TA165
·M41
·I592
no.
R-276

RSC
MASS. INST. TECH.
JUL 28 1972
LIBRARIES

R-276

DESIGN PRINCIPLES FOR A GENERAL CONTROL COMPUTER

by

R. Alonso

J. H. Laning, Jr.

April 1960

INSTRUMENTATION LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Cambridge 39, Mass.

R-276

DESIGN PRINCIPLES FOR A GENERAL CONTROL COMPUTER

by

R. Alonso
J. H. Laning, Jr.

April 1960

INSTRUMENTATION LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE 39, MASSACHUSETTS

Approved: R. B. Woodbury
Associate Director

ACKNOWLEDGMENT

Part of the work presented in this report was done by Dr. Richard H. Battin and Mr. Robert E. Oleksiak. Dr. Battin took part in the discussions on order codes, and authored the final refined version of the "representative" order code. Mr. Oleksiak had a major part in the electrical design of both the fixed and erasable memory systems constructed and operated to date, designed the specialized and necessary Z and P registers, and contributed to the invention of the priority circuit.

This report was prepared under Project DSR 52-156 sponsored by the Ballistic Missile Division of the Air Research and Development Command through USAF Contract AF 04(647)-303 and Project DSR 53-138, Division of Sponsored Research, Massachusetts Institute of Technology, sponsored by the Bureau of Ordnance, Department of the Navy, under Contract NOrd 17366.

This document has been reprinted January , 1962 under National Space & Aeronautics Authority Contract NAS 9-153.

RECEIVED
JUL 26 1972
M. I. T. LIBRARIES

ABSTRACT

A set of techniques is described which permits the design of computers particularly well suited for outer space and other airborne control applications. These techniques are unified by reference to a representative computer.

Some of the properties of the computer are: variable speed, power consumption proportional to speed, relatively few transistors, relatively large storage for program and constants, and parallel transfer of words. Certain features of the input system permit automatic incrementing of counters and automatic interruption of normal computer processes upon receipt of inputs.

The program and constants are stored in a wired-in form of memory which permits unusually high bit densities.

625621

TABLEOFCONTENTS

		Page
Introduction.		7
Chapter I	General Organization of Storage	11
A.	The *Core* Rope (Wired-in Memory)	11
B.	Erasable Storage Registers	17
C.	The Erasable Storage System	23
D.	The Complete Memory System	26
Chapter II	Execution of Computer Instructions	33
A.	Central Registers and Overall Organization	33
B.	The Sequence Generator	39
C.	Representative Order Code System	43
D.	Addition and Multiplication	60
Chapter III	Input- Output Systems	67
A.	Introduction	67
B.	The Priority Circuit	68
C.	The Automatic Interrupt	75
D.	State Inputs,	76
E.	Counter Inputs	80
F.	Output Devices	88
Chapter IV	General Remarks	97
A	Sizes and Weights	97
B.	Reliability Problems.	100
C.	Practical Experience to Date	103
D.	Power and Speed	104

LIST OF ILLUSTRATIONS

Figure		Page
1-1	256 word core rope,	12
1-2	Selection of a core in a core rope	14
1-3	Timing of inhibiting, set and clear currents	15
1-4	Sensing wires on a core rope.	16
1-5	Symbolic erasable register.	18
1-6	Erasable register	20
1-7	Erasable storage system.	22
1-8	System of ten erasable storage registers	24
1-9	Erasable storage selection system.	27
1-10	Complete memory system	28
2-1	Over-all organization of the representative computer	34
2-2	Sequence generator	38
2-3	Control pulse generation by sequence generator	40
2-4	Representative central register system	44
2-5	Average number of cycles for addition.	63
3-1	Priority circuit	69
3-2	Priority circuit system	71
3-3	Input circuits for priority circuit system	72
3-4	Unit M1 for input to priority circuit.	77
3-5	Circuitry for state input	79
3-6	Ideal counter register	82
3-7	System for counter inputs	83
3-8	Counter inputs -- system for dealing with positive and negative increments	87
3-9	Output switch circuit	89
3-10	Gating of a pulse generator by an output switch	90
3-11	Generator of several pulse rates,	92

Figure Page

3-12 Details of an output rate register. 93

3-13 Pulse rate generation and examples 95

4-1 Sixteen word erasable storage system 99

4-2 Physical characteristics of components 101

DESIGN PRINCIPLES FOR A GENERAL CONTROL COMPUTER

INTRODUCTION

The subject of this report is a medium-speed, general-purpose digital computer designed to serve as a controlling element for a deep space probe or other complex airborne system; it is a continuation of the work presented in Report R-235:*. Major features of the design are the small size and weight of this computer, its rugged construction, low power consumption, and a combination of circuits and logical organization which allows many sophisticated operations to be accomplished quite simply. All of this can be achieved by circuits composed of magnetic cores, diodes, and transistors, with the latter being few in number relative to comparable existing computers.

The computer is not presented in the context of a specific application, but rather as a collection of useful techniques and approaches, especially with regard to inputs and outputs. To assist the reader in visualizing reasonable areas of potential uses, sizes and speeds for a representative machine are nonetheless discussed These would vary with the control application, but are meaningful in indicating the kinds of problems the computer can best solve. The remainder of this Introduction summarizes the principal characteristics of this machine.

---

* Chapter 10, Vol III, A Recoverable Interplanetary Space Probe MIT Instrumentation Laboratory Report R-235, July 1959.

Computer memory consists of a wired-in section for program and constants plus a smaller volatile memory for variable items. Both are random access in nature, with all transfers occurring in parallel. The wired-in memory provides for storage of information at a density between ten and one hundred times that achieved with the volatile type of memory (see Chap. 4, Section A of this report). As a result, the proposed computer is well suited for the solution of a large class of airborne and spaceborne control problems which require the versatility afforded by long and intricate logical or arithmetic programs, or require the storage of many constants, but which use relatively few variables. The representative machine possesses about 4000 words of wired-in storage and 128 words of volatile storage, all words being of the order of 23 bits in length, plus a parity bit. Estimates of a representative computer's size and weight are 0. 37 ft^3 and 20 lb.

Various input-handling techniques are presented which assist efficient computer utilization. The input lines are not scanned; when an input line becomes active, the computer interrupts its present activity to tend to that input, returning to its former task after the input has been dealt with. This mode of operation is particularly advantageous in the case in which there are many slowly varying inputs. (A scanning system requires that computer speed be increased with an increase in the number of input lines.) However, the computer is also well equipped for the counting of rapid trains of input pulses; each such pulse costs only two clock cycles* on the part of the computer for its processing. These techniques employ the efficient time Sharing of a small amount of common equipment and thereby permit a relatively complex control situation to be handled by a small computer.

*A clock cycle is composed of two clock pulses, α and β.

The order code for the representative computer is of the single address type and is one in which 3 to 5 clock cycles are required for the operations involving transfer of information from one point to another or for logical operations. Addition and multiplication, however, require an average of about 27 and 750 clock cycles respectively; in fact, multiplication is accomplished by a subroutine. At a nominal 100 KC clock rate, multiplication requires 7. 5 ms, addition 0. 27 ms, and many other operations only 0. 04 ms.

Computer speed is variable, and can be controlled both by external events and by the computer program itself. When functioning at full speed, the computer will consume power on the order of 20 watts, exclusive of input and output equipment. The electrical design, however, permits the computer to function at any clock cycle rate from a nominal maximum of 100 KC to zero cps, with a power consumption proportional to speed. Hence it is possible to take advantage of long periods during which the computer need not operate very rapidly, and save power. While in this semi-dormant status, the ability of the computer to process input information instantly on demand is in no way impaired; thus the effective operation speed can be exactly that dictated by the input requirement. This property is clearly useful in the case of a deep space probe.

The computer is expected to be long-lived, in the sense of no catastrophic failures. This is because the number of active components is low (of the order of 1500 transistors), and the power dissipated in each of these is well within the components' ratings. Tne possible sophistication and length of the computer program can be used for checking computations and to provide alternative actions in the case of failures of parts of the controlled system. Thus the survival capabilities of the controlled system could be increased significantly, especially in the case of temporary malfunctions.

The remaining chapters in this report discuss in some detail the logic and circuits for this computer. An effort is made to lead the reader progressively from the simple and separate elements of which the machine is composed, taward the complexities of the overall logical organization; the report is therefore best read in sequence. Many details are passed over in the process; the first rudimentary version of the machine is just now being assembled, and many changes are certain to be made. The overall organization and logic, however, appear to be sufficiently unique and promising in their possible applications to justify a report at this time.

CHAPTER 1

GENERAL ORGANIZATION OF STORAGE

A. The Core Rope (Wired-In Memory)

The term "core rope" is used to denote the device used as the wired-in or "fixed" storage for this computer and as the selection switch controlling operation of the volatile or "erasable" storage. The term is derived from the physical appearance of the array of cores and wiring; a rope of 256 cores is shown in Fig. 1-1. By means of the rope wiring pattern, an arbitary binary address can be translated into the switching of a single corresponding rope core, which may in turn cause other actions such as read-out of an associated erasable (volatile) storage register. Additional sensing-wiring through the core permits the read-out of a wired-in binary number upon resetting the core, providing a medium for fixed storage.

The core rope, which was invented by Olsen of Lincoln Laboratory, and which is also known as the Diamond Switch, the Rajchman Selection Switch, and the Linear Selection Switch, makes use of cores as saturable pulse transformers. The cores are small permalloy ribbon cores, with a fairly square hysteresis loop. A small 8-core rope is shown in schematic form in Fig. 1-2. There are two pulse generators, one providing a ground at point (1) at TIME 1, and the other providing a ground at point (2) at TIME 2. The duration and timing of these grounds are shown in Fig. 1-3. The lines marked A, $\bar{A}$, B, $\bar{B}$, C, and $\bar{C}$ are called Inhibitors. The slanted line at the intersection of a core and a wire shows that the wire in question threads that core. The

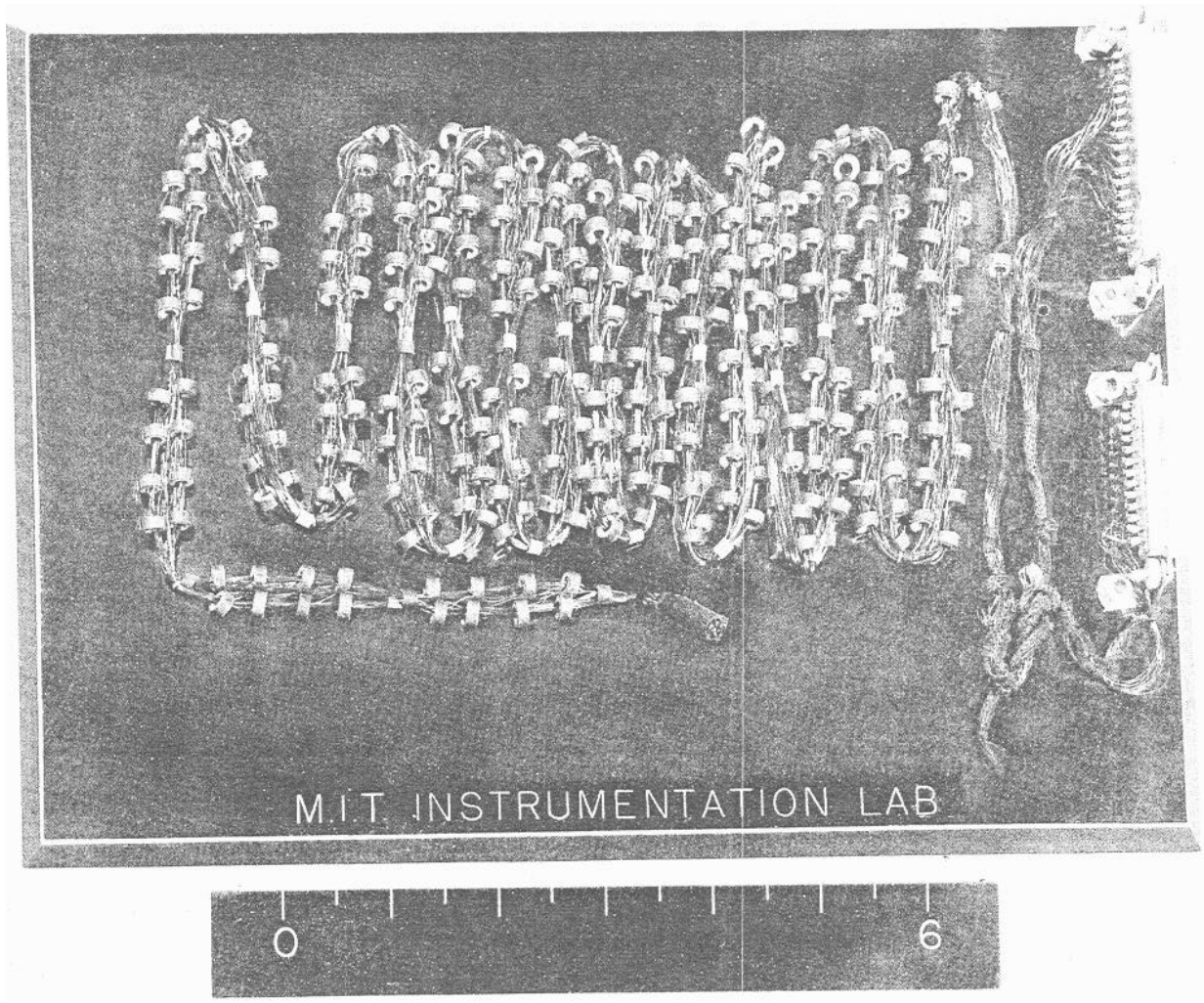


Fig. 1-1 256 word core rope

direction of the slant relative to the current in the wire indicates the direction of threading.

All eight cores shown in Fig. 1- 2 are initially in state ZERO. The Set line puts a current through all the cores in a sense that drives all cores toward state ONE. A current-carrying Inhibit or, on the other hand, prevents those cores which it threads from switching from ZERO to ONE. If it does not carry current, i. e., if it is not connected to point (1) by virtue of the position of the appropriate switch*, then it has no effect on any core. Each pair of Inhibitors in Fig. 1-2 may be regarded as one binary digit position in a 3-digit address; one and only one Inhibitor from a pair is active at any time. It is easy to verify from the wiring pattern shown that each of the eight possible combinations of inhibitors connected to the pulse source corresponds to a single core being switched from ZERO to ONE by a current through the Set line, all other cores being inhibited by at least one inhibiting current.

The Clear line, which carries current at TIME 2, tends to drive all the cores toward state ZERO. However, there is only one core to switch, the core that is in state ONE at that time.

Thus far we have considered how a core is selected, but not how information is stored in that core. This is accomplished by means of an additional set of windings on the cores of the rope, as shown in Fig. 1-4. As before, a threaded core is shown by means of a slanted line at the intersection of core and wire. For example, core number (7) is threaded only by Sense line d; core (6) is threaded by all Sense lines. Upon selecting and switching a core (say (4)), all Sense lines which thread it will

*Symbolized in Fig. l-2 as a relay contact, but electronic in practice.

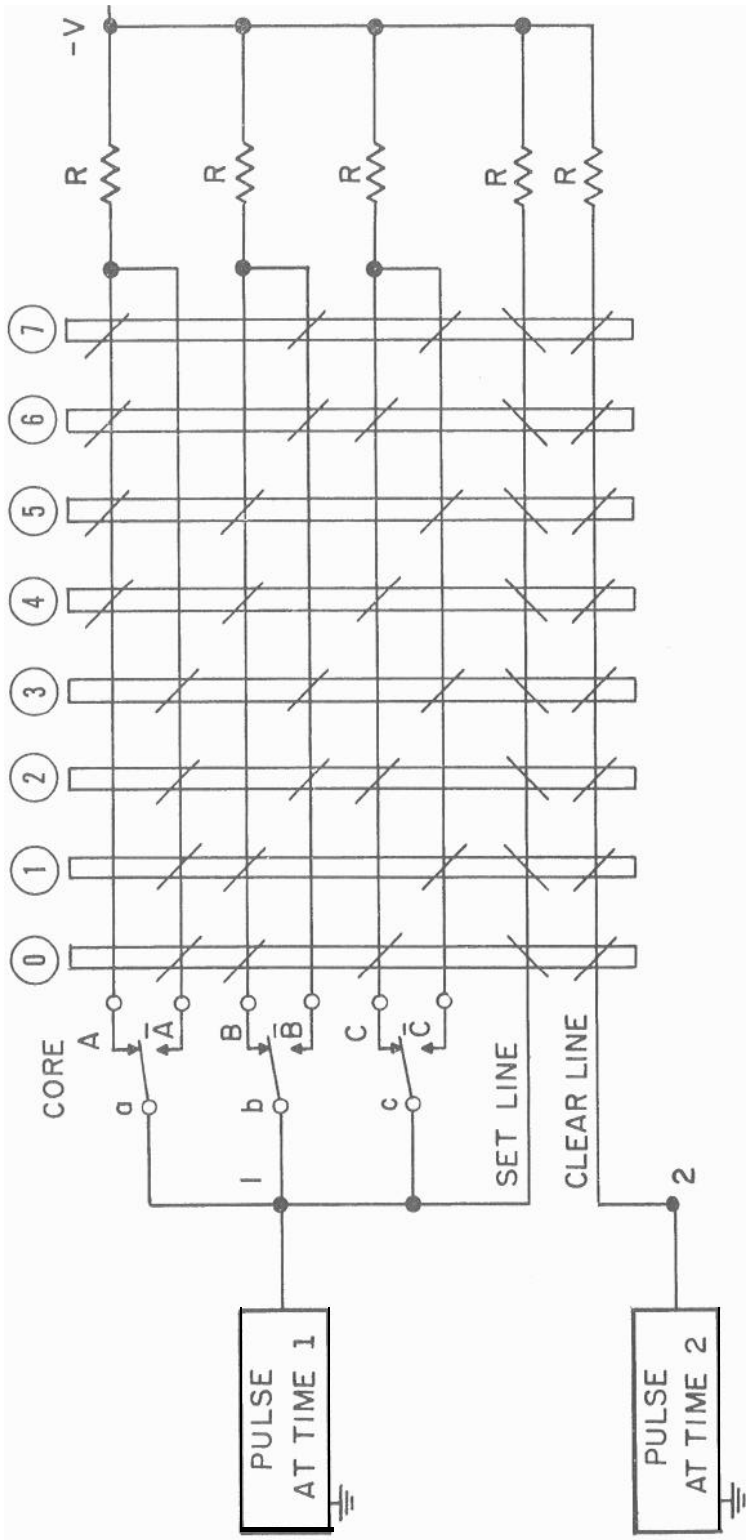


Fig. 1-2 of a core in a core rope

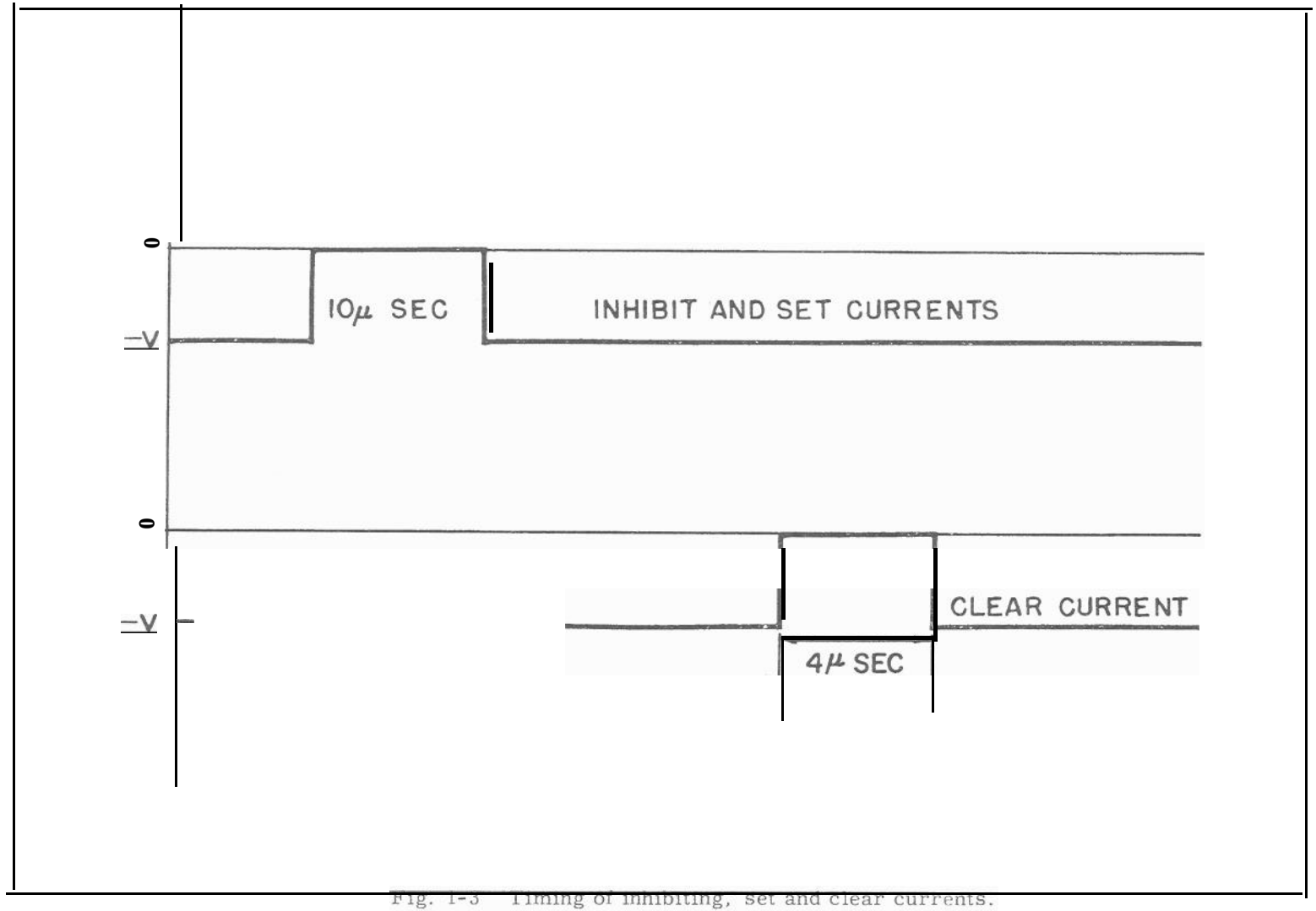
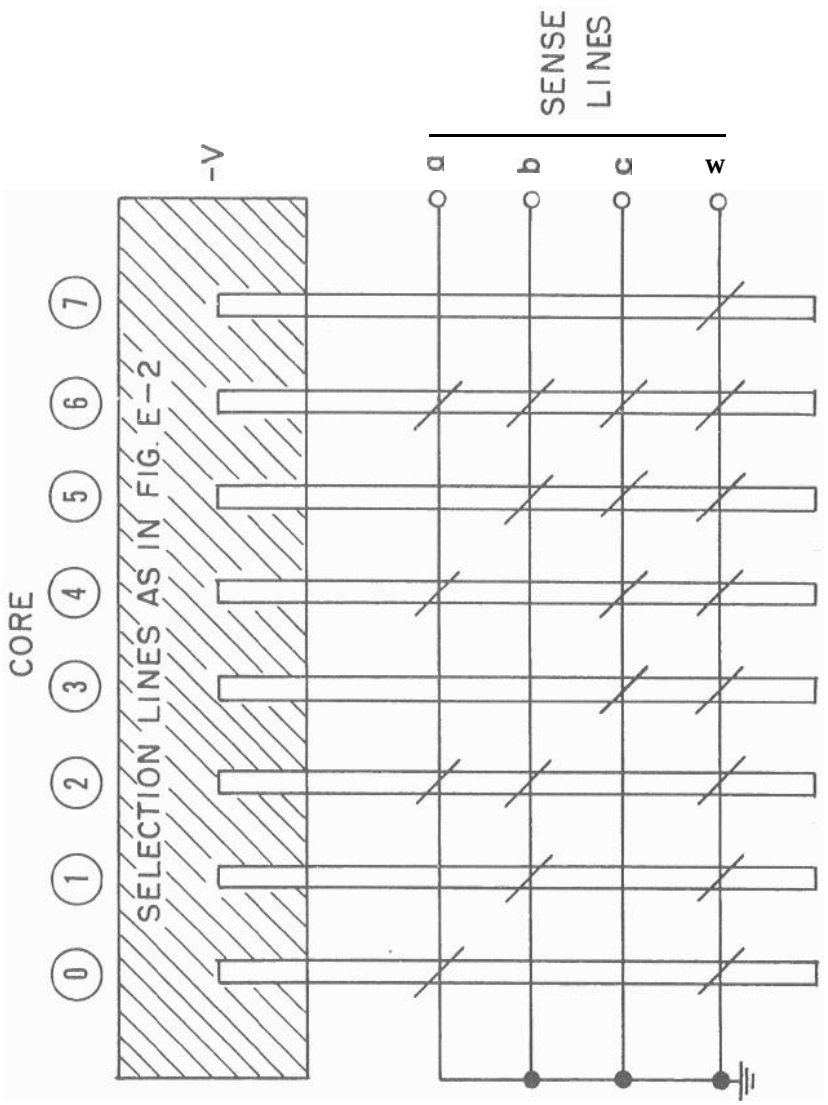


Fig. 1-3 Timing of inhibiting, set and clear currents.


SENSE LINES
a
b
c
w
-V
CORE
0
1
2
3
4
5
6
7
SELECTION LINES AS IN FIG. E-2

have an emf induced on them, and the remaining lines will not. Thus a pattern of ONE's and ZERO's in the resultant word can be specified by prescribing which lines do and do not thread the corresponding core.

In practice, the signals are observed at Clear time rather than at Set time, because the interference or noise induced by the currents in the many Inhibit lines masks the signals induced by the Set current. Nevertheless, the engineering problems of eliminating or minimizing the various sources of interference of this nature have provided one of the main obstacles that was necessary to overcome in the course of developing this computer. The situation is felt to be well under control for the memory sizes presently considered, but might constitute a problem in extending this technique to memories of significantly larger size.

Clearly, the size of the word stored per core is arbitrary, depending merely on the number of Sense lines. As a last remark, the whole core rope system may be thought of as a coding switch, with a correspondence between inputs (inhibiting currents) and outputs (emf's on Sense lines), rather than as a "memory" in the traditional computer sense. It is noted that the above method for selecting a core and making it switch is very economical, for it requires the logical minimum of 2N Inhibitors for selecting one out of 2^N cores.

B. Erasable Storage Registers

Each core of the rope just described may be regarded as a register, since it stores a word. However, that word cannot be erased and a new one put in its place because of the wired - in nature of the information, and hence these are known as "fixed registers". In order to store variables, it is necessary to have registers such that their contents may be changed; these will be called "erasable registers". The two kinds of memories will

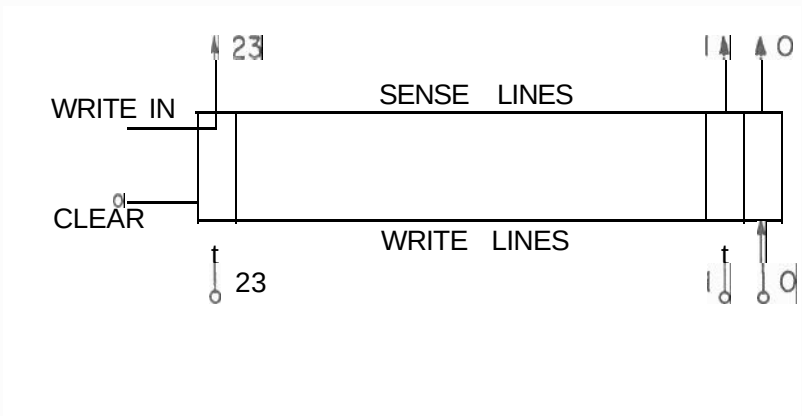


Fig. 1-5 Symbolic erasable register.

be referred to as the F memory and the E memory, or often simply as F or E. An operational description of an erasable register is shown in Fig. 1-5. Transfer of a word into and out of a register is done in parallel, i. e. , at one pulse time. In an erasable register, a separate core is required for each binary digit to be stored, in contrast to the fixed register (rope core) which stores many digits per core.

Shown in Fig. l-6 is a three-bit erasable register. When the control core is made to switch from a ZERO to ONE, transistor T1 saturates, and cores (0), (1), and (2) are cleared to ZERO. Those cores which are at ONE will induce an emf in the corresponding Sensing lines in the process of being cleared; ideally, those which are already at ZERO will not. Thus the contents of the register may be read out; e. g. , to the special buffer register shown in the figure. Note that the process of reading out from an erasable register leaves all its cores at ZERO. If the original contents are to be preserved they must be written back into the register by a subsequent operation.

To write* a word into an erasable register it is necessary to provide a current source on those writing lines for which a ONE is desired. This is indicated in idealized form by a set of relay contacts in the buffer in Fig. l-6. Writing occurs when the control core is switched from ONE back to ZERO. At this time transistor T2 acts as a gate to permit current to flow from ground to the point marked (-V). A ONE is therefore placed in those cores which correspond to closed switches in the buffer register. The diodes of the Write lines prevent interactions between different registers.

A system of three erasable registers of three bits each,

*To write into a register is physically the same as to set all the register's cores; the convention used here is that whole registers are written into, while individual cores are set. Both are cleared.

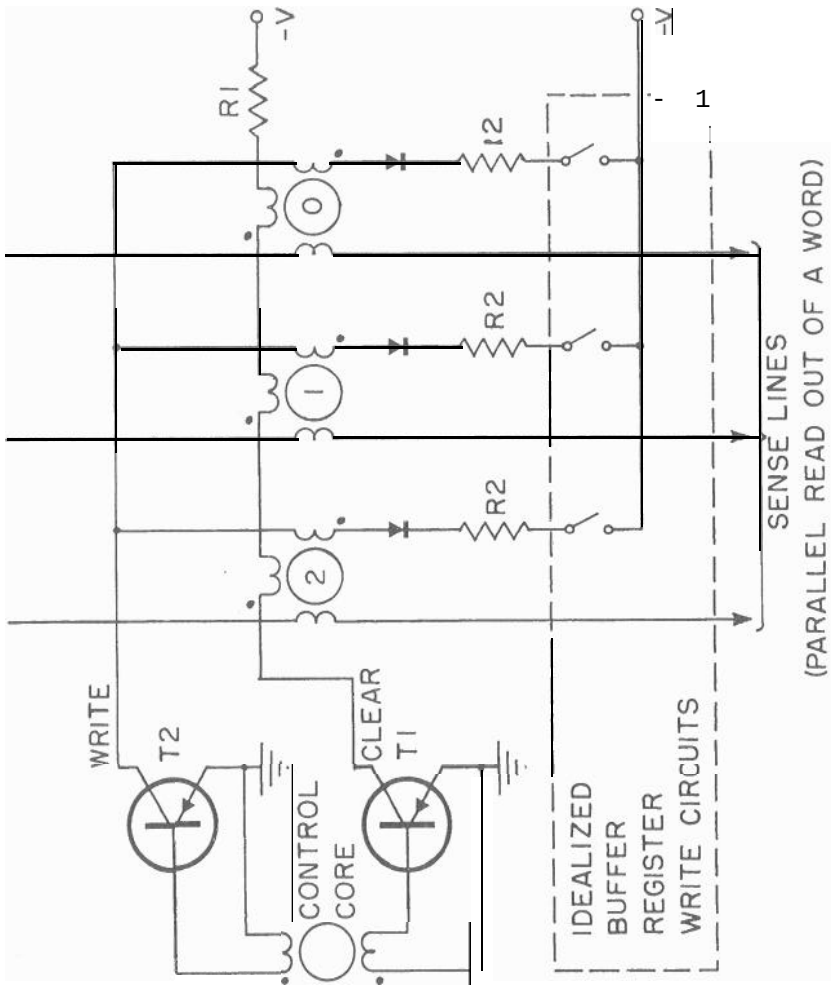


Fig. 1-6 Erasable register.

together with a buffer register, is shown in Fig. 1-7. It is seen that the buffer itself is simply another erasable register, separated from the other three by suitable sensing and writing amplifiers. In order to transfer a word from a register to the buffer, it is necessary to have time coincidence between a pulse on the Clear line of the register and a pulse on the Write line of the buffer. To transfer a word from buffer to register, the process must be reversed; i. e., the Clear line of the buffer and the Write line of the register are pulsed at the same time. The active elements in the Clear and Write lines of each register in Fig. 1-7 are the respective transistors T1 and T2 in Fig. 1-6; thus each erasable register is composed of an array of cores and diodes plus two transistors.

The results of pulsing a register's Clear line without simultaneously pulsing the buffer's Write line will be to leave all the cores of the cleared register at ZERO, and the cores of the buffer unaffected. If the buffer is cleared and the register Write line is pulsed, without having previously cleared the register, then the contents of that register will be the "logical sum" or bit by bit meet of ONE's from the word previously in that register (word A), and the word just cleared from the buffer (word B). In other words, the register will contain a ONE in those positions in which either word A or word B (or both) had a ONE.

Another possibility is that of transferring information between the buffer and two or more registers at the same time. If, at the time the buffer is cleared, the Write lines of the several registers are impulsed, the contents of the buffer will be transferred into each register (assuming them to have been cleared previously). Correspondingly, if two or more registers are cleared at once, the resultant information written into the buffer will be the logical sum of the contents of those registers.

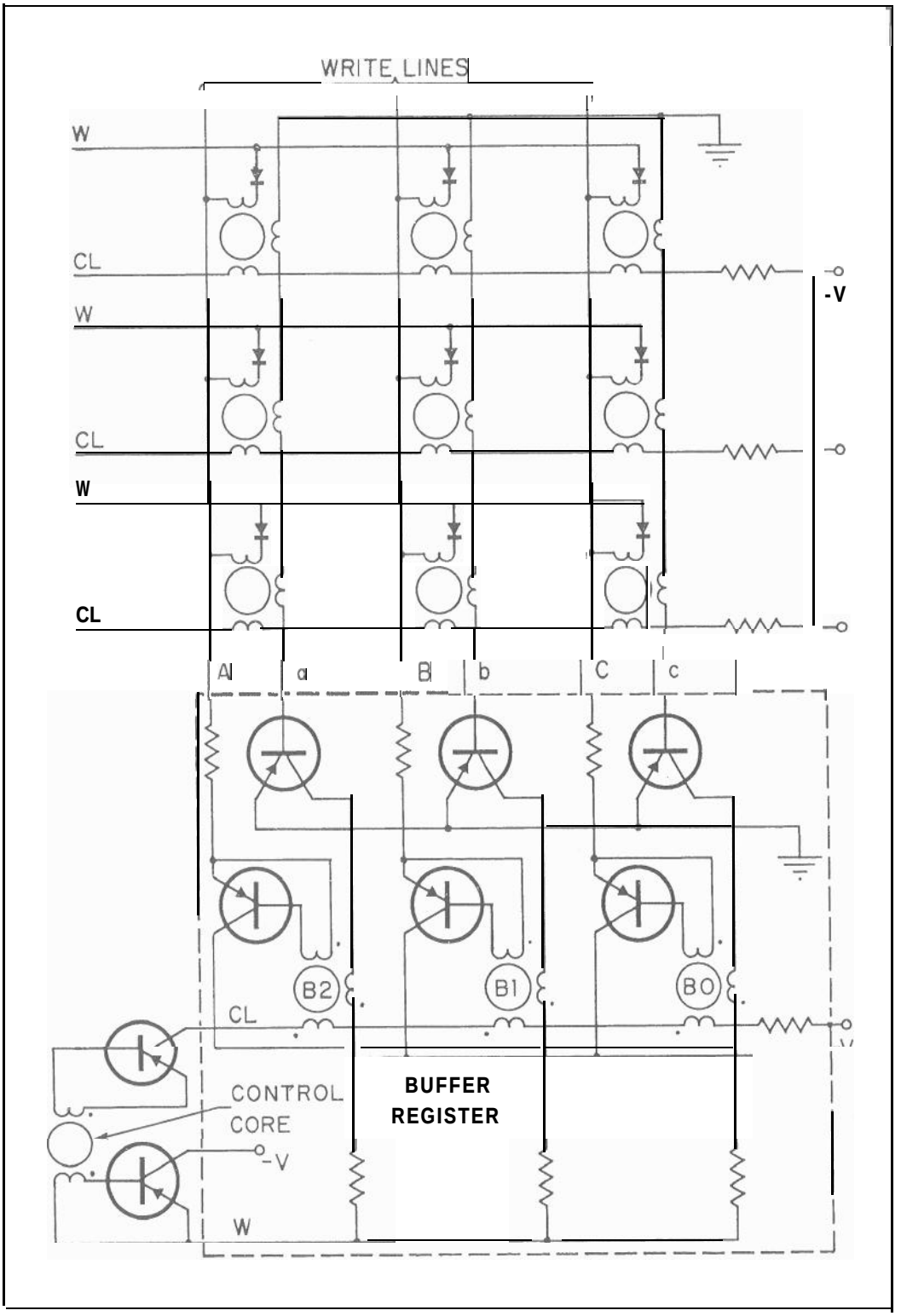


Fig. 1-7 Erasable storage system.

C. The Erasable Storage System

Shown in Fig. 1-8 is a ten-register erasable storage system, identical in operation to the small system of Fig. 1-7, but with additional useful notations illustrated. Examples are provided of several operations which can be performed on a word by use of a register with special rearrangements of the Write and Sense lines. For the sake of definiteness, the number of digits and the structure of the computer word are taken to be those for the representative machine. Each normal word has 24 binary digit positions numbered from 0 (low order bit) to 23 (high order bit). Digit 0 is customarily the parity bit (see Chap. 2, Section A) and, for numerical quantities, digit 23 is the sign bit.

The Write lines in Fig. 1-8 are represented by the arrow at the left, and directly over the arrowhead is, for each register, the list of Write lines (W) versus the bit positions of the register (R). Similarly the right hand numbers S and R refer to the Sense lines which thread the registers, and to their correspondence. For example, Register 1 has lines 23 through 0 connected to bit positions 23 through 0, on both the Write and Sense lines. This is of course the normal configuration; the wiring rearrangements in Registers 2 to 5 are the exception. Unless a register has markings to indicate the contrary, it should be assumed that it is a normal register (e. g. , Registers 6 through 10).

Registers 2 and 3 illustrate how wiring can be used to break a word into separate parts for manipulation by the computer. Transfer of a word from the buffer simultaneously into these two registers places digits 12-l in Register 2 and digits 23-13 in Register 3. When Register 2 is subsequently transferred back into the buffer, only the low order digit portion of the original word will be involved; Sense lines 23-13 pass through no cores in Register 2, and will hence carry no pulses (i. e. , ZERO's) to the sensing amplifiers. The clearing of Register 3 will cause digits 23-13 of the original word to appear on Sense lines 11-1, with the

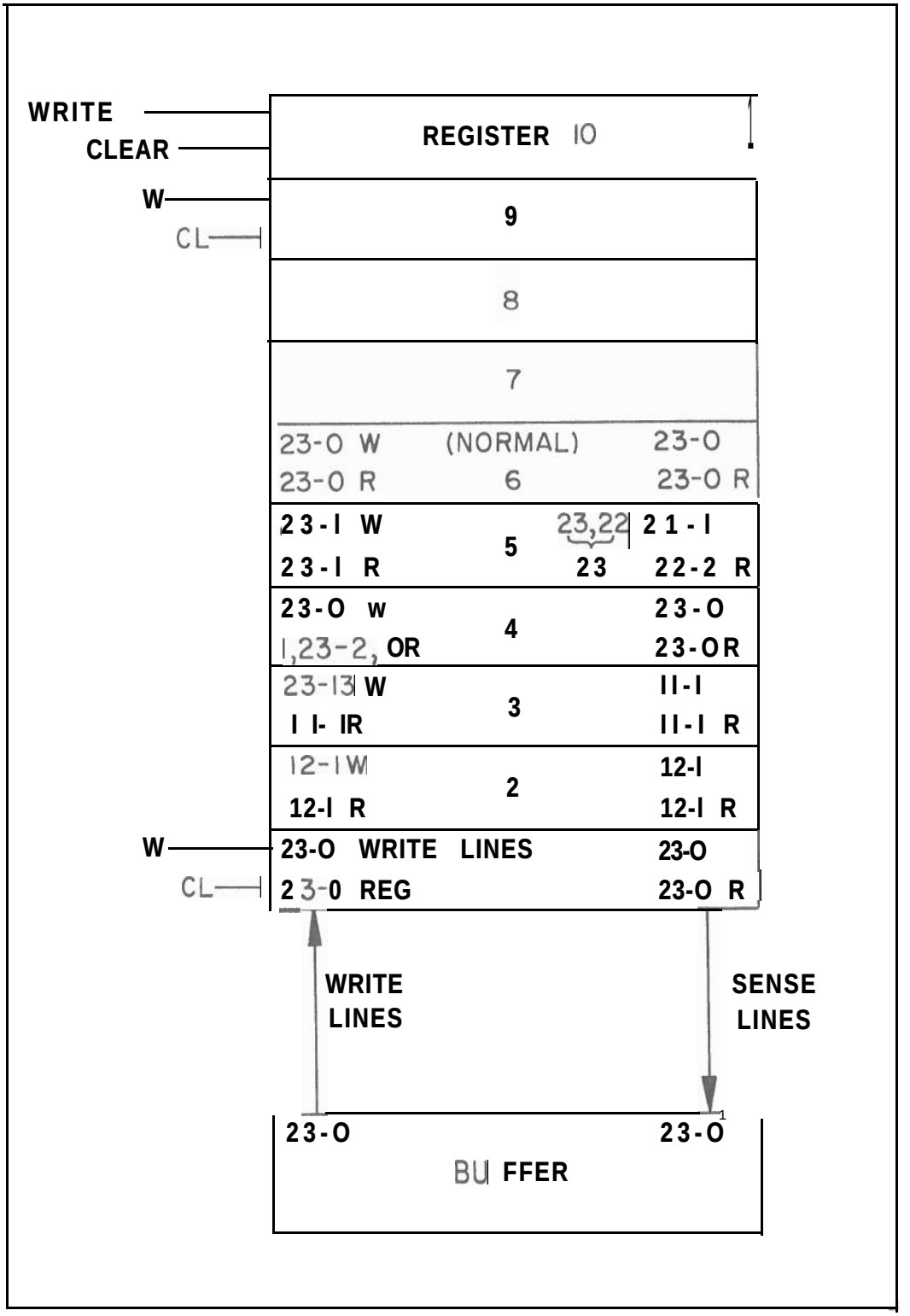


Fig. 1-8 System of ten erasable storage registers.

result that these digits are effectively shifted twelve positions to the right by the wiring of this register. Note that the parity bit (0) is omitted from the wiring of these registers, as it would lose its significance here.

As a further example of wiring rearrangements, Write line 23 is connected to bit position 1 in Register 4, Write lines 22-l are connected to bit positions 23-2 respectively, and digit 0 is left invariant. The net result of writing a word into Register 4 is to shift it left one bit, with end-around carry, preserving the parity bit in the process. Further transformations may be obtained by manipulation of the Sense lines versus the register bit positions, as in Register 5. Here, Sense lines 23 and 22 both are impulsed by the clearing of core 23; Sense lines 21-1 are connected to bits 22-2. The result is to shift the word right by one digit, with the new digit position 22 being filled with a ZERO if the word is positive (digit 23 ZERO), or with a ONE if the word is negative.

The method for selecting a register is shown in Fig. l-9. The figure shows a more complicated arrangement of the clearing and writing operations than is illustrated in Fig. 1-6. The added complexity provides greater flexibility and is further made necessary by timing considerations which are discussed in Chap. 1, Section D.

An erasable register is selected by means of a core rope, such as the one described in Chap. 1, Section A, where each core of the rope corresponds to a register. As before, all rope cores start at state ZERO. Selecting a core results in a single core (say core x of Fig. l-9) being set to a ONE; nothing else happens at this time. When the rope core is reset by a pulse on the CL E line (for Clear E) and previously called the rope Clear line), core x is switched back to ZERO, which causes transistor T1 to fire. The firing of T1 in turn clears the erasable storage Register 100 (hence the name CL E), and sets core y to

ONE. All the cores in the same vertical column as core y (the writing cores of the E registers), started at ZERO; hence after CL E only core y from among the writing cores is at ONE. The subsequent pulse on line W E (for Write E) drives core y from ONE to ZERO, causing transistor T2 to switch, and allowing the contents of the concurrently cleared buffer to be written into Register 100. As can be seen, writing into a register controlled in this fashion implies having cleared it previously.

D. The Complete Memory System

The memory system for this computer consists of fixed storage (F), erasable storage (E), and a special register (S) called the address register, by means of which storage is addressed. There are, in addition, other special registers of the erasable type called "central registers", which are used under direct control of the computer logic to execute the various instructions; the buffer B is an example*. These are discussed separately in Chap. 2, Section A; the "memory system" will be defined as the collection of registers which are controlled through the address register S.

The address register consists of two parts, denoted by the symbols Sd and S_C, which stand for the "direct" and "complement" parts of S. Each part is itself an erasable register of a length (12 bits, for the representative computer) sufficient for storing addresses of all computer words. The direct part of S stores the pertinent address. whereas S_C stores the ONE's complement of this address. The clearing of register S (i.e., the simultaneous clearing of Sd and Sc) by the control pulse CL S, causes the establishment of currents in the Inhibitor and Set lines (Fig, l-2) required for switching a rope core associated

*To be quite accurate, S should properly be called a central register rather than part of the memory system.

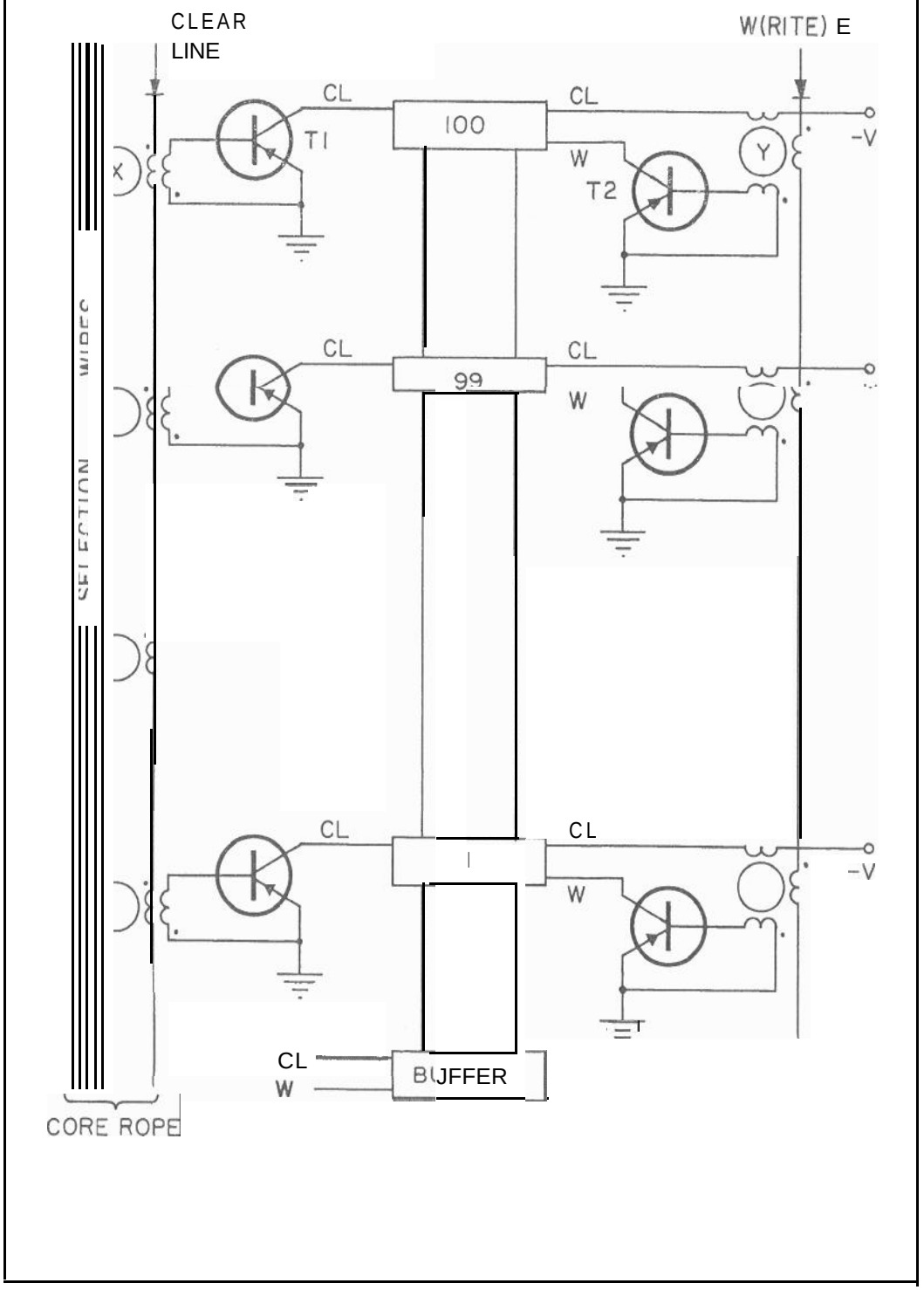


Fig. l-9 Erasable storage selection system.

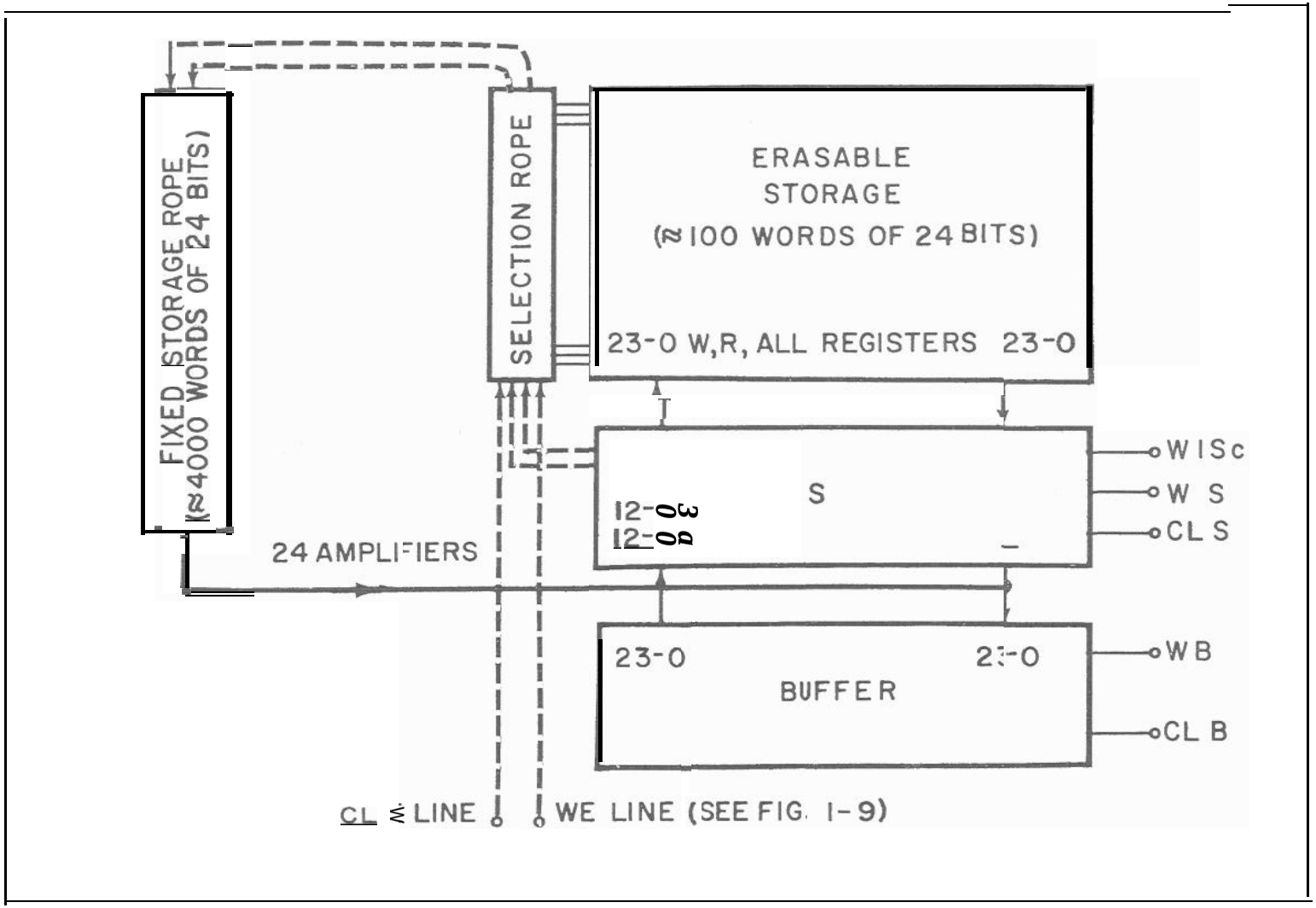


Fig. 1-10 Complete memory system.

with the address placed in S. These selecting currents are represented by the dotted lines of Fig. l-10, and are seen to be common to the fixed storage rope and the erasable storage selection rope. The two ropes thus differ only in their purpose and name, and not in the method used for addressing or selecting a core.*

Each bit of Sd and Sc has a current generating circuit connected to it, which drives a corresponding rope Inhibit line. The control pulse CL S clears all cores of both Sd and Sc to ZERO, generating thereby a current pulse in each Inhibit line for which the corresponding core contained a ONE. The pulse CL S itself initiates the current through the Set line. Each pair of corresponding cores in Sd and Sc is associated with a pair of Inhibit lines. The fact that the contents of Sd and Sc are complements thus ensures that exactly one of each Inhibit pair is active.

Register S has several peculiarities. It is not connected to the Sense lines, and hence CL S does not result in the transfer of a word from S to the buffer. Furthermore, S requires a special control pulse WlSc, which results in all the cores in Sc being driven to ONE. This pulse is applied before transferring the address from the buffer to S, which is done by the pulse WS (and CL B, of course). The latter is the gating pulse which permits writing into both Sd and Sc. The Write lines are connected to Sd in the normal fashion for an erasable register. For Sc, however, the sense of the windings is reversed from the normal direction for writing, so that a current (i.e., a ONE on the Write line) causes the core to be driven from ONE to ZERO. Thus any digit position on the Write lines which contains a ONE

*By suitable rearrangements of the inhibit wires, it would be possible to have the addresses for erasable registers distributed amongst those for fixed storage in any manner whatever, if this were desirable, even though the erasable registers were themselves physically grouped together.

causes that digit in Sd to be set to ONE and in SC to ZERO; conversely, a ZERO (no current) leaves Sd and SC unchanged, i. e. , at ZERO and ONE.

It is now possible to list the sequence of pulses involved in reading from and writing into the complete storage system, and in so doing to explain the timing associated with Fig. 1-9. Each computer clock cycle is divided into two equal intervals called α time and β time. The computer runs on a two-beat system which, at α time, information flows into (or towards) the buffer and, at β time, towards erasable storage. Most control pulses have associated with them a specific time, either α or β, during which they may occur; they may not occur at the other time. For example, CL B can occur only at β, and WB at α. A more accurate statement would be that CL B may not occur at α, and WB may not occur at β, for it is not true that a pulse CL B occurs every β time or a pulse WB every α time.

Assuming an initial address to be in register S, then the following sequence of control pulses will result in a word being read out to the buffer from the register with that address, placed back where it came from if it came from erasable storage, and also placed in S. Each line shows the time, α or β, associated with these pulses.

Time	Pulses	Comments
α	CL S	
β	(wait)	No control pulse occurs
α	CL E, WB, WlSd	Reset rope, ONE into SC
β	CL B, WS WE	

Although it may not seem so at first sight, this four -

pulse sequence takes care of both possible cases of reading from fixed or erasable storage. CL S selects a core in the rope regardless of whether it belongs to F or E, driving it to a ONE. CL E resets that core to a ZERO. If the core is associated with a register in E, then that register is cleared, its contents admitted to the buffer by virtue of the pulse WB, and the corresponding writing core (e.g., core y of Fig. 1-9) is set to ONE. When the buffer is cleared at the next β time, the pulse WE resets the writing core to ZERO, permitting the contents of B to be written back into the E register from which it came. The word is also admitted into S by the pulse WS; it is noted that Sd and Sc have been set to ZERO and ONE by the successive pulses CL S and WlSc prior to the pulse WS.

If the core switched by CL S belongs to F, then the subsequent CL E and WB causes a wired-in word to be read into the buffer through a set of amplifiers* (Fig. 1-10). The last row of pulses again causes the buffer to be cleared and S written into. WE here has no effect, for no writing core has been set to a ONE; hence none of the E registers is affected.

The sequence for replacing the word in an E register by a word previously stored in B is very similar, namely:

α: CL S
β: (wait)
α: CL E, WlSd
β: CL B, WS, WE

The only difference is the absence of WB from the second α pulse. If this pulse is absent, the word cleared from the E

*The main electrical distinction. between F and E signals is that those from F are weak and occur in the midst of noise, while signals from E are strong and free from interference. Hence the amplifiers.

register is lost. The pulses occurring at the following β time will cause the word previously placed in B to be transferred into the E register. It was assumed that the address originally in S was an E address, but the wired-in nature of F is such that nothing would happen if the address were an F address.

The control pulse sequences illustrated in this section accomplish quite elementary and restricted objectives, and are introduced primarily to discuss principles of communication with the computer memory. In the next chapter, after a discussion of central registers, consideration is given to the formation of control pulse sequences into computer orders in the accepted sense. In particular, an order code is given in more or less complete form for the representative computer, and possible variants are also presented.

CHAPTER 2

EXECUTION OF COMPUTER INSTRUCTIONS

A. Central Registers and Overall Organization

The overall organization of the computer, shown in Fig. 2-1, includes storage, discussed in Chapter 1, input-output equipment, discussed in the next chapter, together with the sequence generator and central registers which comprise the main logic for executing computer instructions. In this chapter the central registers and sequence generator are discussed, first in terms of generalities, and subsequently in somewhat more specific form in relation to the order code for the representative computer. In spite of the latter discussion, however, it should be emphasized that we are here concerned with a diverse class of possible machines rather than primarily with the specific computer selected as "representative"; this should be evident from the number of alternatives which are presented from time to time.

Central registers are, by definition, all those registers which are not addressed by means of the address register S, but whose control pulses are provided directly by the sequence generator. Registers S and B, for example, have previously been described as central registers; the pulses CL B, WS, etc., are provided by the sequence generator, as are the others listed in Chap. 1 Section D as necessary for reading from memory. The outputs from the sequence generator are, at pulse times α and β, pulses which bear names like CL B, CL E, WE, WS, WlSc, etc.

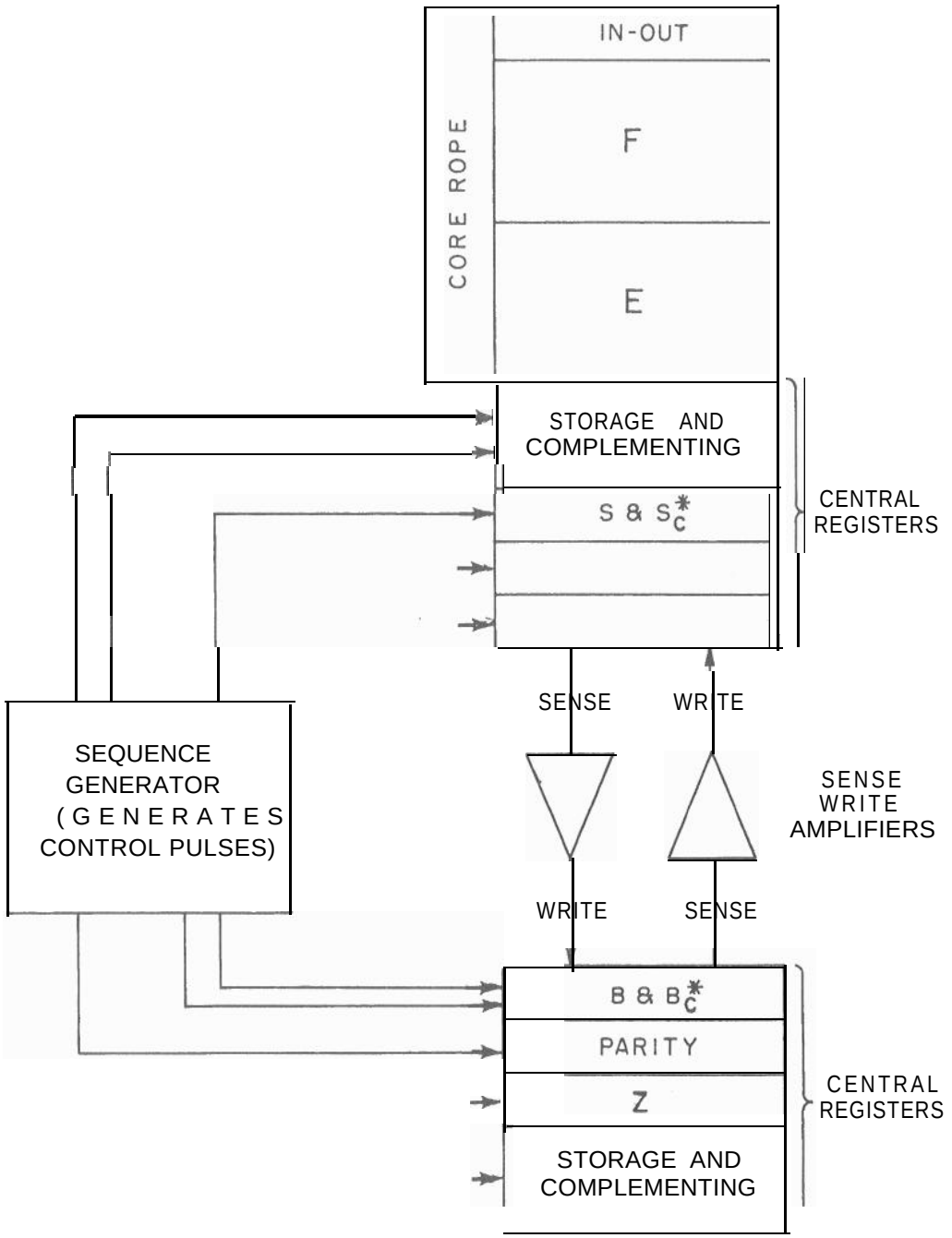


Fig. 2-1 Over-all organization of the representative computer.

Stopped Reading Here 6/30

ese control pulses go the central registers; however, a
E have other purposes. What follows is a brief
of the various kinds of central registers and a possible central register organization.

The overall computer organization is shown in Fig. 2-1. There are two sets of Sense and Write lines, one for the transfer of words from the core rope side to the B side, and one for the transfer of words from the B side to the rope side. Central registers may be on either side of the Sense- Write amplifiers and are of three kinds: special-purpose registers which have extra circuitry, complementing registers, and ordinary storage registers. The special-purpose registers are the address register S discussed previously, the **Z** register, the parity register P, and the buffer B. These are discussed separately in subsequent paragraphs. As in E storage, ordinary storage registers may occur with rearranged wiring to secure various special effects.

The "rope side" of the Sense- Write amplifiers is henceforth called the α side of the register system; i. e. , the side on which registers are cleared at α time. The "B side" is correspondingly called the β side. The division of the computer into an α side and a β side, together with the electrical properties of the erasable storage scheme, helps explain the working of the computer on a two-beat system and why each control pulse has a particular time (α or β) at which it occurs.

At α time information flows from the α side to the β side and at β time from the β side back toward the α side. It would appear that both actions could take place simultaneously, because separate sets of Sense-Write amplifiers are used. Consider, however, the signals from the sensing windings on the α side. These result from certain cores on the α side being cleared from ONE to ZERO at α time. If other cores on the α side are being driven from ZERO to ONE by a simultaneous writing operation, a reverse emf is induced in the corresponding sense lines

which cancels the desired signals where overlapping occurs. As a result, there is a significant coupling from the write side of one set of amplifiers to the sense side of the other set which prevents them from being active at the same time. Various electrical schemes could eliminate or avoid this restriction; however, the present arrangement appears to offer a simple and reliable system.

The Z register is a special-purpose register that adds "one" to a number read into it, so that a transfer into and out of Z converts the number N into N + 1. It is used, on the one hand, to store and increment the address of successive computer instructions when these occur in sequence. On the other hand, it is used on a time-shared basis to increment the various program or input counter registers (see Chap. 2, Section C and Chap. 3, Section E respectively). The electrical nature of the Z register is fairly well established; however, at the moment of writing, there are one or more unsettled questions concerning its logical integration with the rest of the computer. These relate to the treatment of input situations in which pulses to be counted may be either plus or minus, and to the length of the Z register.

The parity register, P, accepts inputs on all 24 Sense lines and computes whether the number of ONE's in the word being read into it is even or odd. Its output is one bit long and occupies digit position 0; if the number of ONE's in the word is even, then the parity bit will hold a ONE. A parity system is employed in which the total number of ONE's in a word including its parity bit, must be odd to pass the parity test. When such a word is read into P, the contents of the parity bit will be zero, so that an alarm will not be triggered when P is subsequently cleared and tested. If, on the other hand, a 23-digit word is read into P with its parity bit missing, the parity register computes and stores the correct value in digit position 0. Thus the parity may be transferred to E storage along with digits 1-23 by

clearing P at the same time the remainder of the word is stored.

The buffer register B is a special central register used as an ordinary register in communicating with storage and also used in the decoding of computer instructions. It is composed of two parts, Bd and Bc, as is the address register S. The direct part, Bd, is a full-length register; however, the complement part, Bc, occupies only the digit positions used by the order code (digits 13-18 for the representative computer). The latter bit positions of both Bd and Bc are wired to generate Inhibit pulses in a special short core rope, for selecting a sequence in the sequence generator. They operate in the same way as Sd and Sc, being cleared by the special pulse CL Bc. The pulse called CL B actually should be entitled "CL Bd" since it leaves the complement portion of B untouched. Also, the pulse CL Bc clears corresponding digits in both Bc and Bd and further provides a gate to permit the Inhibit currents to flow. The latter is necessary since Bd is used for other purposes.

Central storage registers are identical to ordinary erasable storage registers. They each require two control pulses, CL register and W register(for "Clear" and "Write"). Complementing registers are similar to storage registers but with one extra common winding added. This winding is used to set all the cores in the register to ONE, and is pulsed by a control pulse W1 Register(for "Write Ones"). The polarity of the windings which connect each core of a complementing register to a Write line is reversed relative to the polarity of a storage register. Thus if a word is written into a complementing register which has been set to all ONE's, the register will then hold the bit by bit complement of the word.

The number and arrangement of central registers of the storage and complementing types is a function of the order code desired, and of the way in which the order is translated into a sequence to be generated by the sequence generator. In

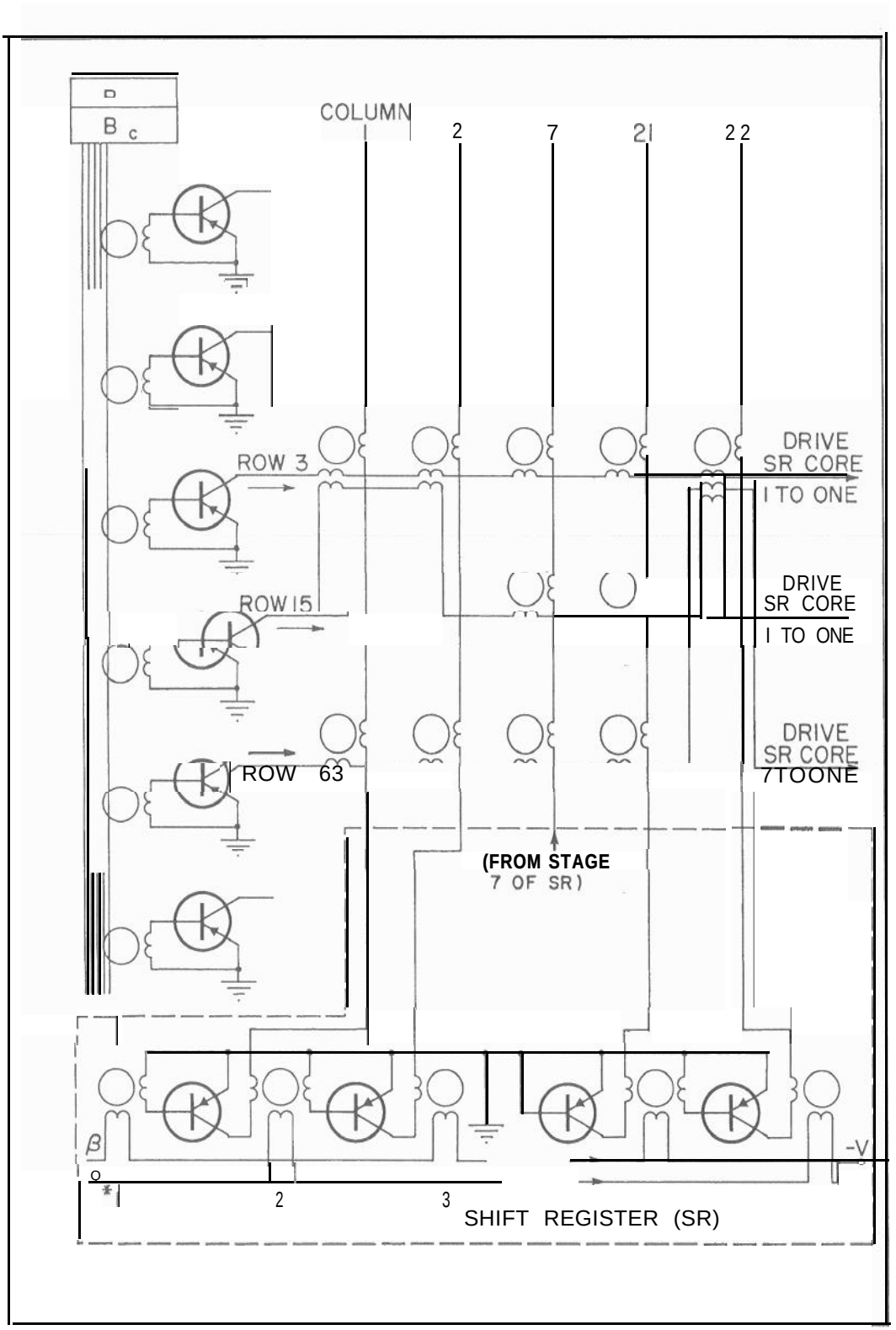


Fig. 2-2 Sequence generator.

general, the more powerful the order code, the larger the number of such central registers. These registers are in themselves no more expensive than those in erasable storage, but addressing them must be done through the sequence generator which is a costlier process than addressing through a rope. Later in this report specific arrangements of central-registers will be discussed in order to illustrate how certain things can be done and what alternatives there might be. This discussion is necessarily tied to a discussion of the order code and of the sequence generator. The address register S, the "add one" register Z, the parity register P, and the buffer B are common to all computers under discussion.

B. The Sequence Generator

The sequence generator forms what is commonly called the "logic" of a computer. Other common names for it are program counter and microprogram store. The basic idea is that of associating a sequence of control pulses with each kind of order stored in F or in E. The sequence involved in transferring the contents of some register X into X and central register A, which bears the name "clear and add X" or CA X, is one such order. That involved in storing the contents of A in X, called "transfer to storage X" or TS X, is another. By means of the sequence selection circuits (mentioned above in connection with the buffer), each combination of bits in the order code digit positions of an instruction is made to correspond to the particular pulse sequence for execution of that instruction. For reasons which will become clear later, all sequences produced in the sequence generator are called "zero level sequences".

The order structure of the representative computer is such that bit 0 (numbering from low-toward high-order digits) indicates the parity of the word (cf. Chap. 2, Section A), and the next twelve bits (bits l-12) are for any address which forms

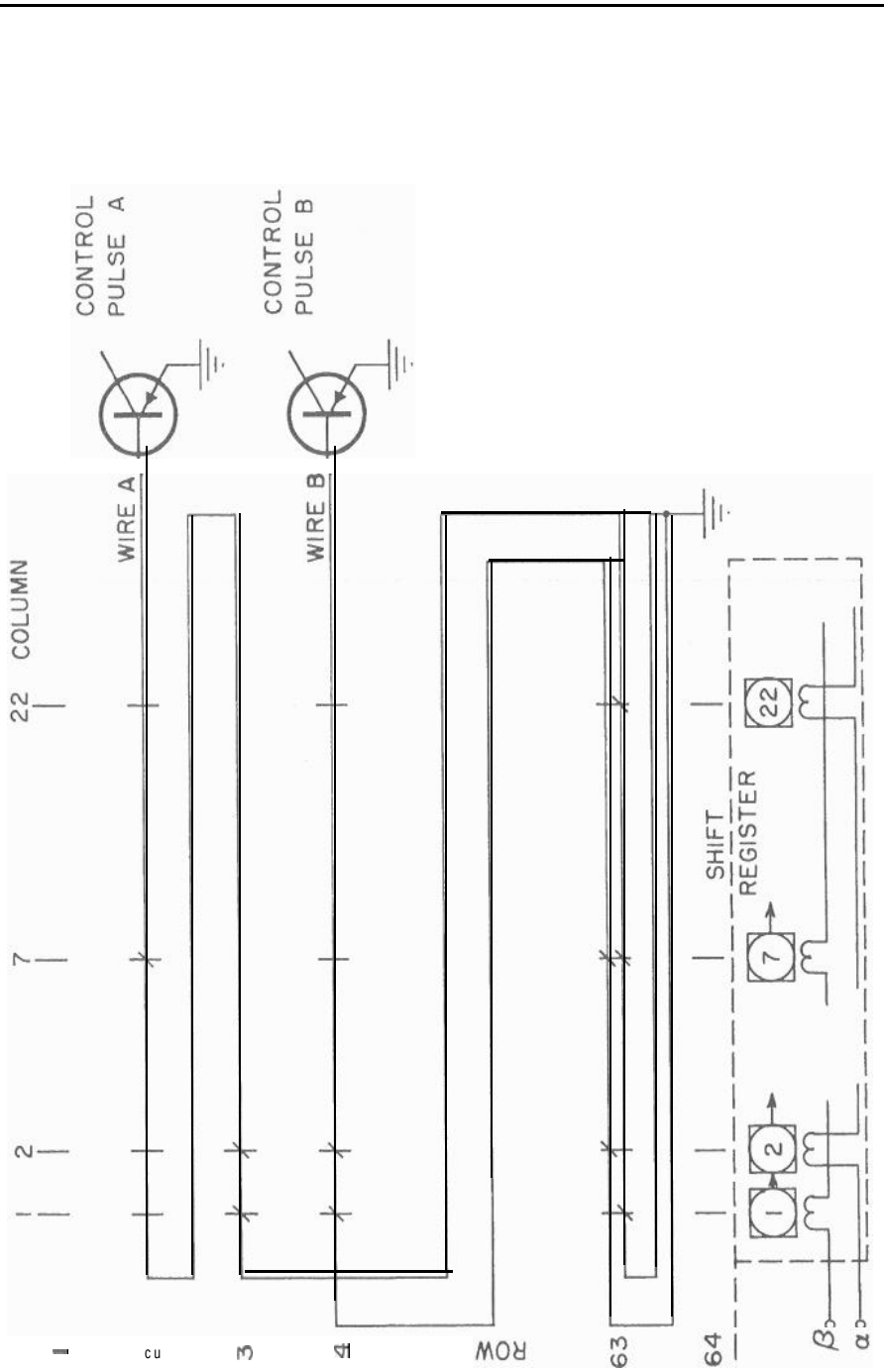


Fig. 2-3 Control pulse generation by sequence generator.

part of the instruction (e. g. , the "X" in CA X). Digits 13 to 18 form the symbol or code for an order, i. e. , the "name" of a sequence. The remaining five digits may have other optional uses which, in effect, cause them to act like part of the order code.

The sequence generator is a complicated device, and hence a simplified case will be discussed first. Consider a rectangular matrix, made of cores, with 64 rows and 22 columns (Fig. 2-2, 2-3). At the bottom of each column there is a core - transistor circuit, symbolized in Fig. 2-3 by a square box with a number inside it, which, when fired, drives all cores of that column to ZERO and drives to a ONE the core in the next transistor- core element. These 22 elements form a shift register. All the cores in the matrix start at ZERO. Selecting a sequence means setting a row of cores to ONE's and placing a ONE in the first position of the shift register (Fig. 2-2). Hence only one core of each column may be switched to ZERO by the corresponding shift register element*. As the shifting pulses α and β (Fig. 2-2 and 2-3) shift the single ONE in the shift register through all positions from (1) to (22), all the matrix cores set to ONE (forming a row) are cleared to ZERO one by one. Each control pulse has a wire associated with it, which threads (or "not threads") the cores of each row. Thus when a row of cores switches in sequence, the wire has a pattern of outputs and "no outputs" impressed on it.

For example, consider control pulse A (Fig. 2-3) if the selected row is (2). The only output (i. e. , the only control pulse) will occur at shift time 7, i. e. , when the ONE travelling down the shift register is being transferred from position (7) to position (8). If the sequence selected had been sequence (= row) (3), then there would have been outputs at time 1 and 2. Notice further

*All cores of that column are driven to ZERO, but only one switches, since only one of the cores of that column was switched to ONE in the first place.

that sequence (1) involves neither pulses A nor B, while sequence (63) involves both.

There are several variants of the matrix just described which enhance the flexibility of the sequence generator and permit certain economies . The particular use of these variants depends directly on the properties of the sequences, which will be discussed later: and hence the variants are presented here as "tricks" which. can be used if certain situations arise.

The most obvious variant is illustrated in Fig. 2-2 and consists in modifying the meaning of "row". Selection of a sequence means setting to a ONE some cores of the matrix, where each column will have at most one core set to ONE. Many sequences may have common parts; for example, they must end with an identical set of control pulses, for calling forth the next instruction. Then She same core in the matrix may be set by the various sequences, as in the case of the core of row (1), column (22), Fig, 2-2. In fact, that core is the only core present in column (22). Also, it is possible that a sequence may require a waiting period of one or more pulses, in which case the sequence selecting wire which sets a "row" to all ONE's will simply not set any core of the pertinent columns to a ONE. For example, sequence (15) (row 15) does not set any core of column (21).

A further "trick" consists of the use of the sequence selecting mechanism for specifying the starting point in the shift register. In Fig. 2-2, sequences (3) and (15) start at shift position (= time) 1, while sequence (63) starts at time 7. This scheme is useful in avoiding unnecessary waits and delays. Thus, the sequence CA X (clear A and add the contents of register X into A) and the sequence CA* X, in which X is now interpreted as the location of the address of the data, are such that the set of pulses for CA X is a subset of those for CA* X. CA X starts at time 7; CA* X starts at time 1 and is identical to CA X from time 7 on, The wire which selects sequence CA X drives core 7 on the shift

register to ONE, thus avoiding a wait of 6 pulse times.

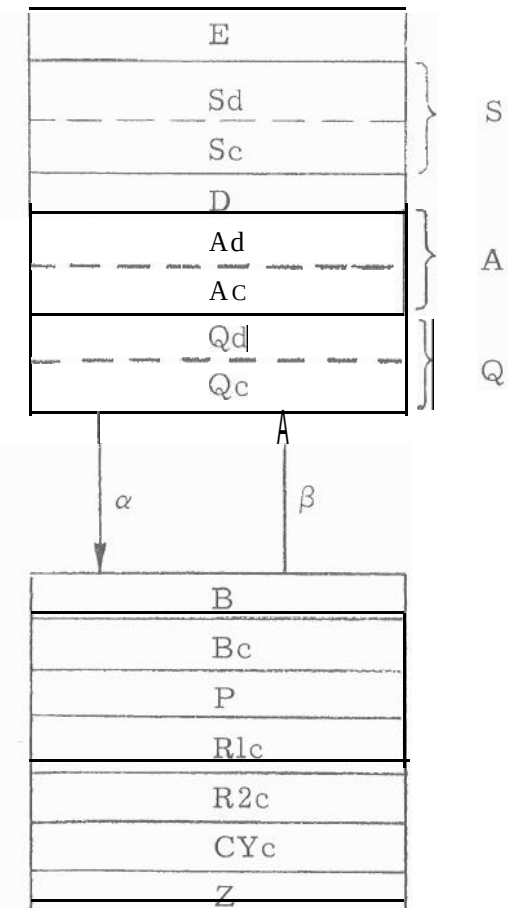
A third and final variant of the basic scheme consists of providing for branches, This is done by using a sequence to test the state of a core, e. g. , the core in which the sign of the quantity presently in A is stored. This result is then used to select one of two sequences, i. e., to set to ONE's the cores of one of two "rows". This is discussed further in the next section.

C. Representative Order Code System

For purpose of illustrating the variety of computer operations which can be accomplished by rather simple wiring and logic, a representative system of central registers and instructions, is described in this-section. The system is not completely described here; details are not sufficiently settled to make this possible. As show below, however, some relatively sophisticated operations can easily be achieved.

The central register system shown in Fig. 2-4 contains registers S (Sd, Sc), B and Bc, P, and **Z** which have already been discussed. In addition are shown ordinary register D, registers A and Q, each of which consist of a direct and complement part, complementing registers Rlc and R2c, together with register CYc which simultaneously complements a word and cycles it left one digit with end-around carry. Four control pulses affect register A: WAd, and WAd for writing respectively in the direct and complement portions, CL A, which clears both halves simultaneously, and WlAd to place ONE's in Ac. Register Q possesses corresponding properties. Registers Rlc, R2c, and CYc, possess three control pulses each for writing, clearing, and placing ONE's in the cores before writing.

To illustrate an entire sequence of steps comprising one relatively simple order, consider the instruction "Clear and Add X", CA X, as described previously. Control pulses for CA X are given in Table 2-l. Each instruction must bear the responsibility for calling forth fro-m storage the next instruction in



Register	Digits
S	12-1
A, Q	23-1
B, Rlc, R2c, CYc, Z	23-0

Fig. 2-4 Representative Central Register System

TABLE 2-l

CA X, Clear and Add X, C(X) into A, 8 pulses

1 α CL S, CL A, CL D
2 β (no action)
3 α CL E, WB, WP, WlSd
4 β CL Z, WS, WD
5 α CL S, CL D, WZ, WlR2d
6 β CL B, CL P, TP, WAd, WE, WlBd
7 α CL E, WB, WP, WR2c, WlSd
8 β CL B, WE, WS, WD, CL Bc, CL P, TP

sequence, after its own action is complete, so that the control pulses form a never ending sequence. A convention which states at what point of the process the "new" order takes over the "old" must be adopted here. The operation CL Bc, shown in the last line of Table 2-1, sets the appropriate line of cores in the sequence generator core matrix for the new instruction; it is therefore reasonable to define the α - pulse time which immediately follows as the first pulse of the next order.

To follow the actions described by the control pulses in Table 2-1, it is necessary to know that certain of the central registers are assumed to have been left with particular contents by the previous order. Line 8 in the table is the last line executed by every instruction; at the time of its execution, the next order is in B. By the action of line 8, the order is written back into E (if it came from E), into D, and the address portion is written in S. Thus at the time of initiating CA X, in line 1 of the table, the last line of the previous instruction has placed the address X in S and the entire order in D. The effect of line 1 for CA X is to clear D, to clear the accumulator A in preparation for later receiving the contents of X, and to initiate read-out of address X through setting the corresponding rope core.

Line 3 of Table 2-1 transfers the contents of X into the buffer, B, and writes them also into the parity register P. The Z register at this point contains the address of the next instruction, N + 1; line 4 transfers this address to S as the initial step in the action of calling the next instruction forth. The number n + 1 is also saved in register D, from where it is cleared in line 5 back into Z. The result leaves N + 2 in Z as the address of the instruction after next. Neither of the actions CL S or WlR2d interfere with this transfer since S has no sense wires and since the sense wires for R2d are in use only at β time. The control pulses WlR2d in line 5 and WR2d in line 7 serve the purpose of saving in R2d the complement of each instruction

executed; this feature is required only in a very special situation to be discussed later. It is noted that WlR2c prepares R2d to receive a new word just as effectively as clearing an ordinary register. All cores are left at ONE's, regardless of their previous status.

In line 6 the pair of pulses CL B, CL P places on sense lines 23-O the full word, including its parity bit, that was read from storage. The parity bit comes from digit 0 of B in this instance; the contents of the single bit called register P are properly a ZERO at this time since an entire 24 digit word from storage, with presumably correct parity, was read into P in line 3l Thus the pulse on digit 0 sense line is the logical sum of a ZERO and digit 0 of B. The control pulse labelled TP (Test Parity) activates a test to verify that P does indeed contain a ZERO as it is cleared An alarm, as yet specified, or an error diagnosis procedure is to be initiated if this is not the case.

Line 6 writes the original word from storage back into E if required, and writes digits 23-l of the word into part Ad of the accumulator A. Thus the contents of register X have been placed in A, and X is left unaltered. The next instruction is already in the process of being called forth by the pulse CL **S** of line 5 which sets the appropriate rope core. The occurrence of the pulse WE in line 6 after the pulse CL S in line 5 may offer some confusion, until it is recalled that the write core for an erasable register is set only when the rope core is reset. Thus the rope core for the next order, i. e., core N + 1, may be set prior to the time at which the erasable storage write core (if any) associated with register X is reset.

The pulse WlBc places ONE's in Bc to enable the complement of the order code to be obtained through the WB pulse in line 7. The fact that WlBc occurs simultaneously with CL B would at first glance appear to violate basic principles. Register

Bc contains no sense windings, however: except those connected to the logic decoding rope. Further, CL B leaves Bc unaffected; only WB is common to B and Bc.

The principal action of line 7 in Table 2-l is to call forth the next instruction from storage, writing it directly in the buffer B (and in Bc, as well), in the parity register, and in the form of its complement into R2c. Line 8 has already been discussed briefly above; the new order is written back into E if relevant, into D, its address portion into S, its parity is checked, and the pulse CL Bc transforms its order code into appropriate action in the sequence generator.

Considerable detail has been brought into the analysis of the pulse by pulse action of a single elementary order, CA X, because in so doing a considerable portion of the logic of most of the rest of the orders is also presented. In the following paragraphs and tables other typical orders are presented, generally with no more than a few sentences of description attached to each. These descriptions are made somewhat more concise by use of a few simple abbreviations and conventions. The order being executed is understood to have been found in register N, and the next order is located in register N + 1 unless otherwise noted. The address portion of the order is denoted by X, as in CA X above. The contents of any register are denoted by the symbol C( ) with the symbol for the register inserted in the parentheses; e. g., the contents of A are written C(A). The system of arithmetic used here is that in which the negative of a word is simply its ONE's complement, denoted by following that word with a prime: - C(X) is the same as C(X)'. Finally, a register X, A, or Q is understood to be left unchanged by the action of an order, unless a change is indicated specifically.

The order TS X, to store the contents of A, has been mentioned previously. As seen from Table 2-2, it differs from CA X only in that the contents of A are transferred to B and P

in line 1, instead of being discarded, and the contents of X are discarded in line 3 instead of being transferred to B and P. Note that CL P in line 6 is here not accompanied by the parity test pulse TP as it was before. Instead, the correct parity bit for the word read into P in line 1, on digits 23-1, is supplied as the output of P on digit 0. The physical action of the parity circuit is the same in the two cases, but its function is different. Digit 0 of B is ZERO in line 6, since only positions 23-l were filled from A in line 1.

TABLE 2-2

TS X., Transfer to Storage X, C(A) into X, 8 Pulses

1 α CL S, CL A, WB, WP
2 β (no action)
3 α CL E, CL D, WlSc
4 β CL **Z,** WS, WD
5 α CL S, CL D, WZ, WlR2c
6 β CL B, CL P, WAd, WE, WlBd
7 a CL E, WB, WP, WR2c, WlSd
8 β CL B, WE, WS, WD, CL Bc, CL P, TP

The instruction CS X, to clear A and place the negative of C(X) in A, is seen in Table 2-3 to differ from CA X only in the pulse WlAc in line 2 and in the replacement of WAd by WAd in line 6.

TABLE 2-3

CS X, Clear and Subtract X, C(X) into A, 8 Pulses

1 α CL S, CLA, CLD
2 β WlAc
3 α CL E, WB, WP, WlSd
4 β CL Z, WS, WD
5 α CL S, CL D, WZ, WlR2d
6 β CL B, CL P, TP, WAc, WE, WlBc
7 α CL E, WB, WP, WR2c, WlSc
8 β CL B, WE, WS, WD, CL Bc, CL P, TP

If, the first line of control pulses for CA X, the pulse CL A is omitted, the result is an order LA X as described in Table 2-4, called "Logical Add X". The effect of this order is to place in A the word which is the union or logical sum of the previous C(A) with C(X); i. e. , the word whose digits are ONE in any digit position where either or both of C(A) and C(X) contain a ONE. The application of an order of this type occurs principally in such logical processes as masking, in which only selected digits of a word are to be used in an operation. A more suitable order for this purpose is that for forming the logical product of C(X) with C(A); however, the operation LA X is so simply achieved that it is probably preferable unless a very rapid masking order is needed.

It is noted that LA X writes in the cores of Ad. If the order just preceding LA X were a clear and subtract operation, the cores of Ad would all contain ZERO's and the cores of Ac would contain the real "contents of A". It must be recognized that the pulse CL A places on the α sense lines a ONE wherever either Ad or Ac contains a ONE; i. e. , the logical sum of C(Ad) and C(Ac). Thus the effect would be the same as if it were Ad rather than Ac that contained the non-zero information. For all logical purposes, A may be regarded as a single row of cores in which a logical sum is formed as the writing takes place.

TABLE 2-4

LA X, Logical Add X, C(X) U C(A) into A, 8 Pulses

1 α CLS, CL D
2 to 8: The same as in Table 2-l

An operation having frequent convenience as a storage-saving device in arithmetic, but which in addition has an unique logical function in the present computer, is that of exchanging

C(X) with C(A). The logical function has to do with counter registers that receive input pulses from an external source. All counting (which uses the Z register) occurs at a definite place in the sequence generator shift register at which the Z register is vacant; specifically, this is between lines 4 and 5 in the previous tables. If the contents of an externally pulsed counter register are to be examined without risk of missing a pulse in the count, it is necessary to be able to pick up the counter contents and to reset the counter to a known value simultaneously in a single order. Even the order CA X, when applied to a counter register, is not a safe way of examining the register as the order is now written, since the pulses CL E and WE are separated by the time at which counters are incremented. The order EX X, Exchange with X, described in Table 2-5 does achieve the desired result, permitting the contents of the counter to be placed in A and those of A in the counter during a time period in which no incrementing can occur.

TABLE 2-5

EX X, Exchange with X, C(X) exchanged with C(A), 12 Pulses

1	α	CL S, CL D
2	β	(no action)
3	α	CL E, WB, WlSd
4	β	CL B, WD
5	α	CL A, WB, WP
6	β	CL B, CL P, WE
7	α	CLD, WB
8	β	CL Z, WS, WD
9	α	CL S, CL D, WZ, WlR2c
10	β	CL B, WAd, WlBd
11	α	CL E, WB, WP, WR2c, WlSc
12	β	CL B, WE, WS, WD, CL Bc, CL P, TP

No parity check is made on the original contents of X in the exchange order; for counters, in the present mode of operation, a parity check is not kept. An alternate counter operation method in which the **Z** register would be on the a side of storage would permit parity to be preserved.

The exchange order is the first one that has been encountered which involves other than eight pulse times for its execution. Its beginning is identical to that for LA X and its ending is identical to all orders considered. In the interest of economy of cores and windings in the sequence generator core matrix, the shorter orders are therefore stretched out to match the twelve columns of cores required for EX X. This operation is not done in a way so as to cause the other orders to waste four pulse times of inactivity, but requires the shorter orders to skip over several positions of the shift register. The result permits each instruction to proceed with no waste of time, and yet permits many cases of sharing of matrix cores between different orders. The numbering of the lines in the tables shown here is for convenience only, and does not correspond to core matrix columns.

The instruction "Edit by X" is perhaps unique to this particular computer, and is motivated by the fact that many of the operations such as shifting are handled by special write wiring in registers in E storage. The function of this order, in brief, is to accomplish in one order of twelve pulse times what would otherwise require the two orders TS X, CA X and sixteen pulse times. Its operation is described in the table which follows.

TABLE 2-6

ED X, Edit by X, C(A) into X, then C(X) into A, 12 Pulses

1 a CL S, CL D, WB
2 β (no action)
3 α CL E, WlSd
4 β CL B, WS
5 α CL S, CL A, WB, WP
6 β CL B, CL P, WE
7 α CL E, WB, WP, WlSd
8 β CL Z, WS, WD
9 α CL S, CL D, WZ, WlR2d
10 β CL B, CL P, WAd, WE, WlBc
11 α CL E, WB, WP, WR2c, WlSd
12 β CL B, WE, WS, WD, CL Bc, CL P, TP

Orders dealt with up to this point have been chiefly concerned with moving information around from one point to another. Consider now the order TC X, transfer control to X, which causes the next instruction executed to be that in register X. In the process, the next consecutive address (i. e. , the address one greater than that of the location of TC X) is left in central register Q from which it may be stored by TA X, below. This pair of orders provides convenient entry and exit procedures for closed subroutines.

TABLE 2-7

TC X, Transfer Control to X, N + 1 into Q.

Next Operation from X, 6 Pulses

1 α CLQ
2 β CL Z, WQd
3 α CL S, CL D, WZ, WlR2d
4 β WlBc
5 α CL E, WB, WP, WR2c, WlSd
6 β CL B, WE, WS, WD, CL Bc, CL P, TP

It is noted that the order itself goes into the Z register in line 3, rather than merely the address, and a similar statement is true for certain branch orders considered later. This can, in turn, result in order codes being attached to the addresses which enter Q and thence are stored by the TA order. The effect of this, if recognized in programming, can cause little harm.

The transfer address order, TA X, serves an additional purpose beyond that of storing the contents of Q. This other purpose is the transfer to Q of the contents of R2c, thus two TA orders following each other will store successively the original contents of Q and of R2c. Otherwise, TA X closely resembles TS X in operation.

TABLE 2-8

TA X, Transfer Address to X, C(Q) into X, then C(R2c) into Q, 8 Pulses

1 α CL S, CL Q, WB, WP
2 β CL R2c, WQ
3 α CL E, CL D, WlSc
4 β CL **Z,** WS, WD
5 α CL S, CL D, WZ, WlR2d
6 β CL B, CL P, WE, WlBd
7 α CL E, WB, WP, WR2c, WlSd
8 β CL B, WE, WS, WD, CL Bc, CL P, TP

We now come to consider a group of three branching orders, that cause a transfer of control to occur or not, depending upon the outcome of a test. Representative of such orders is BNZ X, branch non-zero to X, which tests central register A and acts as a TC X order if A contains any ONE 's. Electrical details of the branching action have been considered only in a preliminary way; however, the principles are relatively simple. The original setting of a row of cores in the sequence

generator core matrix for BNZ X skips certain columns (i. e. , shift register positions). These are filled in subsequently in one of two alternate ways selected by the result of the testing operation. Thus lines 3 to 7 in Table 2-9 are filled with two choices which depend upon whether C(A) are zero or not. Control pulses describing details of this action are not shown; however, the testing is done as C(A) are written into B in line 1, and the actual action of setting one of two rows of cores is indicated by the word "BRANCH" in line 2.

TABLE 2-9

BNZ X, Branch Non-Zero to X, Acts as TC X if C(A) are Non-Zero, Otherwise Order is Ignored, 10 Pulses

1 α CL S, CL A, WB
2 β CL B, WA, BRANCH

Active branch (non- zero)	Inactive branch (zero)
3 α (no action)	3 α CL E, WB, WlSd
4 β (no action)	4 β CL B, WE
5 α CLQ	5 α CL D
6 β CL Z, WQ	6 β CL **Z,** WS, WD
7 α CL D, WZ, WlR2d	7 α CL S, CL D, WZ, WlR2d
8 β WlBc	
9 α CL E, WB, WP, WR2c, WlSd	
10 β CL B, WE, WS, WD, CL Bc, CL P, TP	

A branch order closely related to BNZ X is the order BMN X, branch minus to X, which tests the sign (digit 23) of A and acts as TC X if this bit contains a ONE denoting a negative number. The formal table of control pulses for BMN X is identical to that of Table 2-9 for BNZ X. The sole difference lies in one control pulse, not shown in the table, which deter-

mines the nature of the test. For BNZ X, all 23 bits of C(A) are tested, whereas for BMN X only the single high-order bit is involved.

A further branch order, similar to BNZ X except that the control pulses CL A, WB, CL B, WA are omitted in lines 1 and 2, is that for testing for overflow of a program counter. Certain erasable storage registers may be set aside as counters by provision of extra circuits described in Section E of Chapter 3. Those used by the program for its own bookkeeping purposes differ from those for external pulse inputs in the source of the input to be counted and in the disposition of any overflow pulses. Extra digit positions in the instruction word are so wired that a ONE in that digit causes an associated program counter to increment by one after the order is executed. For this particular class of counters, any overflow pulses are used to set an overflow core which may be tested at any later time. The "branch no overflow" instruction, BNO X, is the means for this test. This instruction acts as TC X provided than no. overflow pulse has occurred, and is otherwise ignored (except that it clears the overflow core).

One final branching order of frequent convenience is that designated as TRSP X, "test register X and skip (next order) if plus". If C(X) are negative, the next order is taken from N + 1; if C(X) are positive, the next order is taken from N + 2. Further, the absolute value of C(X) is left in Q from which it may be stored by TA.

TABLE 2-10

TRSP X, Test Register X and Skip on Plus, 10 Pulses

If C(X) plus, next order from N + 2 and C(X) into Q; If C(X) negative, next order from N + 1 and C(X) into Q

1	CL S, CL Q, WlRld	
2	CL Z, WD	
3	CL E, WB, WP, WRlc, WlSd	
4	CL B, CL P, TP, WE, BRANCH	
	Active branch (negative)	Inactive branch (positive)
5	CLD, WB	5 CL D, WZ
6	CL B, WS, WD	6 CL Z, WS, WD, WlQd
7	CL S, CL D, WZ, WlR2c	7 CL S, CL D, WZ, WlR2c
8	CL Rlc, WQd, WlBc	8 CL Rlc, WQc, WlBd
9	CL E, WB, WP, WR2c, WlSd	
10	CL B, WE, WS, WD, CL Bc, CL P, TP	

The order CANP X stands for "clear and add, with no parity (check)". Its purpose is to provide a way to operate with registers for which the parity bit may be incorrect or absent; e. g. , the register for shifting right one digit, described in Chap. 1, Section C, or an input register. The pulse sequence for CANP X is essentially the same as that of CA X in Table 2-1 except that the pulse WP is missing from line 3 and the pulses CL P, TP are absent from line 6.

The automatic interrupt, discussed in Chap. 3, Section C, is an interruption to the normal sequence of computer operations caused by various input circuits which require attention. It occurs, when required, just before execution of the last line in an instruction so that the next order has been called from storage but not yet transferred to the sequence generator. Instead of the next order, a special "interrupt" sequence of control

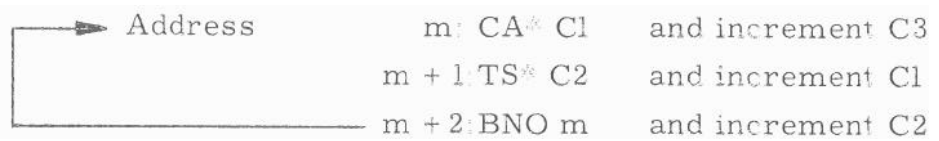
pulses is generated which, in addition to several specialized logical actions, stores away the contents of selected central registers required to permit the normal instruction sequence to resume later without any apparent discontinuity. These contents are stored in other central registers, not shown in Fig. 2-1; probably two such registers can be made to suffice.

The interrupt sequence is not itself a programmable order, however there is at present in the order code system an order RSM X, "resume" (X is irrelevant), which provides a control pulse sequence for resuming operation after an interrupt. The effect of RSM X is, first, to test whether another interrupt request has occurred while the actions called for by the original interrupt were in progress (no more than one interrupt is processed at a time). If not, the contents of the central registers are restored and the operation of the main program is resumed.

As any programmer knows, one of the most important computer capabilities required for compact and elegant programming is that of address modification; i. e. , the capability of writing an instruction whose effective address at the time of execution depends upon other actions of the program itself. In this computer the problem is solved in terms of quite simple circuits by providing the capability for so-called "indirect addressing" for every order. The effect of indirect addressing on an instruction is to replace the address mentioned in the instruction by the address contained in that location. For example, let an asterisk following an order code indicate that indirect addressing is to be applied, and let the contents of register X be the number Y. Then CA* X is essentially synonymous with CA Y. If X is an erasable register (otherwise the operation is pointless) then its contents Y may be given different values by operation of the program, and the single wired-in instruction CA* X may thereby have a corresponding variety

of effective addresses

When combined with the use of program counters and the previously mentioned capability of incrementing these automatically under control of selected digits in the order code- the indirect addressing capability permits some quite compact. programs For example, let. it be desired to move N consecutive words in storage locations K to K + N - I to the new locations L to L + N - 1. To do this we place the address K in a counter C1 address L in a. counter C2 and place hhe number -(N - 1) in a counter C3. Following this preliminary operation the actual execution is accomplished with a loop of three orders.



The electrical details of the indirect address feature are rather simple. Of the six bits specified for the order code, four are used for the fourteen orders thus far discussed. One more is reserved for the "extended operations" to be discussed shortly, The remaining one is for indirect addressed orders. I's effect- in brief, is to precede the control. pulse sequence normal to t.he order in question by another of four pulses in length which replaces the contents of D and S by the word. obtained from storage in the address specified by the order, The control pulse sequence for this purpose 6s given in Table 2-11 It is necessary in addition to cause the control sequence shift register to start at a position four steps preceding the beginning point for normal orders. This is accomplished through a relatively simple wiring in the sequence generator core rope, The whole cost of providing this facility is perhaps seven or eight transistors and a few I-ores. etd

TABLE 2-11

Indirect Addressing

1 CL S, CL D

2 (no action)

3 CL E, WB, WlSd

4 CL B, WE, WS, WD

Normal operation of order follows

The so- called "extended operations" form an inexpensive storage-saving device by providing means for a one-word entry into a number of subroutines. As an example, multiplication exists in most variations of the computer only as a subroutine. To multiply C(X) by C(Y), the following orders might be written:

CA X
TS to standard location
CA Y
TC to multiply routine

The extended operation version of this would replace the last three orders by one order. The problem is that four items of information are required: (1) address X or its contents, (2) address Y or its contents, (3) the address to go to after multiplication is complete, and (4) the fact that multiplication is desired; i. e., an address associated with this particular subroutine. A single order does not have sufficient digit positions to carry all of these data; two, however, can do so by relying on the Z and Q registers to handle the next address, provided that only a few bits are required for item 4.

The action, in brief, is that one bit in the order code causes a special control pulse sequence to become active, which holds C(Z) in Q as in the TC order, preserves the complement of the original order in R2c, and transfers control to a specific address (e. g., address 128) which is generated electrically. There, a short standard program is initiated which files these

data away, including the original order, and finally transfers control to an address derived from the order code portion of the original instruction. The whole operation requires approximately 50 pulse times whereas the three-order version cited previously requires about half of that. Thirty-two of these extended operations are available. By and large they will be used for a variety of instructions found in most computers which are accomplished in ours by means of subroutines, There is some appeal, though, in planning a computer for geometric computations required, say, in space navigation, which may have "vector add" as a computer instruction.

D. Addition and Multiplication

The selection of an order code is dependent on the particular use to which the computer is put. The choices are, as usual, between speed and sophistication on the one hand and simplicity and reduced equipment on the other. Of particular interest in the present case is the method for addition, in which no use is made of an adder. Instead, in the representative computer, addition is carried out by a repetitive logical process within a loop in the sequence generator, in which all operations are conducted by the processes and registers already discussed.

The action is as follows. Let $u \oplus v$ stand for the bit by bit logical sum of two words u and v; i. e. , the word that contains a ONE in each digit position for which either u or v (or both) contains a ONE, To add two numbers x and y, the operations

$$a = x \oplus y$$
$$b = x' \oplus y$$
$$C = b'$$
$$S = (a' \oplus b')'$$

are performed. The word S is the sum and C the carry for a one-digit-at-a-time summation of the words x and y. If C is zero, the process is complete. Otherwise, C is cycled leftward

one bit, with end-around carry, and S and the cycled carry word are used to replace x and y in the above formulae and the addition process repeated. The operation continues repeatedly until the carry word is zero, at which time the corresponding sum S is indeed the algebraic sum of x and y. The number representation is that of ONE's complements, in which - x is the bit by bit complement of x.

The pulse by pulse operation of the loop portion of the control sequence is given in Table 2-12, in which the contents

TABLE 2-12

Addition Loop

	Ac	Ad	Qd	Qc	CYc	R1c	R2c
	x	y	x'	y'	0	1	1
1 α CL A, WR2c, W1CYc, WB, BRANCH							
	0	0	x'	y'	1	1	a'
2 β W1Ac, CL B							
	1	0	x'	y'	1	1	a'
3 α CL Q, WCYc, WR1c							
	1	0	0	0	C	b'	a'
4 β CL R1c, CL R2c, WAc, WQd, W1Qc							
	s	0	S'	1	C	0	0
5 α W1R1c, W1R2c							
	s	0	S'	1	C	1	1
6 β CL CYc, WAd, WQc							
	s	c	S'	C'	0	1	1

of the relevant central registers are shown at successive stages of the loop. Starting with the two numbers to be added, x and y, in Ac and Ad before line 1, and their respective complements in Qd and Qc, in six steps these numbers are replaced by S, C, and their complements, and everything is ready for the cycle to repeat. The BRANCH operation in line 1 corresponds to a test

set up at line 6 of the previous cycle, in which Ad is wired to test for a non- zero carry word as C is written into it. The contents of A are written into B in line 1, so that when the BRANCH operation causes the control to leave the loop, the sum S is found in B.

The time required for execution of an addition by this process is approximately 14 + 6 n pulse times, where n is the number of times S and C must be formed. A digital simulation of this addition process, using computer-generated "random numbers" has produced the plot of average n versus word length shown in Fig. 2-5. For each word length from 3 to 36, at intervals of 3, 4000 samples were taken to produce these data. Results show that for a word length of 23 digits, corresponding to the representative computer (neglecting the parity bit, of course), the average n is very close to five cycles. Thus approximately 44 pulse times, or 264 microseconds at 6 microseconds per pulse, are required on the average. The corresponding maximum time requires 24 cycles, giving 158 pulse times or 948 microseconds. Some care should be taken in making estimates based on the "average" value. The addition of a small negative number and a slightly larger positiue number requires essentially the maximum time, since the carry must propagate all the way along the word one digit at a time.

We note in passing that the addition capability has been obtained quite inexpensively, given that the central registers shown in Fig. 2-4 already exist. Essentially, the only extra circuits required are those required for the branching operations, which involve perhaps eight or ten transistors altogether.

Roughly speaking, there are three ways of doing additions. The fastest and most expensive way in terms of equipment is to add with a parallel adder. With this scheme, addition of y to some x stored in A would require a single sequence, with no loops inside the sequence generator, of about 16 pulse times.

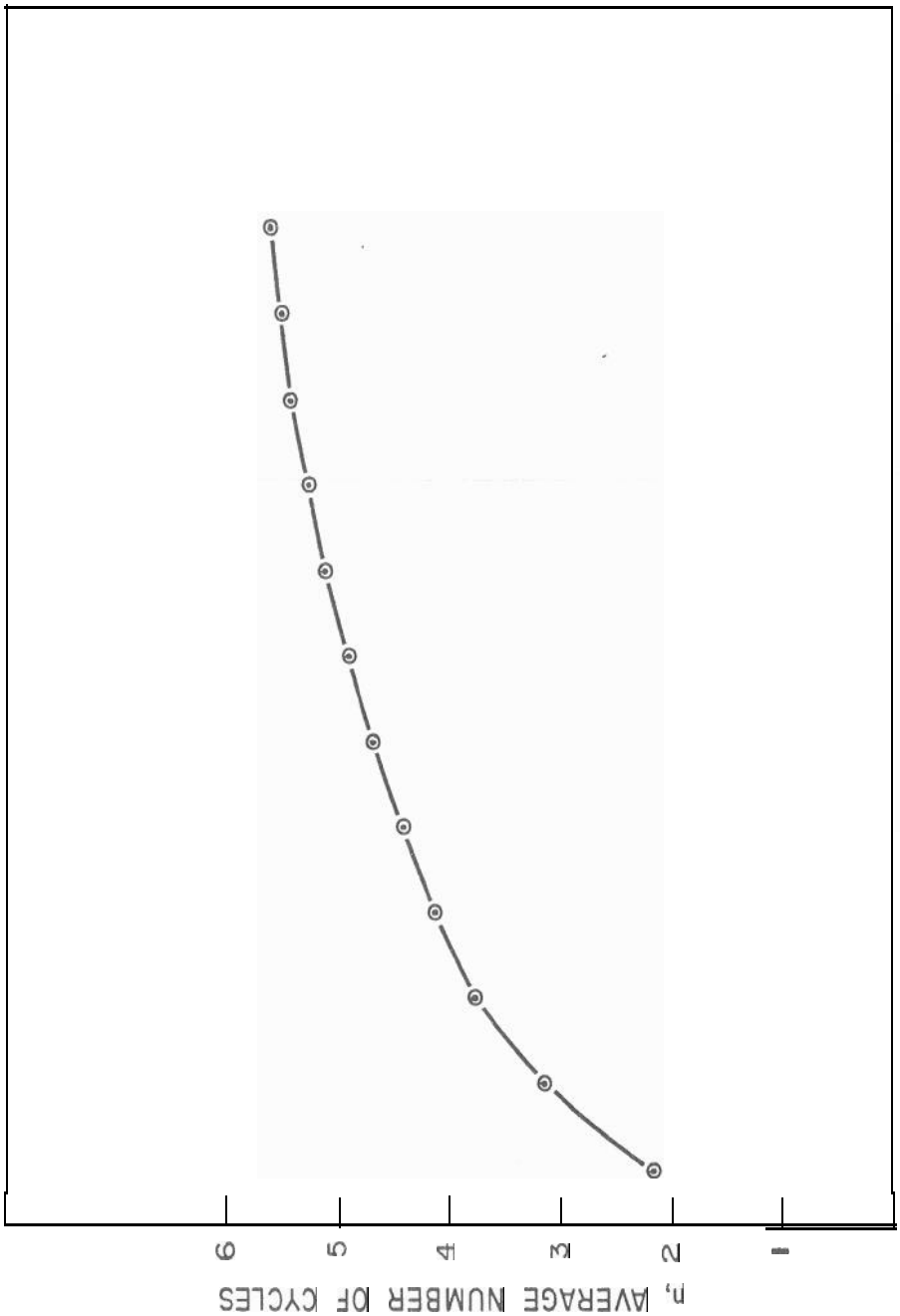


Fig. 2-5 Average number of cycles for addition.

The next fastest way to add was described in the above paragraphs, with a loop which is internal to the sequence generator. This method requires about 44 pulse times on the average, and 158 pulse times maximum. The slowest and most economical method is to perform addition with a program subroutine, using orders of which each is a zero level sequence with no loops inside the sequence generator, and with the orders stored in the core rope. This method is many times slower than the method described above, because in effect each pulse time of the microprogram sequence is replaced by an order which is itself some eight to twelve pulse times long. The saving in equipment consists mostly in the elimination of control pulse circuitry. There is little saving of registers because almost the same number of erasable storage registers must be reserved for addition by the program subroutine method as are used by the microprogram sequence. The difference is that in the subroutine method the registers are addressed by means of the S register, while in the faster microprogram method the registers are addressed directly by means of control pulses.

Multiplication in this computer is almost certain to wind up as an extended operation or subroutine. It should be remarked, however, that a "multiply loop" within the sequence generator is not out of the question, for an application in which the extra speed is necessary. No attempt has been made to draw up the logic for such a loop; however the following time estimates are probably reasonable for its operation. Use of such a loop surrounding a "one-shot" parallel adder circuit would perhaps require 200 pulse times for an average multiplication. If used in conjunction with an inner addition loop of the form discussed above, an average might be 500 pulses. These estimates may be compared to a value of from 1400 to 1500 pulse times for an average multiplication by subroutine methods, using the type of sequence generator addition loop discussed above. If addition, also, is a programmed operation, then of course the multiply order re-

quires much longer. It is thus seen that there are a variety of options open in the selection of a multiplication method. It is expected that most of the presently contemplated applications of this computer will be adequately served by one of the simpler procedures.

One rather unexpected consequence of the comparatively long time required for multiplication is the fact that, in those programs prepared to date, the operation time is easily estimated simply by counting the multiplications. Viewed in another light, the same proposition may be restated by saying that those operations of bookkeeping etc, , which are incident to the preparation of an elaborate subroutine structure turn out to be quite inexpensive as far as computing time is concerned. As an example, the extended operation for vector addition requires about 400 pulse times; that for the dot product of two vectors requires about 5000. The amount of bookkeeping is identical in the two cases. The difference lies solely in the 4500 pulse times required for three multiplications. As a result, as long as this unbalance in operation time remains in this computer, it is quite likely that the compactness provided by use of numerous subroutine operations will more than compensate for the slight extra time this usually involves.

CHAPTER 3

INPUT AND OUTPUT SYSTEMS

A. Introduction

It is very difficult to discuss an input-output system without a clear definition of what the inputs and outputs are to be, and how many, of them there are, and how fast they operate. As was mentioned in the introduction, it is difficult to describe a special purpose computer when the purpose, though special, is vague. In the present case, input-output systems will be discussed as a collection of techniques which the authors believe will be useful, in general, in the control of airborne and space-borne systems.

As a background picture, it might be kept in mind that the origins of this computer were in a program studying guidance and control for an interplanetary probe. Representative problems for that application might include control of the vehicle orientation, through flywheels, in such a fashion as to align a body-fixed telescope tracker with the Sun or a star; or the positioning of a sextant to a precisely measured angle. Other tasks would include monitoring of various discrete signals describing the current states of various items of equipment. In general, various pulse trains are to be counted, representing the quantization of analog signals, and other steady state signals are to be examined. The rates involved never exceed a few hundred cycles per second, but the number and diversity of the data to be processed simultaneously may render an elementary scanning system difficult to achieve. The methods presented in

this chapter represent our approach to this type of problem.

To simplify discussion, it has been assumed that, as far as the computer is concerned, inputs from the outside world consist of on-off switches, mechanical or solid state. An output is similarly defined as a switch closure. The intent is to control such things as the angular position of a motor by on-off servo techniques. This sort of thing can be done if the computer response is fast enough. What follows is a set of schemes which will make the computer fast enough, in some cases at least, by logical means rather than by fast circuitry.

B. The Priority Circuit

The priority circuit (Fig. 3-l) is a novel device which permits certain economies in the handling of inputs. The circuit itself is related both to counters and to shift registers. In Fig. 3-1, let all cores be initially in state ONE. When transistor TA is saturated, core (1) starts to switch, and in so doing it causes transistor T1 to switch. This in turn causes all current to flow through T1, with almost no current flowing through the windings of cores (2), (3) and (4). It is assumed that the saturation resistance of T1 is zero compared to R2. It is further assumed that TA is turned off at the same time that core (1) finishes switching.

The next time an advancing pulse is applied to the base of TA, core (1) will be at zero; cores (2), (3) and (4) at ONE. This time core (2) switches in exactly the same fashion as did core (1).

If all the cores of the chain start at ONE, then the chain behaves much as a shift register with a single ONE travelling down it, in the sense that the numbers of cores in the chain determine the number of pulse times necessary to make the last core switch. In the case of the priority circuit, however, it is possible to reduce the effective length of the chain by not starting with all ONE 's. For example, if core (3) had started at ZERO, then it would be the third advancing pulse instead of the fourth,

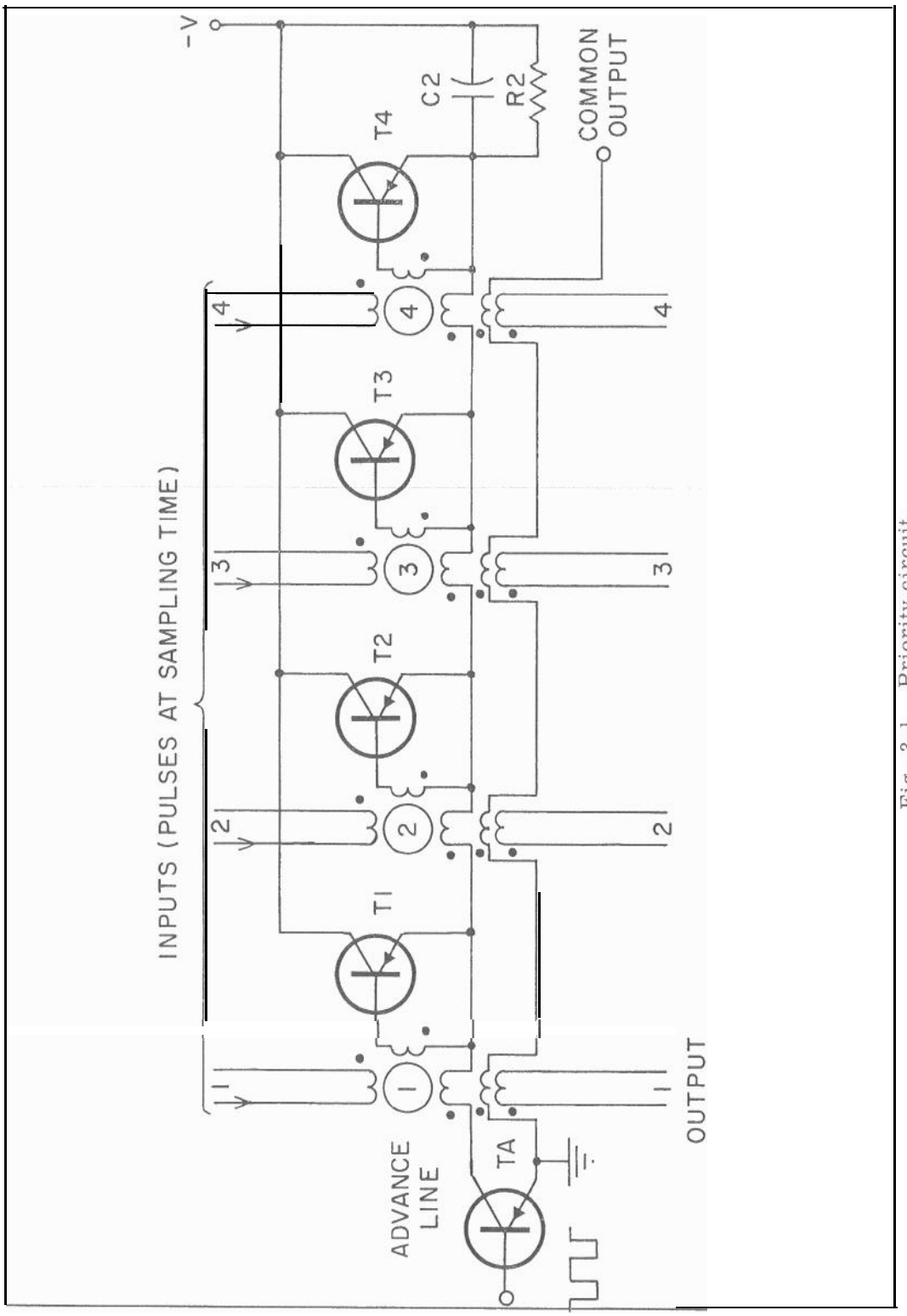


Fig. 3-1 Priority circuit.

which causes core (4) to switch,

Consider next a step of input leads such that a pulse on an input lead sets a core to a ONE (top of Fig. 3-1). Input pulses can only occur at a specific sampling time T_S. A winding common to all the cores in the priority circuit (labelled common output) can be used to sense whether or not one or more inputs occur during a sampling time T_S. If one or more inputs do occur, then the pulse which appears at the common output can be used, after a suitable delay, inversion, and reshaping, to trigger TA, Triggering TA results in driving from a ONE to a ZERO, the first core to be in state ONE, which in turn creates outputs at both an individual output winding, and at the common output. This last pulse, again delayed and reshaped, but this time not inverted, can be used to trigger TA once more. The process, schematized in Fig. 3-2, will continue until all the cores in the priority circuit are at ZERO, and will require as many advancing pulses as there were inputs at T_S (plus one>. The name “priority” arises from the fact that input line 1 is always served (i. e. , core (1) cleared to ZERO) first if it has an input, and input line 2 next, so that line 1 has priority over line 2, and so forth.

The sampling system of Fig. 3-2 is shown in more detail in Fig. 3-3. The inputs to the priority circuit are pulses generated by memory units M indicating that a switch has changed state since the last sampling time. This type of input, rather than a type which gives forth a pulse for every “on” switch every sampling time, permits an input system with some advantages over conventional input scanning systems. Furthermore, the memory units M of Fig. 3-3 can be so designed as to provide information on the actual state of the corresponding switch, and thus avoid the dangers of a pure “change of state” system, which is vulnerable to loss of pulses. A realization of unit M will be discussed later.

It is interesting to compare the priority circuit input

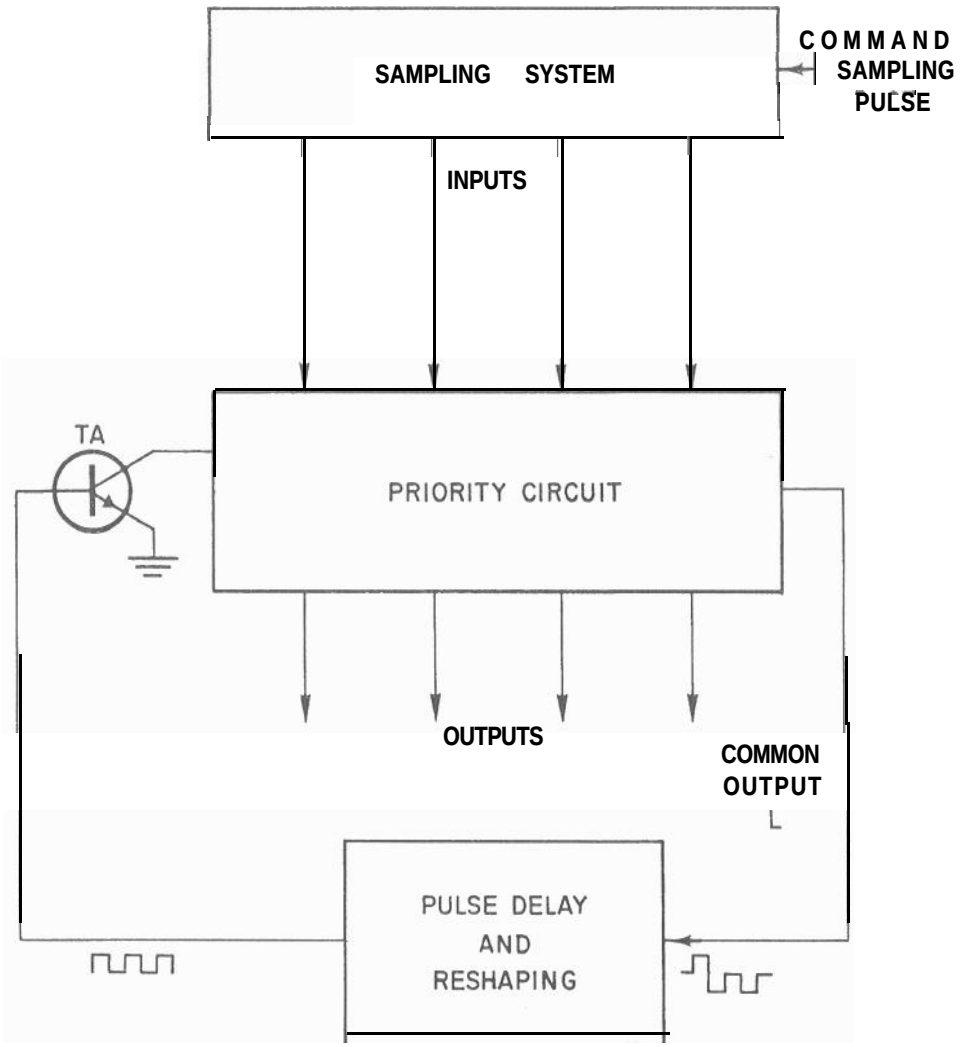


Fig. 3-2 Priority circuit system.

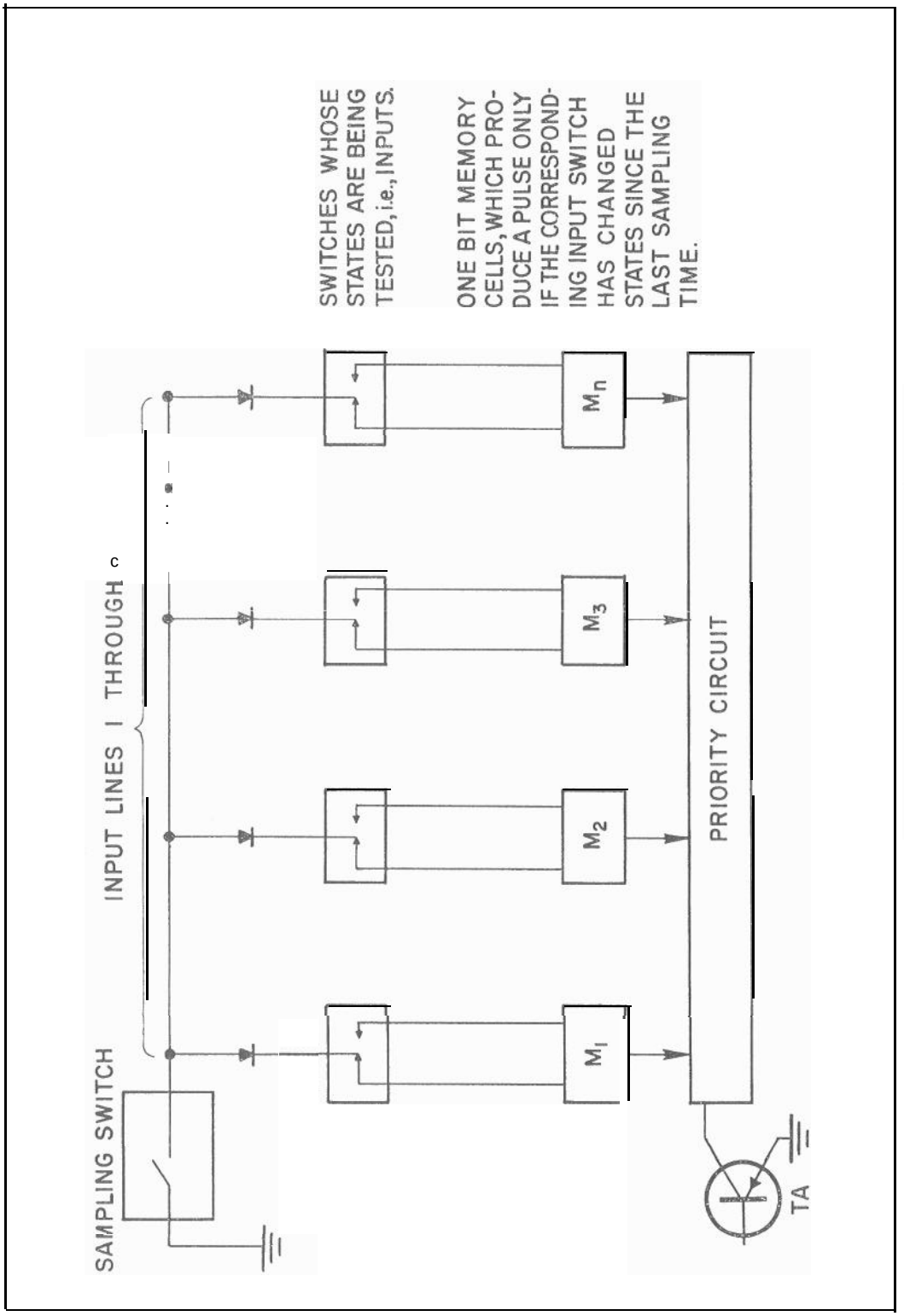


Fig. 3-3 Input circuits for priority circuit system.

system with one in which there is a shift register in place of the priority circuit. proper. For the shift register, if there are N input lines, N advancing pulses by transistor TA are required between sampling times. This second system in effect scans the N lines in between sampling times. If there are K sampling times per second, then the overall pulse handling capability of the second system is KN pulses per second. Assuming some technological limit to the pulses per second which may be handled by a given type of component, then the maximum sampling rate l/K is determined essentially by the number of input lines to be served, Furthermore, each line separately may handle on more than K pulses per second" with this arrangement.

With the priority circuit in place on the shift register, it is no longer necessary that there be at least N advancing pulses on transistor TA in between every sampling time. The number of such pulses is determined by the overall activity of the input lines rather than number of such lines. Hence, advantage may be taken of any properties of the input lines, such as, for instance, that they be many in number but that their average activity be low.

More important perhaps is the ability to increase the sampling rate to keep pace with the fastest of the input channels', without necessarily requiring the advancing pulses on transistor TA to speed up proportionately. Suppose for the sake of definiteness that the processing of one pulse requires 200 microseconds and that, of 30 input channels, 6 may have up to 400 pulses per second whereas the rest may have no more than 10 pulses per second each. A maximum of 2640 pulses per second is ;therefore involved. Now if a scanning system requires 200 microseconds

* 'In speaking of the pulse carrying capacity of a channel in terms of pulses per second, we, of course, are really referring to the reciprocal of this number which is the minimum time interval between consecutive pulses.

to look at each channel, whether a pulse is present there or not, it cannot cope with the situation as described. To do so would require a processing rate of 30 x 400 = 12,000 pulses per second. Of course two separate scanning systems could be employed, one to scan the six high frequency channels and the other to handle the rest.

In a sense, the priority circuit input system acts in this way, except that it adapts itself automatically to varying input requirements. All that is required is to make the sampling rate fast enough to match the fastest of the input channels and, simultaneously, to be able to process pulses fast enough to handle the total load. Thus, in the example cited, a sampling rate of 500 per second would permit ten advance pulses on the priority circuit during each sample period (at 200 microseconds each), which would permit the six 400 cycle lines to be processed once each sample time and still leave time for handling four other pulses distributed among the remaining 24 lines. It could, of course, happen that more than four of the remaining 24 inputs would require processing in a particular sample period. However, the priority circuit would carry this information over until the next sample period, and could be guaranteed to process each channel before the next pulse on that channel arrived. From the standpoint of the tie-in to the computer, discussed in the next section, the priority circuit provides a means whereby the computer gives exactly the minimum possible amount of attention to its inputs, thus permitting highly efficient time sharing procedures.

A careful study of the properties of the priority circuit input system is beyond the scope of this report, but it seems clear that such a system can take advantage of variations in information rate among the input lines, and that it is in no case worse than the scanning system.

The memory unit M mentioned earlier may be realized

by the circuit shown in Fig. 3-4. Upon closure of the sampling switch, cores Al and A2 are driven to ONE for one position of the input switch, and to ZERO for the other. Cores Bl and B2 are driven to ZERO and ONE, correspondingly. Cores A2 and B2 are always in opposite states, and will change states only if the input switch has changed states since the last closure of the sampling switch. A full wave rectifier generates a negative pulse every time A2 and B2 change states, and this pulse constitutes the input to the priority circuit.

It may be necessary to ascertain the state itself, rather than a change of states of the input switch, and cores Al and Bl can be used for this purpose. These cores, which may be part of an erasable storage register, may be both cleared to state ZERO simultaneously, and their outputs sensed. Note that only one of Al or Bl may be at ONE. The next closure of the sampling switch restores the proper core to a ONE, in correspondence with the state of the input switch. This subject is developed further in the next section.

C. The Automatic Interrupt

The automatic interrupt system provides integration of the priority circuit input system with the rest of the computer. The desired overall behavior of the input system is approximately as follows : when an input occurs a signal is given to the sequence generator and the normal chain of events is interrupted at the end of the current sequence. The next sequence (i. e. , order) is not the order determined by the contents of register B, but a special sequence called the Automatic Interrupt Sequence (AI). This sequence transfers the contents of selected central registers for safekeeping into other central registers specifically reserved for this purpose . Then, by mechanisms to be described later, program control is transferred to a subroutine appropriate to the particular input which activated the interrupt. When such action is ended, a Resume sequence restores the contents of the central

registers and program control is transferred back to the instruction displaced by the AI sequence. In short, the occurrence of any input results in the automatic interruption of whatever the computer is doing, preservation of the contents of certain central registers, processing of the input, and then resumption of the previous computer activity. There is an exception, namely that it is not possible to interrupt a program which was entered into because of a previous interrupt. If an input B occurs while processing another input A, or at the same time as another input A with higher priority, then input B is processed after A is processed, and before the computer resumes its previous activity. A further exception lies in the fact that the interrupt action can be inhibited (i. e., delayed) by the presence of a ONE in a designated position in the order code. Assuming this facility to be carefully used by the programmer, certain central registers (e. g., Q and R2c) need not be preserved by the AI and Resume sequences. Thus some equipment savings result.

Roughly speaking, inputs may be divided into two categories: those which indicate the state of a switch; and those which indicate the presence or absence of a pulse, where the state of the switch which generates the pulse is not itself important. For example, the switch which indicates that a rocket motor is on is an input of the first kind. The train of pulses which comes from a flywheel pick-off is an input of the second kind. The point is that inputs of the second kind feed into counters, and that, in terms of input circuitry, cores Al and Bl of Fig. 3-4 can be eliminated. The integration of each kind of input with the computer will be treated separately. The first kind will be called State Inputs, and the second kind Counter Inputs.

D. State Inputs

State inputs are handled by associating one (or more) input switches with an addressable register. This register is a combination of the erasable and the fixed kinds. Such a register,

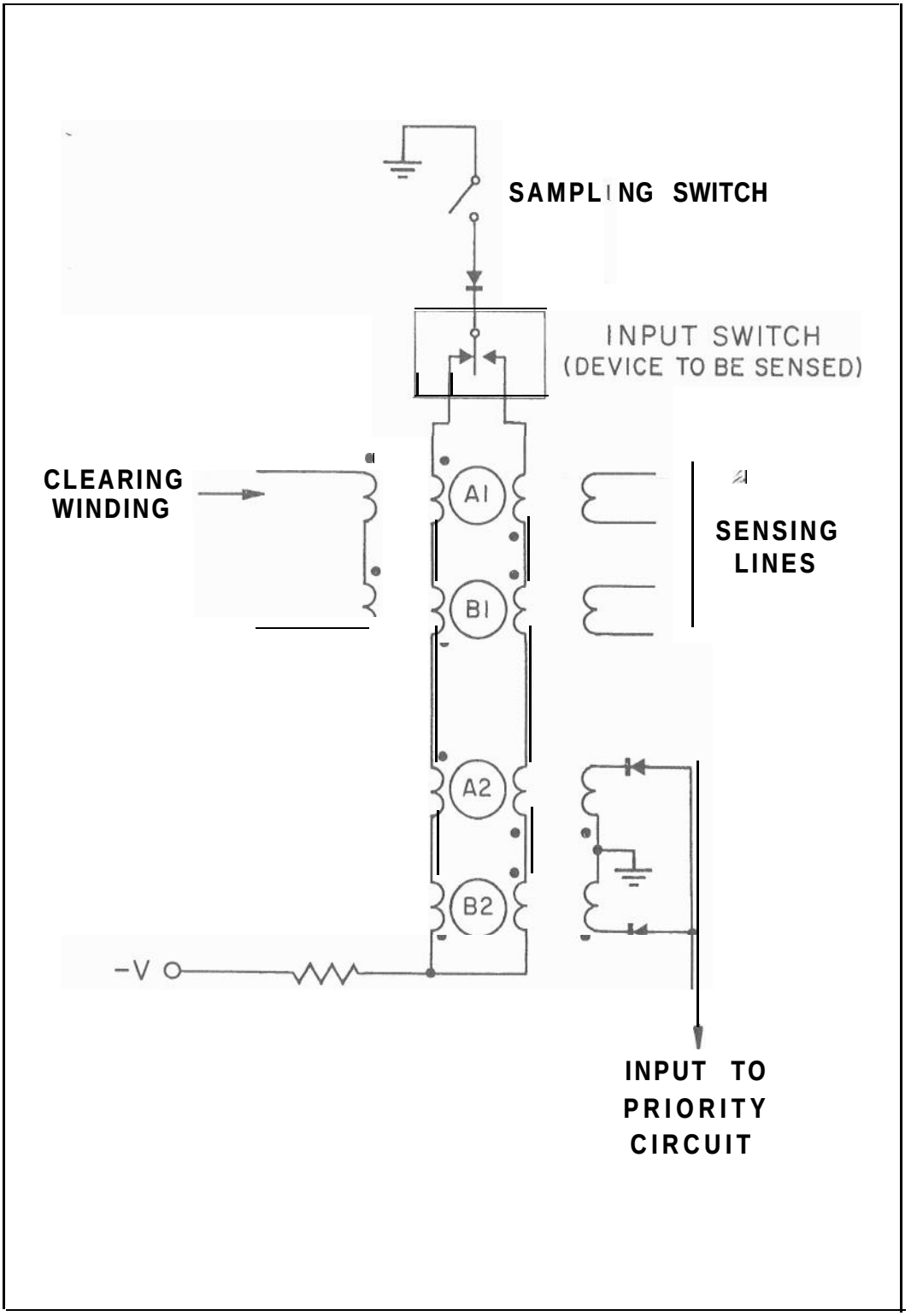


Fig. 3-4 Unit M for input to priority circuit.

with some associated circuitry, is shown in Fig. 3-5. The overall circuit is an elaboration of the circuits of Fig. 3-3 and 3-4. The addressable register itself is composed of cores R1, A1, B1, and R2. Core R1 is a rope core, and can be set to a ONE by having the proper address cleared from the S register, as usual, or by the occurrence of a change of state of cores A2 and B2, as will be explained presently. Cores A1 and B1 describe the state of the input switch, as in the circuit of Fig. 3-4. Core R2 follows the state of R1, and is threaded by some of several sensing lines. (This core could actually be eliminated, by threading the lines through R1 itself.) All of these sensing lines go into the A Sense Write amplifiers shared by E storage.

Operation is as follows: A change of state of the input switch from one sampling time to the next results, at sampling time, in a signal on the base of transistor TP. This signal was previously described as "input to the priority circuit" (Fig. 3-4), and its net result is to drive to a ONE the priority circuit core P. The act of switching core P from ZERO to ONE also results in a signal on line L, which is common to all priority cores, and which goes to the sequence generator. The sequence generator then goes into the Automatic Interrupt sequence which generates, among other things, an Advance pulse. This advance pulse drives to ZERO the highest priority P core, say the one of Fig. 3-5. In so doing, transistor T_R generates a pulse which drives to a ONE the rope core R1, and the register core R2. When the sequence generator produces a CLE pulse during the resulting AI sequence, core R1 is driven to ZERO, and all cores of its associated register, A1, B1 and R2 in this case, are driven to a ZERO. In so doing a word is read into B which consists of the two bits which indicate the state of the switch, and other bits which give the address of the subroutine pertinent to the input being processed. The AI sequence then separates the word into two different words, and transfers program control to the input subroutine. When that subroutine is finished, a Resume sequence is generated, and another advance pulse is produced to test the

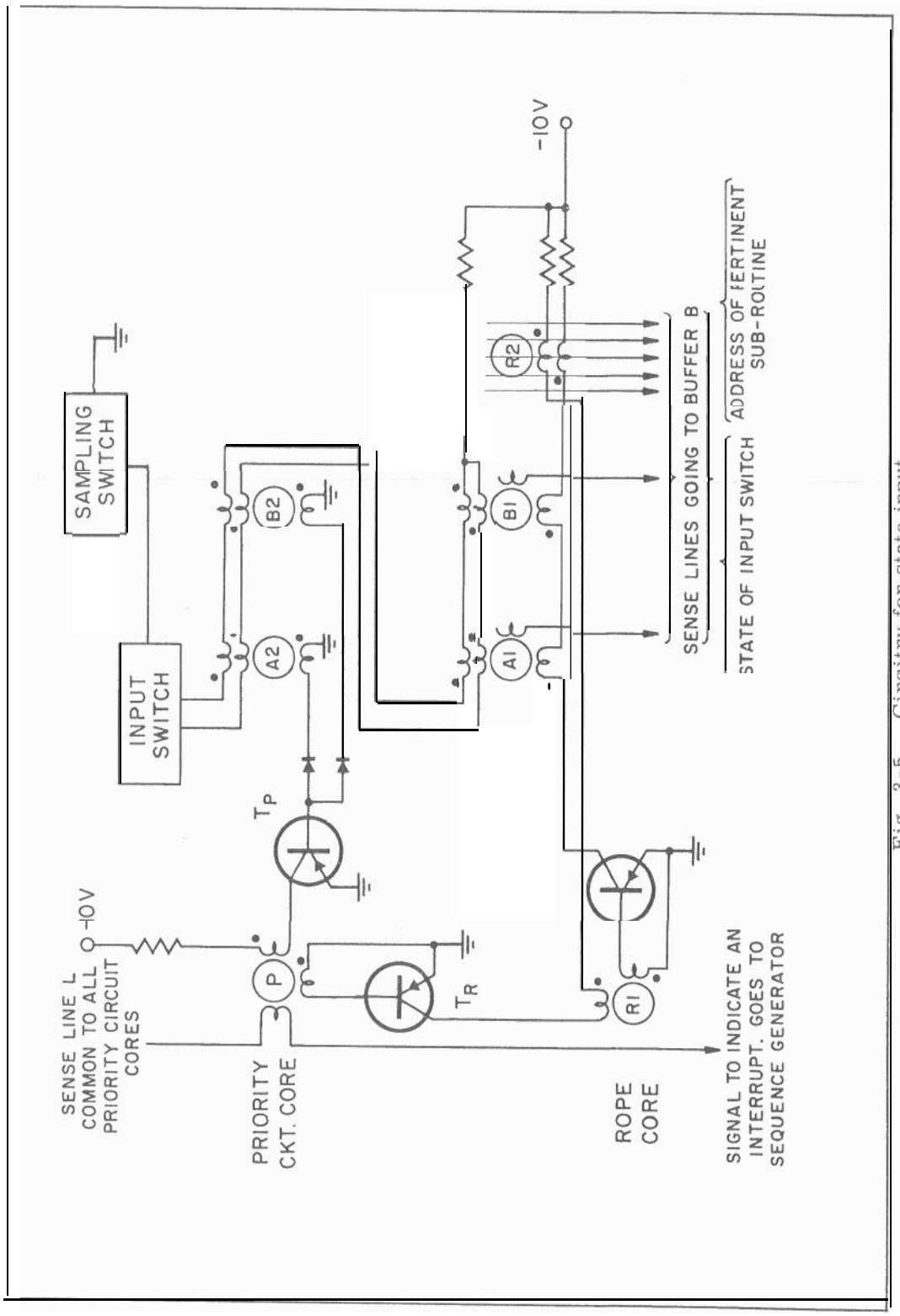


Fig. 3-5 Ciraitry for state input.

presence of other P cores which might have been driven to a ONE. Assuming no other inputs, occur, the computer returns to its previous task.

Cores Al and Bl, cleared by a signal from the transistor associated with Rl, are restored to correspond to the input switch at the next sampling time after action is initiated. In so doing, one of Al and Bl changes state from ZERO to ONE, and for this reason they cannot be used to generate the "change of state" signal to transistor $T_{P'}$. If they were so used, the CLE pulse, which clears the register Rl, would generate a false interrupt signal, setting P to a ONE.

In the example given above, an entire register is used for a single input switch. Obviously, more than one switch may be associated with each register, requiring some additional circuitry for recording which switch caused the alarm. As a final remark on State Input, registers, it should be noted that these registers do not have the means for writing into them possessed by normal erasable storage registers. They can be written into only by means of input switches. The state of the input switch may be examined at any time, under program control, since register Rl can be addressed by normal means. Since no Write busses are provided for cores Al and Bl, however, this interrogation should not be made oftener than once per sample time.

E. Counter Inputs

Counting is an indispensable and frequent activity of control computers of the type under discussion. The sources of things to be counted are both internal and external. In the case of a space vehicle, the external sources are things such as accelerometer outputs, pulses generated by the rotation of a flywheel, a clock which keeps time. Internal sources are various bookkeeping operations necessary for a complex program, such as the number of iterations of a converging process,

or index modification in the case of programs dealing with matrices or vectors,

In either case, a solution which permits programming and computing time economies is to have special registers. These ideal registers have all the attributes of addressable erasable storage registers plus two others: they count pulses on an input lead independently of other computer activity, and they generate a signal upon overflow (Fig. 3-6). The contents of an ideal counter register may be examined at any time by reading said contents into B, and a word may be transferred from B into the counter; this last property allows counters to be preset to any number, under program control.

It is possible to realize these ideal counter registers or ones very nearly ideal by having active counters, with many components. A more desirable solution, where many counter registers are to be employed, is to have a system in which the registers themselves are passive, i.e., cannot do any counting. This requires a central unit which adds one to a number read into it, which is shared by all counter units. It will be recalled that such a unit was described in the section on central registers, i.e., the Z register. This solution is more desirable only because it saves equipment. The properties of the counter registers become slightly less than ideal, because of certain timing hazards which arise from the use of common equipment, and because the overall pulse handling capacity is limited to the capacity of the Z register. However, the end product is a reasonably economic system for a few active inputs, and a very economic system for a large number of slow inputs. As will be seen, there are many similarities between the proposed input counting system, and the more general state input system described previously.

Shown in Fig. 3-7 is a schematized system of two counter inputs and counter registers. When an input occurs, the priority

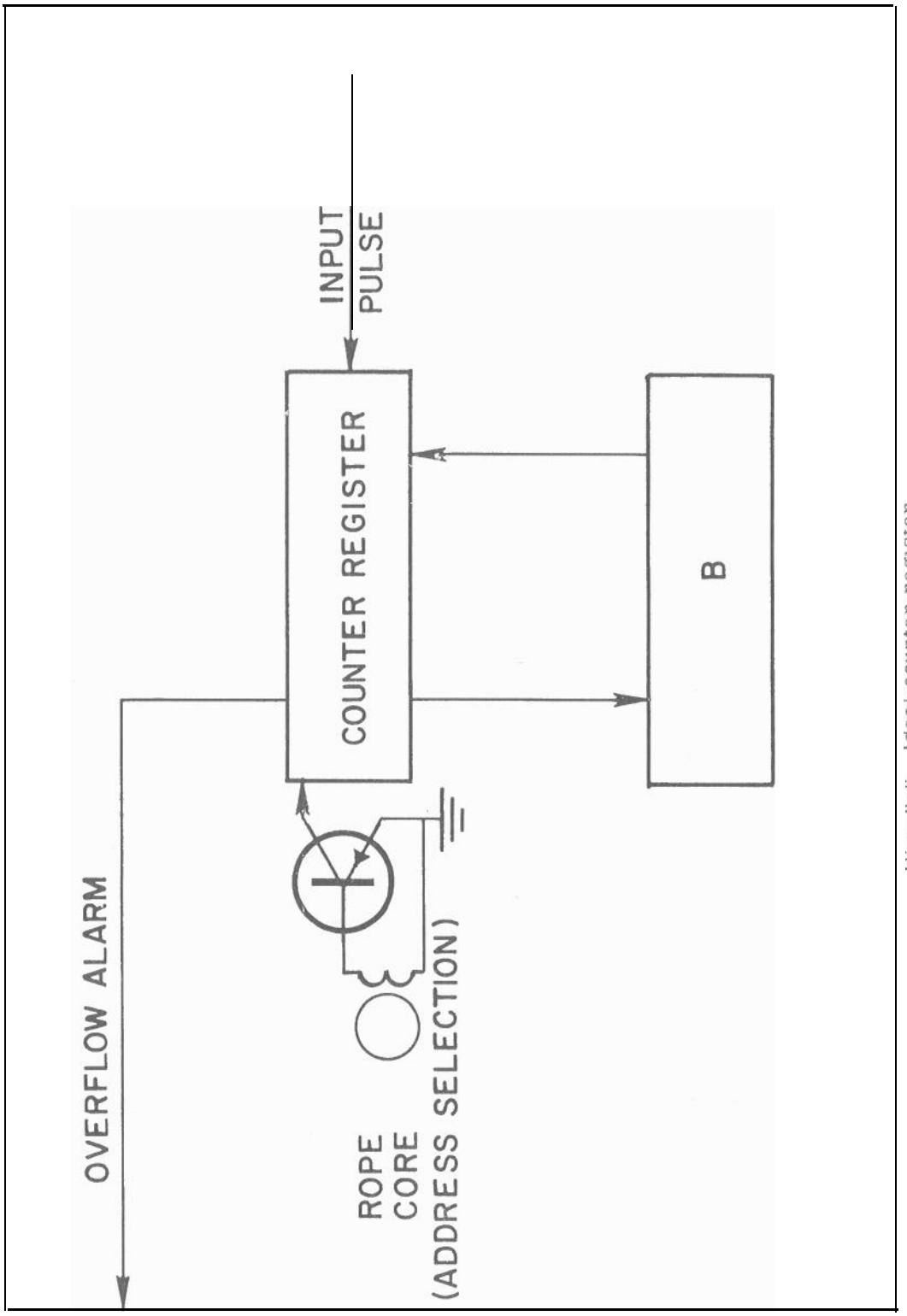


Fig. 3-6 Ideal counter register

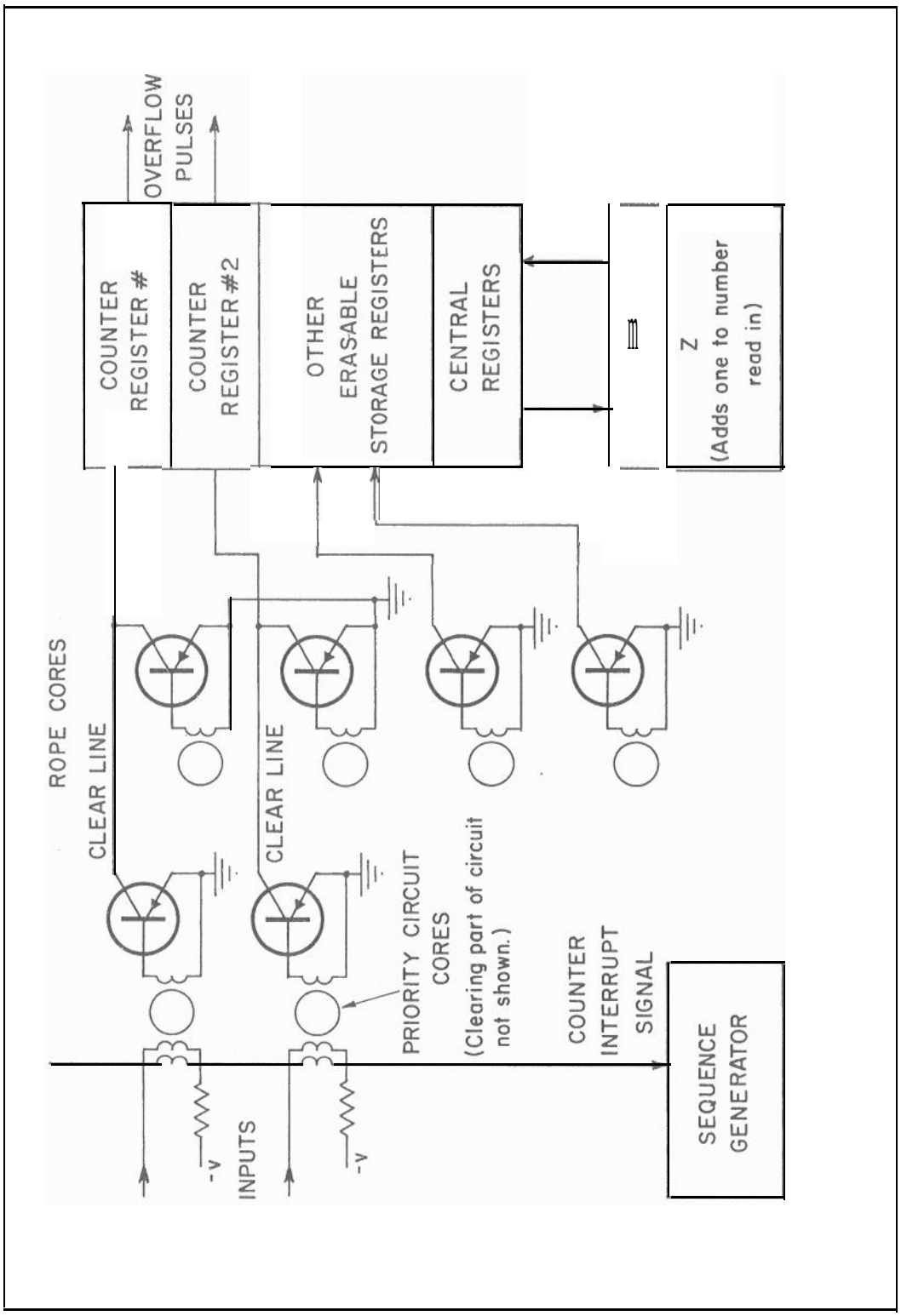


Fig. 3-7 System for counter inputs.

circuit core associated with that input is driven from ZERO to ONE, and an interrupt signal is generated. **As** in the case of state inputs, an interrupt signal causes the sequence generator to enter a special sequence. However, the special sequence which applies to counter inputs is shorter and simpler than that which applies to state inputs. Specifically, an Advance pulse is generated, which clears the priority circuit core to ZERO, and at the same time, a WZ pulse is generated. Clearing the priority core causes the Counter Register associated with it to be cleared to all ZERO's (just as if the proper rope core had been set to a ONE) followed by a CLE pulse. Because of the WZ pulse, the contents of the counter are read into **Z,** where a ONE is added in the least significant place.

The very next pulse time, the pulses generated are CLZ and WE, which cause the incremented count to be placed back in the proper Counter Register. Following this, another Advance pulse is generated to test for unserviced inputs, and the two pulse time process repeated until there are no more priority cores which have to do with counters at ONE. When this condition is met the sequence generator returns to normal functioning. All counters are updated at the end of every sequence, and once per cycle in the addition loop.

It is worth while to look closely at the difference between the processes for state and counter inputs. The most striking difference is in the times required to process each kind of input. **A** state input interrupts the program currently being executed, and transfers control to a special subroutine which deals with that input. This requires that many orders be executed, and since each order is about 10 pulse times long, each state input requires hundreds of pulse times. A counter input, on the other hand, requires exactly two pulse times to be processed. These pulse times are sandwiched between pulses of a normal sequence, when the **Z** register is free. Thus, the

current program is interrupted only in the sense of a short delay being introduced. Further: the circuits may be so arranged that no delay whatever is experienced when there is no counting to be done.

Each Counter Register has a transistor associated with its most significant digit core (not shown in Fig. 3-7), which generates an overflow pulse when the count reaches a certain size. This pulse can become a state input, so that a special subroutine is entered into every time that counter overflows. Alternately, the inputs to a counter can come from within the computer itself. For example, a rope core, whose switching causes the execution of an order, can have a transistor associated with it which feeds a counter. Every time that order is executed, the contents of the counter are automatically increased by one. This permits programs to be efficient in the sense of little time being wasted on some aspects of bookkeeping.

The counters described thus far can only be used for accumulating positive increments. There are basically two alternatives to the problem of positive and negative increments. One of these assigns two counters to each variable capable of increments of either sign, and a separate count is kept both of positive and negative increments. This arrangement, while workable, has the undesirable feature of requiring a subtraction every time the absolute count of a variable is to be examined.

The other solution to the problem of bipolar increments associates two priority input cores with each counter, one for positive and one for negative increments. Then, depending on which input core is excited, either an “Add One” or a “Subtract One” sequence is entered. The “Add One” sequence can be as before, The “Subtract One” sequence can still be done with only the Z register by a double complementing operation, since complementing the contents x of a counter, adding one to the result, and complementing this in turn is the same as subtracting

one from x [(x| + 1)| = x - 1; x = x'] . This scheme has the disadvantage of requiring four pulse times to increment a counter, and also that the Z register be placed on the α side, instead of the β side where it is presently assigned (Fig. 3-8). The new location of Z in turn requires a different incrementing sequence for positive increments than the simple two pulse sequence used in the case of positive increments only. The two sequences are listed below.

Positive Increments

1 CL CNTR, WC1
2 CL Cl, wz
3 CL z, WCl, WP
4 CL Cl, CL P, WE (= W CNTR)

Negative Increments

1 CL CNTR, WC2c, WlC3d
2 CL C2c, wz
3 CL z, WC3c, WlC2c, WP
4 CL C3c, CL P, WE (= W CNTR)

Registers such as C2d are complementing registers. The above four pulse sequences have the advantage of permitting a parity bit in counters, which is updated automatically, provided that the number of bits in a word, excluding the parity bit, is even. The two pulse sequence, of course, disregarded parity completely. The condition of an even number of bits is to insure that the parity of a word and its complement be the same.

A final and most desirable solution to the problem of bipolar counting involves a modified Z register capable of subtracting one (by adding the complement of one), as well as adding one to the quantity entered into it. Such a modified Z register is currently under development, and it may be possible to build one so that it still requires only a single transistor per bit.

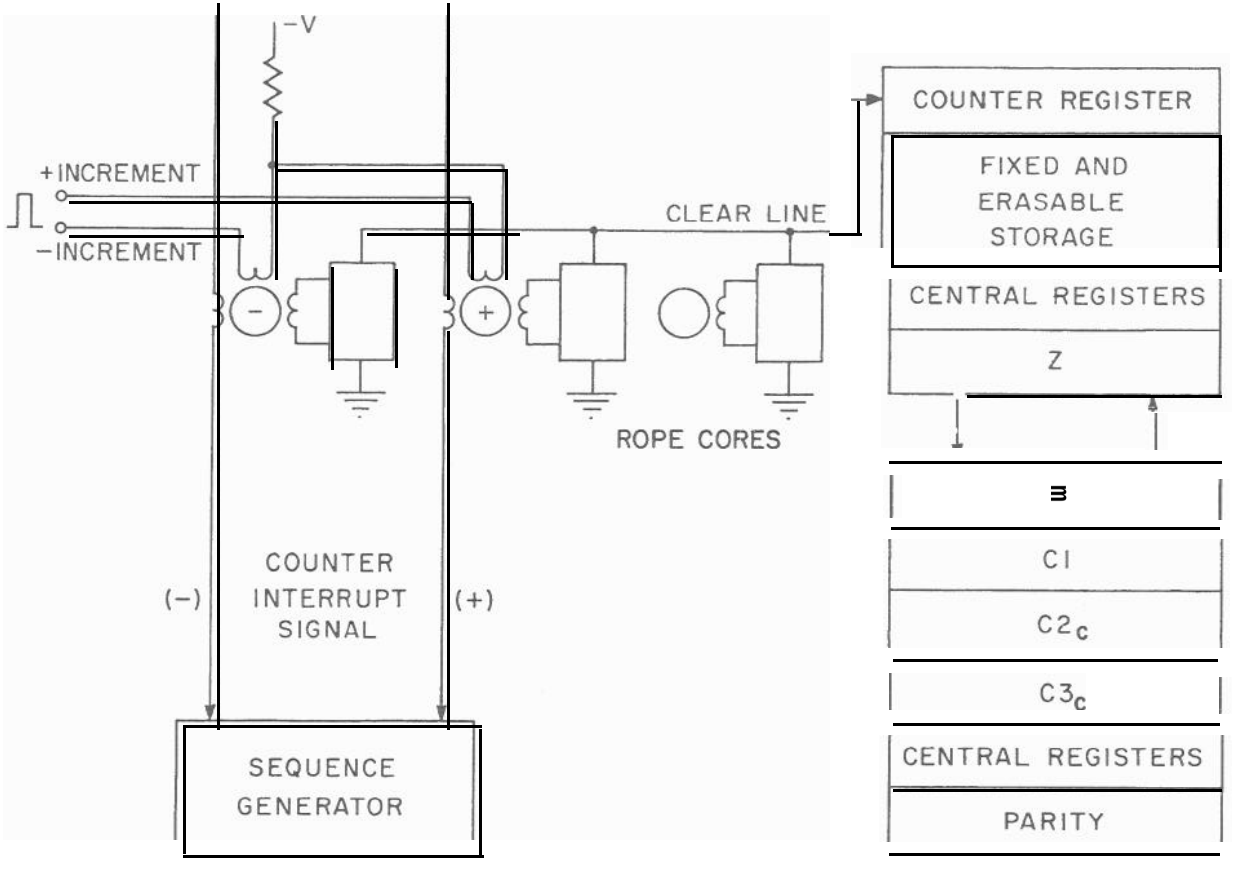


Fig. 3-8 Counter inputs--system for dealing with positive and negative increments.

F. Output Devices

As was the case with input devices, output devices will be discussed without any specific application, and presented as a useful collection of ideas, circuits, and subsystems. No part of the problem of digital to analog conversion will be discussed here, but the generation of power pulses (as opposed to pulses used for signalling purposes only) will be mentioned.

The backbone of the output system is the core rope. Cores in the rope may have transistors or a latch type of solid state device connected to them. The core may be addressed by means of the S register, just as any other fixed or erasable register. Switching one of those cores then generates a pulse, or causes the closure of a switch which remains closed until some other core switches.

The latch devices mentioned above are of particular interest because they approach the ideal relay contact; they are three terminal, four layer transistors which turn on when the base is pulsed positively relative to the emitter (Fig. 3-9), remain on as long as a sustaining current flows through them, and are turned off by pulsing the base negatively relative to the emitter. The sustaining current is of the order of 10 ma. The "on" and "off" signals are of the order of 1 to 5 volts, and about 1 μsec in duration. As shown in Fig. 3-9, these latches can be used to keep a transistor turned on. When the latch is off, the base of the transistor is held at +5 volts. If core 1 fires, the latch turns on, and the base of the transistor is held negative relative to the emitter. The latch is turned off by core 2. Latches may be synthesized out of ordinary transistors, if need be.

An application of the output switch circuit is shown in Fig. 3-10, where it is used to gate a pulse generator. Thus the load may be subjected to trains of pulses, or rate R, under program control. The pulse generator may be common to several output switches.

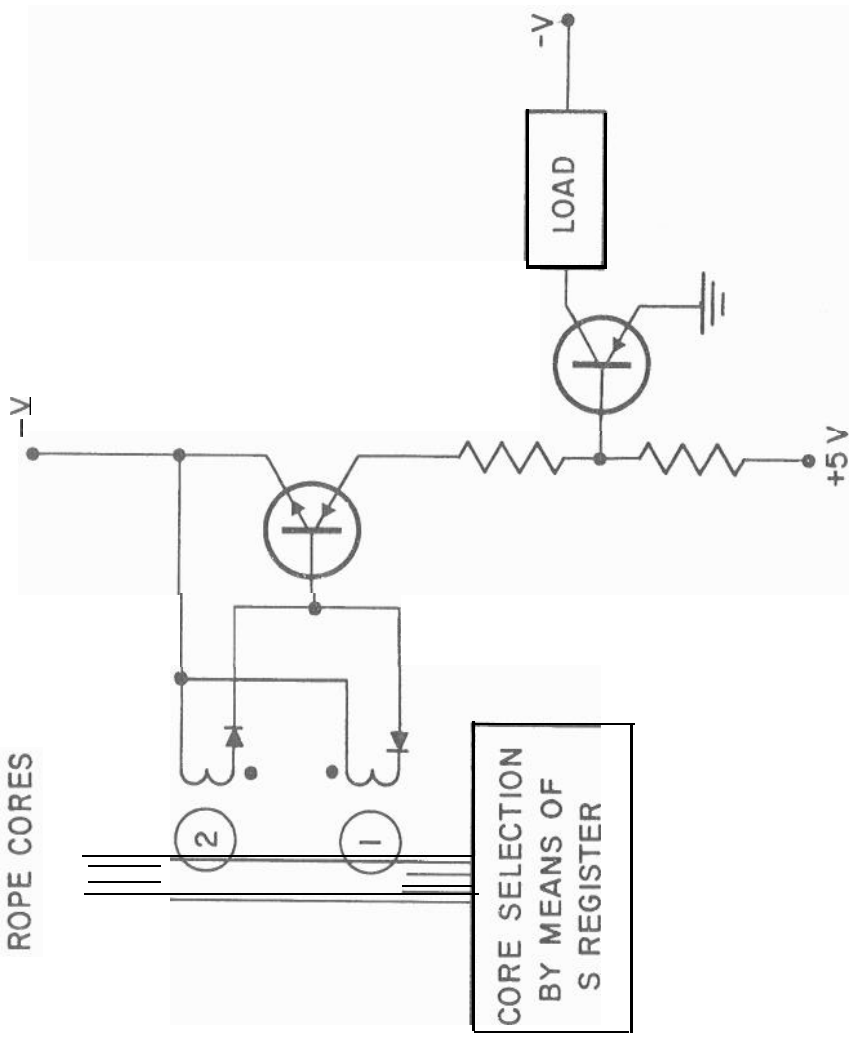


Fig. 3-9 Output switch circuit.

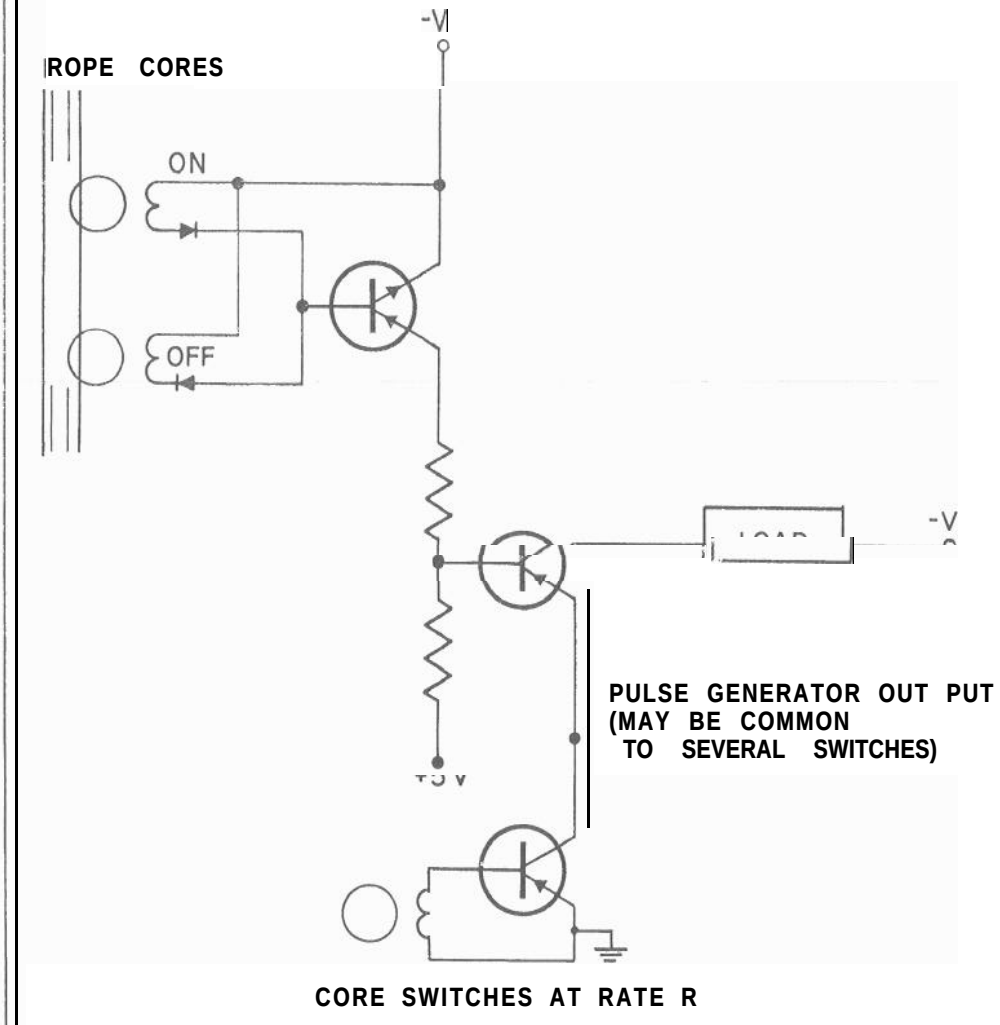


Fig. 3-10 Gating of a pulse generator by an output switch.

An important output problem is that of pulse rate generation. From the point of view of efficient computer utilization, it is desirable that there be a kind of pulse generator whose output rate is set by the computer by simply writing a number into a special register. The intent is to free the computer from the task of regulating pulse rates, The computer sets a rate in a given pulse generator, and this rate is produced independently of the computer until the rate is next set. The premise is that the intervals between settings of the pulse rate are large compared to the intervals between pulses. With this type of pulse generation it is possible to use a medium speed computer for the overall control of many independent pulse rates, each of which varies slowly, but may be of the order of the computer clock rate.

The desired pulse generator features may be achieved at relatively low cost. The basic scheme is outlined in Fig. 3-11. There is a counter associated with a clock (which could be the computer clock), out of which come the various pulse rates $R, \frac{R}{2}, \frac{R}{4}$, etc, each pulse rate appearing on a separate wire. For each independent pulse rate Q to be generated there is an Output Rate Register. These registers are part of the E storage, and their contents can be set by the computer. Each bit position of each register acts as a gate for one of the clock counter rates, as shown in Fig. 3-12. When a ONE is read into the core corresponding to a bit position, the solid state latch tied to that core is turned on, and remains on until the core is cleared to a ZERO. While the latch is on, the base of the transistor connected to the latch is held negative, and the transistor acts as a closed relay contact. While the latch is off, the transistor does not conduct because its base is held positive.

A desired rate Q is generated by combining pulse rates from the clock counter. Thus

$$Q = a_1 R + a_2 \frac{R}{2} + a_3 \frac{R}{4} + a_4 \frac{R}{8} + a_5 \frac{R}{16}$$

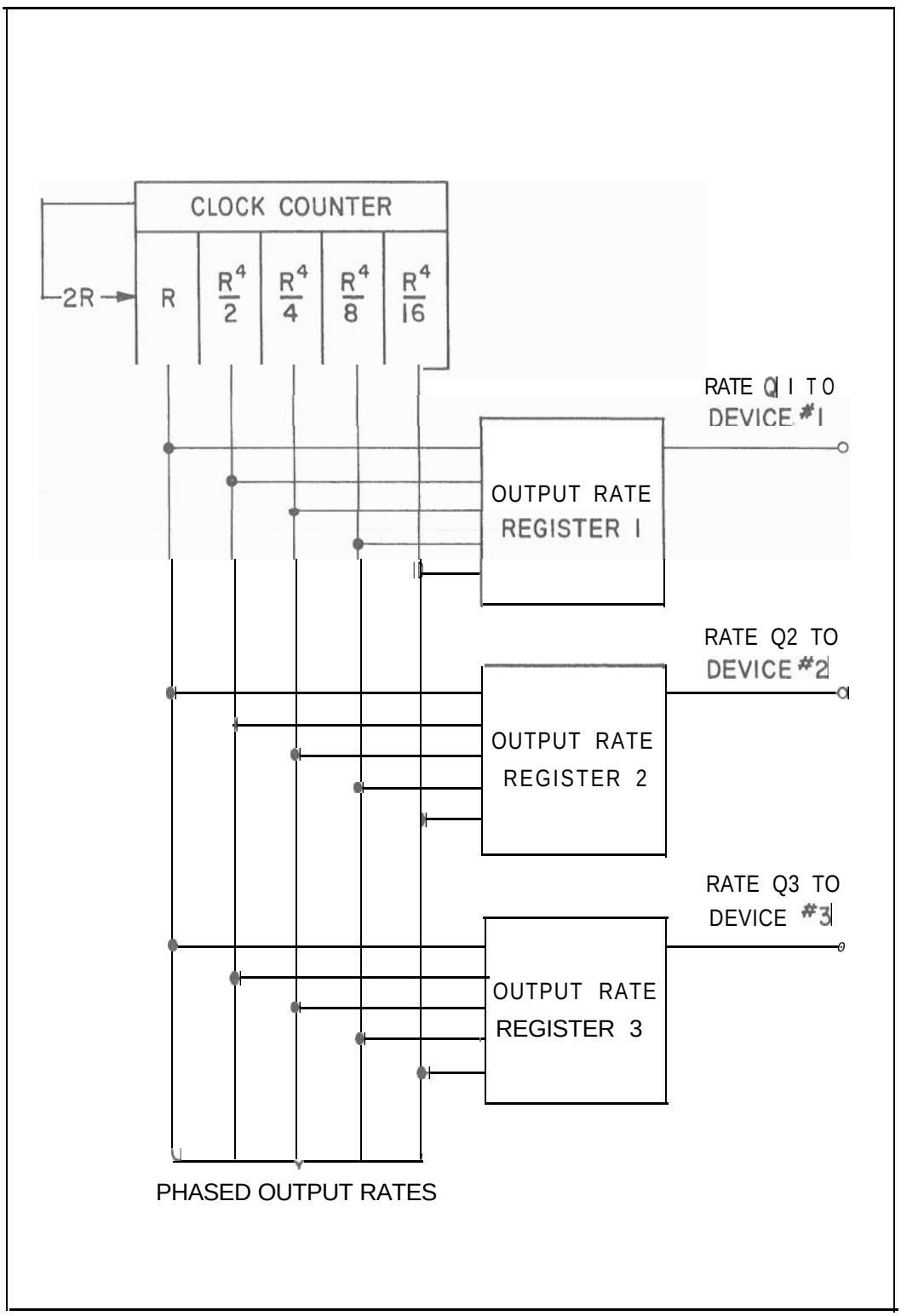


Fig. 3-11 Generator of several pulse rates

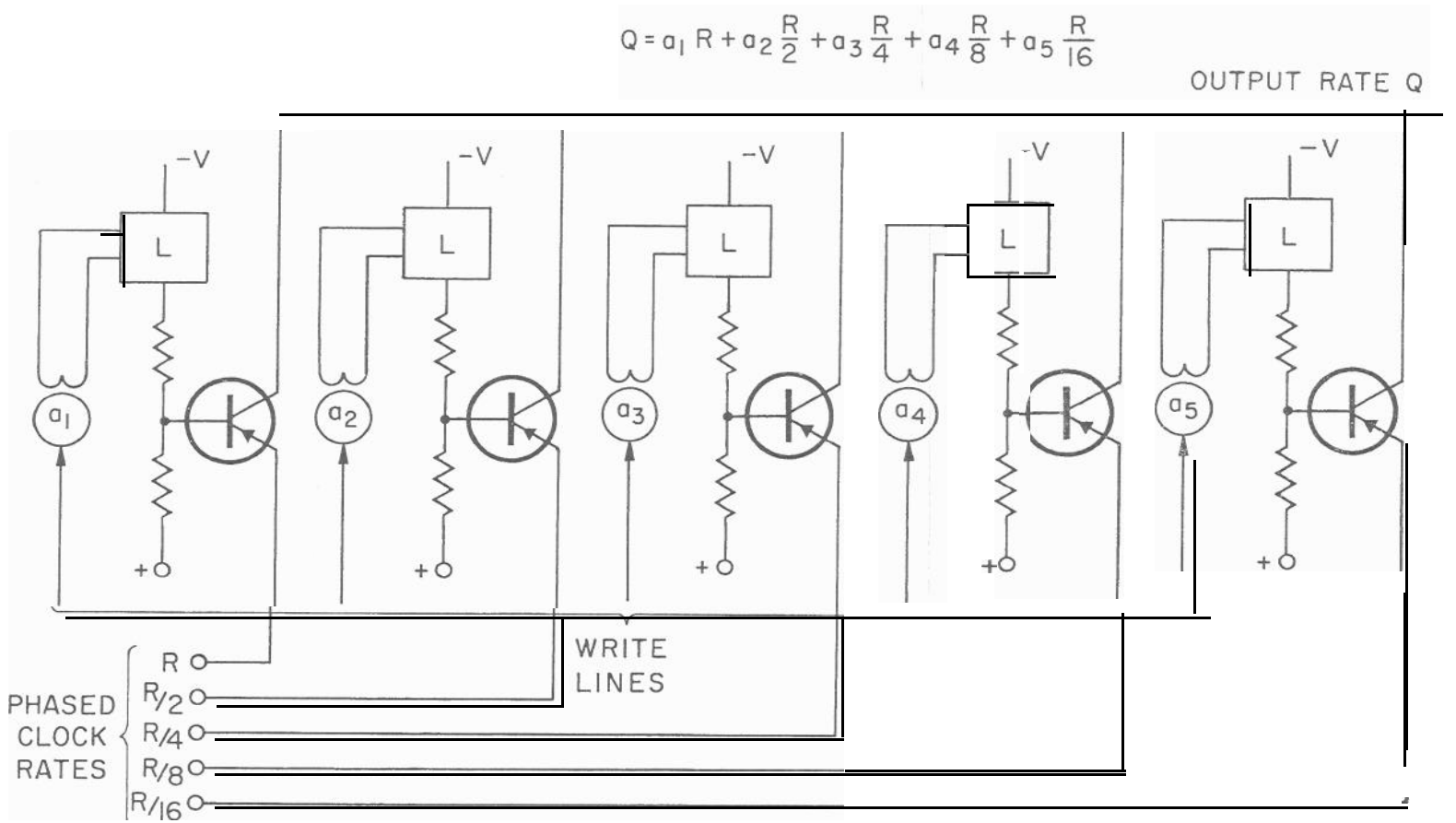


Fig. 3-12 Details of an output rate register.

where each a is either 0 or 1, and corresponds to a bit position. It is necessary to show that the various rates R, R/2, etc., can be generated in such a way that no pulses are lost due to time coincidence; for example, if the desired rate were Q = 5R/8, accomplished by combining rates R/8 and R/2, pulses would be lost if the wires carrying these respective rates had pulses on them at the same time.

The clock counter of Fig. 3-11 has an internal pulse source of frequency 2R, and five flip-flops which successively halve the input rate. Thus, after every input pulse, the state of the Clock Counter can be expressed as a binary number. If the outputs $R, \frac{R}{2}, \frac{R}{4}$, etc., are taken to be the transitions of the halving flip-flops, as they change from state ONE to state ZERO, then there can be phasing problems, since it is possible to have several flip-flops going from ONE to ZERO simultaneously. However, there can be only one flip-flop going from ZERO to ONE at any one pulse time. In fact, there will always be a single flip-flop going from ZERO to ONE, with the exception of the pulse time in which the count goes from all ONES to all ZEROS. Therefore there are no phasing problems in the rates $R, \frac{R}{2}, \frac{R}{4}$ etc., are generated by the ZERO to ONE transistions of the halving flip-flops. The situation is illustrated in the tables of Fig. 3-13, which shows the generation of several output rates.

A variant of the Rate Generator discussed above which could be of some importance is the Rate Multiplier. The output Q of a Rate Generator is Q = AR, where A is the content of the Rate Generator. Thus, if R were a variable rate, Q would be the product of A and R. Rate Multipliers each require a set of counting flip-flops, so that they are fairly expensive in equipment*. However, all these special techniques are available for generating variable pulse rates independently of the main computer activity.

*For a discussion of this subject, see Grabbe, Ramo, Wooleridge, "Handbook of Automatic Controls and Computation".

Count	Clock Counter Output Pulses					Example 1	Example 2	Example 3	
	$\frac{R}{16}$	$\frac{R}{8}$	$\frac{R}{4}$	$\frac{R}{2}$	R	$Q = \frac{5R}{16}$	$Q = \frac{3}{4}R$	$Q = \frac{31}{16}R = 2R$	
0 0 0 0 0								X4----	Missing Pulse
0 0 0 0 1					1			1	
0 0 0 1 0				1			1	1	
0 0 0 1 1					1			1	
0 0 1 0 0			1			1	1	1	
0 0 1 0 1					1			1	
0 0 1 1 0				1			1	1	
0 0 1 1 1					1			1	
0 1 0 0 0		1						1	
0 1 0 0 1					1			1	
0 1 0 1 0				1			1	1	
0 1 0 1 1					1			1	
0 1 1 0 0			1			1	1	1	
0 1 1 0 1					1			1	
0 1 1 1 0				1			1	1	
0 1 1 1 1					1			1	
1 0 0 0 0	1					1		1	
1 0 0 0 1					1			1	
1 0 0 1 0				1			1	1	
1 0 0 1 1					1			1	
1 0 1 0 0			1			1	1	1	
1 0 1 0 1					1			1	
1 0 1 1 0				1			1	1	
1 0 1 1 1					1			1	
1 1 0 0 0		1						1	
1 1 0 0 1					1			1	
1 1 0 1 0				1			1	1	
1 1 0 1 1					1			1	

Fig. 3-13 Pulse Rate Generation and Examples

CHAPTER 4

GENERAL REMARKS

A. Size, Weight, and Number of Components

In this section the size, weight and component quantity of the various subsystems presented before will be discussed.

1. The Core Rope

This subsystem is the most unusual one from the point of view of standard computer technology and packaging. The ropes constructed to date (Fig. l-l) have a core density of 32 cores per $in.^3$. Since each core represents one or two entire words of up to about 50 bits total, the bit density of the core rope could be about 1500 $bits/in.^3$. This density does not take into account connectors, access equipment or mounting of the core rope on some fixture. As a better measure of size, a 4000 word memory, of 24 bits per word, will be postulated.

The core rope for this memory will consist of 2000 cores (two 24 bit words per core) and occupy 160 $in.^3$, complete with mounting boards and end connectors. The forty- eight sensing amplifiers needed in this case require 25 $in.^3$, and the S register and its drivers require 50 $in.^3$. Hence the total volume of a 4000 word system would be 235 $in.^3$, or 0.14 $ft.^3$. It must be emphasized that this size is not a projection based on future improvements on existing packaging techniques, but a calculation based on an already constructed core rope of 256 cores. The unusually large bit density is achieved by the one core-per-word character of the memory, not by refinement of packaging. The

total weight of such a system, again including all connectors and mounting boards, would be of the order of 7 lb. Reasonable refinements on core rope packaging techniques could be expected to reduce both overall volume and weight by one third to one half the figures given above.

2. The Erasable Storage

A board containing 16 erasable storage registers, each of 12 bit capacity, has been constructed (Fig. 4-l). From the existing registers, the following calculations may be made for 24 bit registers. Each register, including the cores and transistors necessary for access to it, would measure 4. 75 X. 5 X. 5 in.^{3}, or 1. 2 in.^{3}. The weight of such a register, not counting mounting boards and connectors, is about 20 gm. Including mounting board and end connectors, and assuming a board size which allows advantage to be taken of the register dimensions (the board of Fig. 4-l does not), it should be possible to have a system of 128 erasable storage registers inside of 160 in.^{3}, and which weighs about 5 lb.

3. The Central Registers

Except for the S, **Z** and Parity registers, all other central registers are electrically similar to ordinary erasable storage registers, and of comparable volume. The S register has already been counted in the discussion on the core rope. The Z and Parity registers require about three times the weight and volume of an ordinary register. Assuming a system of 10 central registers, plus **Z** and Parity; and making generous allowance for the volume required by control pulse circuitry; the central register complex should occupy a volume about as large as that occupied by 25 erasable storage registers, i. e. , about 35 in.^{3}, and weigh about 1. 5 lb.

4. The Clock and Sequence Generator

These two items can fit a volume of less than 100 in.^{3}, and weigh about 2 lb.

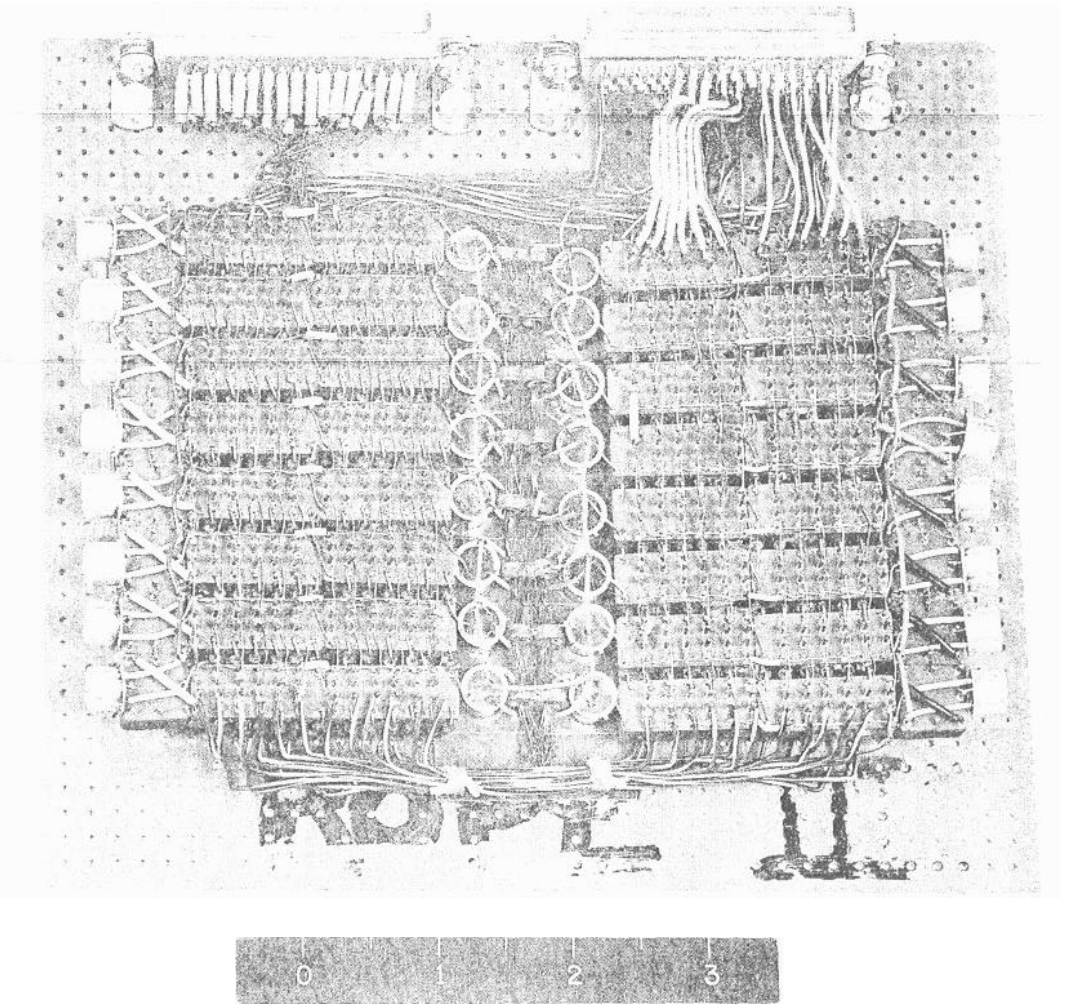


Fig. 4-l Sixteen word erasable storage system

5. Input and Output

An assignment of 100 in.^{3} and 4 lb to the input-output system seems reasonable to this writer. This would represent approximately 32 counter inputs (and counter registers), and 180 state inputs, figured at 6 state inputs in the volume occupied by two erasable storage registers.

6. Computer Totals

The many assumptions and calculations may be summed up as follows: a computer with a word length of 24 bits, with 4000 words of instructions and constants, with 128 erasable registers, and 32 counter and 180 state inputs, would occupy a volume of about 0. 37 ft^3, and weigh about 20 lb. It is interesting to note that these calculations give an overall density of 54 lb/ft^3, which is approximately the density of other airborne digital computers. The various computer totals, including conservative estimates on the number of components, are summarized in Fig. 4-2.

B. Reliability

In discussing the potential reliability of the type of computer proposed here it is necessary to make several distinctions and assumptions. Primarily, it is necessary to distinguish between catastrophic failures e. g. , burning out of a transistor, or opening of a solder joint, or breaking of a wire, and temporary failure due, for example, to voltage variations or a burst of extraneous radiation. It is assumed that a temporary failure does not result in any mechanical or electrical damage.

If there is sufficient program storage, and if speed of computation is not critical, then it is possible to protect against temporary failures by computing things in two different ways, at two different times, and checking results. A parity check system in which each word has a parity bit associated with it serves to detect the large majority of temporary failures and to initiate actions that call upon the computer's error correcting

	Volume in^3	Weight lb	Transistors and Latches made out of 2 Transistors	Diodes	Cores	Resistors	Comments
Rope Including S Register	200	6 0	6ω	——	2000	200	12 bit S register 5 T's/double bit
Sense Amplifiers and α β Amplifiers (48 bits)	35	1.0	480	——	——		10 T's/bit
Erasable Storage	160	5.0	256	3070	3070	25	128 registers, 24 bits each 2 T's/register
Central Registers and Control Pulse Amplifiers	35	1.5	168	30	240	40	12 central registers 60 control pulse amplifiers at 2 T/amplifiers. 48 T's for P and Z
Clock and Sequence Generator	100	2.0	40	80	200	20	
Input - Output	100	4.0	450	1300	1300	400	
	630 in^3 .375 ft^3	19.5 lbs	1454	4750	6810	685	

Fig. 4-2 Physical characteristics

programs.

Catastrophic failures are a more serious concern because of the difficulty in protecting against them. There are possibilities such as duplicating equipment, or having "spares" which can be connected automatically in place of a defective subsystem. These schemes are not attractive because of the expense in terms of volume and weight, and because it is not clear that they necessarily result in a more reliable system. The glib "connected automatically" may imply more reliability problems being generated than solved. The following paragraphs will contain a discussion of the likelihood of catastrophic failures in the proposed computer, without any duplications of equipment.

There is mounting evidence that a computer of the size and number of components of the one proposed here can be built so as to function for years without catastrophic failures. If it is assumed* that there will be no failures of a mechanical nature (poor construction and mounting), then the failures of concern are electrical ones. Of the various components listed In Fig. 4-2, cores may be eliminated as a source of electrical failures, for they may be damaged only by mechanical action or by temperatures above 300°C. The components most prone to failure are transistors and diodes.

The quality of transistors is improving steadily. The MIT Instrumentation Laboratory has had a computer in operation for over 7000 hours, in which there have been no failures among the 500 transistors used in the logic sections. Reports from Lincoln Laboratories** show similar results. Experience with diodes is similar to that with transistors. It is then reasonable

*This assumption does not imply a dismissal of the problem of construction as trivial.

**MIT- Lincoln Laboratories
Tech Report No. 199, ASTIA 214565, D. J. Eckl, P. A. Fergus, R. L. Burke, "Transistor Life Experience", Nov. 1959

to expect a system in which transistors and diodes numbering in the low thousands could be made so as to last for a few years without any catastrophic electrical failure. If the circuits are so designed that they are insensitive to changes in transistor parameters, the lack of catastrophic failures would imply no maintenance. The core-transistor circuitry which is the basis of the proposed computer is insensitive to changes in transistor β's; the power dissipated in the transistors is well within the device's capabilities, and voltages are within breakdown limits.

The most serious remaining reliability problems are physical construction, sensitivity to temperature, and sensitivity to radiation.

Of these, the effects of radiation present in outer space are the least known.

C. Construction and Experience to Date

Various parts and circuits of the proposed machine have been built and tested in order to bring out logical and electrical problems, and to acquire some feeling for the relationship of the paper design to the actual physical system. This section will describe briefly what was built and how it worked. More detailed accounts will be published in future reports.

The most unusual element of the computer is its wired-in memory, and establishing the practicality of large ropes was felt to be a necessary first experiment. As a consequence, core rope systems of increasing sizes were built and operated. The largest system built to date has been one of 256 cores, and it has operated successfully. The basic electrical design problem consists of maximizing the transfer of energy from a pulse on the CLE line to the appropriate sense lines through a selected core, and minimizing the energy transferred through air coupling and through saturated cores. Air coupling was minimized relative to core coupling by essentially simple construction techniques, and by using as much iron in the cores as is practical with available

currents,

The sense lines were threaded through the rope cores in such a way that the noise generated by the CLE pulse through the cores that do not switch cancels out. The uniformity required of the cores is within commercial standards (± 10% variation in output signal for a given switching current). Sampling of the output signal was required only in the sense that the noises induced by the rise and fall of the inhibit currents are blanked out. The conclusion derived from the experience to date is that ropes of up to 4000 cores are perfectly feasible.

The erasable storage registers are of conservative design, and no serious problems have been encountered in that area. Various forms of pulse amplifiers, core-transistor pairs and shift registers have been built and tested; as a result there now exist definite design procedures which lead to sucessful circuits , A sequence generator has been built and operated, and also an 18 core priority circuit. The principles and circuits implied in the Z and P (Parity check) registers have been shown practical.

At this time an abbreviated version of the computer is being assembled, in which there are 32 erasable storage registers and 224 fixed registers (i. e., a 224 core rope). The word length is 12 bits, and there is a four instruction order code. This simple computer is expected to be useful in preliminary system tests.

The experienced acquired to date indicates that the proposed computer is feasible and basically sound. The various design problems encountered do not call for refined circuitry or close component tolerances, and the components used are readily available commercially.

D. Power and Speed

It has been mentioned that with the proposed techniques a variable speed computer results in a correspondingly variable

power consumption. This feature is of obvious value in a space probe, where there is ample time to compute, and hence, where power may be saved. It is also of value in other airborne applications because it is possible to check out the computer at low speed, and not generate appreciable heat. In fact, it might be possible to keep the computer running continuously at very low speed and have a constant check on its functioning.

If the computer's oscillator clock should stop, no cores would switch, and the only currents would be those due to transistor leakage and those due to flip-flops*. The leakage current of a transistor in the "off" condition is of the order of 10 μa. If several transistors are in series, the leakage current is even less. For the representative computer, the total "stopped" current would not exceed 10 ma, exclusive of flip-flop current, which means 0.1 watt of standby power.

It makes sense to speak of the energy (in watt-μsec) required for various operations such as clearing a register of switching a rope core. It turns out that it is possible to estimate the energy required by the actions resulting from the various possible control pulses. All control pulses except CLS and CLE require between 3 and 4 watt-μsec. The inhibit currents generated by CLS have an amplitude of 300 ma, and last for about 5μsec, and are developed across a 10 volt power supply, so that they use 15 watt-μsec. There are 12 inhibit currents, plus a set drive; hence the CLS pulse uses 185 watt-μsec. Similarly, CLE uses 15 watt-μsec. With these figures in hand it is further possible to estimate the energy of the various instructions. For example, the sequence for CA X has two occurrences of CLS and CLE, and 29 of other control pulses. On this basis, per-

*Flip-flops are presently used in the sensing amplifiers and drivers of the prototype computer mentioned in Section 4C. Present plans call for the eventual replacement of these flip-flops by latch circuits which consume essentially no power in the "off" condition.

formance of CA X uses about 520 watt-μsec. Energy required by other instructions is approximately the same. At full speed, i.e., performing an instruction every 48 μsec, the estimated power consumption is approximately 11 watts. If flip-flops are used instead of latches in the Sense Write amplifiers (48 of them) then power consumption increases by about 4 watts. This increase is due to the fact that latches conduct, at most, during 3 μsec out of every 12 μsec clock cycle (the time between successive α pulses), whereas flip-flops conduct all the time at a rate of 10 ma. The total power consumption for the computer, running at maximum speed may then be estimated at 15 watts.