a core is being set to ONE, a voltage appears on the clear winding causing the voltage at some of the emitters to become more negative than the voltage at the collector. (See Fig. 3-6.) A PNP transistor, with the base connected to the emitter through a low impedance, will allow current to flow from collector to emitter. Without the diode in the circuit, current could flow through the shunting transistors and the clear line, causing cores already in state ONE to be partially cleared.

As shown in Fig. 3-5, the Increment Priority Chain, with 14 stages, was too long to operate properly. By placing a capacitor between the base and the emitter of one shunting transistor, the storage time of this stage was increased, allowing the entire chain to operate satisfactorily,

Before going on to the design of the Interrupt Priority Chain, it will be useful to *see* how the Increment Priority Chain is used in the computer, The inputs to the first 13 stages of the Increment Priority Chain are positive or negative pulses to be

counted by the computer. The 14th stage is always set to ONE by Time Pulse 6. The priority chain is located between the 9th and 10th stages of the Clock Shift Register.

Time Pulse 9 generates the clear current for the priority chain. The clear current then propagates along the chain and clears the first stage containing a ONE. An output from Priority stages 1 to 13 sets a ONE into stage 4 of the Shift Register, sets a ONE is the clear core of the appropriate counter register, and selects the positive or negative increment sequence. The Shift Register will continue looping from stage 9 back to stage 4 until all the inputs have been serviced. When stage 14 of the Priority Chain is switched, all the inputs have been serviced and the ONE is then set into stage 10 of the Shift Register allowing the computer to proceed. (See Chapter. 2.)

The most serious pro'blem with the Priority circuit used in the Increment Priority Chain, is that the clear pulse is shortened as it propagates along the chain. The troubles, as associated with this effect, have been largely eliminated in the Priority Circuit used in the Interrupt Priority Chain, *(See* Fig. 3-7.) In this circuit, the output transistor is used to insure that the core finishes switching. The clear pulse is only required to be long enough to insure that the last core in the chain will start to switch. Once a core has started to switch, the ground provided by the output transistor will complete the switching.

In order that the ground provided by the output transistor will not start another stage switching, the output transistor must turn off before the shunting transistor. This could be accomplished by any of several methods: by using a transistor with greater storage time for shunting; by driving the shunting transistor with more turns; by loading the output transistor, or by any combination of the above methods. Since modules, with the same type of transistors, driven by the same number of turns, were available, the method of loading the output transistor was used. The normal load

on the output proved adequate to insure that the output transistor turned off before the shunting transistor.

Since this Priority Circuit is regenerative, it is necessary to gate the output transistors to prevent a stage from clearing itself, due to noise produced by setting a core to ONE. At the desired clear time, the common resistor R1 prevents the clearing of more than one stage.

The Interrupt Priority Chain has operated very well. Using the same circuit, it would be practical to operate chains up to twenty stages long. Longer chains would be made by cascading chains, using the output of the last stage to provide the clear current for the next chain. The clear pulse would be delayed 1 ysec in scanning a chain twenty stages long.

The Interrupt Priority Chain has a more complex relationship with the computer than does the Increment Priority Chain. When an interrupt occurs, the computer leaves the main program and enters the interrupt program. On completion of the interrupt program, the computer may re-enter the main program at the point of departure and continue, or it may enter another interrupt program. (See Chapter 2.)

The circuit operation is as follows. At Time Pulse 13, the clear pulse for the Priority Chain is generated, propagates along the chain and finds a ONE in some stage other than the first or the last. The output from this priority stage sets a ONE into stage 2 of the Clock Shift Register, sets a ONE into the clear core of the appropriate Interrupt Address Register and selects the INTERRUPT Sequence, A control. pulse from the INTERRUPT Sequence then sets a ONE in the first stage of the Interrupt Priority Chain and turns L1 off. When L1 Is off, the Interrupt Priority Chain is by-passed to prevent other interrupts and the output of the Shift Register stage 13 directly sets a ONE into stage 14. The last instruction of the interrupt program causes L1 to be turned on. With L1 on, the next Time Pulse 13 clears the first stage of

the Priority Chain causing a ONE to be set in Shift Register stage **4** and the RESUME Sequence selected. At the next Time Pulse 13, if there are no ONE's calling for another interrupt program, the last stage of the Priority Chain is cleared causing a ONE to be set in Shift Register stage **24** and the computer proceeds with the main program,

Write Latches

Before describing the operation of the write latches, it is necessary to explain the operation of the latch circuit. The latch, similar to a flip-flop, has two stable states. Complemented outputs are not available as in the case of the flip-flop, The main advantage offered by the latch is that power is only dissipated during the ON state. This allows the designer to make use of low duty factors to lower power consumption.

The three latch circuits shown in Fig. 3-8, 3-9, and 3-10 represent different stages of design and are all three used in the computer. The operation of the latch is as follows: All three latches are turned ON by a negative voltage applied to the input resistor. This causes the base of PNP transistor T2 to be drawn negative with respect to the emitter, turning T2 on. The current through T2 then draws the base of NPN transistor T1 positive with respect to the emitter, turning T1 on. The current through T1 then is sufficient to keep the base of T2 negative when the input voltage is removed. The output is taken from the collector of transistor T3 in the circuits shown in Fig. 3-8 and 3-9. In the latch shown in Fig. 3-10, the output is taken from the collector of T2 via diode D2.

To turn the latch off, it is necessary to break the regenerative loop formed by transistors T1 and T2. In the latch circuit of Fig. 3-8, the OFF gate interrupts the flow of emitter current in T1, thereby breaking the loop. In the latch circuits of Fig. 3-9 and 3-10, the loop is broken by clamping the base of T2 positive via diode D1. A complete analysis of latch circuit operation may

be found in Massachusetts Institute of Technology Instrumentation Laboratory Report, E-1054, "Analysis of the PNP-NPN Latch Circuit" September 1961 by Joseph Rocchio.

The greatest problem with the latch circuit of Fig. 3-8 is related to the Off Gate. When more than one latch was operated from the same Off Gate, the voltage at the emitter of transistor T1 would vary, changing the sensitivity of the latch. The transistor T3 was used both to increase the output power and to permit the direct gating of pulses from a common source. When it was later found that T3 could not be used reliably to gate pulsed sources, the power gain alone did not warrant the use of the extra transistor; therefore, the latch circuit in Fig, 3-10 was designed.

The write latches (Fig. 3-11) are a group of 14 latches, one for each bit position in the word, one for the EC bit and one for the OC bit. They are used to amplify and discriminate the signals on the sense lines as they are read from one register, to be written into one or more other registers.

The latches are turned on by outputs of the various memory and register sense amplifiers associated with the given bits. The outputs of the write latches are connected to the write busses. In order to allow the maximum time for writing into registers, the write latches are not turned off until the beginning of the next time pulse. The pulse, used to turn off the latches, is generated by the leading edge of the clock shift register advance pulse.

Indicator lights have been connected directly to the outputs of the write latches. Since the resistance of the cold filament is very low, low current diodes are used in series with the lights to limit the load on the latch. When the computer is stopped, the lights display the last word that has been transferred in the computer. This feature has proven very uneful in debugging the computer.

3.3 Standard Central Registers

The C, T1, T2 and T3 registers are standard central registers. Their only logical function is to provide temporary storage required for the execution of instructions. The central registers are addressed by control pulses from the Sequence Generator,

The circuit diagram of the standard central registers is shown in Fig. 3-12. Note that the central registers are divided into two groups: the sensed output of one group is sampled at an α (odd) time pulse and the sensed output of the other group is sampled at a β (even) time pulse. A register is referred to as an α register or as a β register according to the time at which its output is sampled. This division of the registers is necessary because of the way the bit senselines are connected in series to share amplifiers. The noise induced on the sense lines by writing into a register prohibits the sensing of the outputs from a different register sharing the same set of sense amplifiers. With the two sets of amplifiers, it is possible to read simultaneously the contents of an α register and write into a β register. Conversely, it is possible simultaneously to read data from a β register and write it into an α register. Registers B, C, P, SQ, T1 and T3 are β registers. The α registers include the A, Z and T2 registers, fixed memory and erasable memory.

The sense amplifiers are simple one-transistor circuits. One end of the sense line is connected directly to the emitter and the other end of the sense line is connected to the base through a 51 ohm resistor. This resistor is necessary to limit the current in the sense lines so that partial ONE's are not set in other cores on the same sense line. The 0.5 v output from the 12 turn sense windings easily saturates the sensing transistor. Since only one core on a sense line is cleared at a given time, ZERO noise is completely eliminated by the threshold of the sensing transistor. The sampling of a set of sense amplifiers is done by a common

gate applied to the collectors of the transistors. The isolating diodes in the collectors of the sense transistors are necessary to prevent coupling between the inputs to the write latches due to the reverse voltage which can appear across the transistors when the sampling gate is off.

Using the C register as an example, the operation of a standard central register is as follows: (See Fig. 3-12) Assume that the cores in the C register have been cleared to ZERO. When the data to be written into the C register is on the write busses, a WC (Write C) control pulse (at α time) from the Sequence Generator supplies a minus voltage to the common end of the 40 turn winding. The presence of this voltage together with a ground on the output of a write latch allows current to flow, setting the core to a ONE, When the information stored in the C register is desired, the CLC (Clear C) control pulse (at β time) causes current to flow in the common 4 turn winding, clearing the cores to ZERO. At this time, the outputs are sensed and the write latches are turned on.

3.4 Special Central Registers

Registers B, A, P, S, SQ and Z are special central registers used to perform various logical functions, The description of the S register is given in Section 3.5 because of its relation to the memory.

B Register

The B register is similar to the standard central registers. The sole difference is that it has 14 bits instead of the standard 12 bits. The 2 extra bits are uaed to store the EC and OC bits. Because the Z register and the counters in memory are α registers, it is necessary to have a β register which can store the EC and OC bits *so* that they may be transferred from Z into the counters. The EC bit is cleared by CL EC and the OC bit by CL OC. The rest of the register is cleared by CL B.

A Register

The A register (Fig. 3-13) is used to complement and generate the NOR function. There are three control pulses associated with the A register, The W1A (Write ONE's into A) control pulse causes current to flow in one of the common four turn windings, setting the 11 cores to ONE. The WA control pulse gates the common end of the 40 turn windings *so* that current may flow from the latches which are turned on. Note that the direction of current in the 40 turn winding is such that the cores are cleared to ZERO. The CL A control pulse causes all the cores to be cleared to ZERO. The sensed output of the A register is then the complement of the word on the write busses during the WA control pulse. The NOR function is generated by having two or more WA control pulses occur between the WEA and the CL A control pulses. Since the NOR function may destroy parity, the parity bit is not used in the A register.

SQ Register

The SQ register (Fig. 3-14) is used to decode the two high order bits of the word into the four basic instructions. At Time Pulse 11 the Sc and 10c cores; are set to ONE. At Time Pulse 13, a ONE on write bus 10 sets core 10d to ONE and clears core 10c to ZERO. A ONE on the sign write buss seta core Sd to ONE and clears core Sc to ZERO. The cores are cleared at Time Pulse 14 by SQ. The transistors are connected *so* that the four possible ONE and ZERO combinations are decoded into the four basic instructions. The outputs are then powered to set the cores in the Sequence Generator. The CL B control pulse, which also resets the cores, was necessary because of an oversight in the INTERRUPT Sequence.

The zero test circuit (Fig. 3-15) is used during the branch instruction, The OR function of all the outputs from the write latches except parity is generated by the group of diodes, At the ZT pulse, if any of the latches have a ONE output, the core $\overline{O}$ is

set to ONE; if all the outputs are ZERO, core *O* is set *to* ONE, The test for ZERO pulse clears the cores, causing the outputs to set cores in.the Sequence Generator.

Parity Register

The parity register is used to check and generate odd parity in the computer. Since the computer uses a twelve bit word length (including the parity bit), odd parity requires that each word contain at least a single ONE or a single ZERO. The parity circuit is therefore able to detect the pick-up of all ONE's or the loss of all the ONE's as well as single bit errors.

The basic element in the parity circuit is the "Exclusive OR" circuit shown in Fig. 3-16. Current into input A tries to set Core I to a ONE and inhibit Core II from being set to ONE. Current into input B tries to set Core II to a ONE and inhibit Core I from being set to ONE. The transistor is turned on when either core is switched to ONE. With current into both inputs, neither core will switch. The transistor will be turned on when one and only one input has current.

The parity register is a convergent tree of these "Exclusive OR" circuits, (See Fig. 3-17.) The WP gate allows the word on the write busses to be written into the first stages of the parity tree. The pulses then propagate along the tree with a delay of 0.1 μsec per stage, A pulse on the output of the final stage indicates that the number of ONE's in the word on the write busses is odd. The output from the last stage is used to inhibit Core a from being set to ONE. When Core a is set to ONE, the output transistor is turned on allowing current to flow, setting Core b to ONE. Core b contains a ONE if the number of ONE's in the word is even. When parity is being generated far an eleven bit word from Z or A, the ONE or ZERO in Core b is the correct parity bit for the word. The CL P pulse occurs when the word for which parity has been generated is being transferred back to the α side of the computer. Transistor T1 turns on the parity

bit of the write latches when Core b is cleared from ONE to ZERO, When parity is being checked, a ONE in Core b indicates a parity failure. The TP (Test Parity) pulse occurs at the same time as CL P when parity is checked, allowing transistor T2 to turn on the parity fail latch and to stop the computer,

The parity register will operate correctly only if all bits appear on the write busses simultaneously or if the write gate occurs after the last write latch has been turned on, Since the outputs from fixed memory occur very late and unevenly, the circuit shown in Fig. 3-18 was necessary to delay the Write Parity gate. The circuit is basically a two stage priority circuit with the clear pulse long enough to clear both stages. The 2 psec delay is obtained by using the output of the second stage as the Write Parity gate. It is also necessary to delay the pulse used to set Core a to ONE because of the propagation delay in the parity tree. This delayed pulse is generated by transistors T4 and T5 in Fig. 3-18.

The troubles associated with the parity register are largely due to other sections of the computer. The late arrival of bits from fixed memory requires that the parity register operate at the very end of the time pulse. The variations in the WP control pulse due to the Sequence Generator has made the generation of accurate delays impossible. Because of these problems, the parity register will only operate correctly over a limited voltage range.

The parity register has proven a valuable tool for debugging the computer and programs. The computer console has a switch which can inhibit the parity fail latch so that it is possible to allow the computer to run despite parity failure. It is possible to program a short loop using the register which causes a parity failure so that the operation may be checked using an oscilloscope.

A minor change in the method used to generate and check parity would greatly facilitate trouble shooting. The computer now stops on a parity failure with a ONE always displayed by the

parity bit of the write latches. By having the TP pulse occur at the same time as the WP pulse, the computer would stop with the write latches displaying the word as it is read from the α side of the computer. This would immediately separate true parity failures from failures due to the parity register. This change was not made in Mod 1B because it would have required a change in the Sequence Generator,

The addition of a little extra circuitry to have the parity of the word corrected when a failure occurs instead of always leaving the parity bit at ONE would be worthwhile, As the computer is now organized, a word with incorrect parity and a ONE in the parity bit can only be corrected by reloading the word through the Switch Register.

Z Register

The *Z* register is used to increment the program address and the contents of the counter registers by plus or minus one. The block diagram of the *Z* register is shown in Fig. 3-19. The WZ, ZA, *ZS*, and CL Z control pulses are used to control the Z register. The WZ control pulse causes the word on the write busses to be written into the Z register. If a ZA control pulse occurs at the same time as WZ the word is incremented by plus one. The *ZS* control pulse along with WZ causes the word to be decremented by one. The CL Z control pulse clears the Z register and the result appears on the sense lines. Since incrementing and decrementing can destroy the parity of the word, the parity bit has been omitted in the *Z* register.

The circuit for one bit position of the counter is shown in Fig. 3-20. The resultant bit is formed in cores E_N and F_N. At CL Z a ONE being cleared from core F_N causes a voltage to appear on the sense line in a direction *so* as to turn on the sense amplifier. A ONE being cleared from core E_N causes a voltage to appear on the sense line in a direction opposite to the direction required to turn on the sense amplifier. Note that the sense

winding on E_N has more turns than the sense winding on F_N. When both E_N and F_N are cleared from ONE to ZERO, the resultant voltage is in a direction which will not turn on the sense amplifier and a ZERO is sensed. To sense a ONE, core E_N must have been in state ZERO and core F_N in state ONE,

During a positive increment, cores E_N, F_N and transistor T_{N1} are used. The WZ pulse provides a minus gate *so* that a ONE from the write buss is set into core F_N. Note that the carry from the preceding stage can also set core F_N to ONE. Core F_N stores the function $(X_N \text{ t } C_{N-1})$. The carry (C_N) is generated in core E_N. If the write buss is at ground (ONE state) and there is a carry from the preceding stage, core E_N is set to ONE, turning T_{N1} on to give the carry C_N. Core E_N stores the function $(X_N \cdot C_{N-1})$. When cores E_N and F_N are cleared by CL Z, the sense line output is the result $(X_N + C_{N-1}) \cdot \overline{(X_N \cdot C_{N-1})}$. In the first stage, the ZA pulse is used as the carry from the preceding stage to start the incrementing action.

During a negative increment, cores E_N, F_N, and G_N and transistor T_{N2} are used. The WZ gate allows core F_N to be set to ONE if the write buss is at ground. The borrow pulse from the preceding stage (B_{N-1}) can also set core F_N to QNE. Core F_N stores the function $(X_N + B_{N-1})$. If the write buss is at the ground state (ONE), B_{N-1} also can set core E_N to ONE. Core E_N then stores the function $(X_N \cdot B_{N-1})$. Note that even though transistor T_{N1} is turned on, it has no effect, since the ZA gate is off. When cores E_N and F_N are cleared by CL Z, the sense line output is the result $(X_N + B_{N-1}) \cdot \overline{(X_N \cdot B_{N-1})}$. The borrow, B_N, is generated in core G_N. When the write buss is open (ZERO state), the borrow from the preceding stage sets core G_N to ONE. Transistor T_{N2} is turned on as core G_N switches to ONE to generate the borrow pulse B_N. When the write buss is at ground (ONE), the current from B_{N-1} bypasses the core, leaving it unaltered, The ZS pulse is used as the borrow into the first stage to initiate the borrow propagation. Core G_N has no sense line.

Propagation of the carry or borrow pulse through the eleven bits of the Z register requires 1μsec. The carry or borrow pulse is between 1.2 psec and 1.5 psec long. Since words from β registers are on the write busses a minimum of 3.5 μsec, the amount of time available would allow up to 22 stages in the Z register. The only timing constraint on the Z register is that the ZA and ZS gates do not appear before or last longer than the bits on the write busses.

The ONE's complement number system used in the computer has both positive and negative zeros. When a negative zero (11... 11) is incremented positively, the carry from the last stage is used as an end around carry (E.C.) to generate the correct result (00 ... 01). The low order ONE is generated by using the carry from the last stage to set to ONE a core on the low order sense line. At CL Z time, there are 3 cores on the sense line which are being cleared from ONE to ZERO. The sense winding on this extra core is in the direction to turn the sense amplifier on. The net output of the 3 cores being switched is sensed as a ONE. The carry from the last stage also set the EC + EB core to ONE. At CL Z the transistor on the EC + EB core turns on the EC write latch. When a positive zero (00 ... 00) is incremented negatively the end around borrow (EB) is generated to give the correct result (11 ... 10). The borrow from the last stage is inverted to obtain a ground which is used to set core E_N to ONE in the first stage. The net output on the sense line of the first stage is then ZERO. The EC + EB core is also set to ONE by the borrow pulse of the last stage.

Overflow (01 ... 11 to 10 ... 00) and underflow (10 ... 00 to 01 ... 11) are also sensed. The OF or UF core is set to ONE by a carry from the tenth stage and the sign write buss open, or by a borrow from the tenth stage and the sign write buss at ground. At CL Z the transistor on the OF or UF core is used to turn on the OC write latch.

3.5 Fixed Memory

The basic element of the fixed memory is the core rope.* The core rope is a string of 2^N cores threaded by N pairs of inhibit wires *so* that each core is threaded by a unique combination of wires, one wire from each of the N pairs. When inhibit current is sent through one of each pair of inhibit wires, only one core is free to switch. The core rope is also threaded by a number of sense lines. When a core in the rope is switched, a ONE appears on those sense lines that thread the core and a ZERO appears on those sense lines that do not thread the core. In this way it is possible to store an entire word of fixed storage in a single core.

The S register (Fig. 3-21) is used to generate the inhibit currents to select a particular core in the rope. The nine address bits and their complements are stored in the S register by the WS control pulse. The complement is generated by using the ONE's on the write busses to inhibit the switching of the complementing cores. This method of complementing differs from that used in the A register in that the current used to set the cores to ONE occurs at the same time as the current which tends to clear (inhibits) them to ZERO, If the WS pulse were to begin before or end after the address bits are on the write busses, the complementing cores which should be at ZERO would be partially set to ONE. The cores in the S register are cleared to ZERO by the CL S control pulse, turning on the PNP output transistors. The outputs of the PNP transistors drive the NPN power amplifier providing 300 ma to drive the core rope, The storage of the NPN transistor is used to increase the duration of the outputs to a minimum of 5 psec. These currents typically last 6 psec but they may not exceed 10 psec for computer operation. At the same time, the CL S control pulse is used to generate the set pulse used in the core rope. Note that the NPN power amplifier used

* Design Principles for a General Purpose Control Computer, M. I. T. Instrumentation Laboratory Report: R-276, April 1960.

for the set pulse has a smaller base to emitter resistor as well as diodes in the emitter circuit to insure that the set pulse does not last longer than the inhibit pulses, The' SI control pulse is used to prevent the set or inhibit current when it is desired to clear the S register without addressing the core rope.

The rope reset current is generated by the CL F pulse. In addition to the 300 ma in the reset wire, 15 ma of reset current flows in each of the inhibit lines, The reset current in the inhibit line prevents "ringing" at reset time when the rope outputs are sensed.

A single turn winding is used to sense the outputs of the switching cores. In order to reduce the inductive noise coupling between the reset and sense lines, each sense line passes through an equal (plus or minus one) number of cores in both directions relative to the reset current. The noise contributions from the individual cores cancel and the signal becomes bipolar, The amplitude of the output signal. is approximately 100 mv.

The sense amplifiers used for fixed memory are shown in Fig. 3-22. The circuit is basically a difference amplifier formed by transistors T1 and T2 with the outputs rectified and amplified by transistor T3. The time and amplitude discrimination is accomplished by the gate applied to the emitter of T4. The output of T4 is used to turn on the write latches. The most undesirable feature of this sense amplifier is the dependence of noise rejection on the voltage level of the gate. As the number of ONE's in a word being read out of memory varies, the load on the gate varies causing variations in the rejection level. This problem was solved by increasing the power of the gate and then loading the gate *so* that the variations of sense amplifier load would not cause significant variation of the gate voltage level. The sense amplifiers are also not in keeping with the design philosophy in that they require d-c power.

Despite sense amplifier problems, the fixed memory has operated extremely satisfactorily. The core ropes have proven to be rugged and trouble free. The S register has operated very reliably but a design which did not depend on transistor storage to determine pulse duration would be desirable. Larger core rope memories would require a more sophisticated sense amplifier design.

3.6 Standard Erasable Storage

The standard erasable storage registers differ from the standard central registers in that they are addressed by the core rope instead of the Sequence Generator. See Fig. 3-23. The operation of the standard erasable register is as follows: A core in the rope is set to ONE in the same manner as for fixed storage. When the core is reset the ZERO, the output voltage on the 6 turn winding turns on transistor T1. The current, due to T1 being on, sets the W1 core to ONE and clears the cores in the register to ZERO, The outputs of the register cores are sensed and turn on the write latches. To write back into the register the W1 core is cleared to ZERO, turning on transistor T2 which allows the contents of the write busses to be set in the register.

Forty eight of the erasable registers share sense amplifiers with the α central registers. It was necessary to provide separate sense amplifiers for the remaining thirty two erasable registers because the inductance of eighty cores having 12 turn sense windings in series was too large. It would have been possible using a more complex sense amplifier, but the addition of twelve, one-transistor amplifiers was a more economical solution.

3.7 Special Erasable Registers

Cycle and Shift Registers

The computer has a cycle left register, a cycle right register and a shift right register in erasable storage, as shown in Fig, 3-24. In both cycle registers the write buss connections are

standard, but the sense lines in the cycle left link the next lower order bit position and in the cycle right they link the next higher order bit position. When a word is written into and then read out of a cycle register, the bits (except parity) have been cycled one position.

In the shift right register., the sense lines are standard and the write buss connections have been changed. The sign position write buss is used to set the cores linked by sense lines of the sign bit and the bit ten position. Write busses 10 through 2 are shifted down to the next lower order bit position with the bit 1 write buss not used. Since parity is not conserved in this shifting operation a new parity bit is generated using the sign bit, bit 1 and the old parity bit. The operation of the parity generating circuit is similar to that of the parity register.

Switch Registers

Four erasable registers are used to accept inputs into the computer from a set of forty eight switches on the control panel. The wiring is normal except the set windings on the cores are connected to the switches instead of the write busses (Fig. 3-25). Two of the switch registers are also addressable by control pulses during the engineering sequences. In normal computer operation they are only addressed by the core rope,

Light Registers

The light registers are used both for display and a source of d-c outputs from the computer. The light register circuit is shown in Fig. 3-26. The sense winding on each core is connected to a transistor which turns on a latch as the core is set to ONE. When the register *is* cleared, the latches are turned off.

The latches in the light registers are turned on by writing; the correct word into the light register. To turn all the latches off, it is necessary to write st word of all ZERO's into the register. Since the normal sense lines do not link the light registers

they can not be used for storage. The erasable register, whose address is one less than a light register address, can not be used to store an instruction other than BNZN*.

Counters

The three counter registers are fourteen bit registers which are addressed both by the rope and by the Increment Priority Chain (See Fig. 3-27). The two extra bits are the EC and OC bits. Their outputs are used to set a ONE into the Interrupt Priority Chain. The remaining twelve bits in the preset counter and the PAC counter have normal connections. In the 5 PPS counter, the sign write buss is not connected, and the OC write buss sets to ONE the cores in bit positions 10, 9, 8, 7, 6, 5, 4 and 2 which is the initial condition for the next 5 PPS count. The OC bit is used as the 5 PPS pulse rate. When the counters are addressed by the program, the operation is the same as a standard erasable register. The increment circuit addresses the counter by setting a ONE into core C1, C2 or C3. When this core is cleared to ZERO by CLF, the transistor T3 clears the register and sets a ONE into core C4. When C4 is cleared the contents of the write busses are written into the counter.

Control Registers

The control registers (Fig. 3-28) are standard erasable registers with latches connected to some of the bits. Unlike the light registers, a word written into a control register may be sensed when the register is cleared. The bits which are connected to latches use two cores. One core is used as standard erasable storage; the other core is used to turn the latch on when the core is set to ONE and to turn the latch off when the core is cleared to ZERO.

---

* Note that it is conceivable that a TSN Resume would be stored in this address. When the TSN Resume was executed the light register would be turned off and a parity failure would occur. See instruction control pulse sequences.

The rate magnitude register has latches in bit positions 10, 9, 8, 7, and 6. The outputs of these latches are used to gate the outputs of the pulse generator (f/2, f/4, f/8, f/16, f/32) to form integral pulse rates from 0 to f - f/32. This rate is used to set the preset counter increment core to ONE and as command. pulses to external equipment.

The rate sign control register has latches on the sign bit and on bit P0. The outputs of these latches are used to gate the pulse rate onto one of two output lines.

The 5 PPS control register has a latch on the sign bit. When this latch is on, the twenty four pulse per second rate (f/16) is allowed to set a ONE into'the 5 PPS position of the increment priority chain.

The 5:1, 1:1 ON/OFF control register has latches on the sign bit and bit 10. When the latch connected to bit 10 is on, the positive and negative pulses coming to the computer set a ONE in the appropriate cores in the Increment Priority Chain. If the latch connected to the sign bit is on, the positive and negative incoming pulses set five cores in the Increment Priority Chain to ONE.

'Resume and Normal Stop

Resume and normal stop (Fig. 3-29) are addressable computer functions. When the rope core at the resume address (octal 764) is switched, the transistor T1 generates the resume pulse for the interrupt priority chain. If the normal stop rope core is addressed, the transistor T2 turns off the RUN latch causing the computer to stop.

3.8 Special Circuits

Pulse Generator

A binary counter (Fig. **3-30)** is used to generate the vari-. ous pulse rates required by the computer. Thirteen stages are

needed to count the 100 kc clock rate down to 12 pulses per second. Slower rates are generated using the counter registers. The output pulses from the binary counter are generated by the ZERO to ONE transition of a counter stage. There is only one such transition occurring in the counter at any given time, allowing the outputs to be combined for various pulse rates. This combined output rate is also fairly uniform, with the pulse density differing by a maximum factor of two.

The circuit of a counter section is shown in Fig. 3-31. This circuit is basically a two stage priority chain. The section of the binary counter is in state ZERO when cores 1 and 2 are at ZERO. The section is in state ONE when core 1 is at ONE and core 2 is at ZERO. Assuming both cores are at ZERO, the input to A sets core 1 to ONE, turning on transistors T3 and T4. Transistor T3 shunts the input current away from core 2 and transistor T4 provides the pulse output from the section. The inputs occur at the β phase of the clock and the clear current on winding EF of core 2 occurs at α time. Since core 2 is still at state ZERO, the clear current has no effect. The next input pulse is able to set core 2 to ONE turning transistor T2 on. The input to the next section of the counter is taken from the collector of transistor T2. At the next α time, core 2 is cleared to ZERO, turning on transistor T1 which in turn clears core 1 to ZERO.

Decoder

The computer receives two asynchronous square waves from the optical encoder with 0 and -28 v levels (Fig. 3-32). The magnitude information of the encoder motion is given by the changes of state in both square waves and the direction of motion is given by the phase relationship of the square waves. The decoder translates these signals into synchronous trains of positive and negative pulse according to Table 3-1.

The circuit shown in Fig. 3-33 limits the signal to 0 and -10 v, increases the rise time of the transitions and provides both the normal and complemented outputs. The decoding circuit: is shown in Fig. 3-34, When A changes state, core α switches, turning on two of the four output transistors (T1 and T4 for A changing to 0 v, T2 and T3 for A changing to -1.0 v), and B or $\overline{B}$ provides a ground for one of these transistors, yielding the positive or negative increment pulse, The operation of core β is similar, with the function of A and B interchanged.

The -V gate is used to synchronize the decoder with the computer, The gate is initiated by Time Pulse 6 and feedback taken from the positive and negative increment outputs insures that the gate is long enough to allow complete switching. If A or B changes state near the end of Time Pulse 6, the gate is held on Time Pulse 7 by the feedback from the outputs. This is permissible because the only constraint on the output is that it cannot occur at Time Pulse 9 when the Increment Priority Chain is scanned,

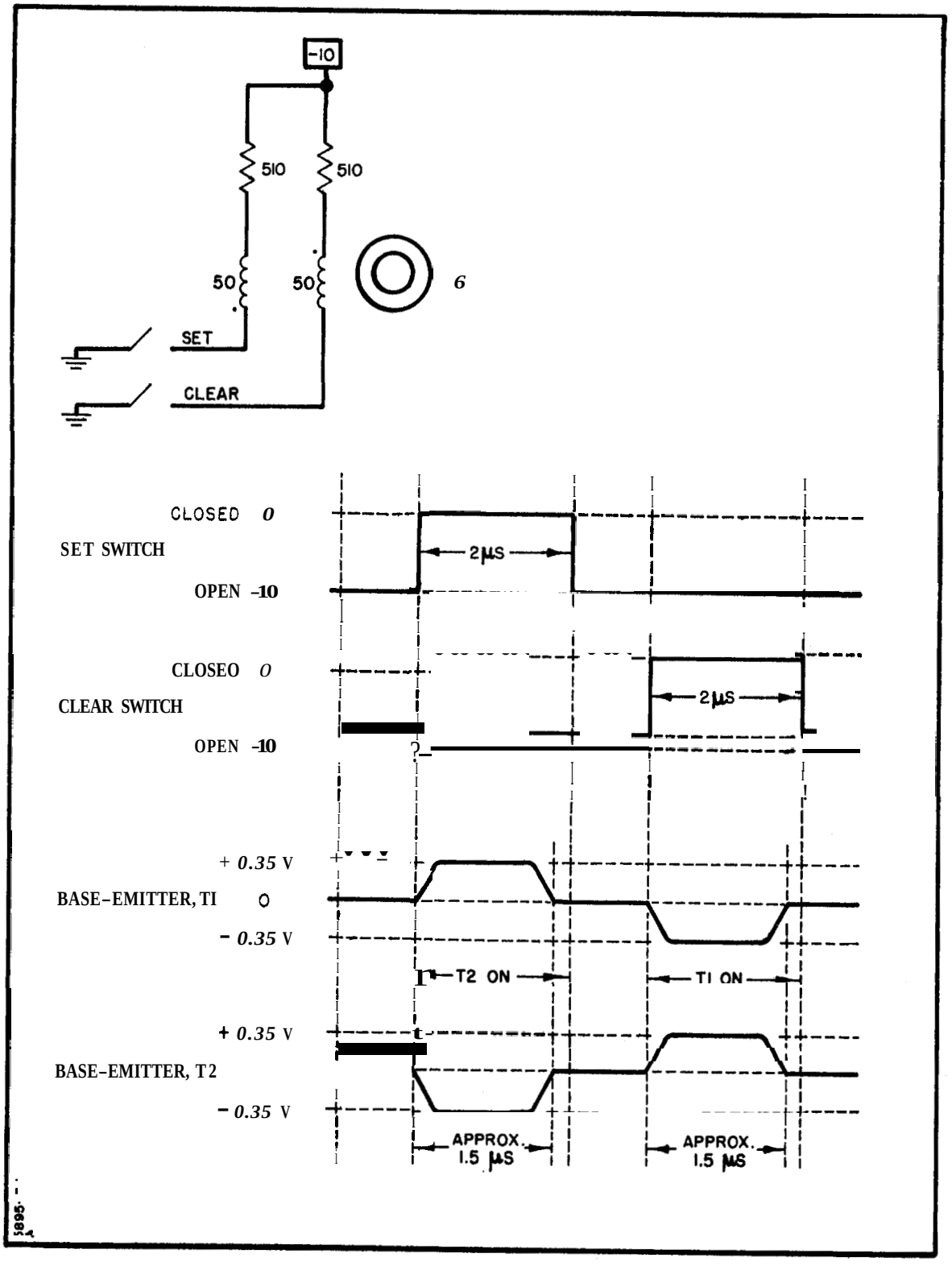


Fig. 3-1 Basic circuit configuration

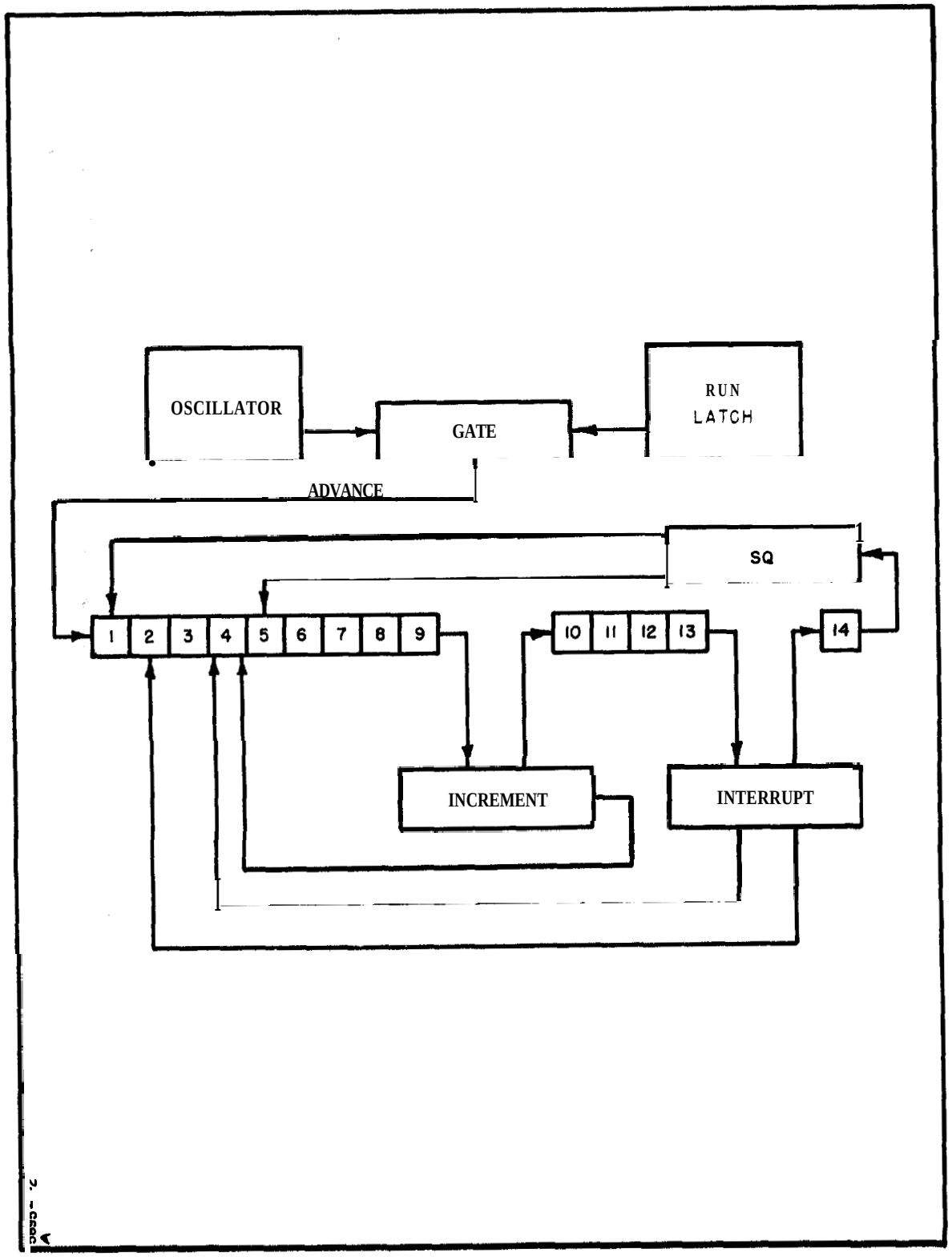


Fig. 3-2 Clock block diagram

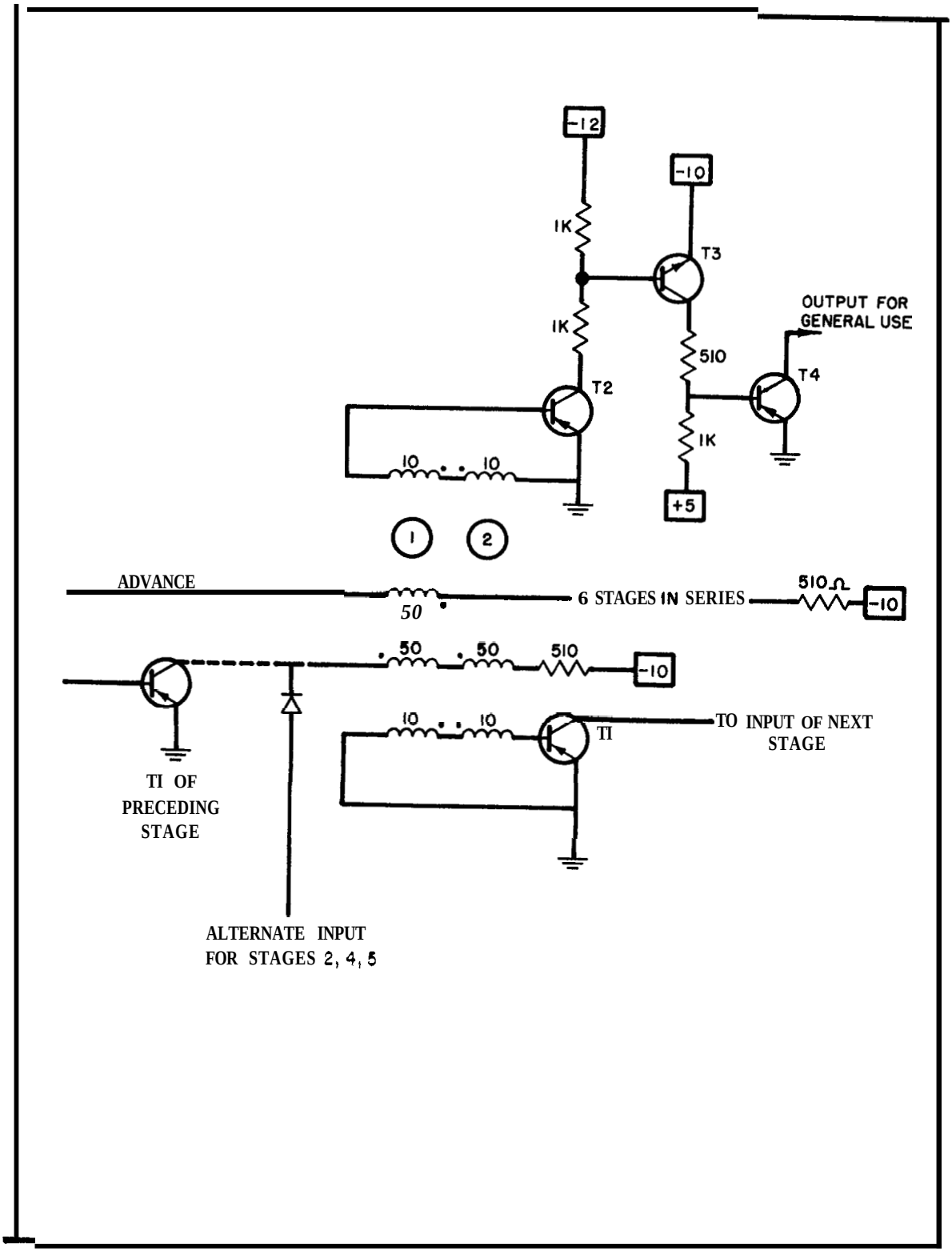


Fig. 3-3 One stage of clock shift register

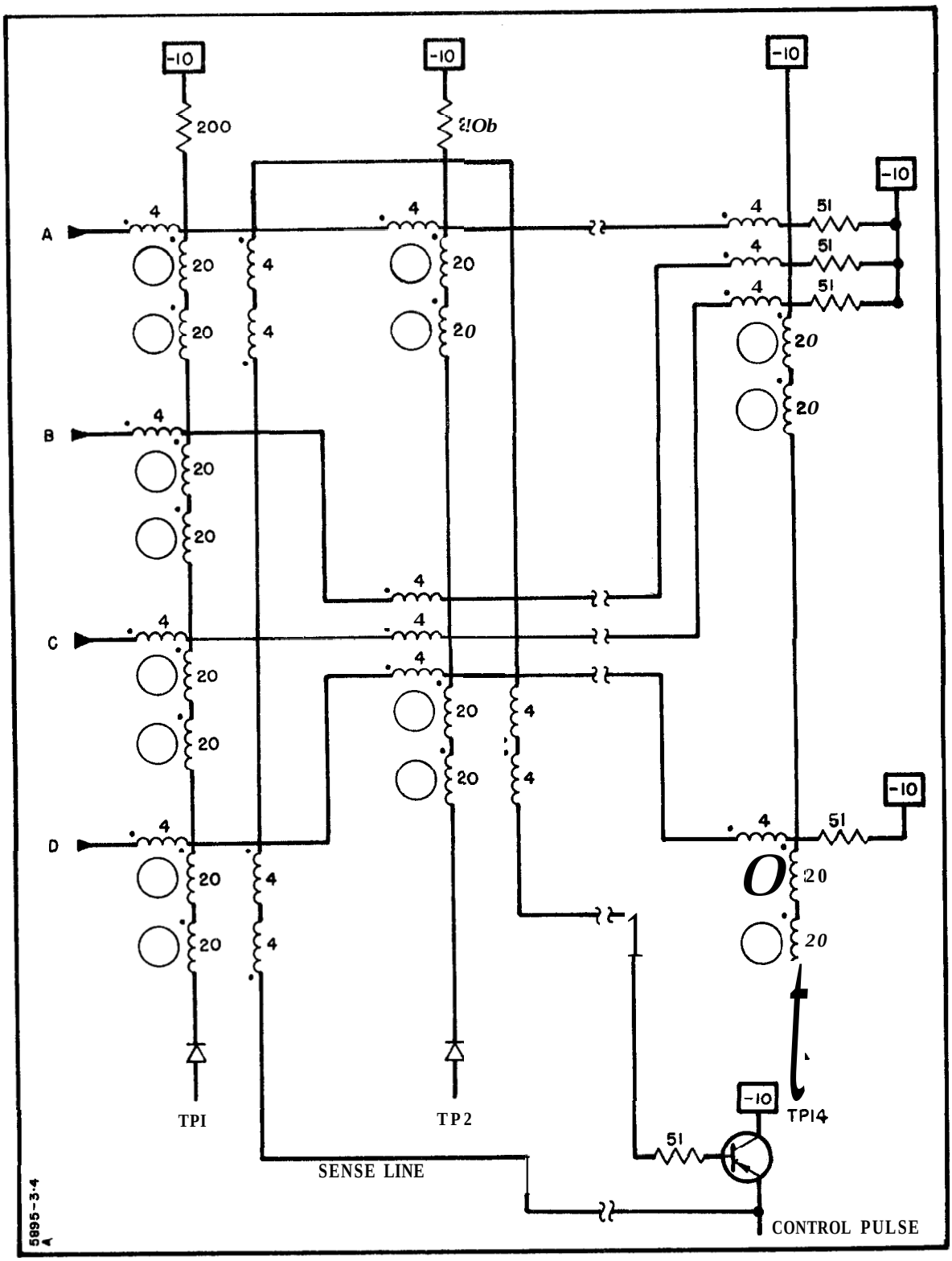


Fig. 3-4 Sequence generator

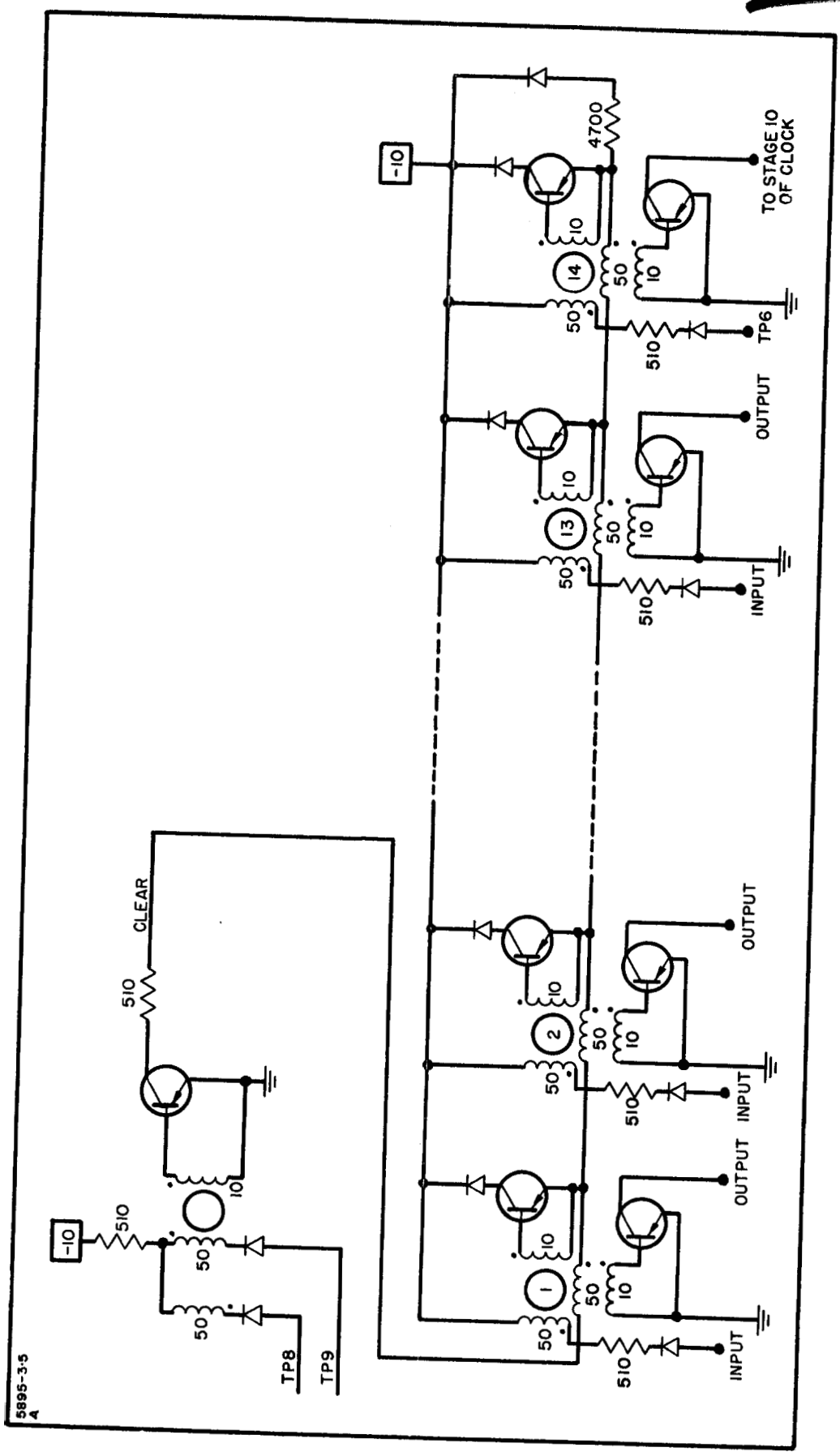


Fig. 3-5 Increment priority chain

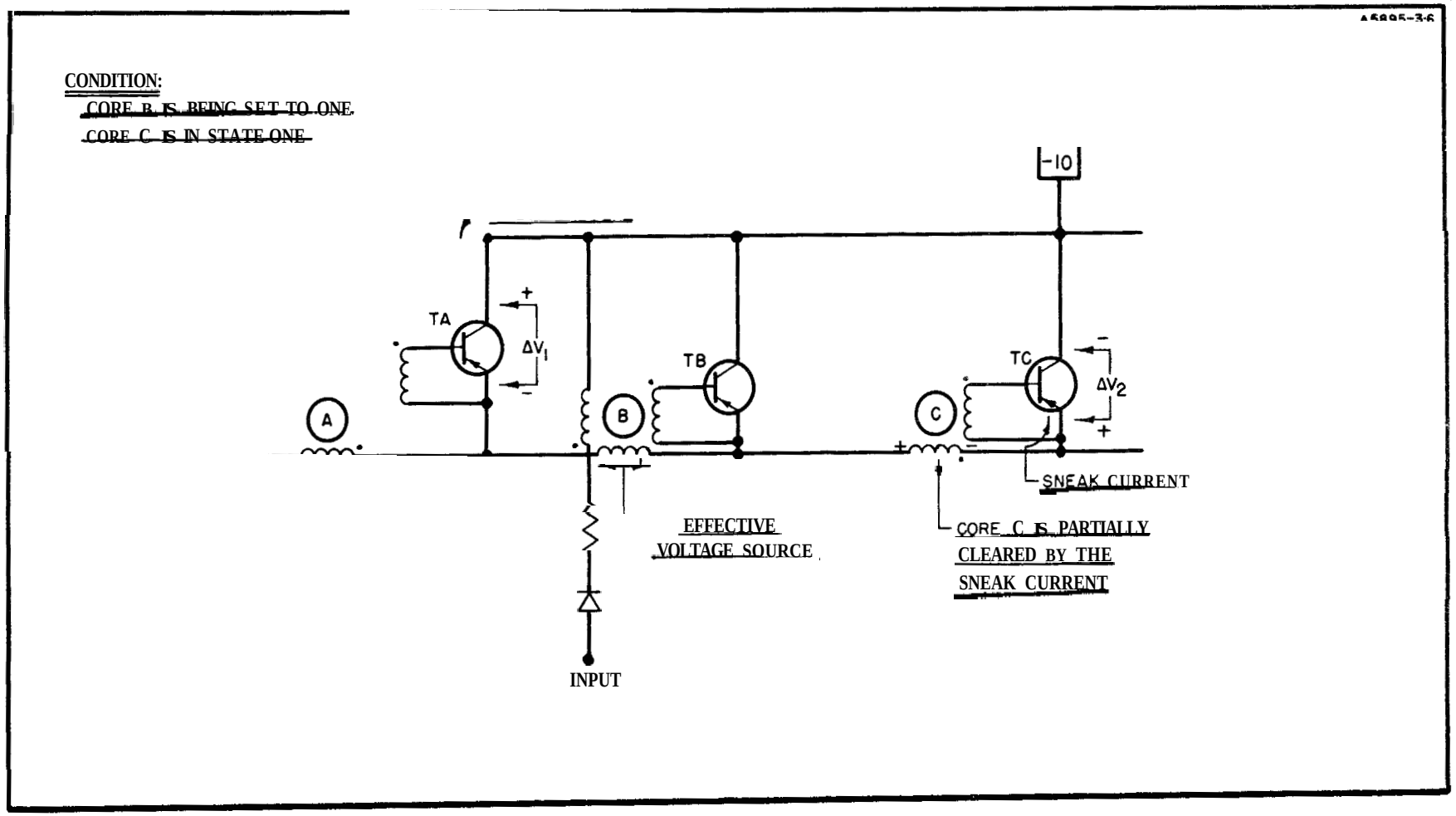


Fig. 3–6 Sneak current paths in priority chain

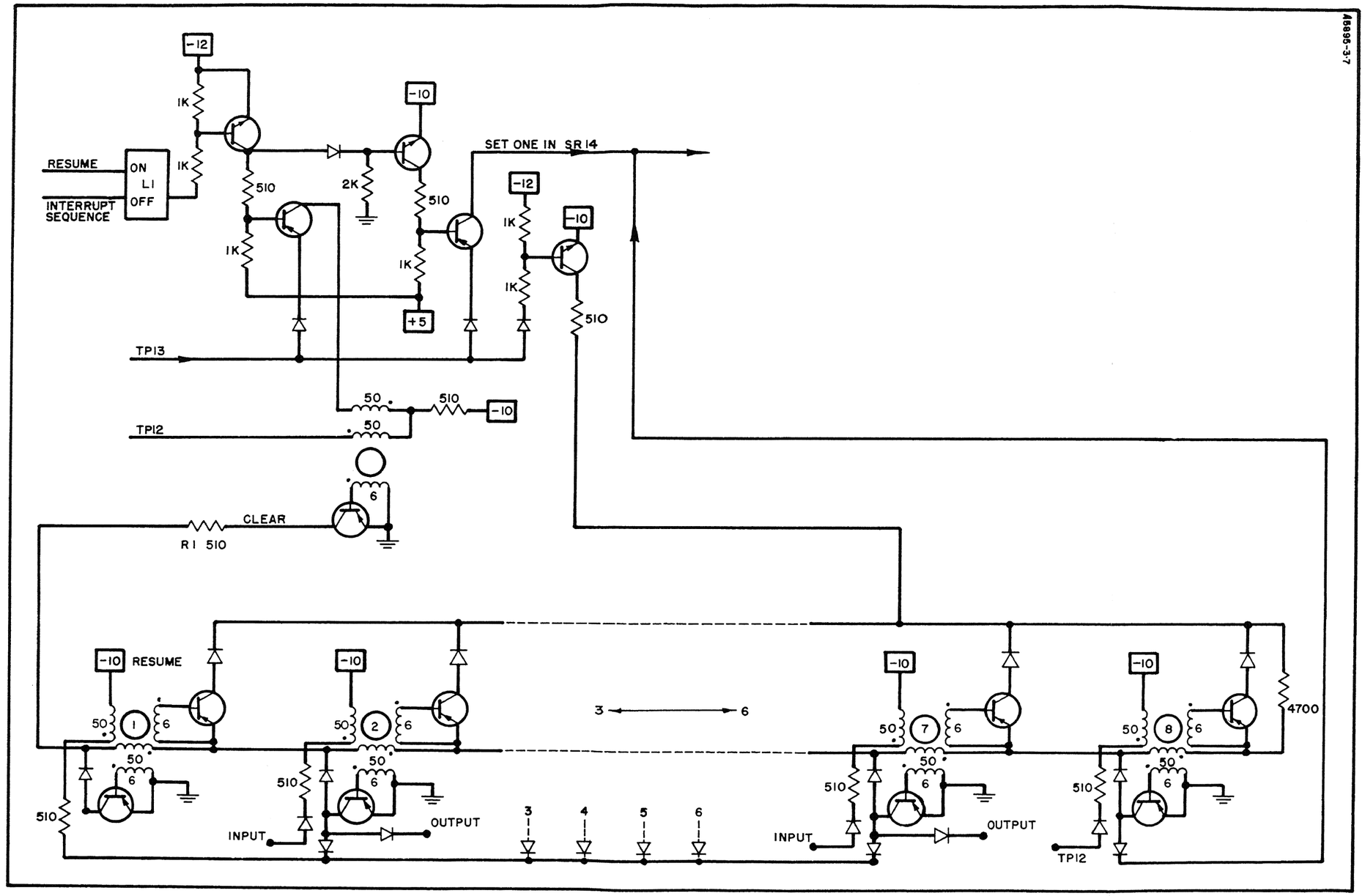


Fig. 3-7 Interrupt priority chain

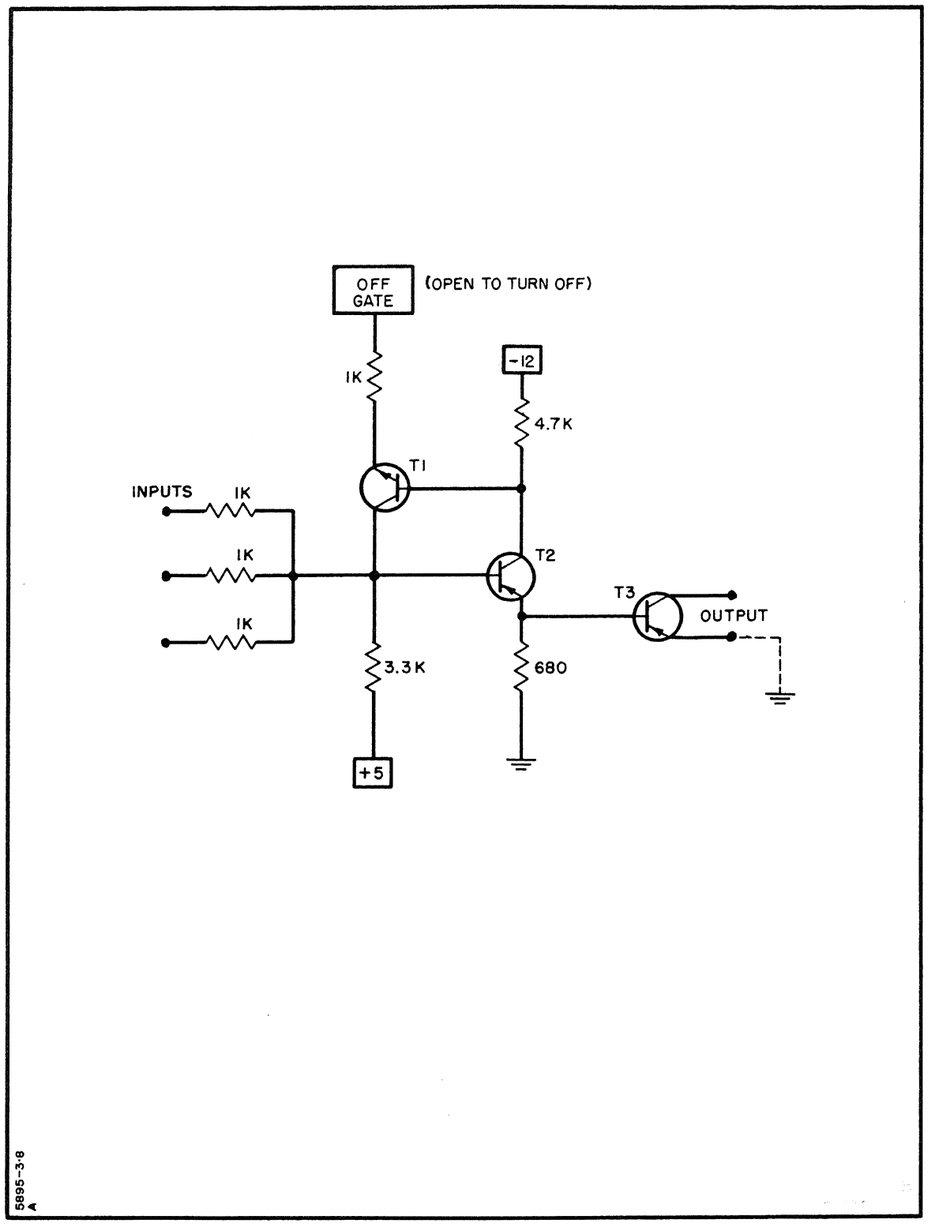


Fig. 3-8 Latch 1

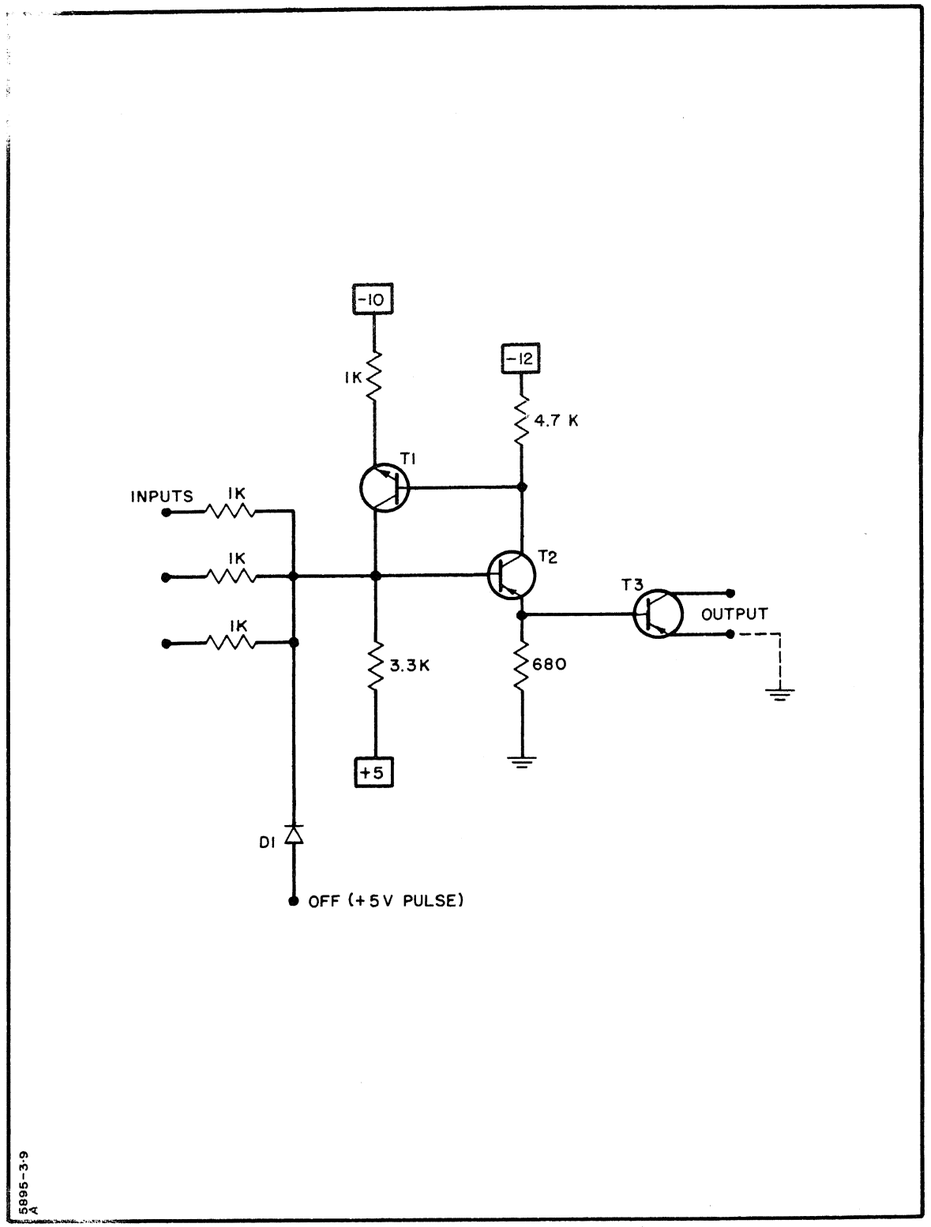


Fig. 3-9 Latch 2

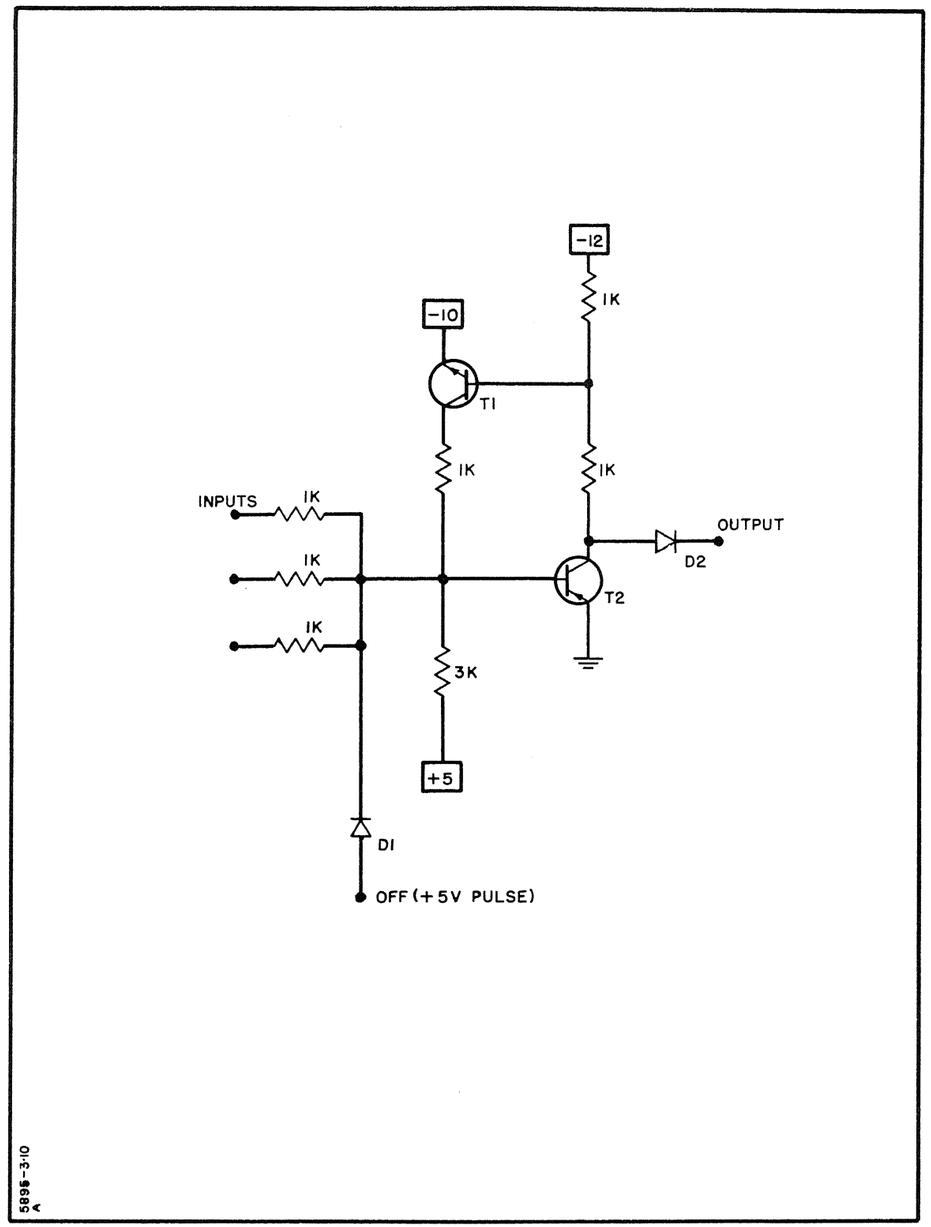


Fig. 3-10 Latch 3

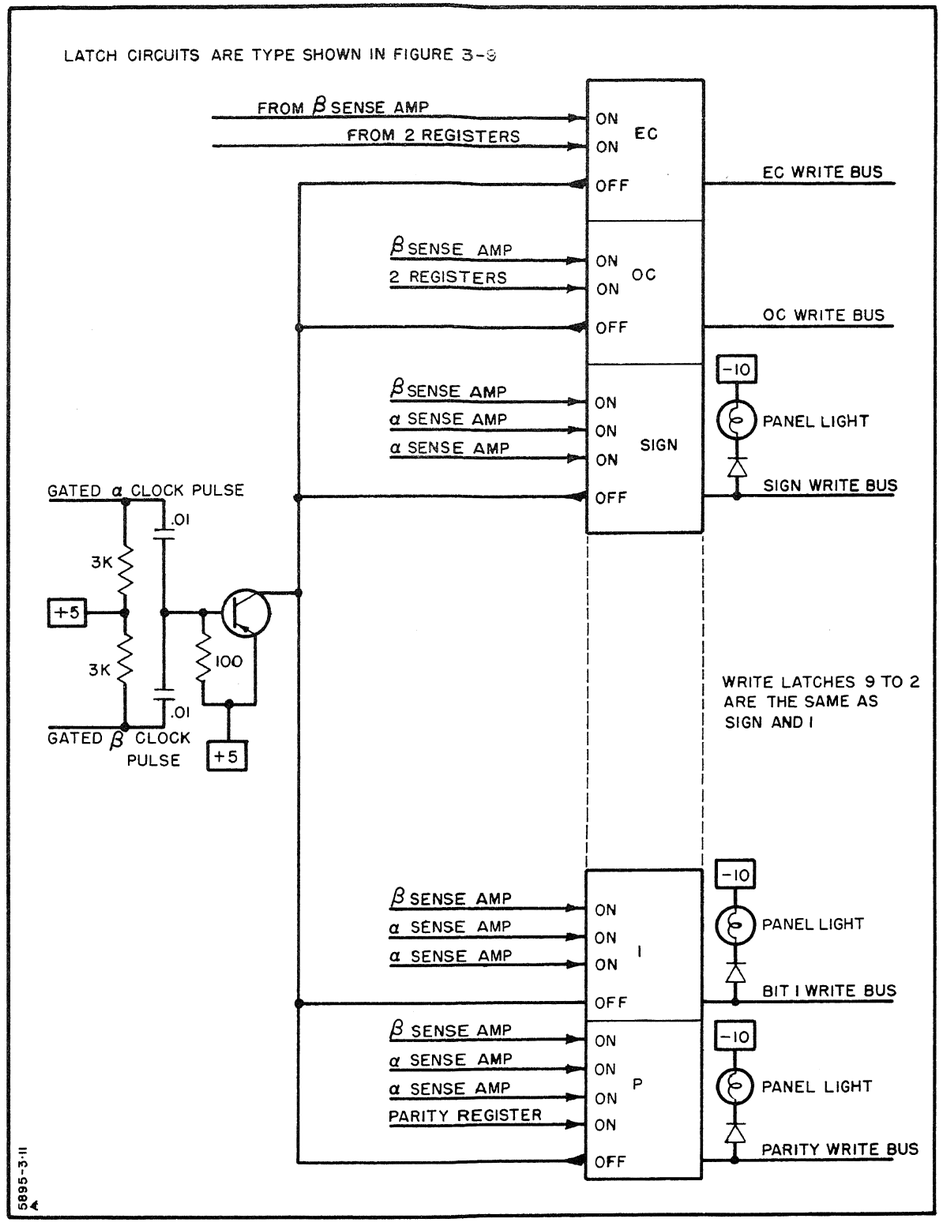


Fig. 3-11 Write latches (latch circuits are type shown in Fig. 3-9)

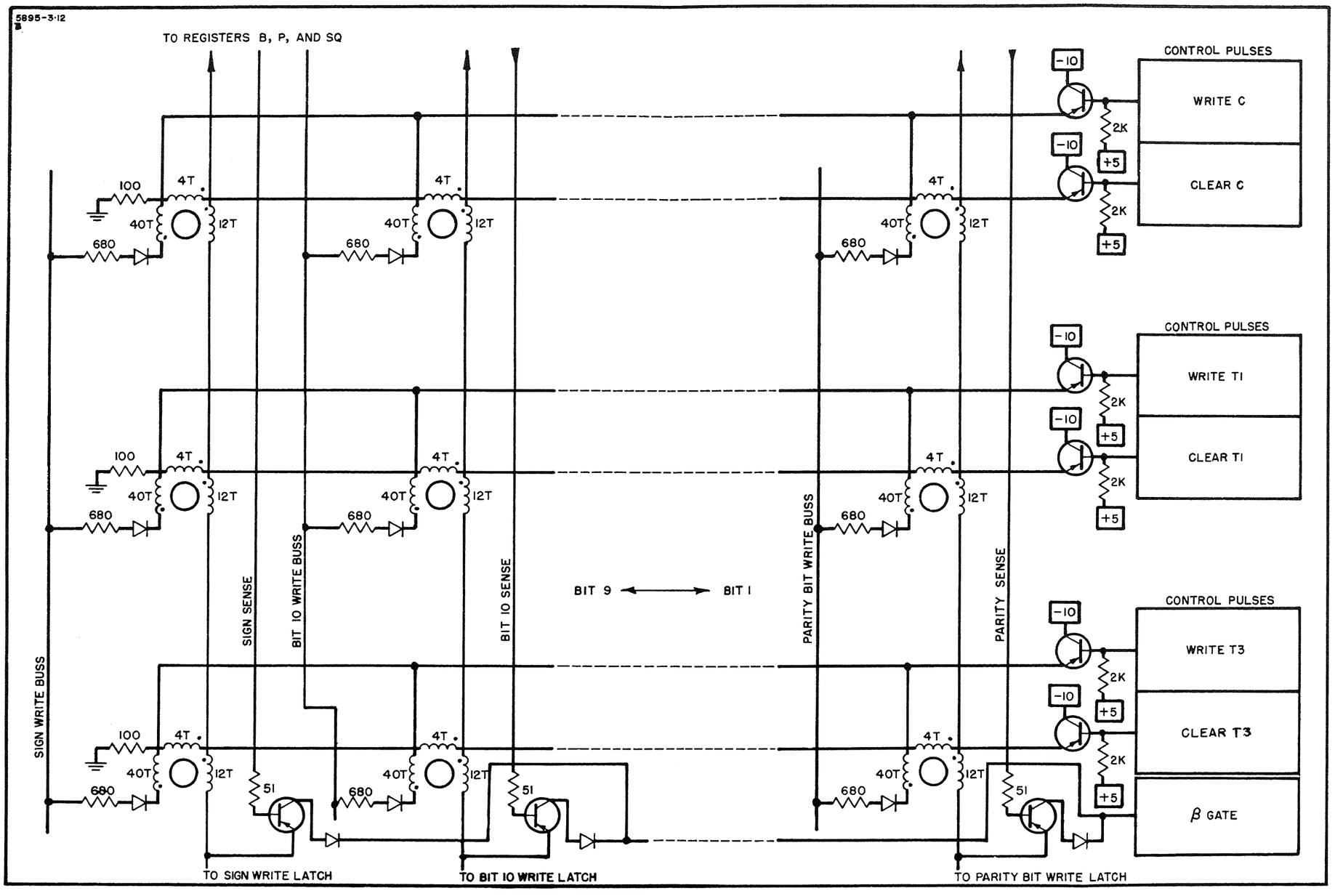


Fig. 3-12 Standard central registers

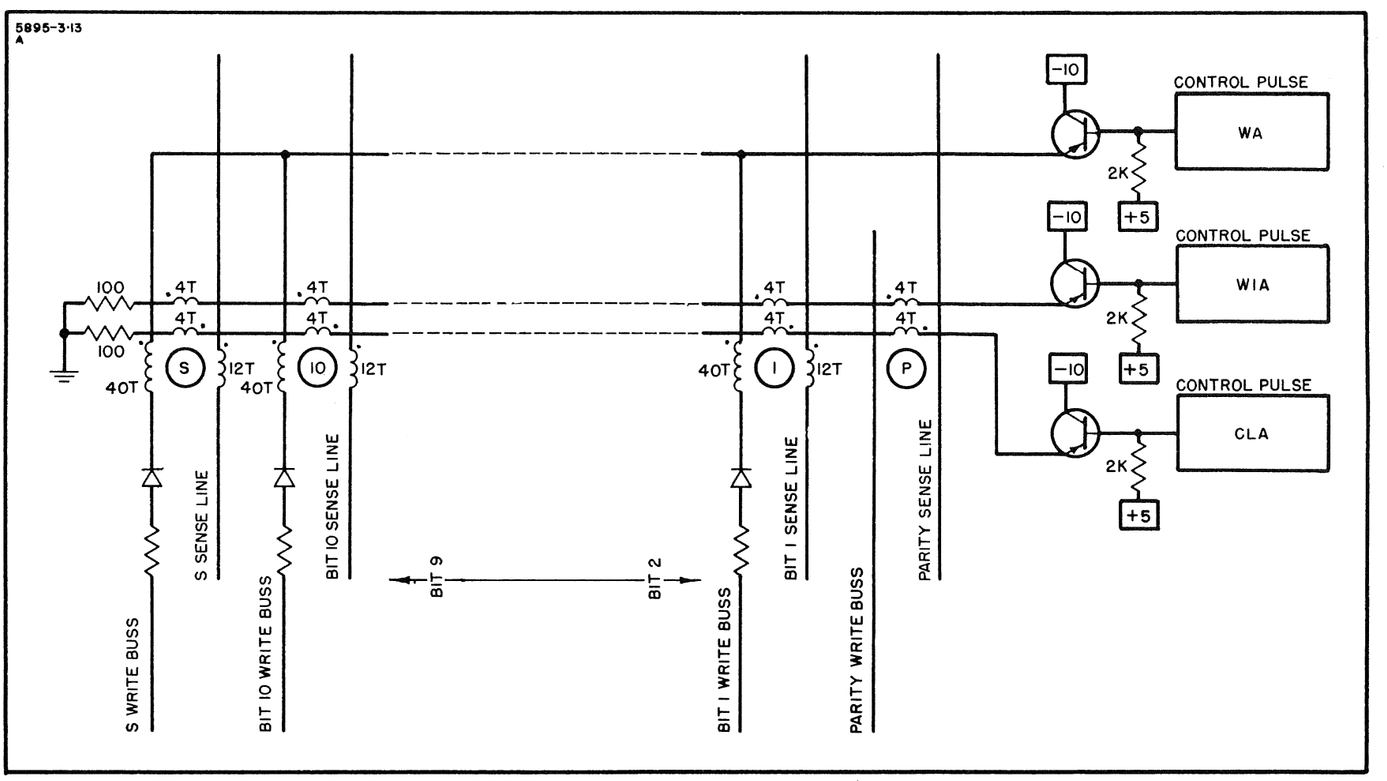


Fig. 3-13 A register

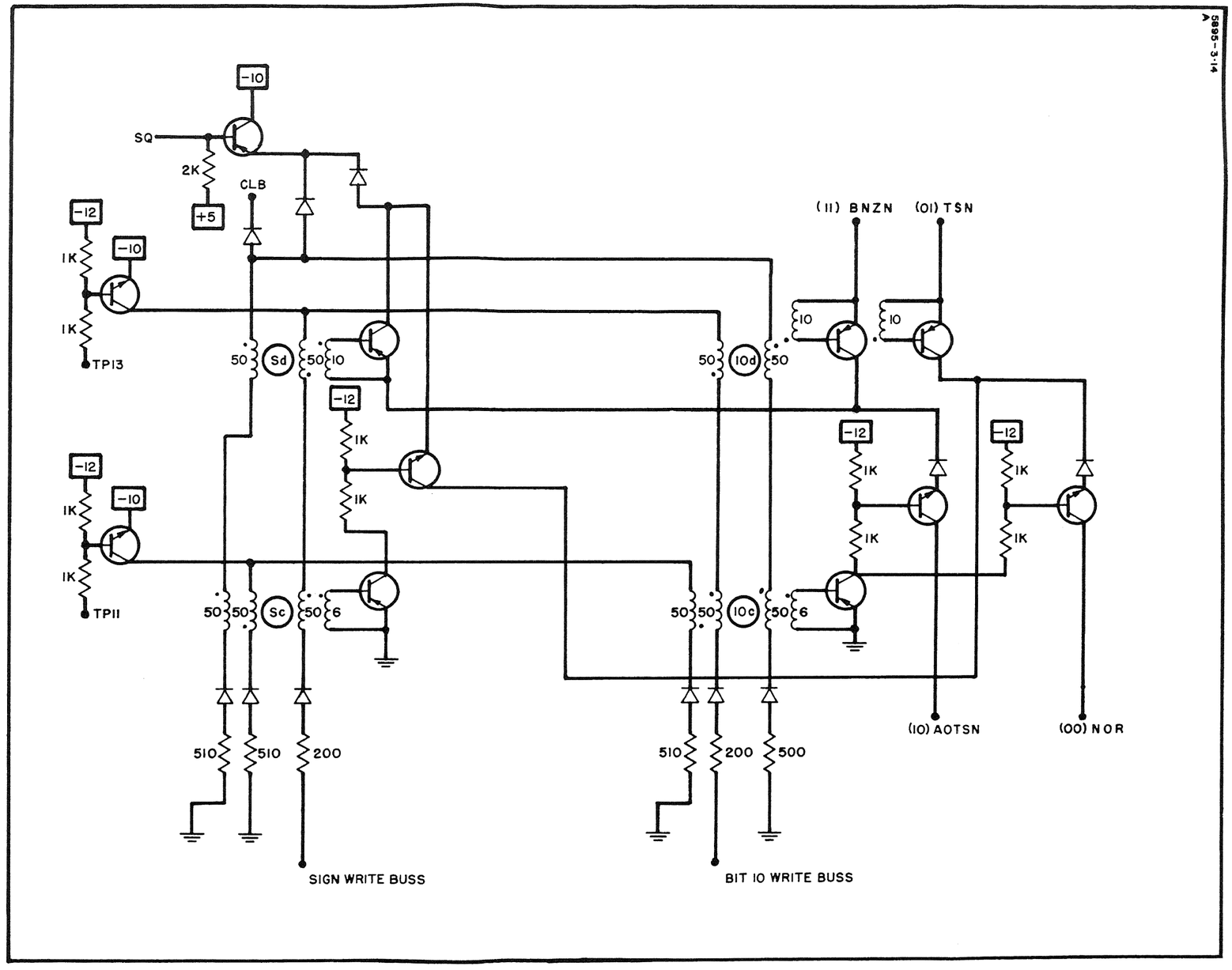


Fig. 3-14 SQ register

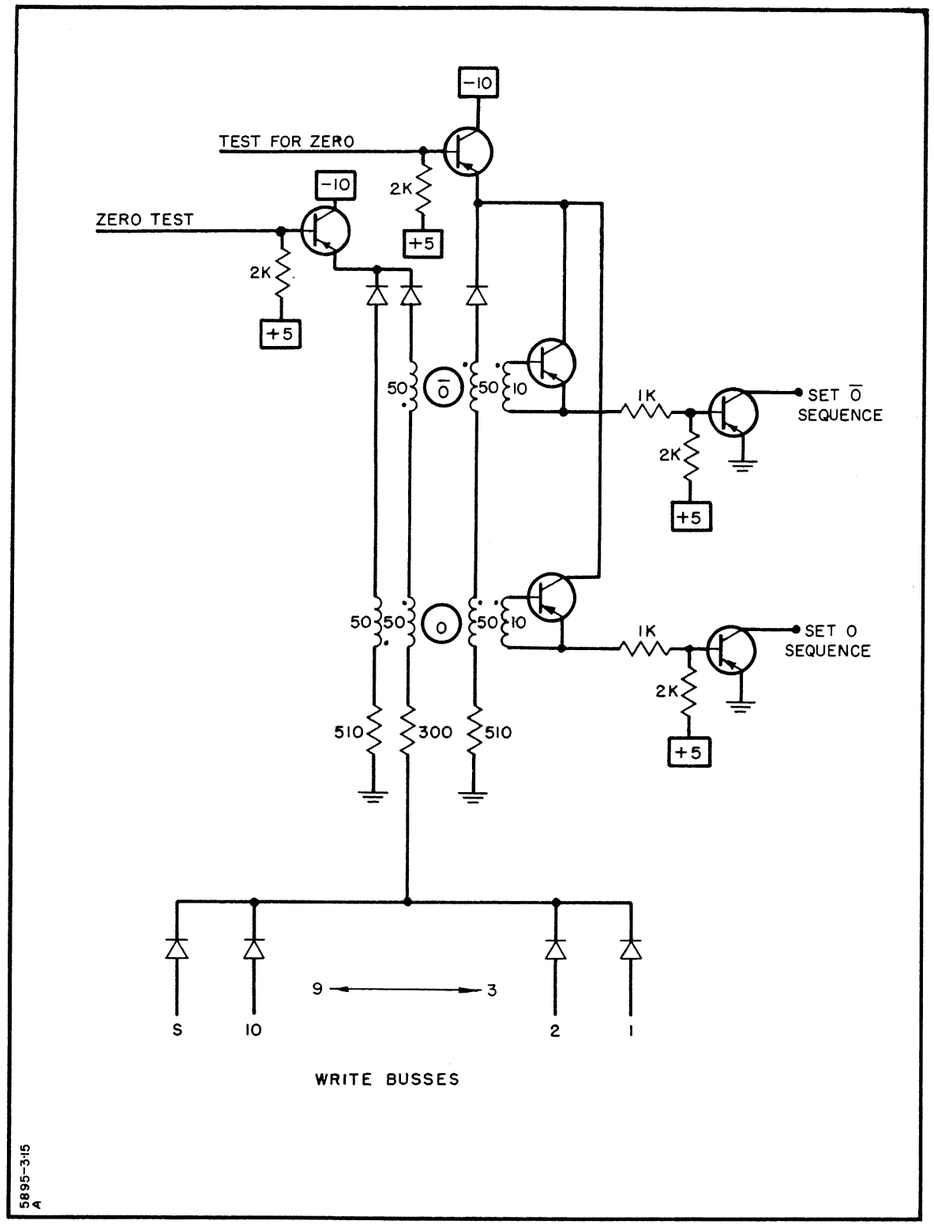


Fig. 3-15 Zero test

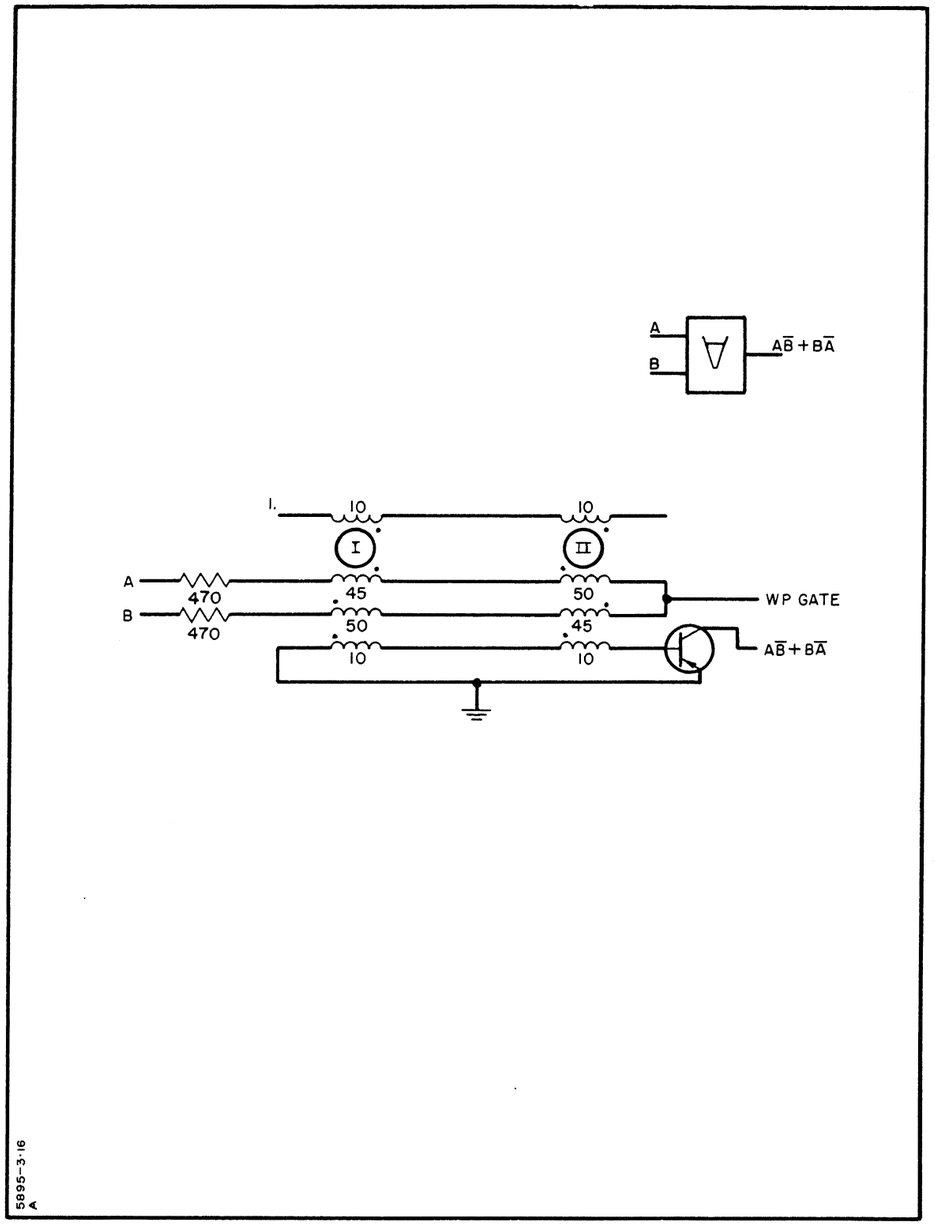


Fig. 3-16 "Exclusive Or" circuit

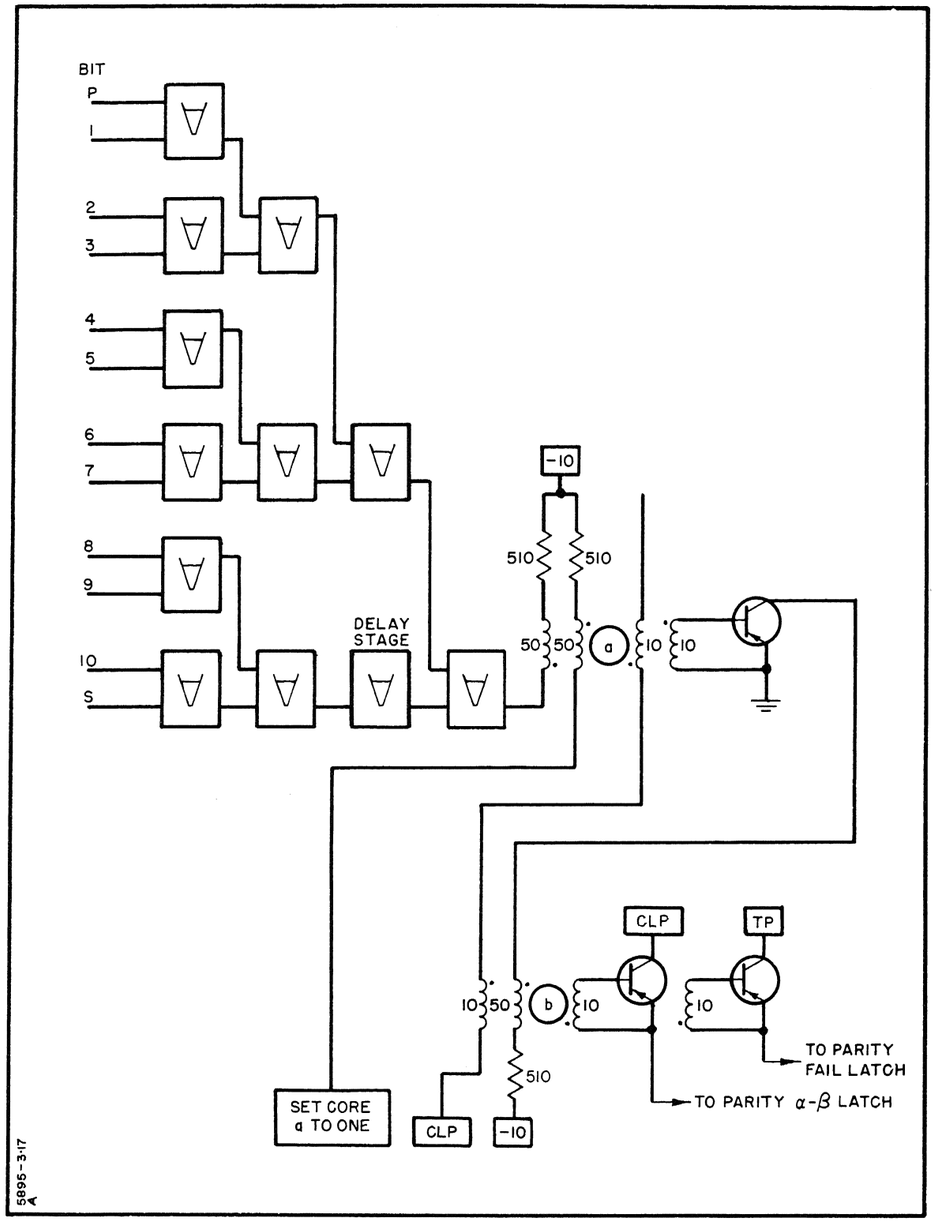


Fig. 3-17 Parity tree

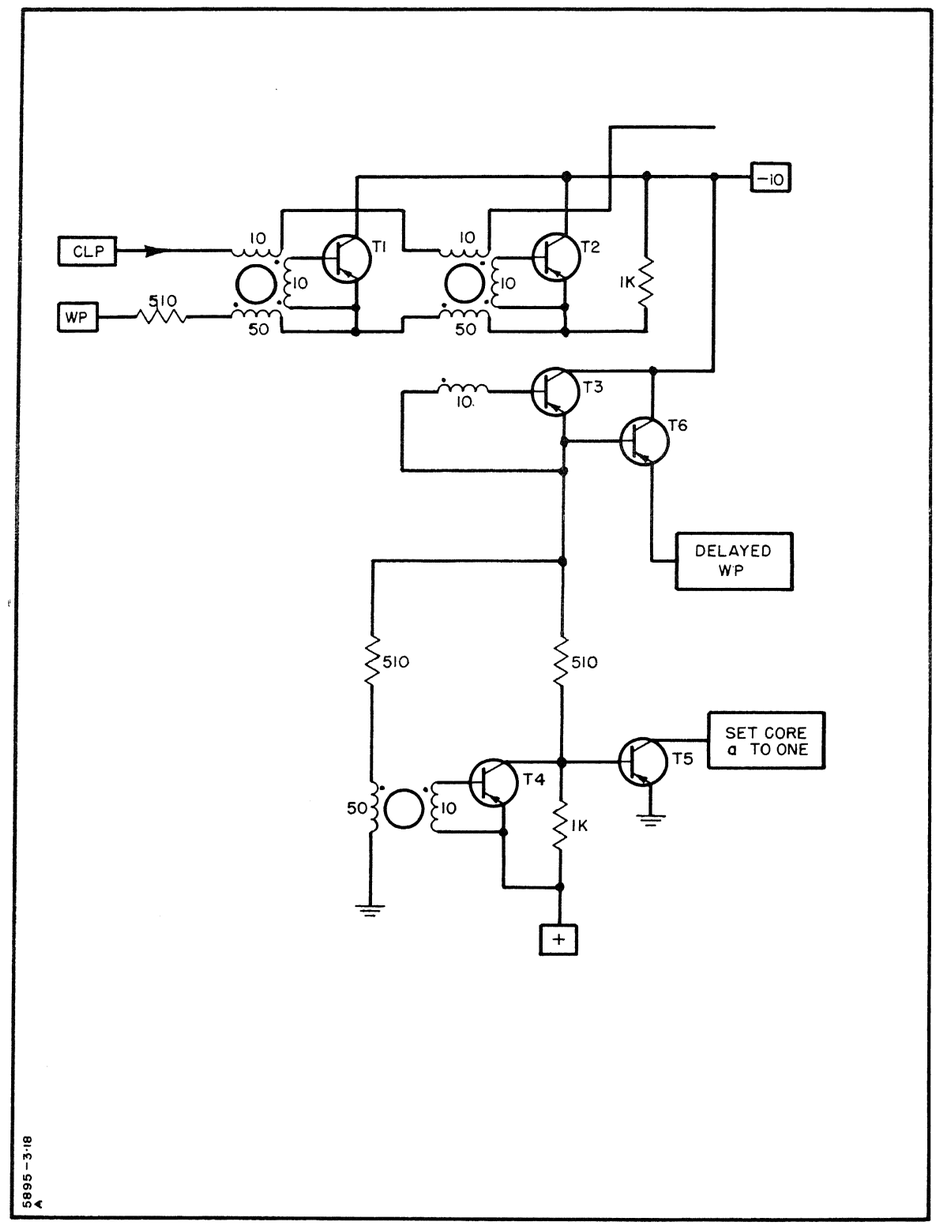


Fig. 3-18 Delayed WP

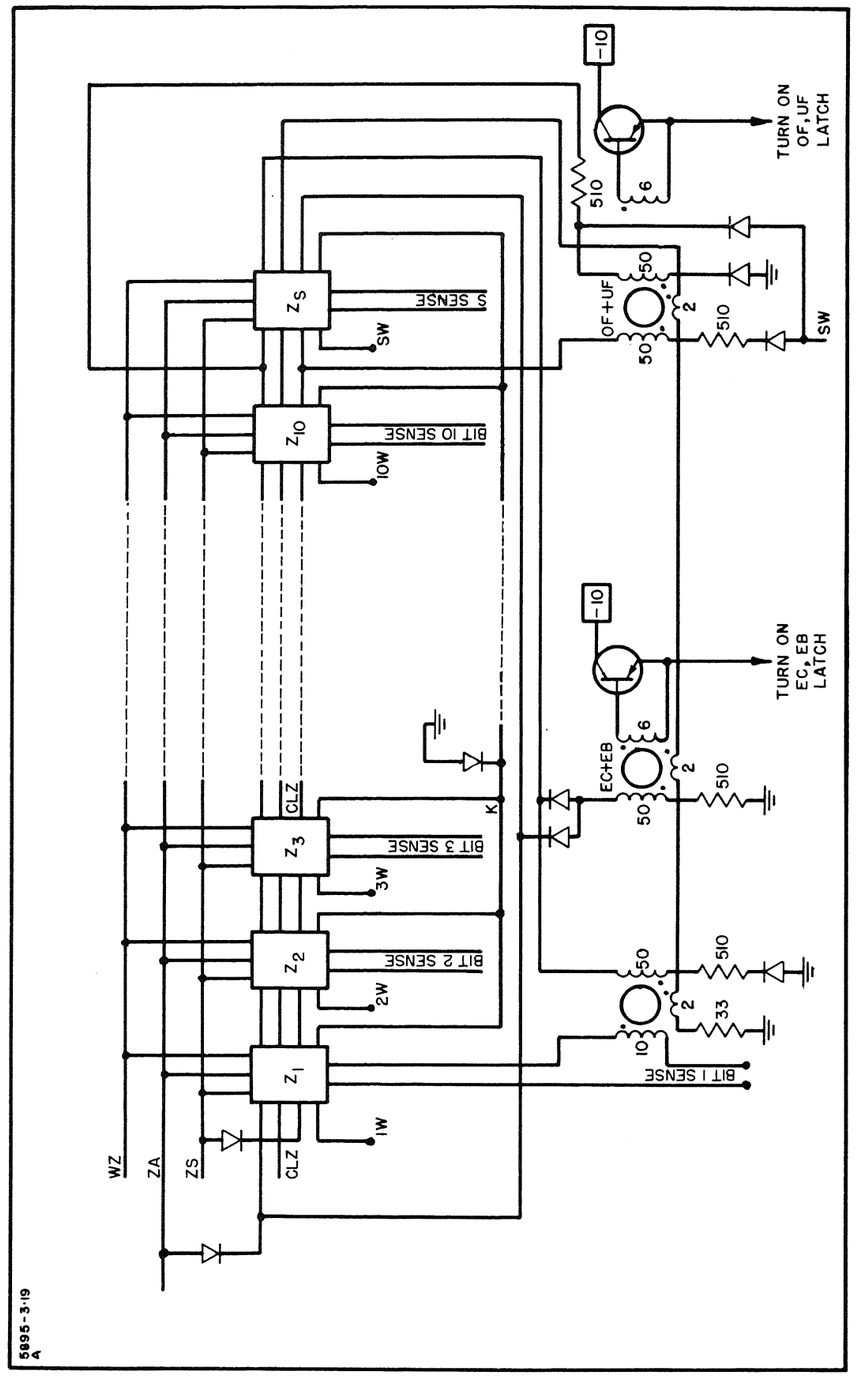


Fig. 3-19 Block diagram of Z register

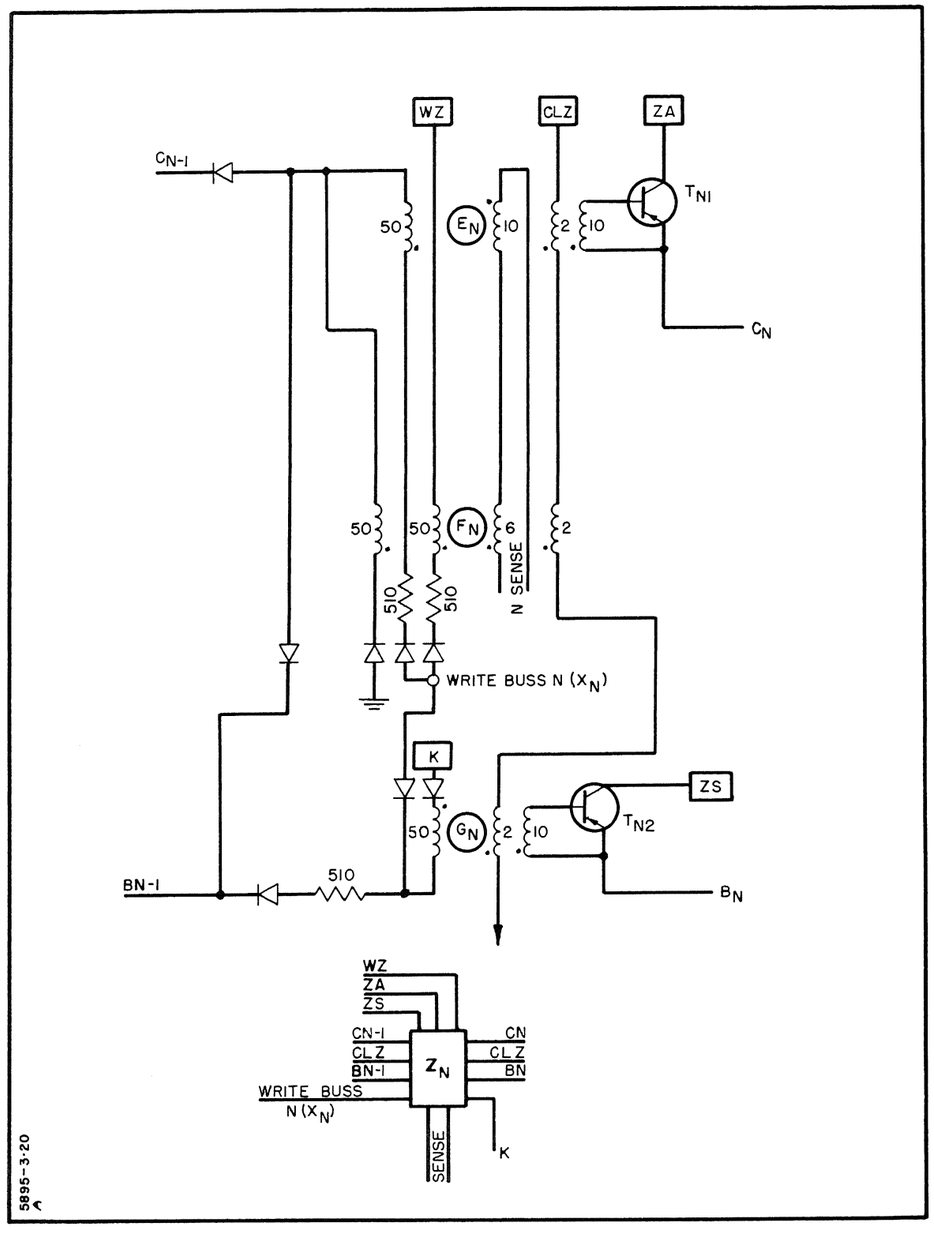


Fig. 3-20 Z INC + OR -

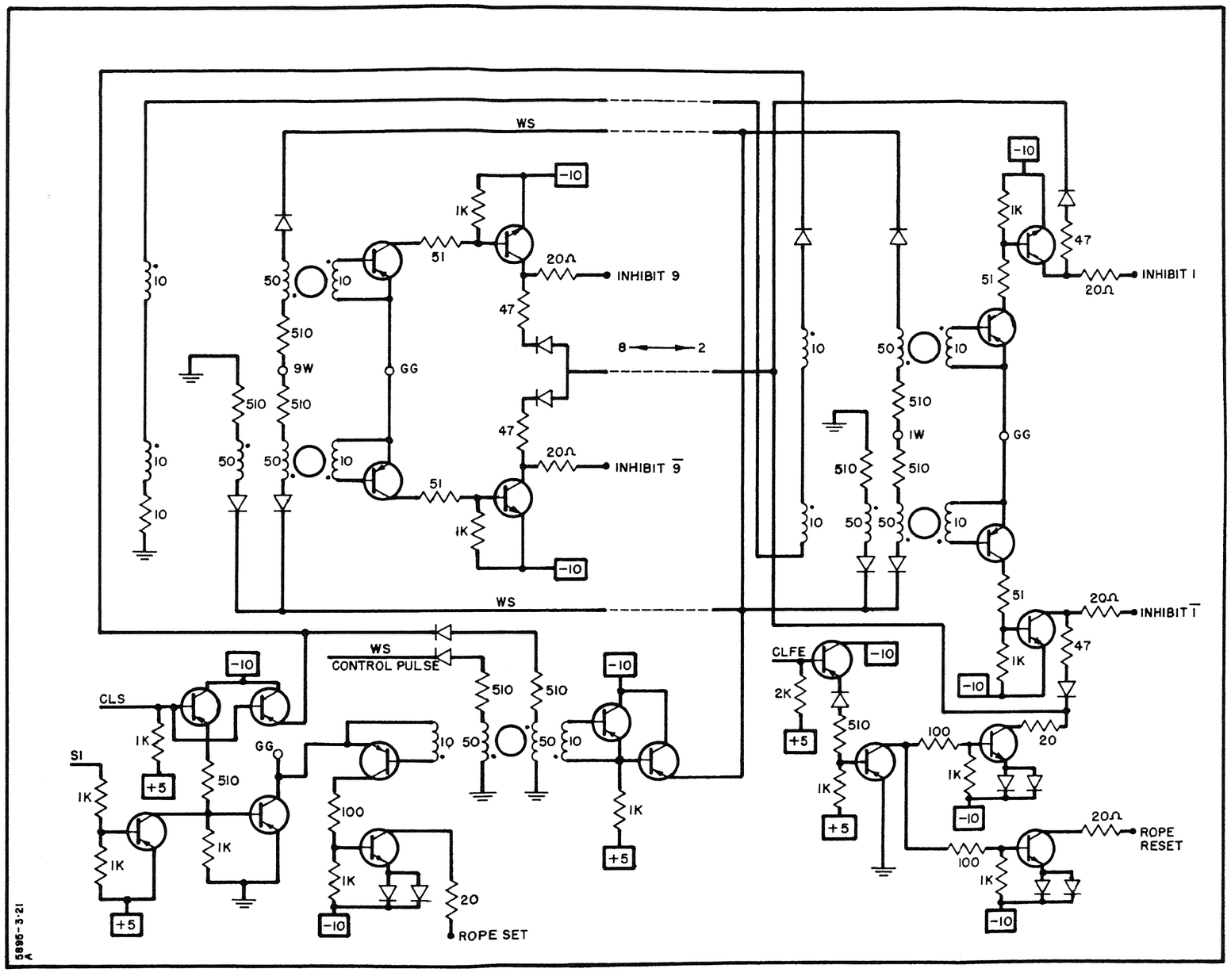


Fig. 3-21 S register

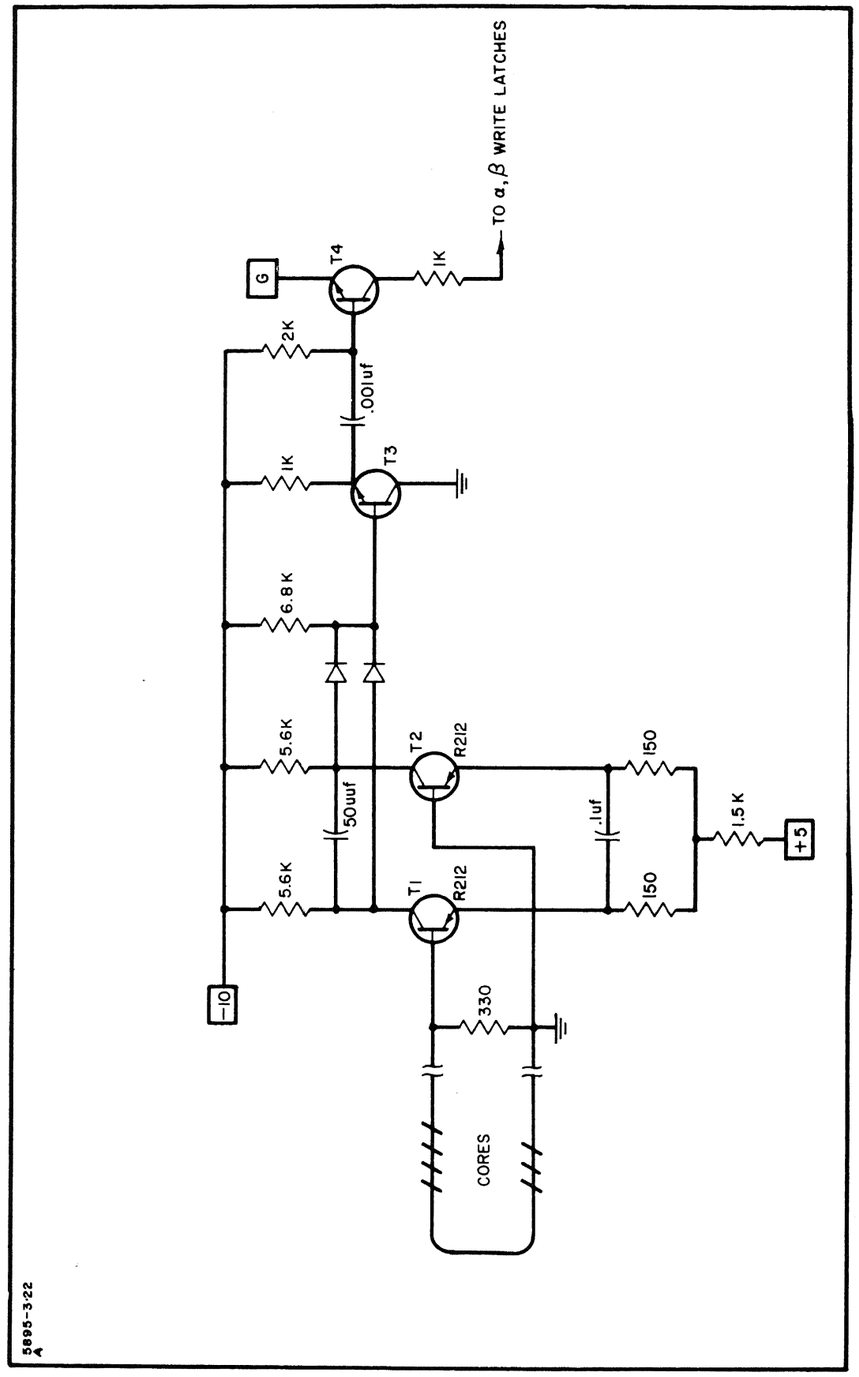


Fig. 3-22 Rope sense amplifier (1 of 12)

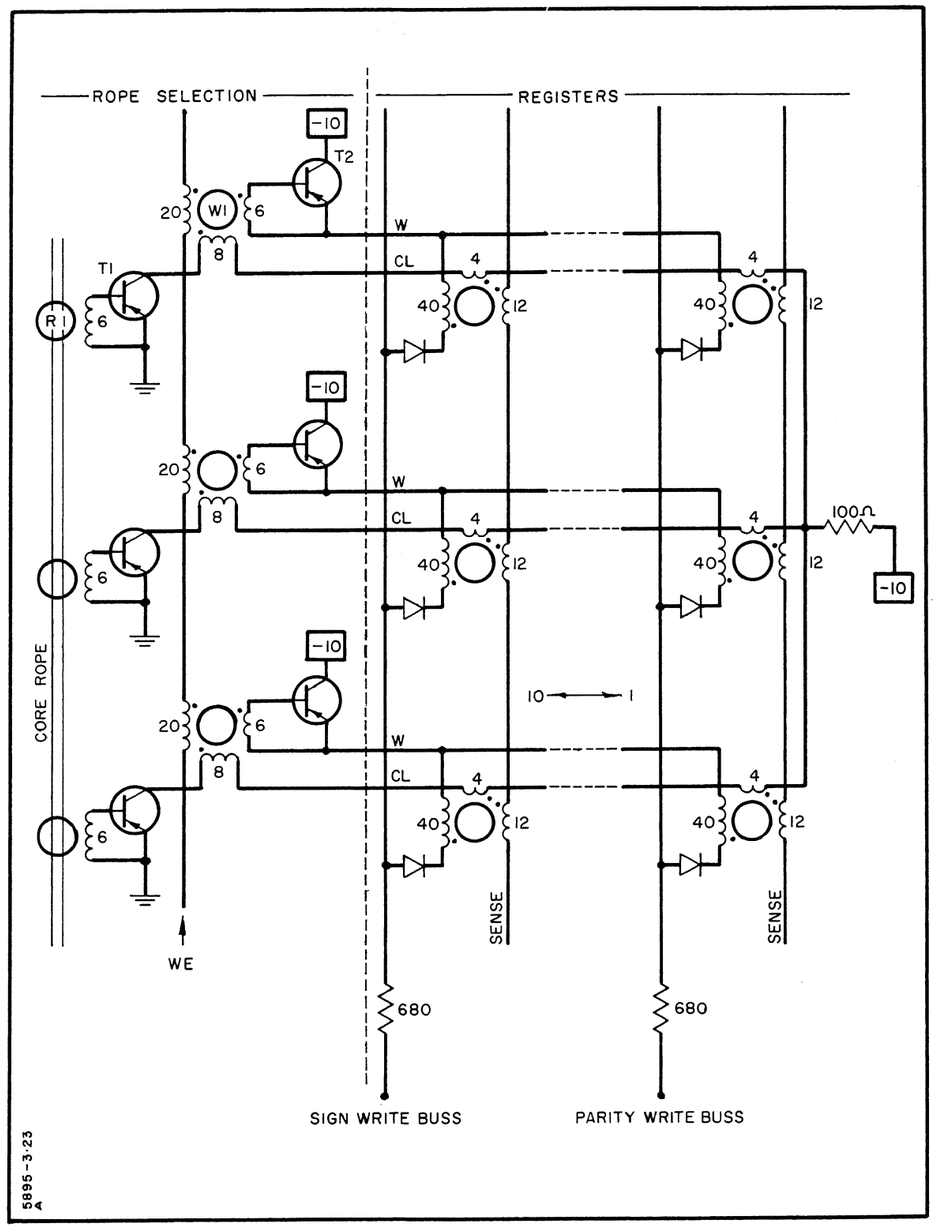


Fig. 3-23 Standard erasable register

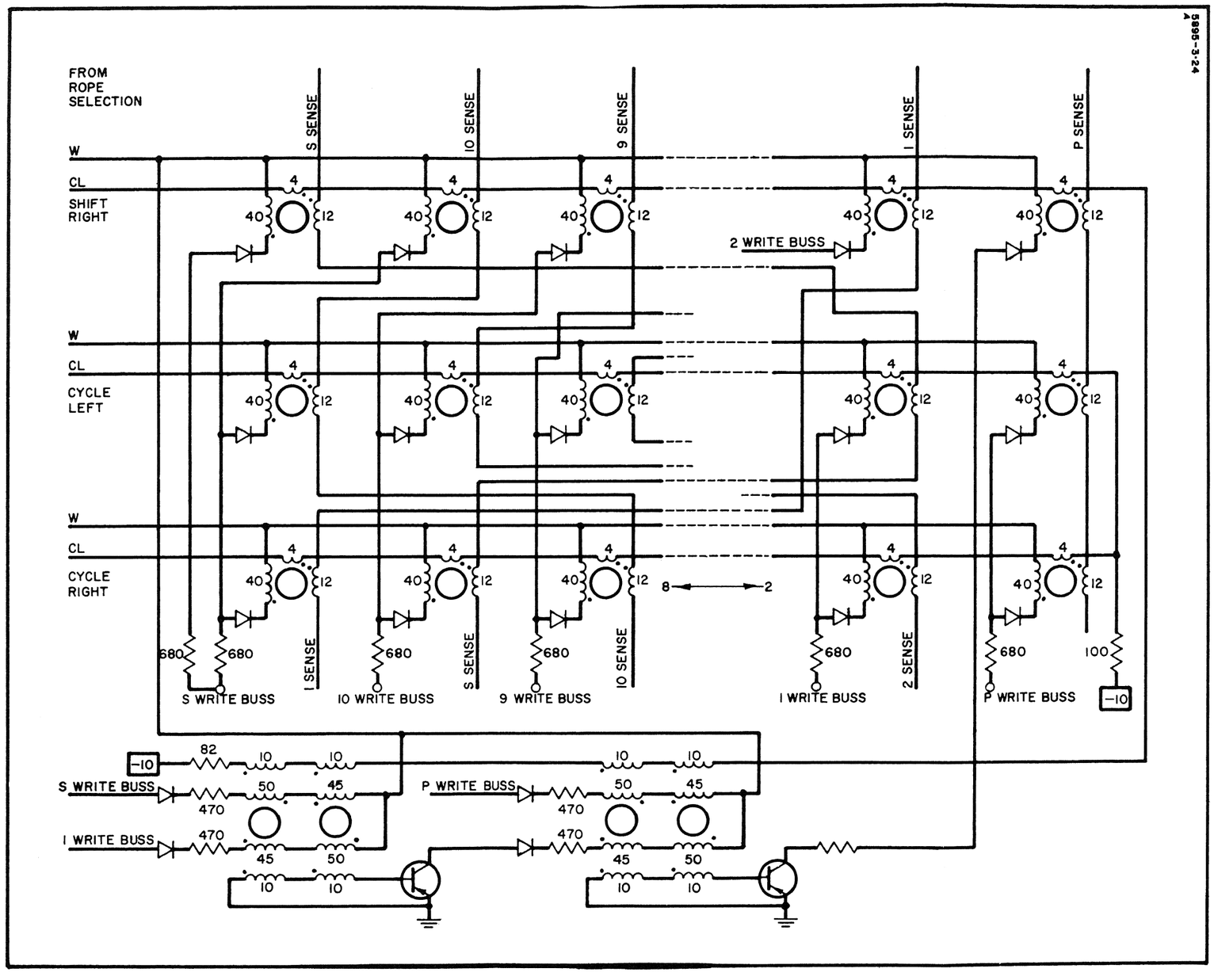


Fig. 3-24 Cycle and shift registers

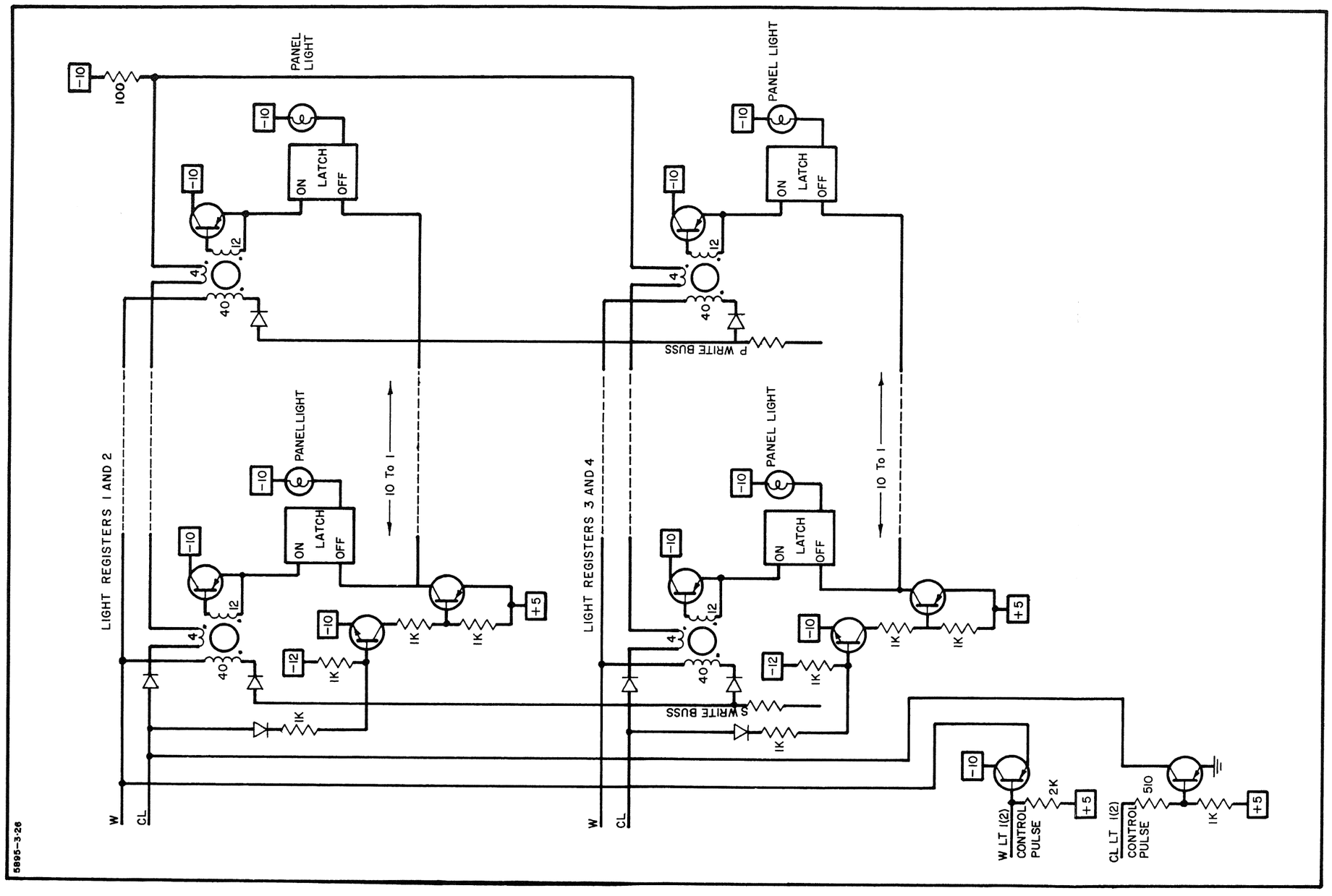


Fig. 3-26 Light registers

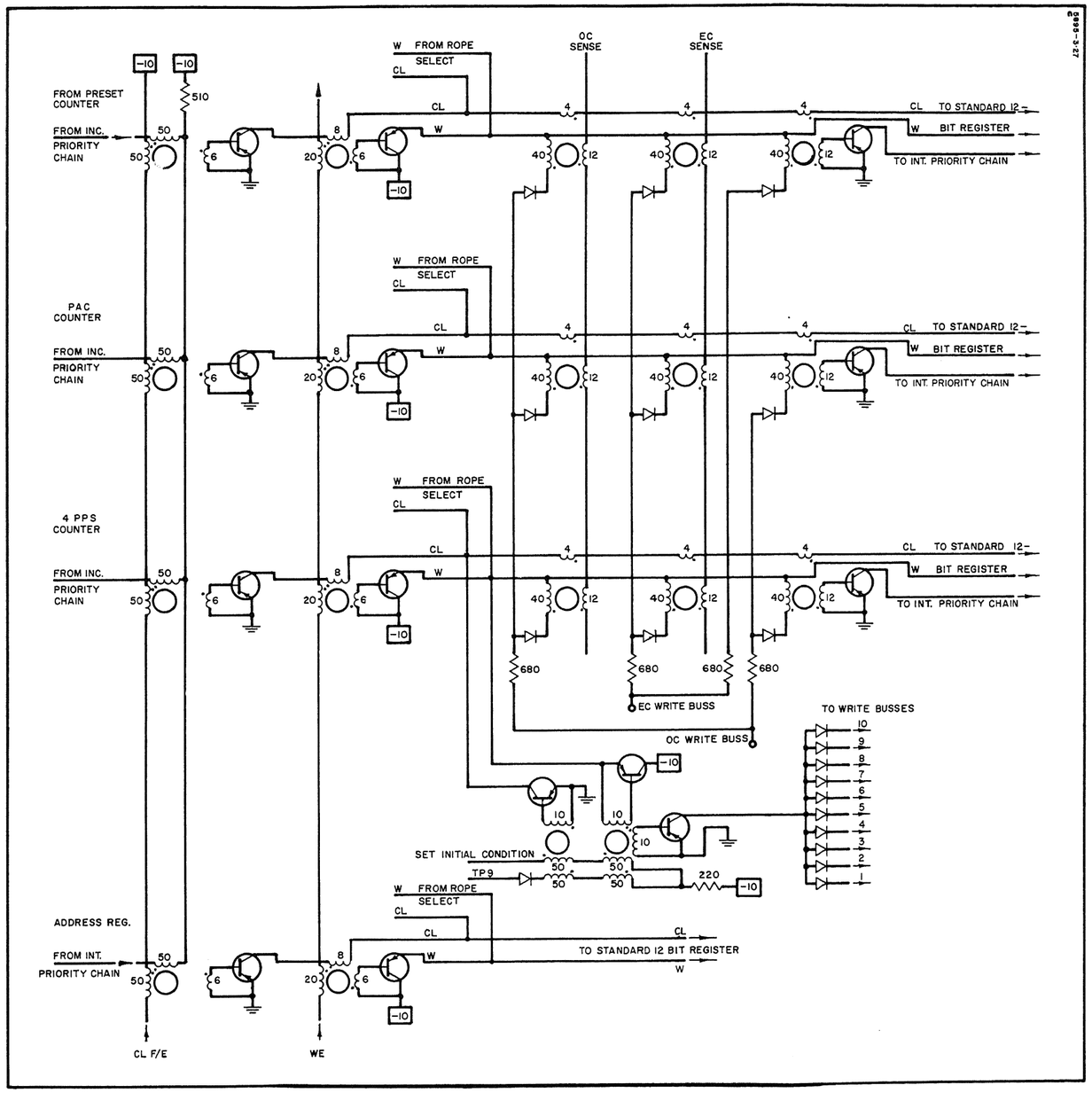


Fig. 3-27 Counter and address registers

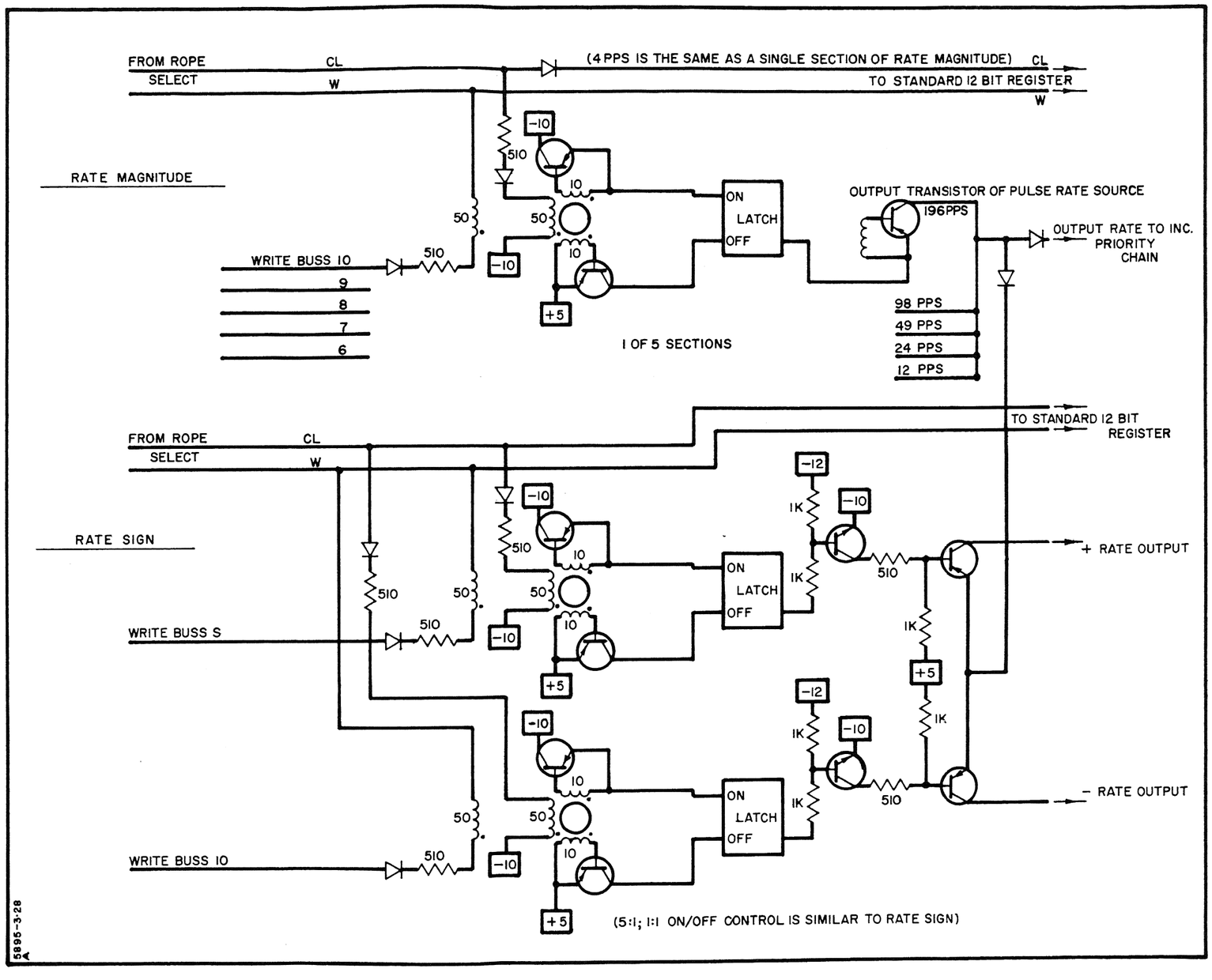
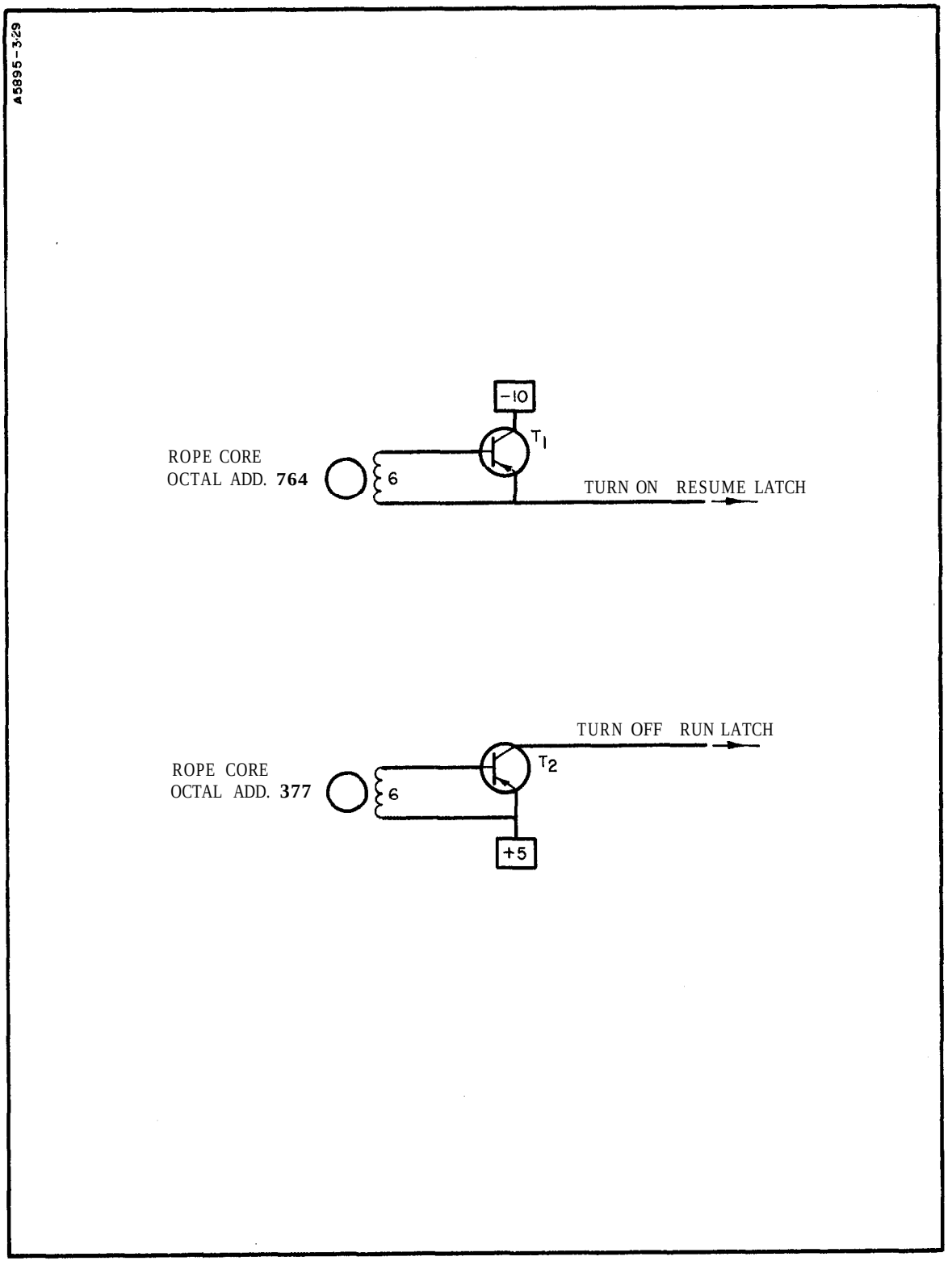


Fig. 3-28 Control registers



Fig, 3-29 RESUME and normal stop

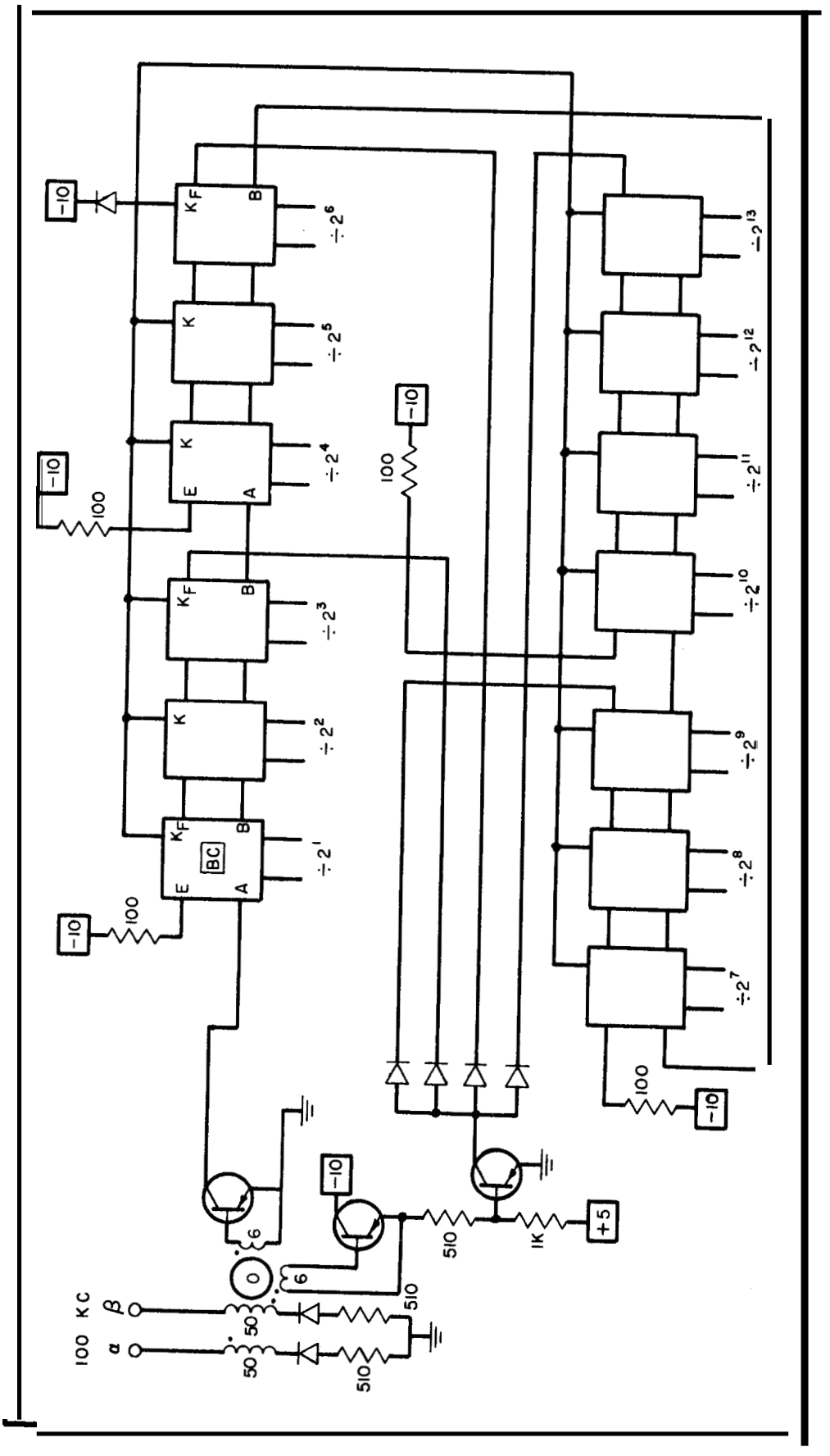


Fig. 3-30 Pulse rate source

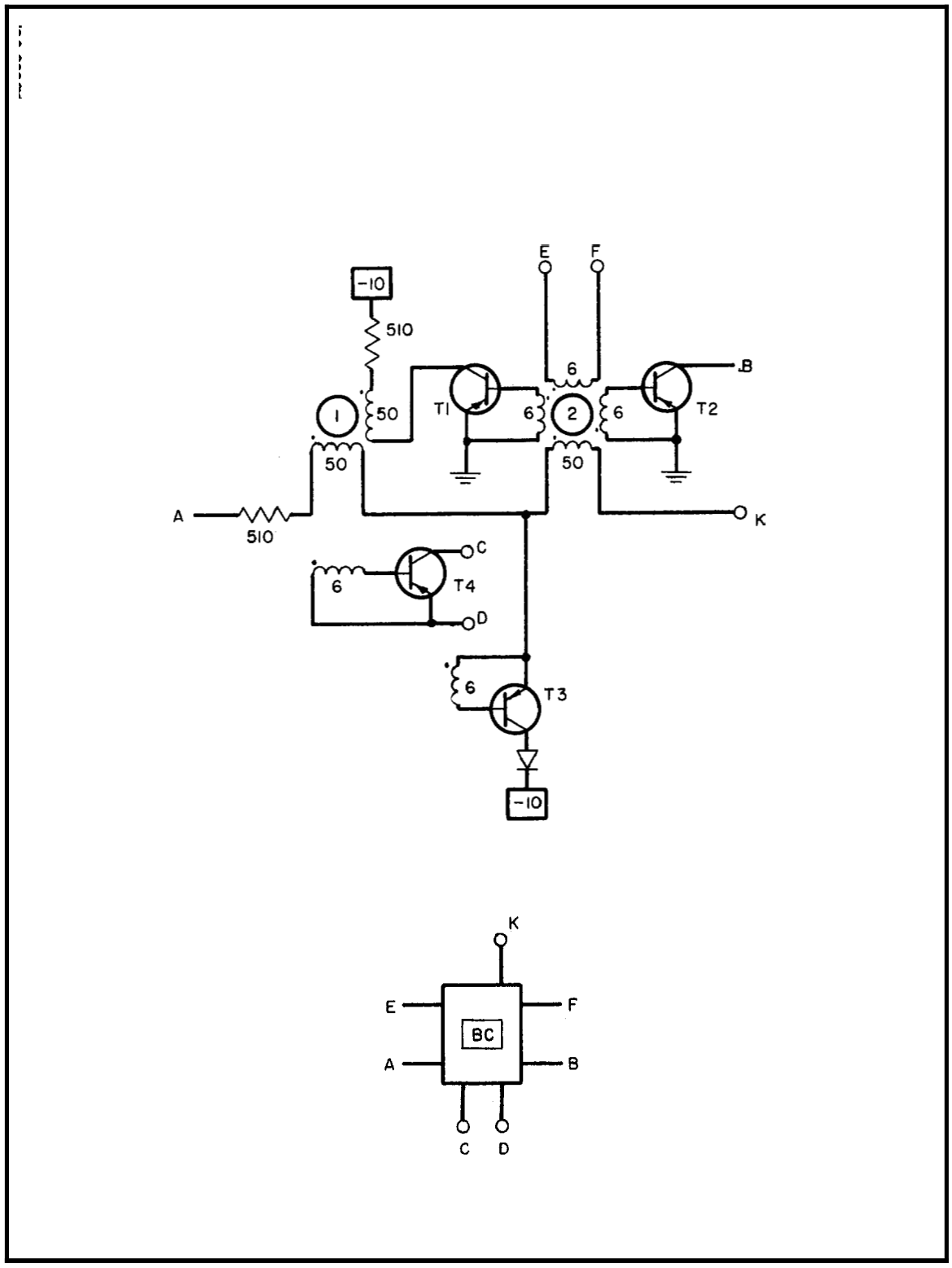


Fig. 3-31 Counter section

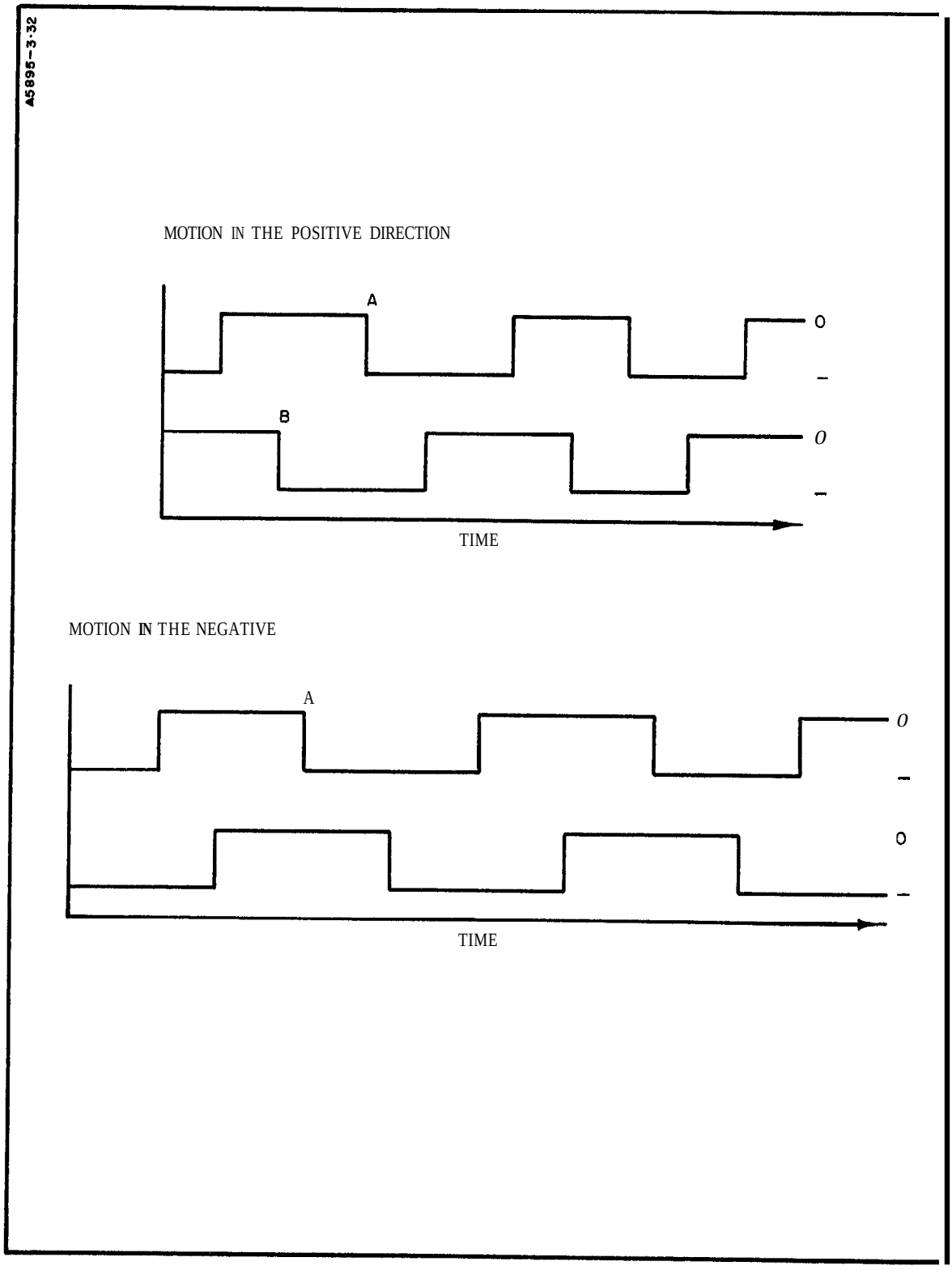


Fig. 3-32 Photoelectric encoder output

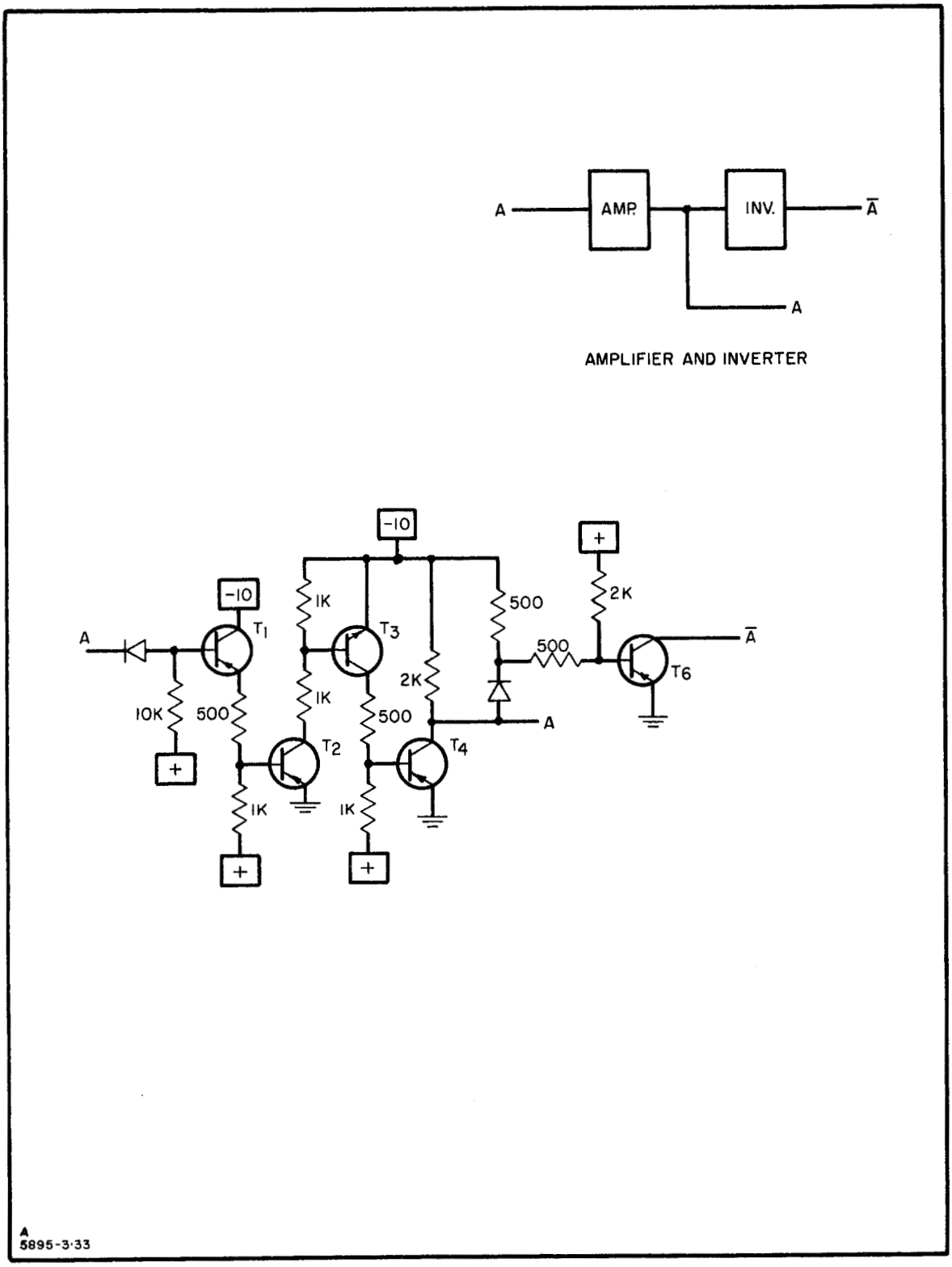


Fig. 3-33 Amplifier and inverter

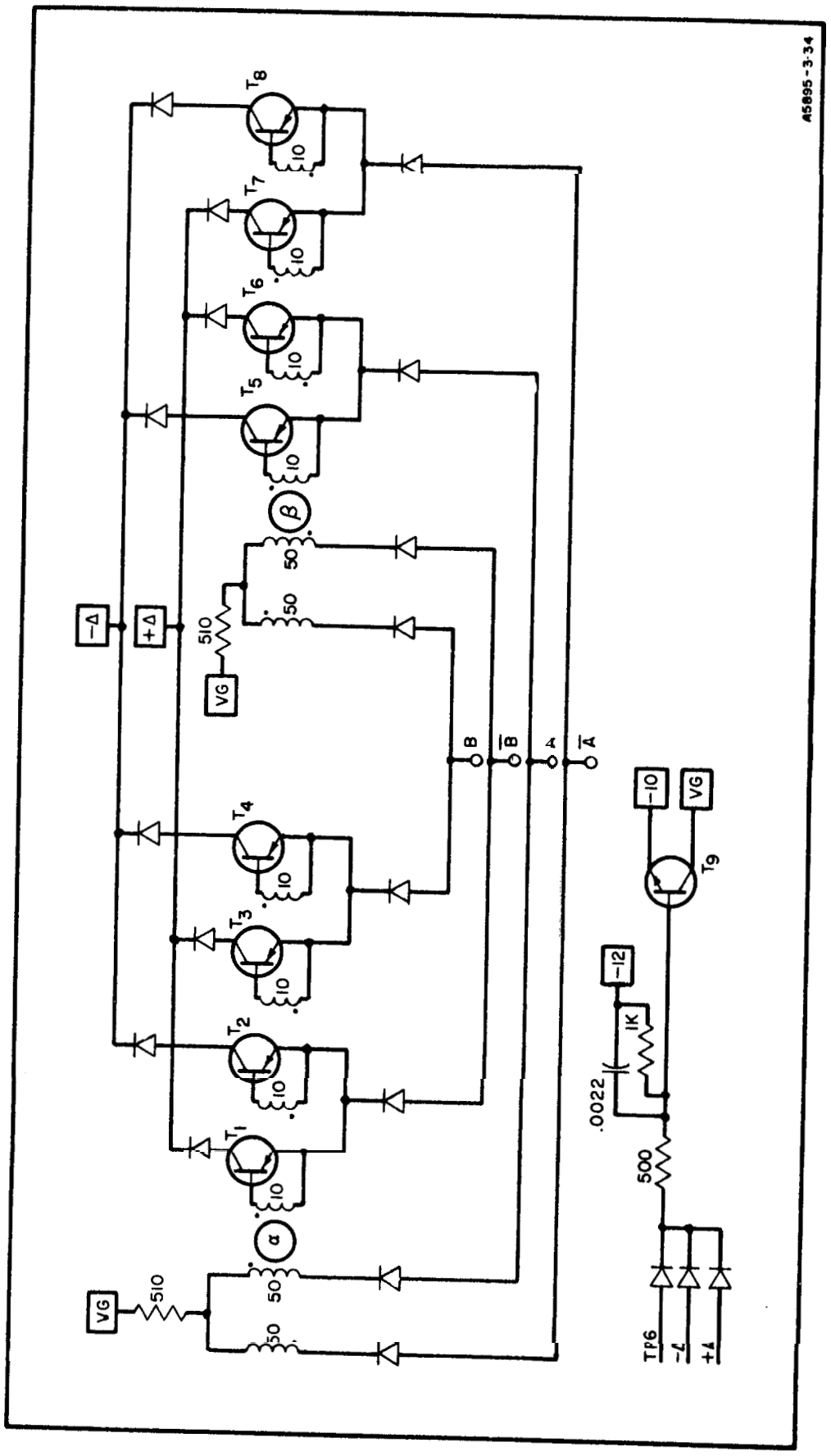


Fig. 3-34 Decoder

Table 3-1 Formation of Positive and Negative Pulses

Change in A	State of B	Increment Pulse
0 to -28	0	
0 to -28	-28	+
-28 to *0*	0	t
-28 to *0*	-28	-

Change in B	State of A	Increment Pulse
0 to -28	*0*	+
0 to -28	-28	-
-28 to 0	0	-
-28 to *0*	-28	t

CHAPTER 4
CONTROLLED DEVICES
by H. E. Maurer

CHAPTER 4

CONTROLLED DEVICES

4.1 Stepper-Motor Control

A stepper-motor is controlled in rate and position (Fig. 4-1) by the Mod 1B-1, 1B-3, and 1B-5 programs of the Mod 1B computer.[1] Fig. 4-2 shows an assembled fixture containing stepper-motor, speed reducer, optical encoder, d-c amplifiers, and shapers. The pointer is driven from the output shaft and serves as a convenient visual indicator of output shaft position. Fig. 4-3 shows the assembled counter, driving circuits, and power supply packaged with other test equipment[2] for convenience of use by Mod 1B.

The stepper-motor logic[3,4] is a three- bit binary cyclic counter and driving circuits which ground the proper combination of motor stator windings to provide ± 45^{o} positioning of the stepper-motor shaft. Input pulses arrive at the counter as ordered by a computer program. Motor shaft angle is reduced by 20:1 and then monitored by an optical encoder. The encoder output is decoded and then accumulated through. the increment circuits of the computer in an addressable register. Five input: pulses to the counter produce 64 output pulses from the decoder for a gain of 12.8 to 1.

A discussion of the role of the computer is found in Chapter 2.

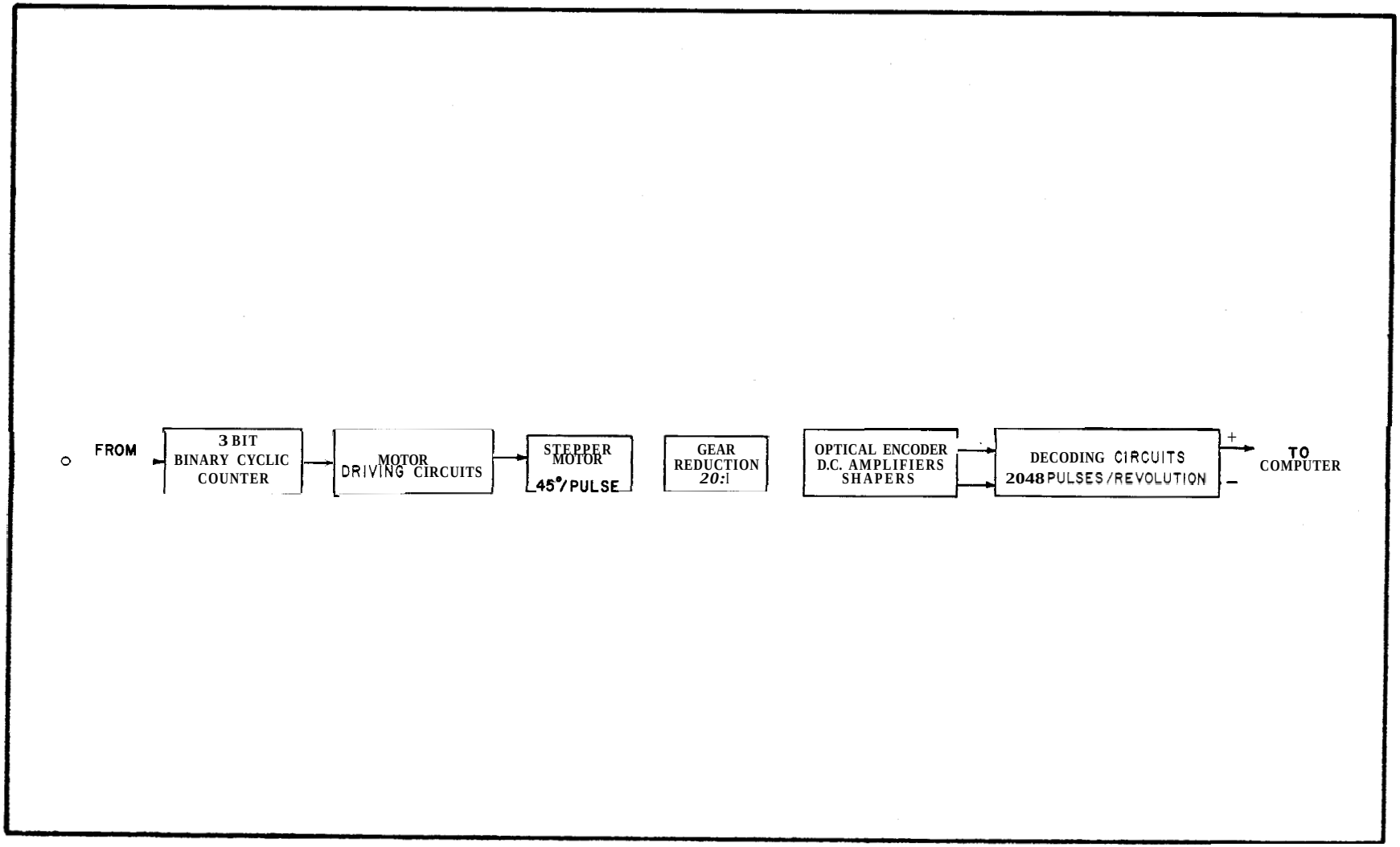


Fig. 4-1 Stepper-motor control

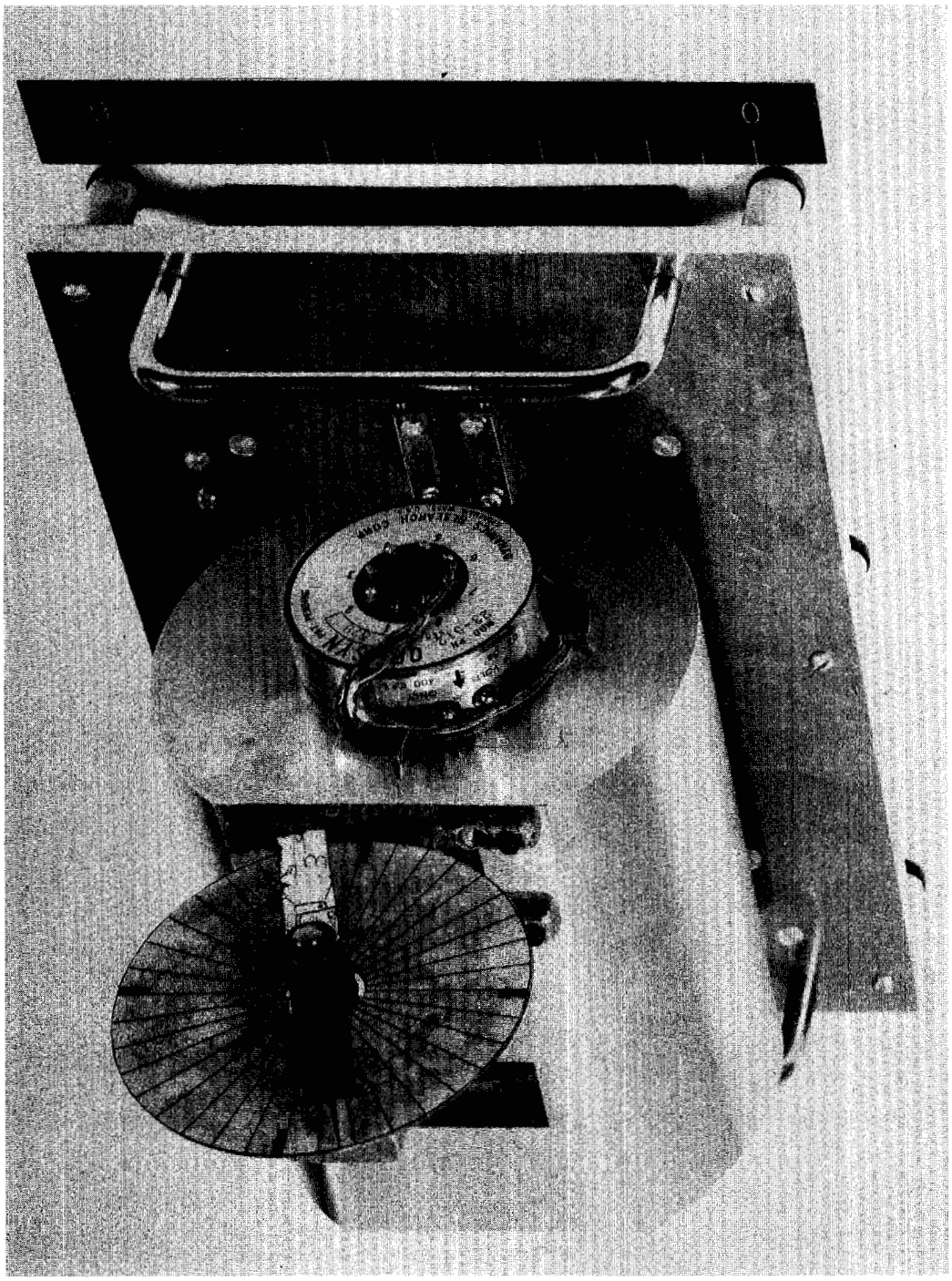


Fig. 4-2 Stepper-motor control assembly

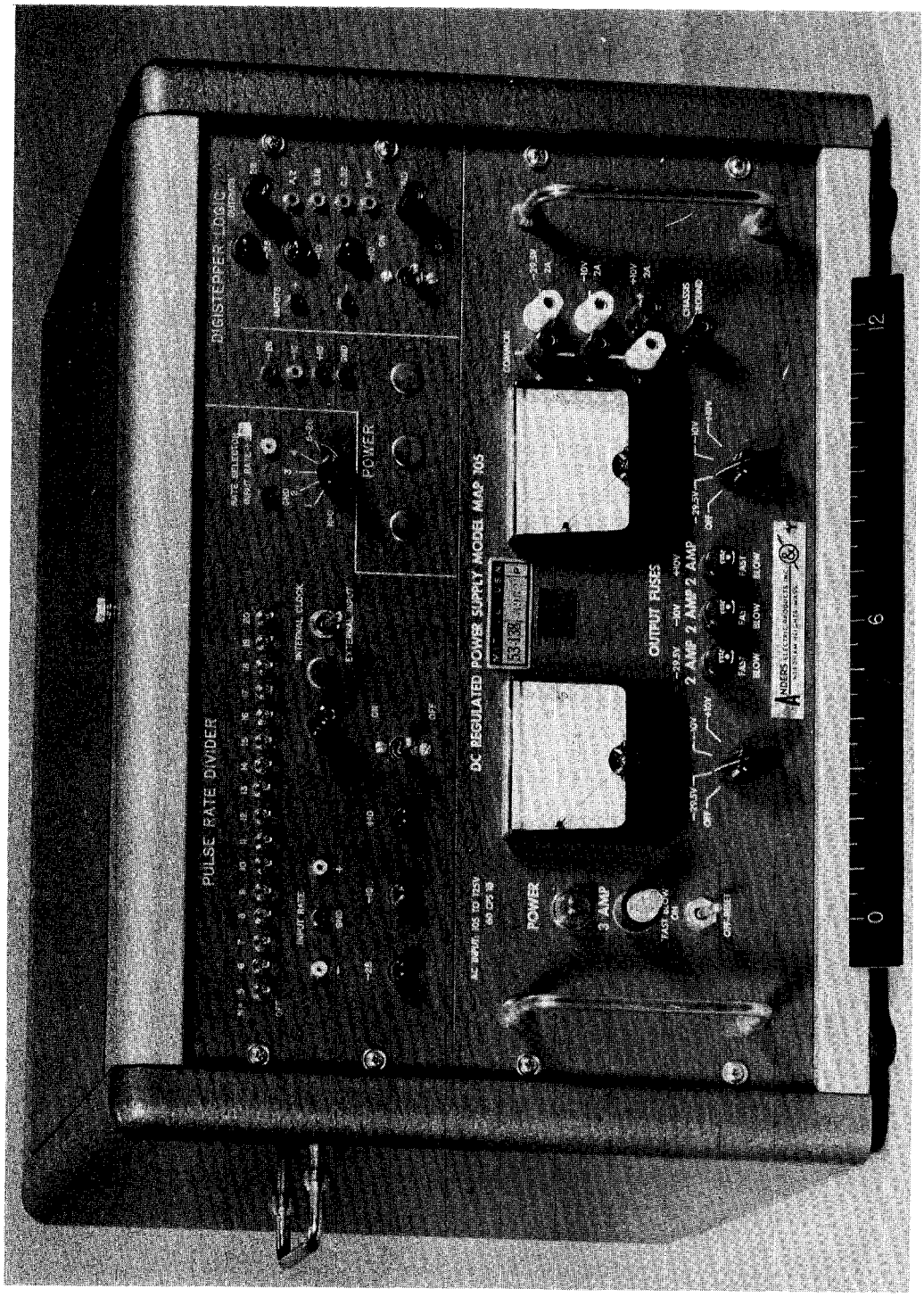


Fig. 4-3 Stepper-motor logic assembly

4.2 Counter and Driving Circuits

The three-bit binary cyclic counter and stepper-motor driving circuits (Fig. 4-4) were available from a previous design[4] which used standard ULB logic circuits (Fig. 4-5).

The counter accepts pulse inputs on one of two input lines. Each pulse represents a desired increment of shaft angle. A pulse on the plus input line causes the counter to count up, for example, *000,* 001, 010, . . . 110, 111, *000,* while a pulse on the minus input line causes the counter to count down, for example, *000,* 111, 110 . . . 010, 001, 000.

The driving circuits (Fig. 4-3) are designed of the same logic circuits as are the counter but the four NOR gates driving the motor windings use power transistors. Each driving NOR gate and stator coil combination comprise an "and" logic function. The E-blocks sort out the counter contents and deliver the proper inputs to the four "and" gates *so* that each of the eight states of the counter determines one of the eight discrete stable positions of the stepper-motor shaft.

4.3 Speed Reducer

The choice of a speed reducer (Fig. 4-6) was based on convenience of computer programing and program time consumed rather than quality of the reducer. A speed reduction of 20:1 meant that all necessary multiplication and division in the interrupt programs could be accomplished by simple shifts rather than time consuming subroutines. A larger gear ratio would have provided additional torque to accelerate the encoder.

With a computer possessing multiply and divide instructions a speed reducer could be used which has the desirable high step-down gear ratio and anti-backlash gears.

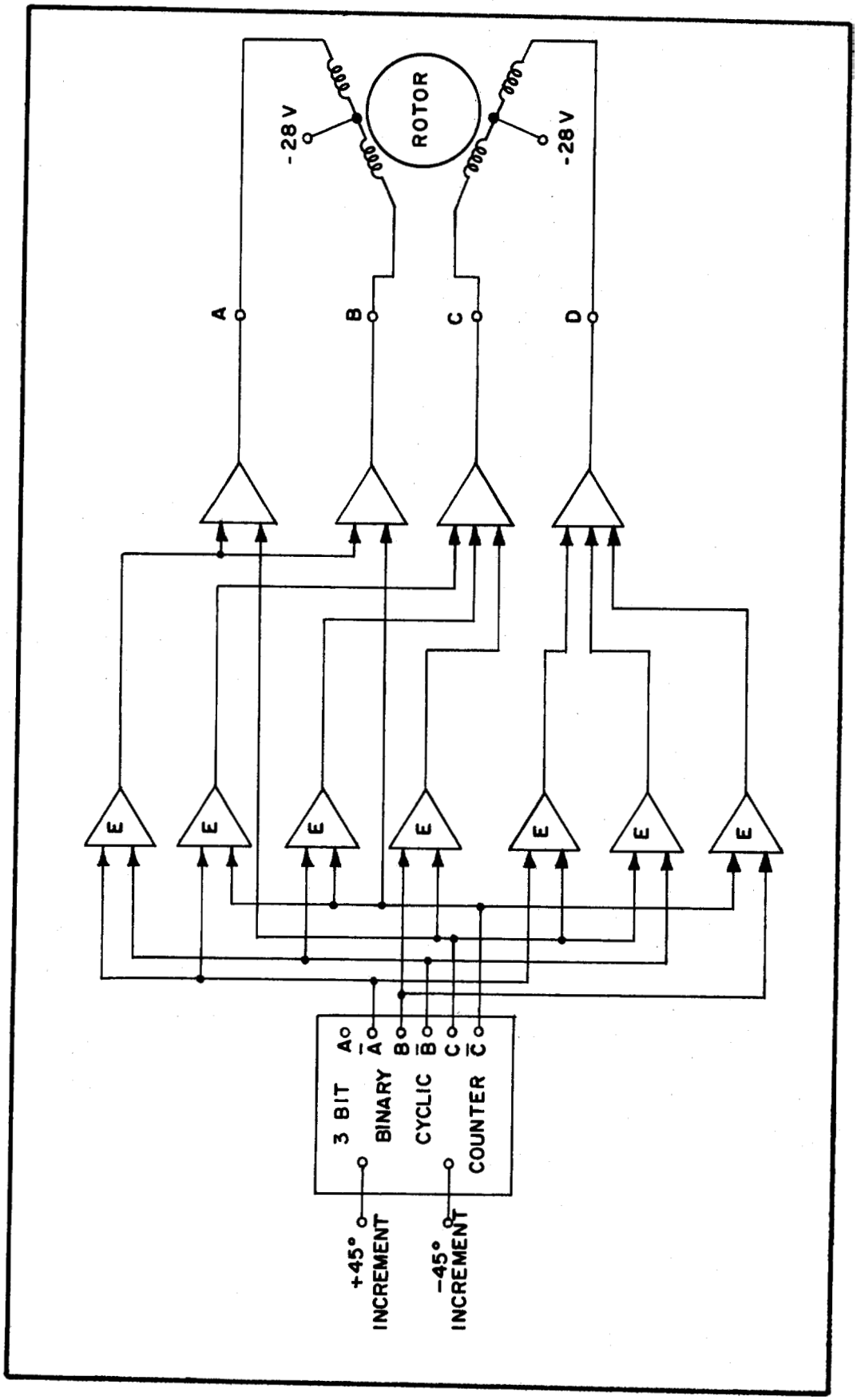


Fig. 4-4 Stepper-motor with driving logic circuits

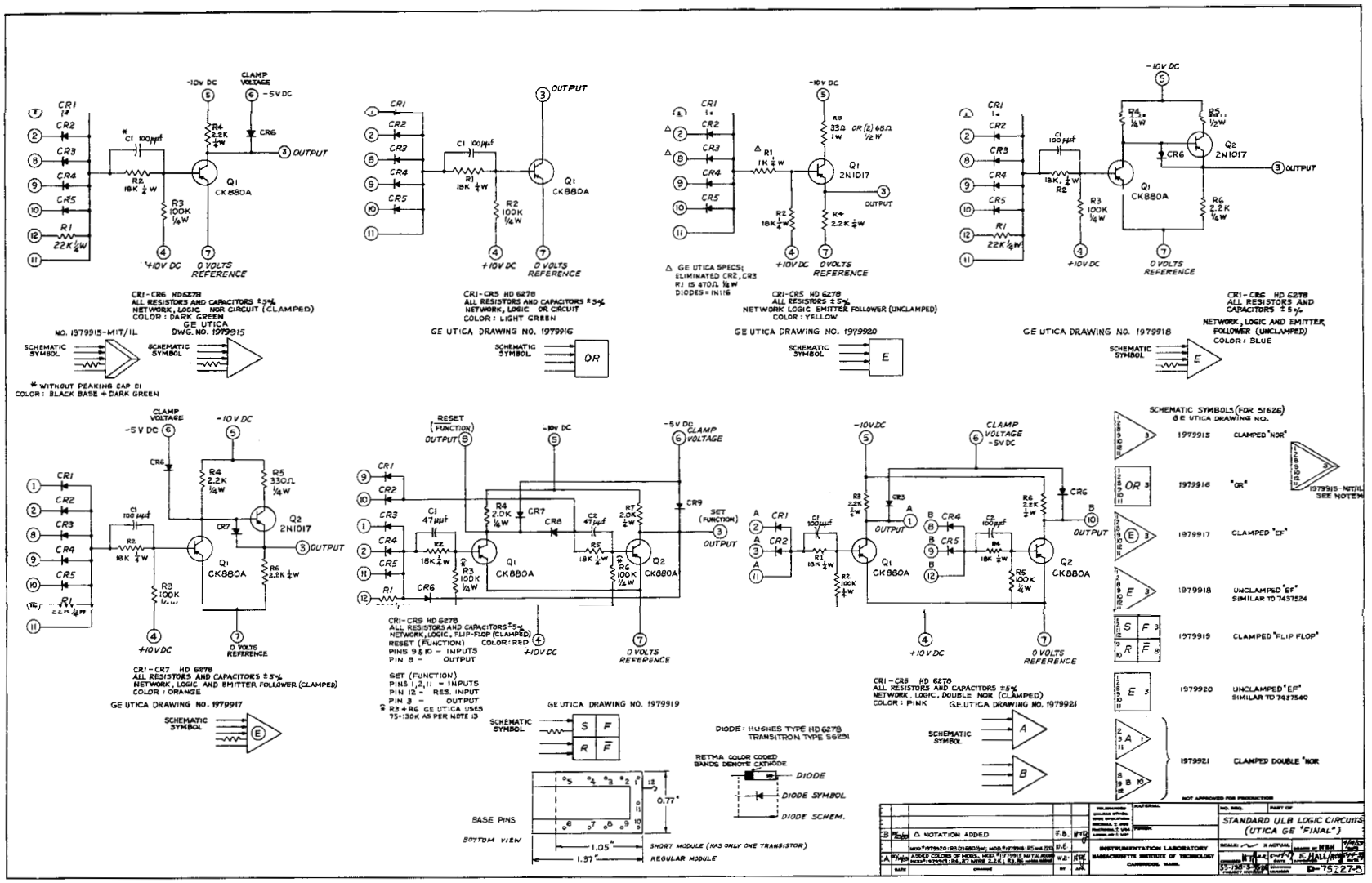


Fig. 4-5 Standard ULB logic circuits (UTICA GE "FINAL")

Fig. 4-6 Stepper-motor, speed reducer, and optical encoder

4.4 Stepper-Motor

The stepper-motor (Fig. 4-6) has a permanent magnet rotor and two centertapped stator windings (Fig. 4-7). The stator windings are separted by 90 mechanical degrees and each winding has a voltage source connected to the centertap. Assuming magnetic symmetry, grounding the proper combination of winding terminals of the two stator windings should produce magnetic fields at 45^{o} intervals.

Actually, the magnetic fields were far from uniform and the 45^{o} intervals were not achieved with any'accuracy. However, eight stable positions of the rotor were achieved consistently.

4.5 Encoder, Amplifiers, Shapers and Decoder

The shaft encoder (Fig. 4-6) is an optical[5], incremental encoder named Optisyn[6] with 512 slits and solar cells for increased stability. The output is two voltage waves in quadrature, each approximating a sinusoid with peak-to-peak amplitudes of about 200 mv. The output load impedance should be 2000 ohms.

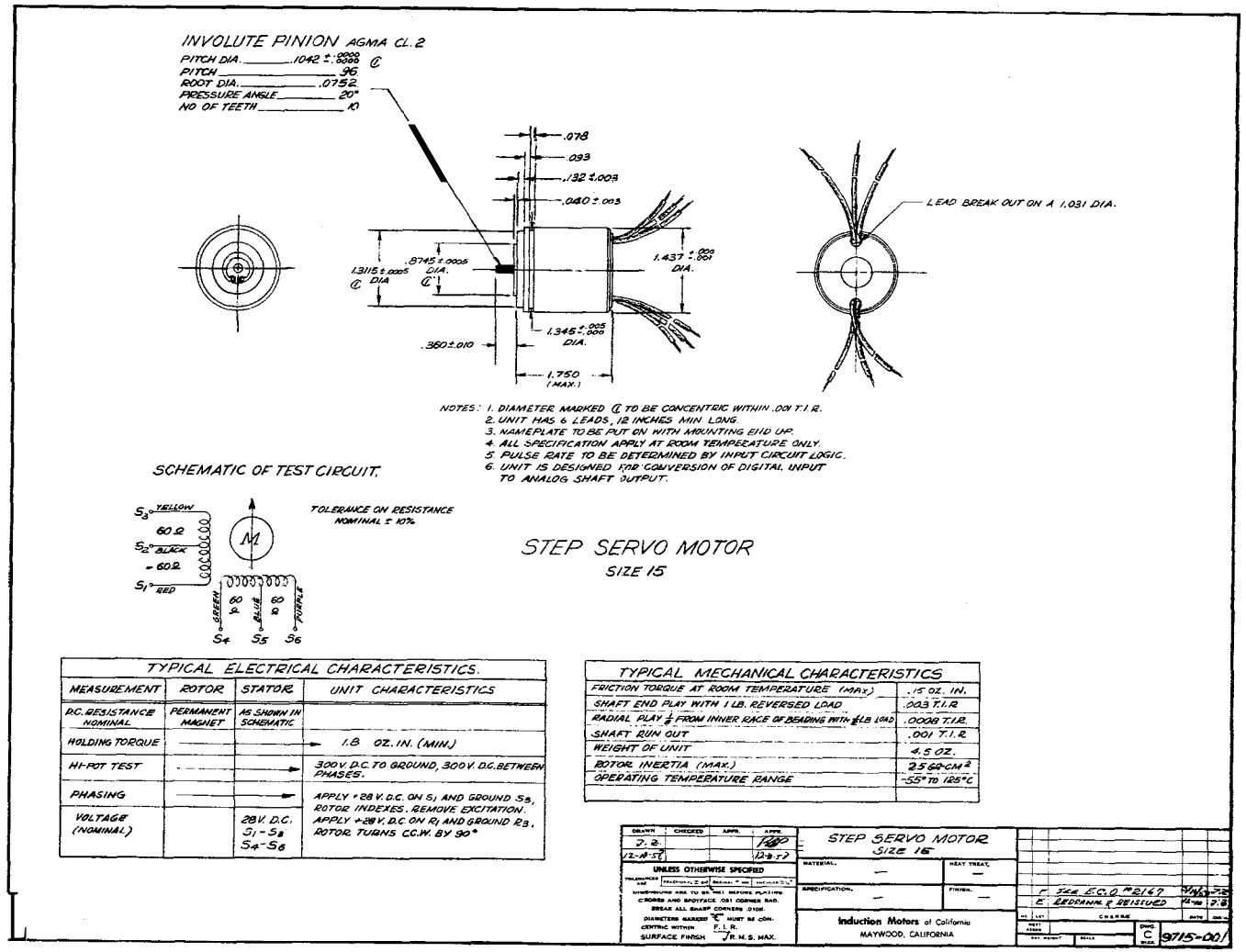
If the null crossings at the zero differential level* are considered for both output waveforms, then ideally a null crossing by a waveform should be followed 90 electrical degrees later by a null crossing of the quadrature waveform. The maximum deviation from 90^{o} of any null crossing of one waveform with respect to an adjacent null crossing is 25 electrical degrees.

Generally it is necessary to amplify the Optisyn's outputs with a d-c saturating type amplifier (Fig. 4-8) followed by shapers (Fig. 4-9) to provide the sharp-edged, flat-topped voltage waveforms required to operate the decoding circuits.

The decoder is designed of circuits similar to those used in Mod 1B proper and is discussed in Chapter 3.

---

*Zero differential level is defined as the voltage above or below ground that produces null crossings 180 electrical degrees apart. The zero differential level is within ±30 mv of the true zero level.



TYPICAL ELECTRICAL CHARACTERISTICS.			
MEASUREMENT	ROTOR	STATOR	UNIT CHARACTERISTICS
D.C. RESISTANCE NOMINAL	PERMANENT MAGNET	AS SHOWN IN SCHEMATIC	
HOLDING TORQUE	———	———	1.8 OZ. IN. (MIN.)
HI-POT TEST	———	———	300 V. D.C. TO GROUND, 300 V. D.C. BETWEEN PHASES.
PHASING	———	———	APPLY +28 V. D.C. ON S_1 AND GROUND S_5, ROTOR INDEXES. REMOVE EXCITATION. APPLY +28 V. D.C. ON R_1 AND GROUND R_3, ROTOR TURNS C.C.W. BY 90°
VOLTAGE (NOMINAL)		28 V. D.C. S_1-S_3 S_4-S_6	

TYPICAL MECHANICAL CHARACTERISTICS	
FRICTION TORQUE AT ROOM TEMPERATURE (MAX.)	.15 OZ. IN.
SHAFT END PLAY WITH 1 LB. REVERSED LOAD	.003 T.I.R
RADIAL PLAY $\frac{1}{8}$ FROM INNER FACE OF BEARING WITH $\frac{1}{2}$ LB LOAD	.0008 T.I.R.
SHAFT RUN OUT	.001 T.I.R.
WEIGHT OF UNIT	4.5 OZ.
ROTOR INERTIA (MAX.)	25 GM-CM²
OPERATING TEMPERATURE RANGE	-55° TO 125°C

DRAWN	CHECKED	APPR.	DATE
J. R.			12-8-57

UNLESS OTHERWISE SPECIFIED
TOLERANCES ARE TO BE HELD BEFORE PLATING
CROSS AND BROACH .031 CORNER RAD.
BREAK ALL SHARP CORNERS .010R.
DIAMETERS MARKED ℄ MUST BE CONCENTRIC WITHIN F. I. R.
SURFACE FINISH √R.M.S. MAX.

STEP SERVO MOTOR
SIZE 15

MATERIAL — HEAT TREAT —
SPECIFICATION — FINISH —

Induction Motors of California
MAYWOOD, CALIFORNIA

NO.	LET	CHANGE	DATE	CHK.
	F	PER E.C.O. #2167	11/6/57	
	E	REDRAWN & REISSUED	12-8-57	J.R.

DWG. C 9715-001

Fig. 4-7 Step servo motor size 15

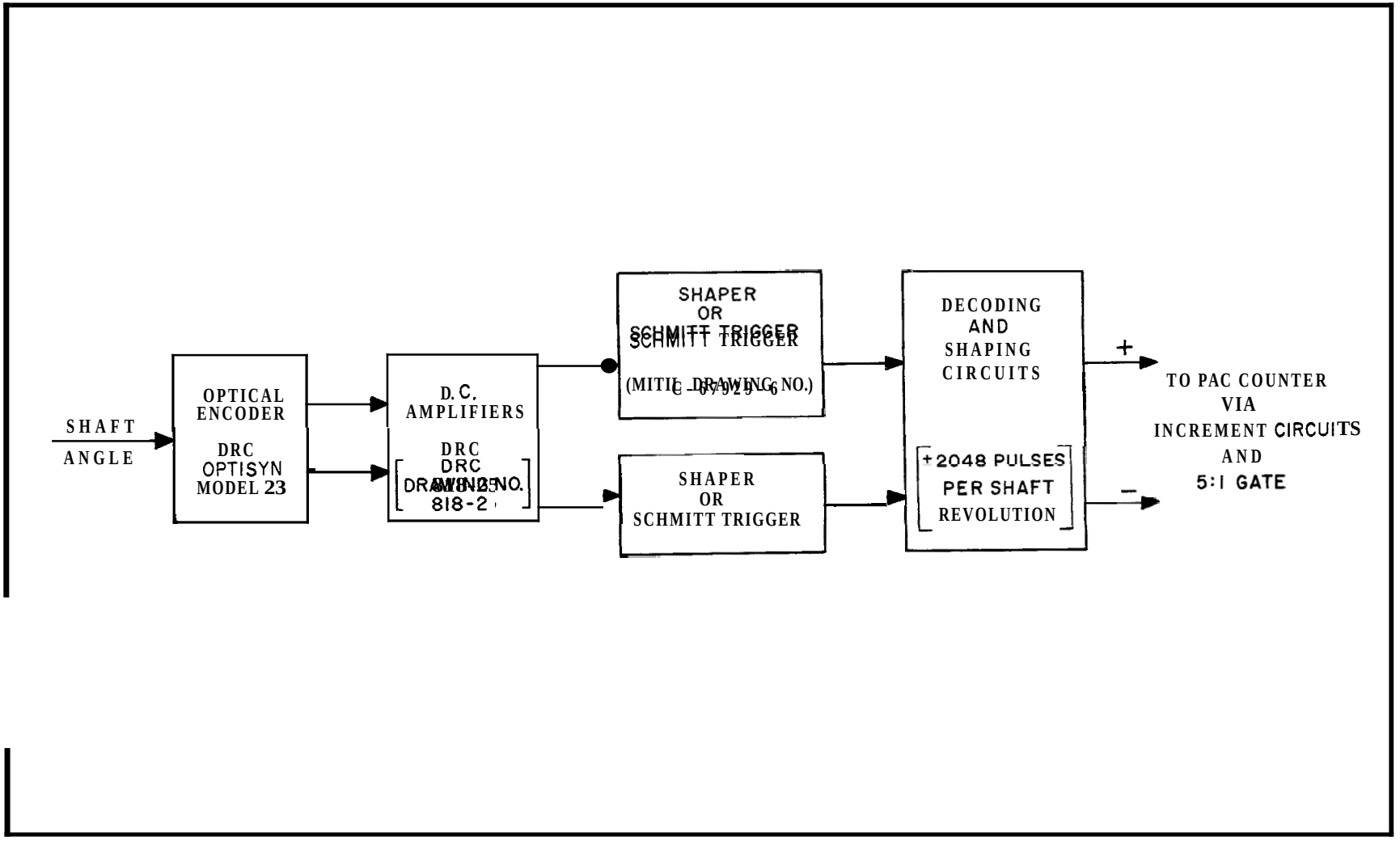
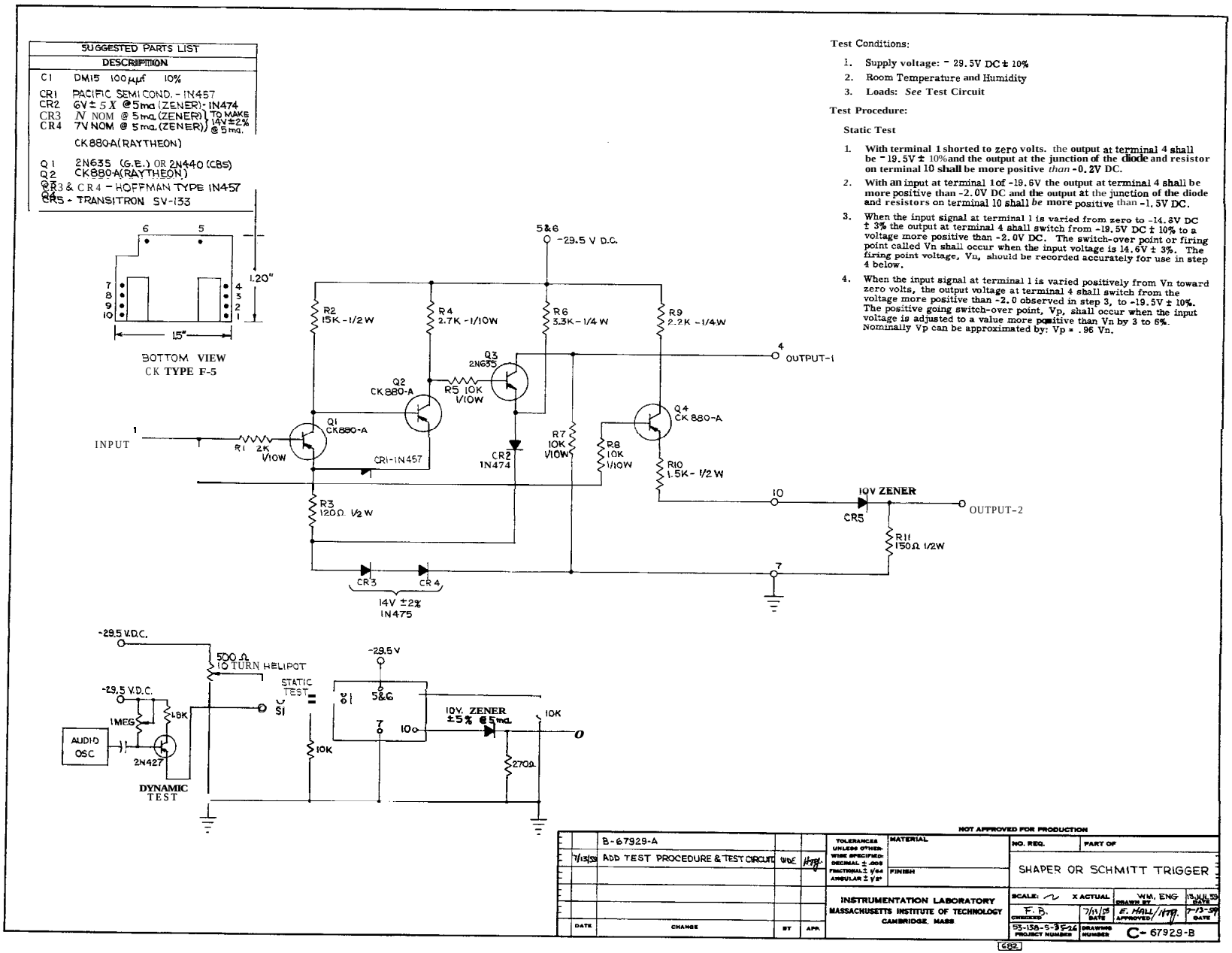


Fig. 4-8 Optical encoder and decoder

CONFIDENTIAL



Test Conditions:

1. Supply voltage: - 29.5V DC ± 10%
2. Room Temperature and Humidity
3. Loads: *See* Test Circuit

Test Procedure:

Static Test

1. With terminal 1 shorted to zero volts. the output at terminal 4 shall be - 19.5V ± 10% and the output at the junction of the diode and resistor on terminal 10 shall be more positive *than* - 0.2V DC.
2. With an input at terminal 1 of -19.6V the output at terminal 4 shall be more positive than -2.0V DC and the output at the junction of the diode and resistors on terminal 10 shall *be* more positive than -1.5V DC.
3. When the input signal at terminal 1 is varied from zero to -14.8V DC ± 3% the output at terminal 4 shall switch from -19.5V DC ± 10% to a voltage more positive than -2.0V DC. The switch-over point or firing point called Vn shall occur when the input voltage is 14.6V ± 3%. The firing point voltage, Vn, should be recorded accurately for use in step 4 below.
4. When the input signal at terminal 1 is varied positively from Vn toward zero volts, the output voltage at terminal 4 shall switch from the voltage more positive than -2.0 observed in step 3, to -19.5V ± 10%. The positive going switch-over point, Vp, shall occur when the input voltage is adjusted to a value more positive than Vn by 3 to 8%. Nominally Vp can be approximated by: Vp = .96 Vn.

NOT APPROVED FOR PRODUCTION

	B-67929-A			NO. REQ.	PART OF
7/13/59	ADD TEST PROCEDURE & TEST CIRCUIT	WDE	HHH		SHAPER OR SCHMITT TRIGGER
DATE	CHANGE	BY	APP.		

TOLERANCES UNLESS OTHERWISE SPECIFIED: DECIMAL ± .005, FRACTIONAL ± 1/64, ANGULAR ± 1/2°
MATERIAL
FINISH

INSTRUMENTATION LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE, MASS

SCALE: ~ X ACTUAL | WM. ENG | 15 JUL 59 DATE
F. B. CHECKED | 7/11/59 DATE | E. HALL APPROVED | 7-13-59 DATE
53-158-5-3-26 PROJECT NUMBER | DRAWING NUMBER C- 67929-B

682

Fig. 4-9 Shaper or Schmitt Trigger

4.6 A High-Inertia Wheel Control Application for Subsequent Mod 1B Programs

A high-inertia wheel driven by a hysteresis motor is to be controlled in rate and position by the 1B-2 program and in rate by the 1B-4 program of the Mod 1B computer (Fig. 4-10). The assembly on the right in Fig. 4-11 contains the high-inertia hysteresis motor, incremental encoder, encoder pickoff modules, d-c amplifiers, and shapers while the assembly on the left contains the latching circuits, amplifiers, and relay switching circuits. The d-c amplifiers, shapers, and decoding circuits are the same as those used with the Optisyn encoder for monitoring stepper-motor shaft angle.

The stepper-motor and flywheel controls differ in that an input pulse to the Stepper-Motor Logic results in a discrete angle of rotation of the stepper-motor shaft while closing one of the relays of the switching circuit of the hysteresis motor control results in a constant angular acceleration of the flywheel. This idea came from ideas about controlling the position of a spacecraft by controlling the angular momentum of flywheels[7].

4.7 Rate Control of Flywheel

Rate control of the flywheel is effective at speeds below synchronous. The spin (rate) computer program will set up an interrupt at equal time intervals which correspond to the contents of Erasable Register 1/8S. During each such interval the program measures flywheel speed and decides whether plus, minus or zero torque should be applied during the next interval. Zero torque is applied if a reduction in rate of less than 8 rps is required. Response of the flywheel should be slow enough that a negligible portion of computer time is used for flywheel control. For example, consider controlling the hysteresis motor whose speed-torque characteristics are shown in Fig. 4-12. If the motor is to be controlled to within 8 rps in 1/8 sec intervals, then the acceleration

should be much less than 64 rev/sec^{2*}. The motor has a synchronous speed of 200 rev/sec (12,000 rpm) so if the motor had an acceleration of 64 rev/sec^2 it would accelerate from rest to synchronous speed in 3.1 sec. Actually, since with an excitation of 35 v the flywheel requires more than 120 sec to reach its final speed, the computer has ample time to control it. Faster response times could be achieved by mixing torque levels.

4.8 Position Control of Flywheel

The turn (position) interrupt program is capable of controlling the hysteresis motor output shaft position to within an angle represented by one output bit** from the decoding circuits. The turn program determines output bits-to-go by comparing ac-, cumulated bits with desired bits. A relay is then turned on, torque applied, until half the bits-to-go have accumulated (the wheel has traversed half the bits to go) and then reverse torque is turned on until a periodic automatic interrupt program indicates that the wheel has stopped or reversed direction. The procedure is repeated until the goal is close enough (within eight bits) that an inching routine can be used. Once the bits-to-go is zero, the plus and minus torque relays are turned off, and the turn routine is terminated with an option to restart turn with a new setting (bits desired).

4.9 Relay Switching Circuit

Power is delivered to the hysteresis motor through a two-relay switching circuit (Fig. 4-13). Fig. 4-14 shows the assembled switching circuit. The relays are activated by latches which in turn are controlled by the flywheel control program. The latches could be part of a Light Register. The switching circuit performs the following functions.

* $\frac{\Delta\omega}{\Delta t} = 8/1/8 = 64$ rev/sec2

**One output bit represents 90 degrees using the encoder shown in Fig. 4-19.

	A	$\overline{A}$
B	No Power to Motor	Torque Reversed from Case AB
$\overline{B}$	Power to Stator Windings	No Power to Motor

A → Relay A 011
B → Relay B on

The relay contacts switch inductive circuits, causing voltage transients. Fig. **4-15** illustrates a desirable response of the latches and relays for inputs with periods of the order of the relay response time. With a motor connected, voltage transients can, for example, cause the latch outputs not to follow the latch inputs (Fig. **4-1**6). The addition of filters (Fig. **4-13)** across the proper relay contacts can reduce the voltage transients to an inoffensive level (Fig. **4-17).**

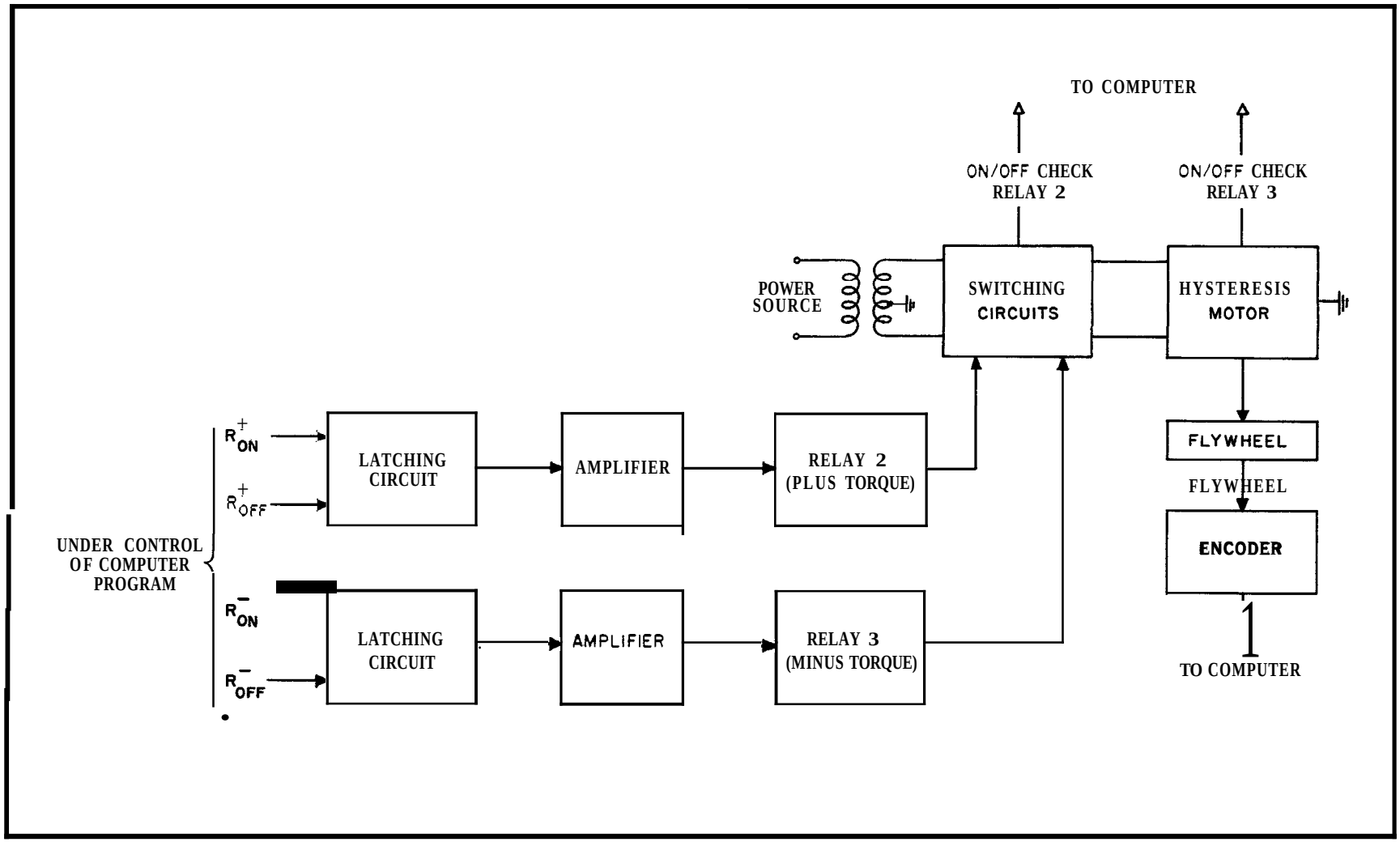


Fig. **4–10** Block diagram of programed hysteresis motor control

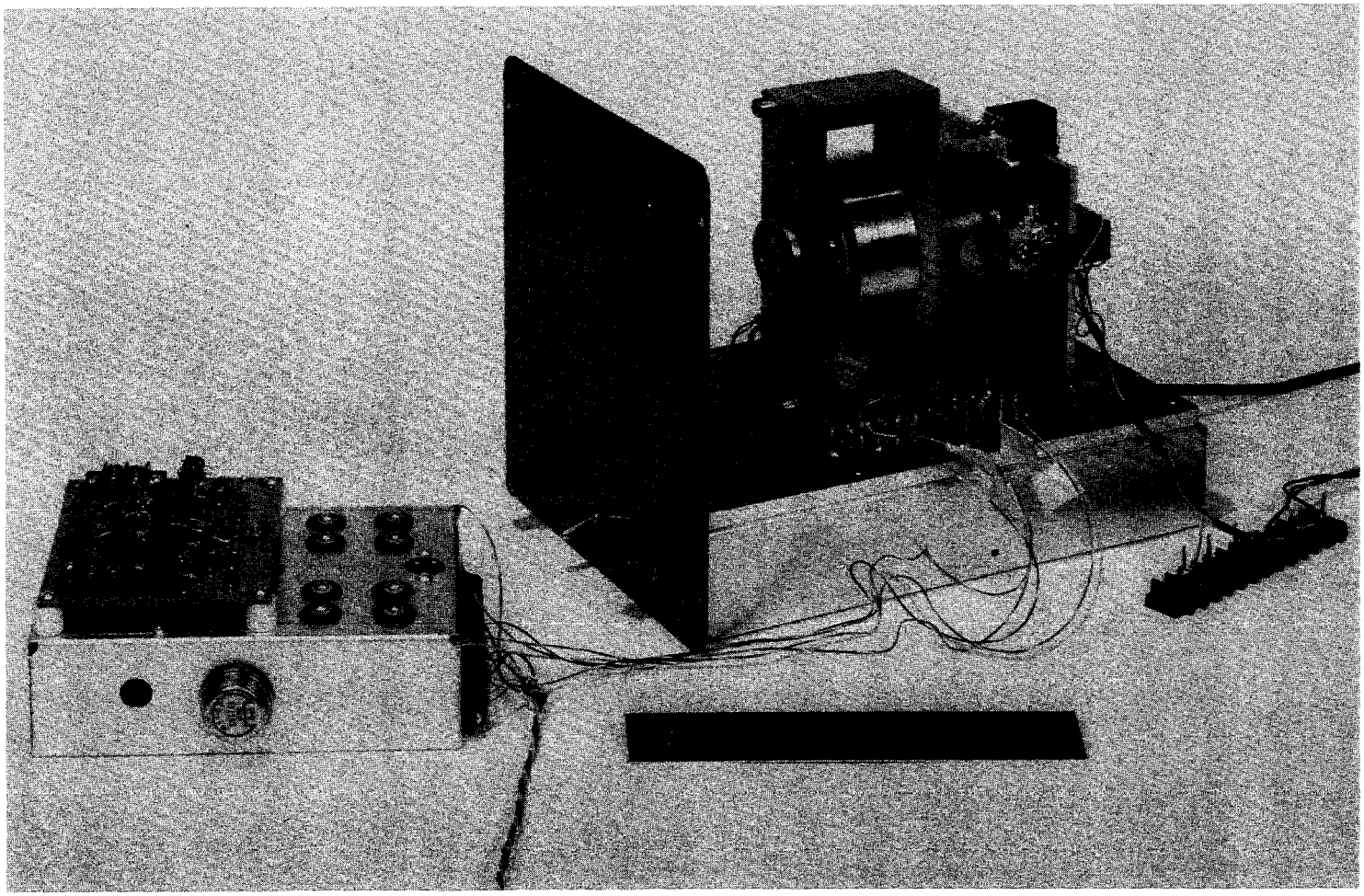


Fig. 4-11 Flywheel, switching circuit and encoder assemblies

CONFIDENTIAL

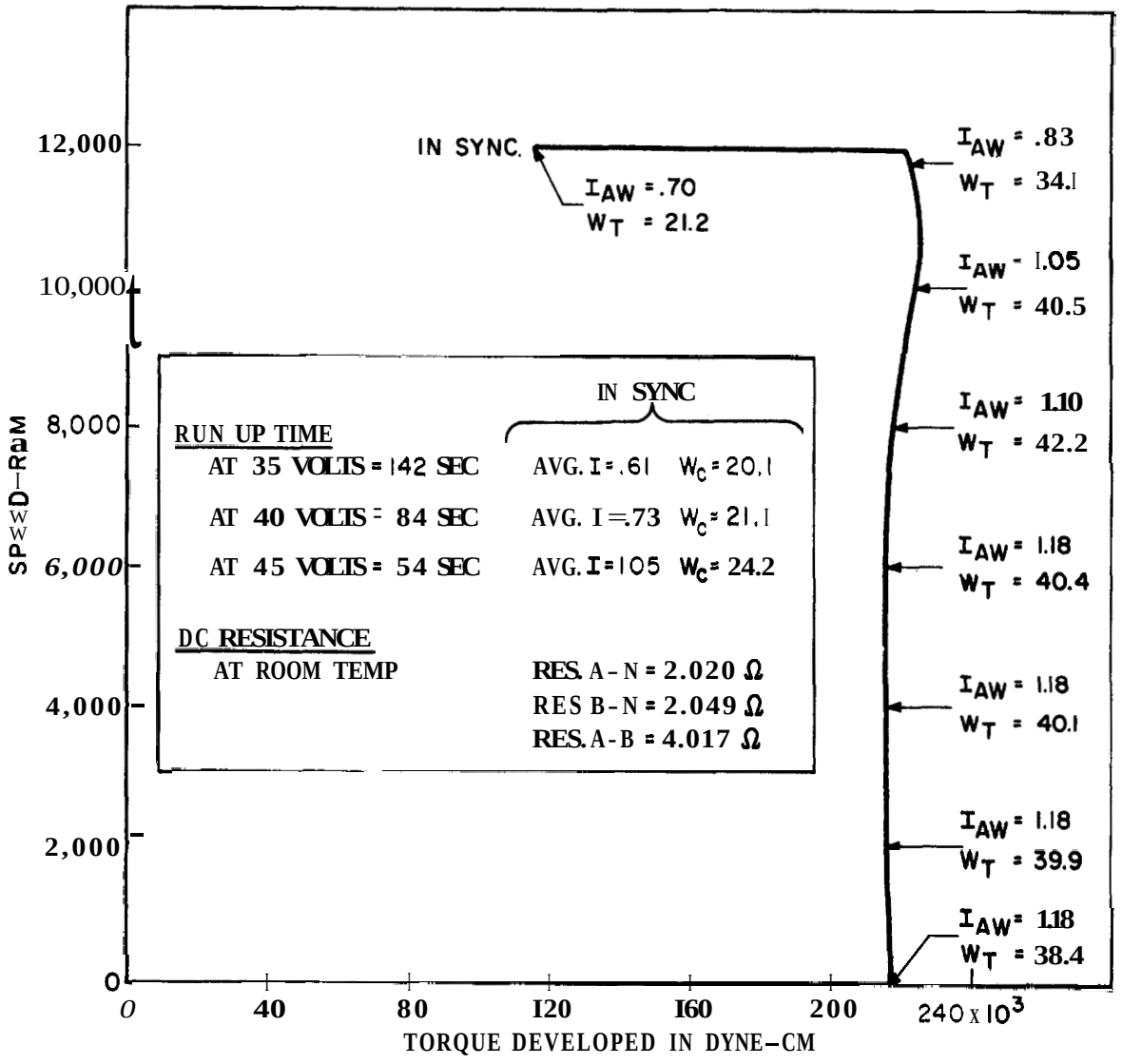


Fig. 4-12 Speed-torque characteristic for the MT 27-198 motor tested in the IL #2 fixture with an inertia of 9850 gm cm^2 40 volts/phase, 2ϕ - 400 cps excitation

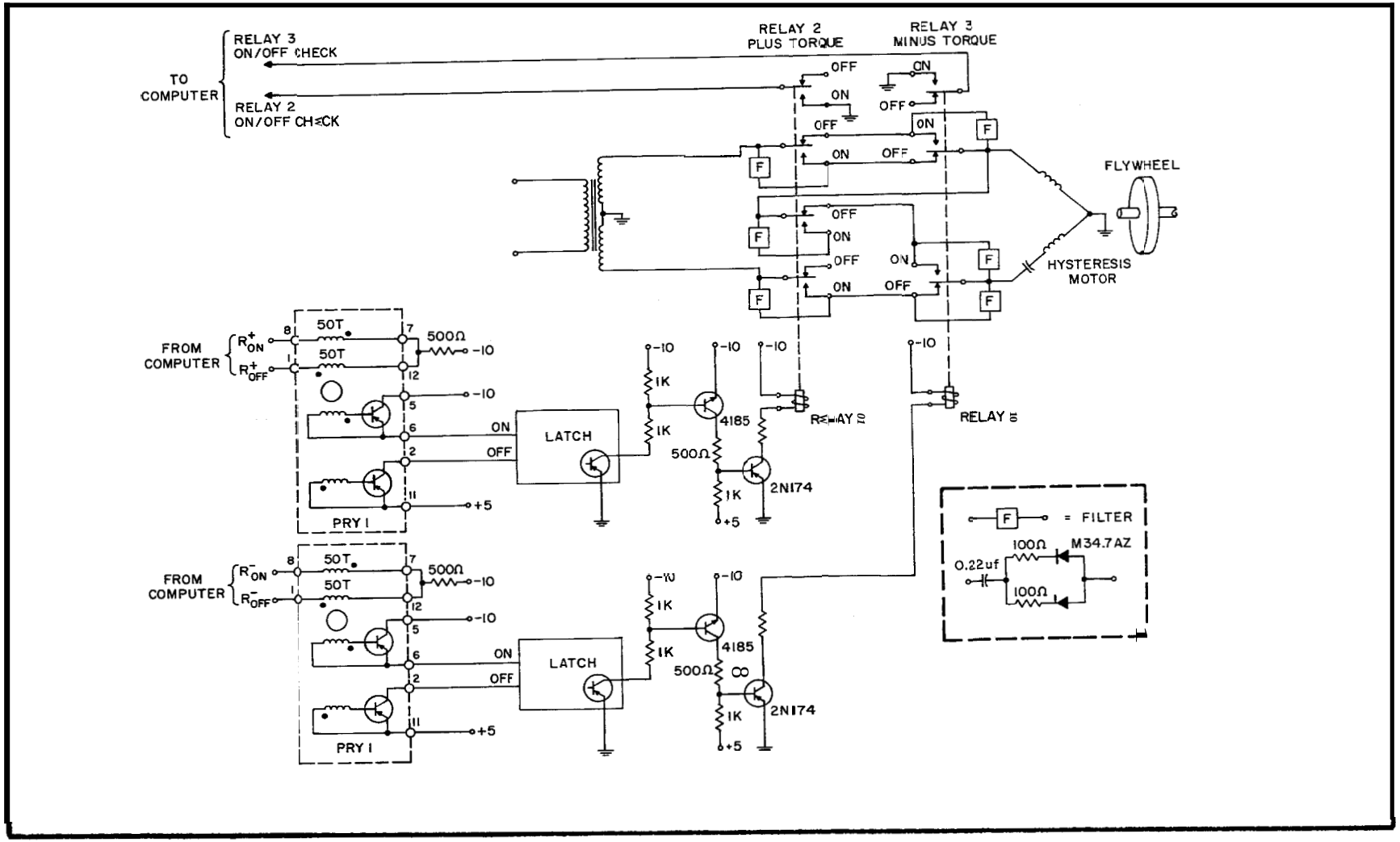


Fig. 4-13 Hysteresis motor control circuit diagram

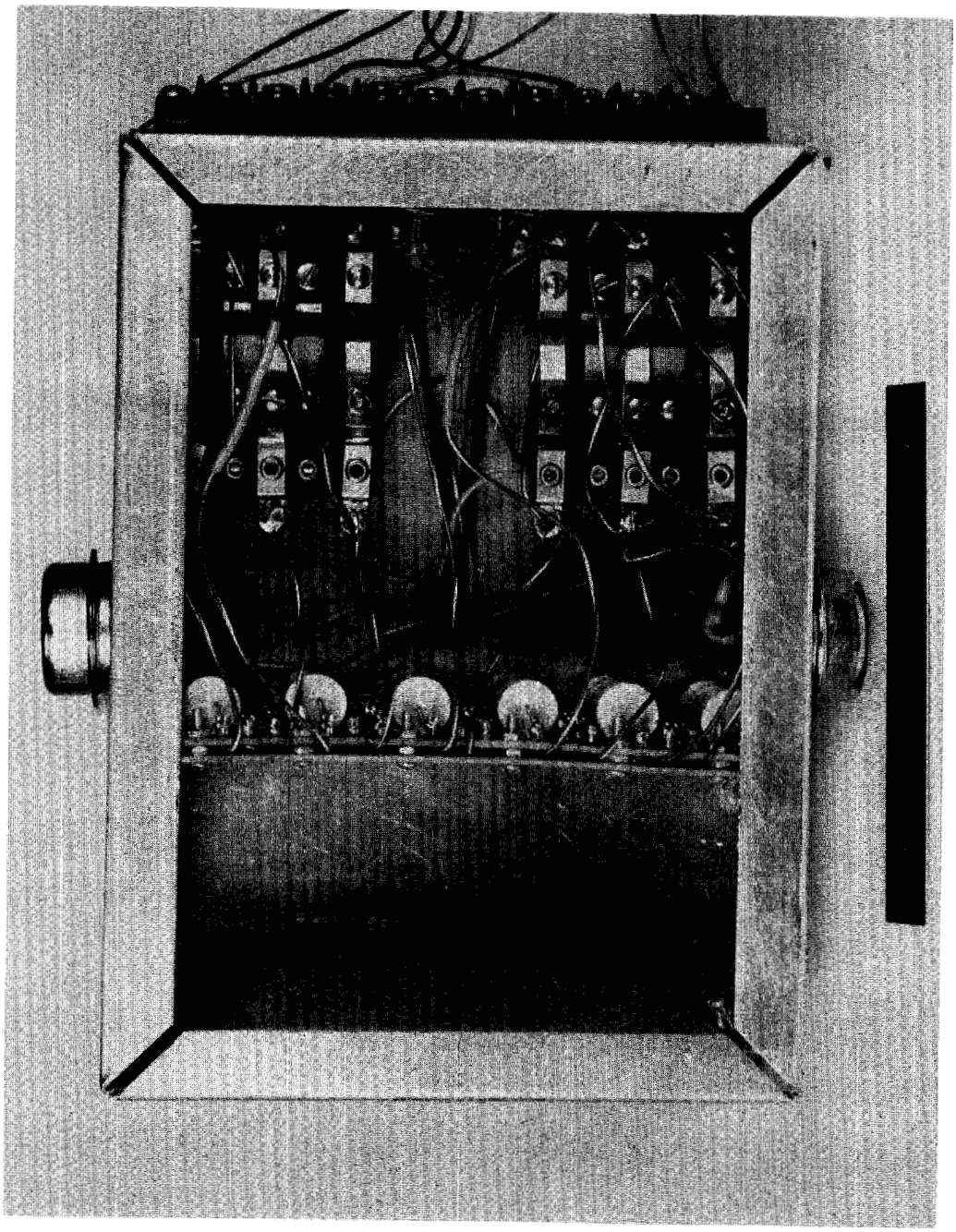


Fig. 4-14 Switching circuit assembly

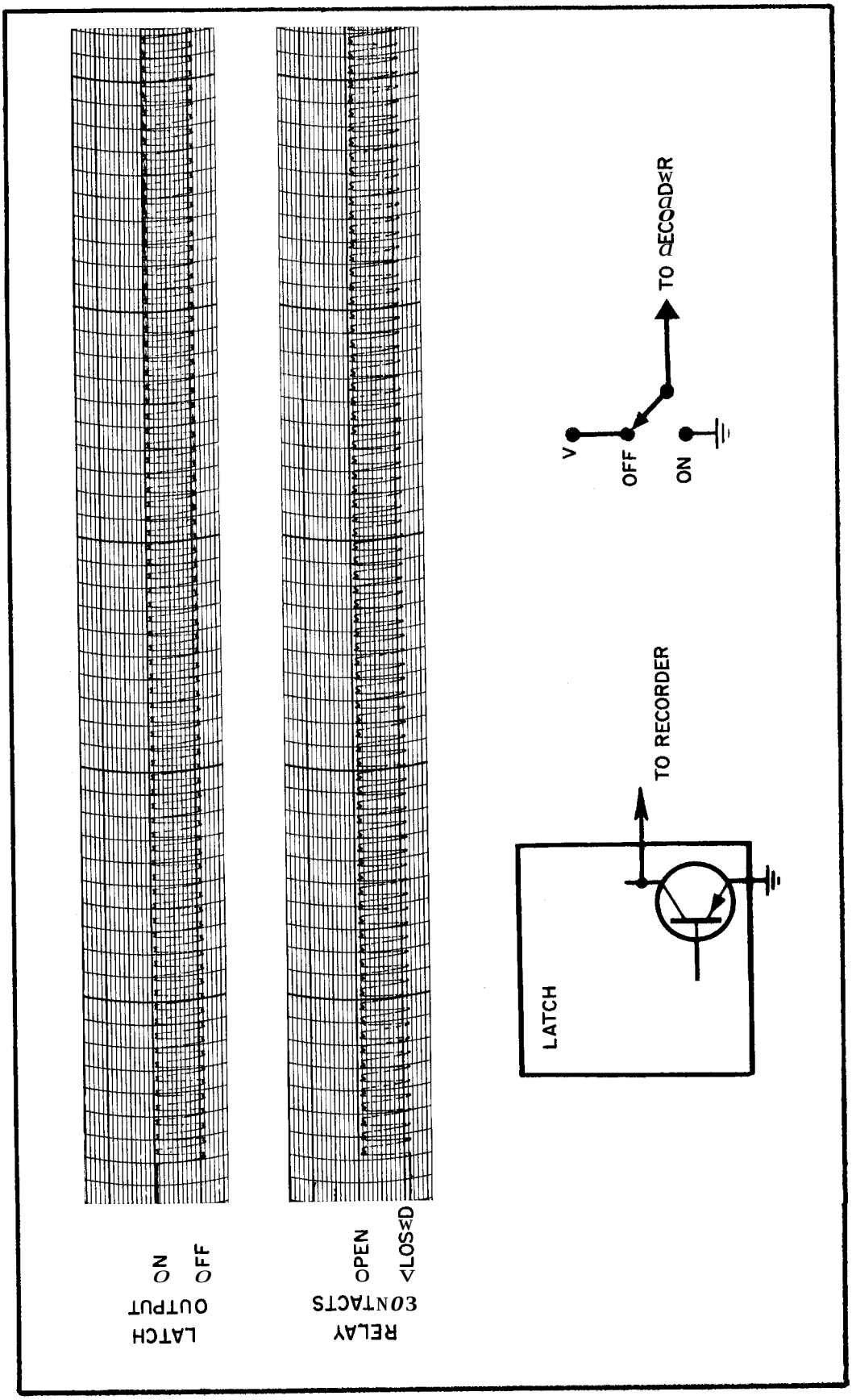


Fig. 4-15 Latch and relay contact outputs of hysteresis motor control circuit no filters, no motor.

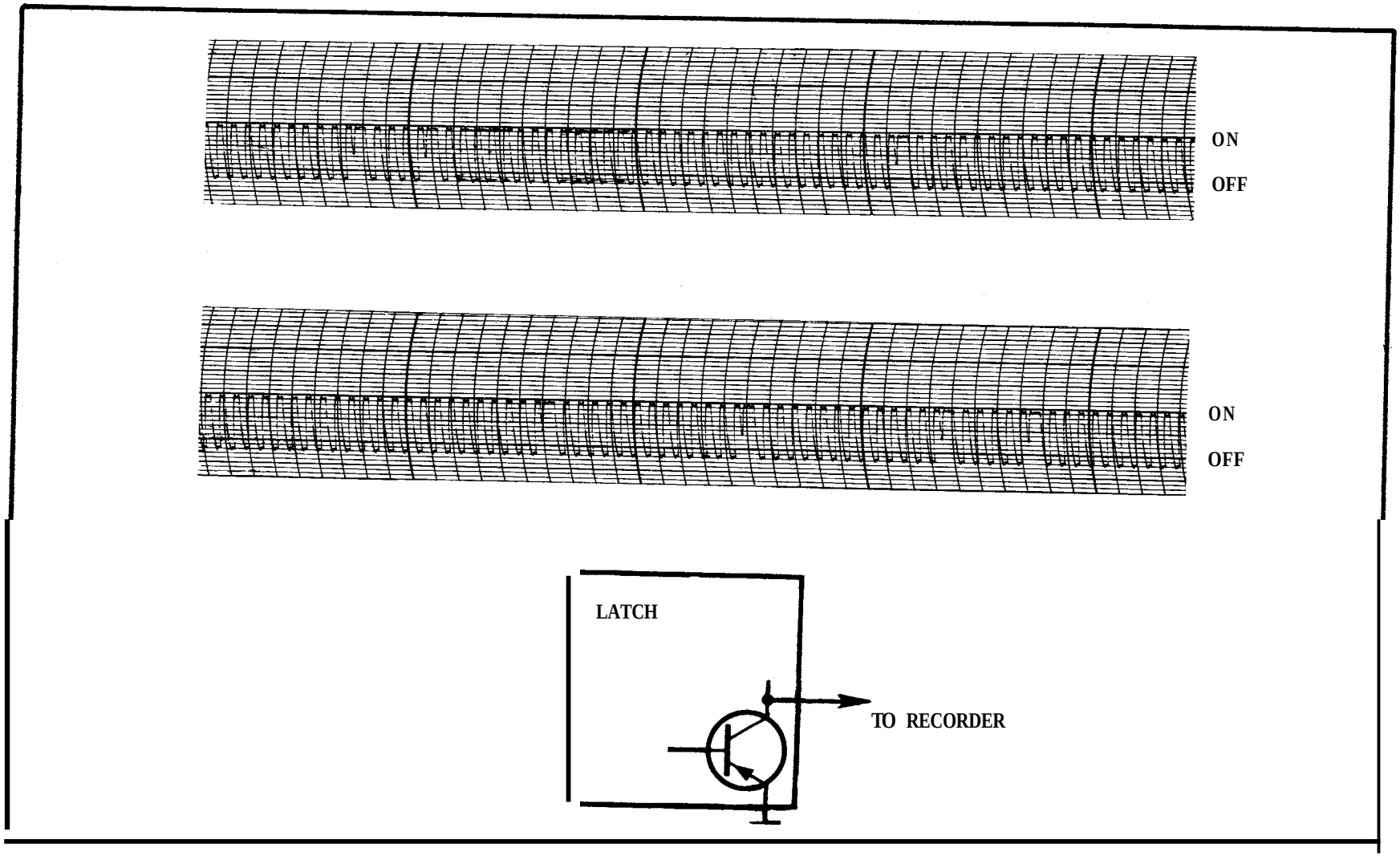


Fig. **4-16** **A.** latch output of hysteresis motor control, no filters

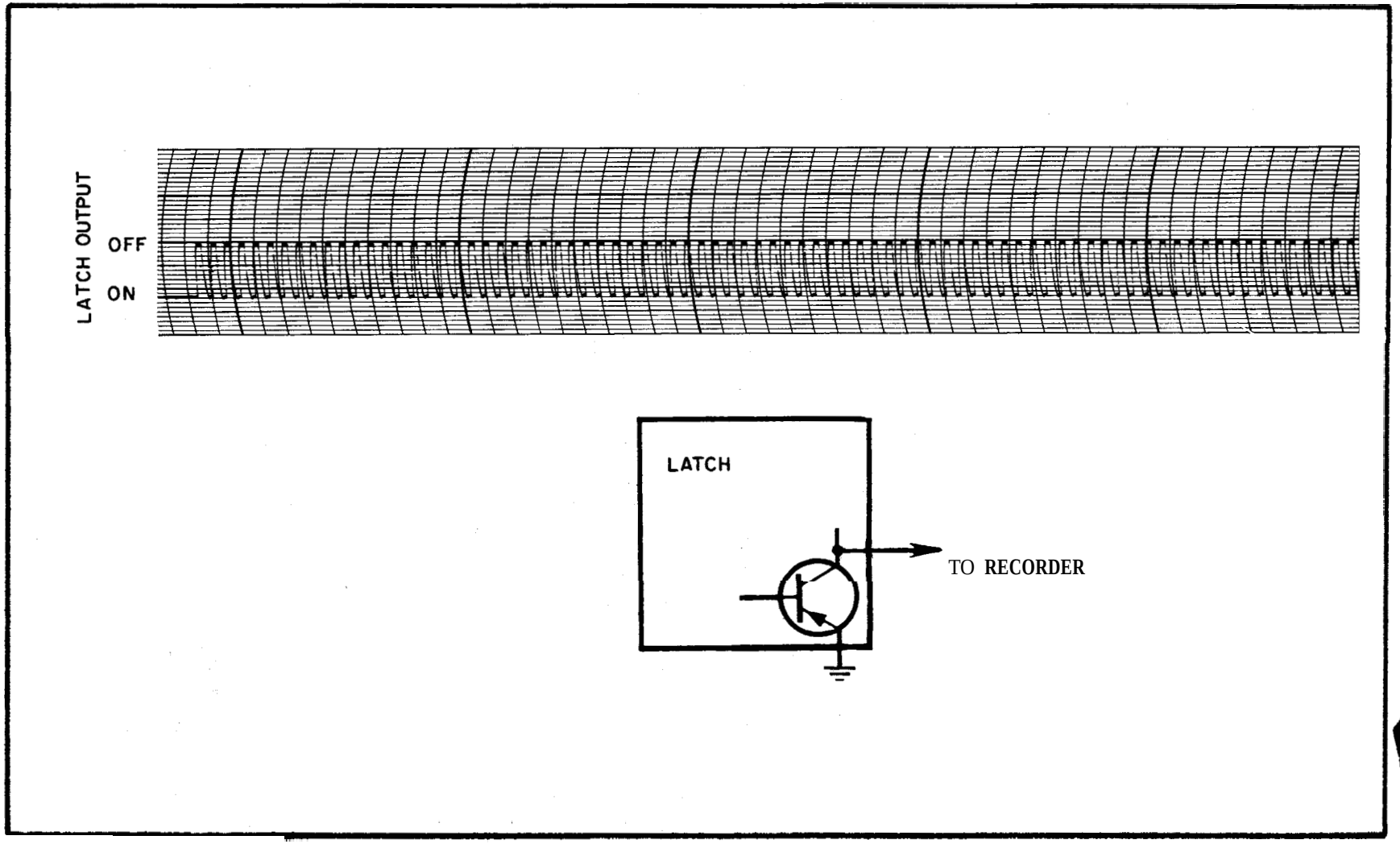


Fig. 4-17 A latch output of hysteresis motor control circuit

4.10 Optical Encoder

An optical shaft angle encoder is used with hysteresis motor MT 2, 7-198. Fig. 4-18 shows the encoder, consisting of a circular disk, two lamps, and two optical modular pickoffs.

The disk is divided into transparent and opaque sections with a zero slit provided for marker information, if desired (Fig. 4-19). The lamps are mounted to maintain between them an angular separation of 90^{o}.

Optical pickoff modules, with each potted module containing a solar cell and d-c amplifier, are paired on the opposite side of the disk with the lamps. A separate lamp and optical pickoff module can be used with the zero slit to provide a marker pulse on a third line.

With the shaft driven at a constant angular rate, the outputs from the optical pickoffs are two square waves which are 90^{o} out of phase. Regardless of shaft angular rate there are four possible combinations of the two waveforms at any one time[4]. It can be shown that each change of state of each waveform, when combined with information on the previous state of the two waveforms, indicates whether the shaft has moved +1 or -1 increments. Each increment represents 90^{o} of rotation of the motor shaft; that is, four pulses per revolution.

4.11 Decoding

The decoding circuits accept the output voltages from the two optical pickoffs and provide pulses on one of two separate output lines, which one depending upon direction of rotation. The increments are then accumulated through the increment circuits of the computer in an addressable register.

The d-c amplifiers, shapers, and decoding circuits are the same as those used by the Stepper-Motor Control (Fig. 4-8).

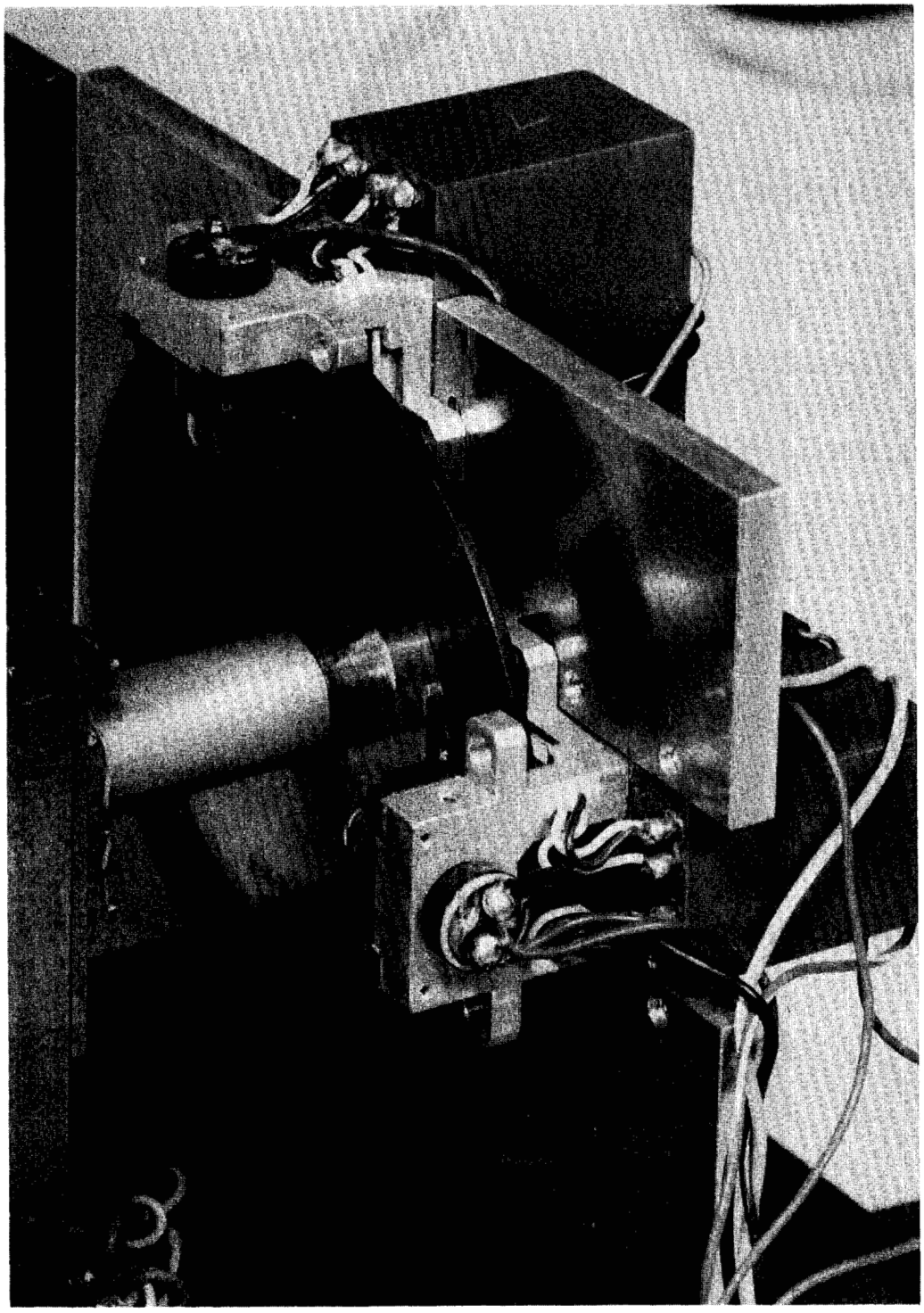


Fig. 4-18 Encoder assembly

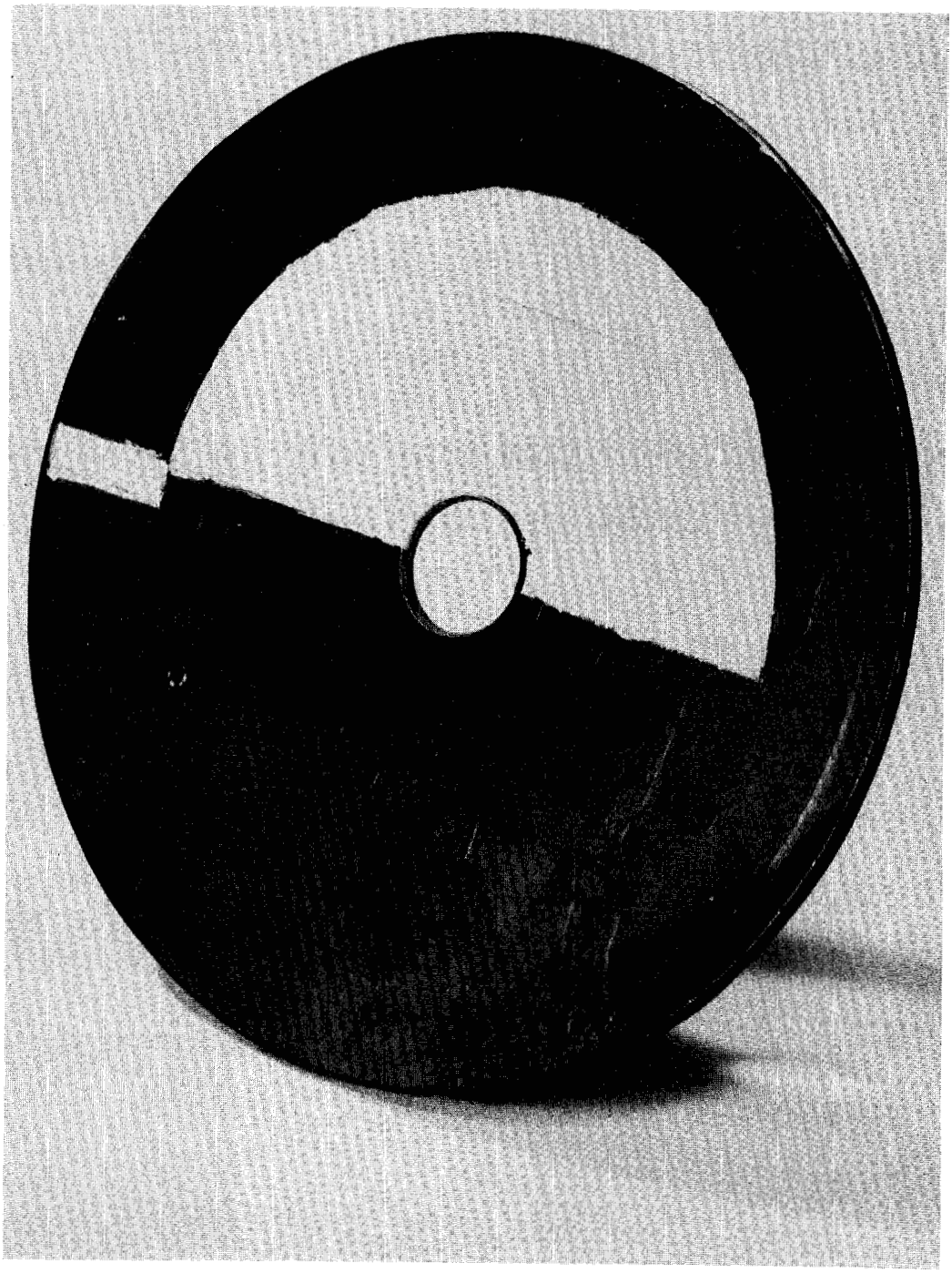


Fig. 4-19 Encoder disk with zero slit

BIBLIOGRAPHY

1. Alonso, R., Laning, J.H., Jr., Design Principles for a General Control Computer, Report R-276, (Instrumentation Laboratory, Massachusetts Institute of Technology, April 1960).

2. Maurer, H.E., Pulse Rate Divider and Stepper-Motor Logic, Report E-971, (Instrumentation Laboratory, Massachusetts Institute of Technology, November 1960).

3. Swonger, Claron W., A Study of the Operation of Polyphase Svnchronous Motors as Stepper-Motors, Report R-205, (Instrumentation Laboratory, Massachusetts Institute of Technology, June 1959).

4. Swonger, Claron W., The Performance of a Two-Phase Motor as a Stepper-Motor, Report E-763, (Instrumentation Laboratory, Massachusetts Institute of Technology, September 1958).

5. Seward, Harold, A Photoelectric Shaft-Digitizer of Improved Stability, Report E-614, (Instrumentation Laboratory, Massachusetts Institute of Technology, February 1957).

6. Optisyn Specifications for Model 23, (Dynamics Research Corporation, July 1961).

7. A Recoverable Interplanetary Space Probe, Report R-235, Vol. II, (Instrumentation Laboratory, Massachusetts Institute of Technology, July 1959).

APPENDIX A
REGISTER ASSIGNMENTS
by A. I. Green

APPENDIX A

REGISTER ASSIGNMENTS

Octal		Octal	
740,	Temporary Storage	760,	Switch Register #4
741,	Temporary Storage	761,	Cycle Left Register
742,	Temporary Storage	762,	Shift Right Register
743,	Temporary Storage'	763,	Light Register #4
744,	Constant Storage	764,	Resume Address
745,	Constant Storage	765,	Preset Counter
746,	Constant Storage	766,	5 P.P.S. Counter
747,	Constant Storage	767,	P_{AC} Counter
750,	Constant Storage	770,	P_{AC} Interrupt Address Register
751,	Constant Storage	771,	5 P.P.S. Interrupt Address Register
752,	Constant Storage	772,	Preset Interrupt Address Register
753,	Constant Storage	773,	START Interrupt Address Register
754,	Constant Storage	774,	Kate Magnitude Control Register
755,	Constant Storage	775,	5 P.P.S. ON/OFF Control Register
756,	Constant Storage	776,	Rate Sign Control Register
757,	Temporary Storage	777,	5:1, 1:1 Control Register

APPENDIX B
BIT ASSIGNMENT OF SPECIAL REGISTERS
by A.I. Green

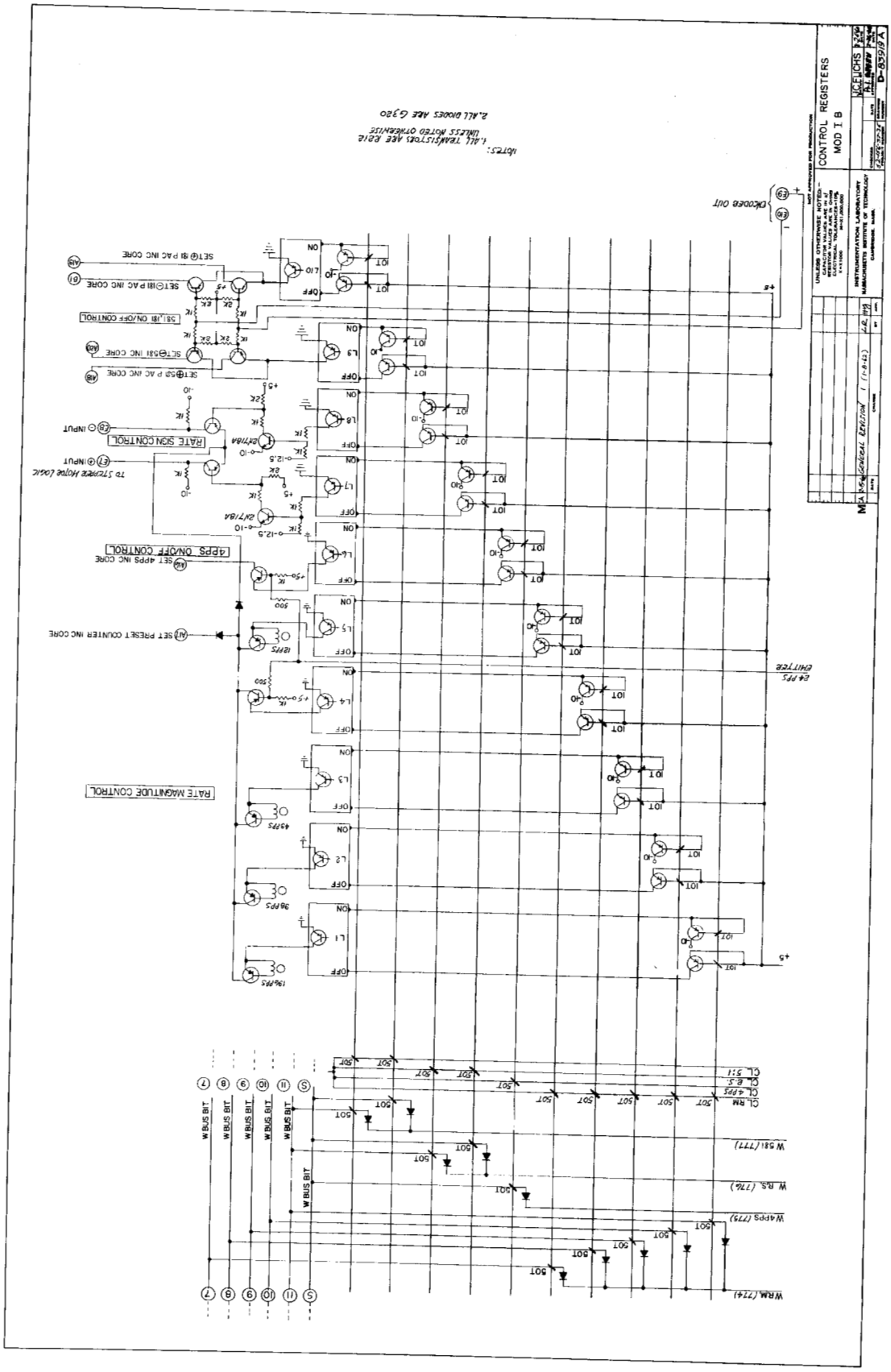


Fig. B-1 Control registers

APPENDIX C
CORE PACKAGES USED IN MOD 1B

by R, E. Oleksiak