336200
47

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LINCOLN LABORATORY

TX-2 TECHNICAL MANUAL

LINCOLN MANUAL NO. 44

Volume 1

MAY 1961

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology, with the joint support of the U.S. Army, Navy and Air Force under Air Force Contract AF 19(604)-7400.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

LEXINGTON MASSACHUSETTS

PREFACE

This manual documents the organization and logic of the Lincoln Laboratory TX-2 computer. The manual is intended to serve primarily as an aid to the computer maintenance personnel and, in order to be of maximum value, should be supplemented by a file of TX-2 block and wiring schematics. The TX-2 "Red Book", which describes the TX-2 circuits and packages, should also be available for reference. The manual by itself will also be useful to programmers and other computer operators who desire more detailed information about the operation of the computer than is available in the TX-2 programming manual.

The first volume of the manual gives a simple general description of the organization of the computer. The principal emphasis is on the general specifications, organization, and component parts of the computer. The second volume covers in detail the logic and timing of the computer. Each element, i.e., the Memory Element, Control Element, Program Element, Exchange Element, Arithmetic Element and In-Out Element, is discussed separately. Descriptions of the individual in-out units will be added in the future to complete the chapter on the In-Out Element. The final volume lists and describes all the possible control, memory, and instruction timing cycles. It is arranged primarily for convenient use during computer maintenance operations.

Acknowledgements are due to the technical writers from Jackson & Moreland, Inc., who prepared the text and figures for the manual; and to Madeline Higgins at Lincoln Laboratory who retyped the several versions of the manuscript and, with the help of Anna Nagy, prepared the final text for printing.

J.M. Frankovich
Group 51

June 1961

TX-2 TECHNICAL MANUAL

TABLE OF CONTENTS

VOLUME I

VOLUME II

VOLUME III

CHAPTER 1

INTRODUCTORY DESCRIPTION

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTORY DESCRIPTION

1-1 INTRODUCTION

TX-2 is a large-scale, high-speed, general-purpose digital computer designed and built by the Digital Computers Group at the Massachusetts Institute of Technology Lincoln Laboratory, Lexington, Massachusetts. It is an experimental computer and contains both solid-state and electron-tube devices. Provisions have been made so that additional circuitry can be readily incorporated into the existing framework in order to increase the usefulness of the computer.

1-2 FUNCTION OF COMPUTER

TX-2 is currently used as a research tool in scientific computations and in data-handling and real-time problems. It differs from conventional general-purpose computers in that it permits an exceptionally high degree of flexibility in programming and in the use of input-output devices. This increase in flexibility, needless to say, is accompanied by a marked increase in design complexity. In many respects the machine is unique and one that has no counterpart at present.

1-3 SPECIAL FEATURES

To enhance programming flexibility, several types of special control have been incorporated into the TX-2 computer. For example, the configuration of the basic computer word is under program control and allows operands to be divided into subwords during the execution of instructions. This enables the same operation to be performed simultaneously upon several subwords. The result is an over-all increase in the effective speed of the computer.

The multiple-sequencing feature allows automatic switching in the computer among various program sequences. The individual in-out devices are associated with particular program sequences in a manner such that several devices can operate simultaneously. A particular program sequence is then started when the associated device needs attention. By assigning priority numbers, it is possible to determine which program sequences can start at a given time. A low priority sequence can also be interrupted when a higher priority sequence needs to be started. Data can be thereby transferred between the computer and the in-out device having the highest priority first, and then between the computer and the next lower priority device, etc.

With overlapped memory operations, instruction words and operand words can sometimes be obtained simultaneously from memory, thereby effectively halving the memory cycle times.

Other features of TX-2 are index registers and deferred addressing. These features, now common in many computers, serve as further aids in programming. Index registers allow the programmer to effectively modify the base address section of the instruction word without actually modifying the instruction word itself. By means of deferred addressing (or substitute addressing), it is possible to go to an intermediate address in memory in order to obtain the desired address of an operand.

1-4 COMPUTER STRUCTURE

TX-2 is a parallel binary computer with a basic 36-bit word length. The words in the Memory Element also have a parity check bit and a programming meta bit. Any word in the Memory Element can be used for instructions, operands, or deferred addresses. The internal memory is all random access and consists of 69,632 registers of parity checked magnetic-core memory and about 70 additional toggle switch, plugboard, and other miscellaneous registers. Approximately 200,000 instructions can be executed per second. Instructions are of the indexed single-address type. A fixed-point, signed-fraction, one's complement binary number system is used.

1-5 BASIC ELEMENTS

Fig. 1-1 illustrates the basic elements that comprise the TX-2 computer and Fig. 1-2 shows the principal registers and transfer paths. The basic elements are: the Memory Element (ME), the Arithmetic Element (AE), the Program Element (PE), the In-Out Element (IOE), the Exchange Element (EE), and the Control Element (CE). Not shown are the various power supplies, coders, decoders, counters, alarms, indicators, logic nets, and other computer elements that contribute to the over-all system design.

1-6 BASIC PRINCIPLES OF OPERATION

Programs and data are usually read into the Memory Element via a paper tape reader. The initial read-in programs are placed manually in the plugboard or toggle switch storage.

During the execution of a typical instruction in a program, the instruction is obtained from the Memory Element register addressed by the P register and placed in the N register in the Program Element. Here the instruction is interpreted. If an operand is required its address is determined and placed in the Q register. The operand is then obtained from the Memory Element and placed in the M register in the Exchange Element. The operation specified by the instruction is then performed upon the operand. The final result is either: (1) left in one of the registers of the AE, EE, or PE, or (2) transmitted to one of the IOE devices, or (3) placed back in the register of the ME from which the operand was obtained.

1-7 COMPUTER CHARACTERISTICS

Tables 1-1 through 1-3 summarize the general characteristics of the TX-2 computer. The actual significance of such terms as meta bit, two-phase clock, and 9-bit quarters will be explained in the chapters that follow.

1-8 COMPONENT LOCATION

A floor plan of the TX-2 computer installation is shown in Fig. 1-3, along with the physical location of the principal registers, memories, and in-out devices. The entire facility occupies approximately 1500 square feet, including a separate room for the air conditioner and the primary power supplies. Although the U Memory has not as yet been constructed, its future location is indicated.

TABLE 1-1

TX-2 CHARACTERISTICS

CENTRAL COMPUTER	
General	38-bit word length, including a memory parity check bit and a programming meta bit Parallel word transfers Synchronous timing 5 mc two-phase clock
Instructions	One single-address instruction per word Addresses are indexable; double indexing is possible Indirect addressing, on all instructions, can be iterated indefinitely.
Operands	Operands can be given a subword configuration based upon 9-bit quarters: One 36-bit operand — Four 9-bit operands Two 18-bit operands — One 27-bit and one 9-bit operand Most instructions specify a subword configuration, and the arrangement and number of the subwords used
Arithmetic	1's complement binary number system, fixed-point arithmetic From 200,000, 36-bit additions per second to 800,000, 9-bit additions per second From 50,000, 36-bit multiplications per second to 400,000, 9-bit multiplications per second From 14,000, 36-bit divisions per second to 200,000, 9-bit divisions per second Arithmetic instructions longer than one memory cycle can be executed concurrently with nonarithmetic instructions
Memory	S Memory: 65,536 registers of magnetic-core memory with 6.4-microsecond cycle time and 4.0-microsecond access time T Memory: 4,096 registers of magnetic-core memory with 4.4-microsecond cycle time and 2.0-microsecond access time 16 registers of toggle-switch memory 32 registers of plugboard memory 5 registers in the Arithmetic and Exchange Elements Each register, except the last five, has an additional meta (i.e., trapping) bit All core memory registers are parity checked Instructions and operands can be obtained concurrently Addressing for a total of 131,072 registers of memory provided

TABLE 1-1 - cont'd

TX-2 CHARACTERISTICS

CENTRAL COMPUTER	
Special Memories	X Memory: 64 register magnetic-core index register memory, 19-bit word, with 3.6-microsecond cycle time and 0.6-microsecond access time CF Memory: 32 register magnetic-film configuration memory, 10-bit word, with 0.8-microsecond cycle time and 0.2-microsecond access time
Area Requirements (approximately)	1500 square feet of floor space 20 kw of power 20 tons of air conditioning (primarily for S Memory)
Checking Facilities	Marginal checking Test programs

TABLE 1-2

TX-2 CHARACTERISTICS

IN-OUT SYSTEM		
Photoelectric Tape Reader (PETR)	200-2000 lines per second	
High-Speed Tape Punch	200 lines per second	
Display Scope	20 to 80 microsecond per point	
Lincoln Writer	Keyboard:	Up to 10 characters per second
	Tape Reader:	19 lines per second
	Tape Punch:	20 lines per second
	Printer:	10 characters per second
Xerox High-Speed Printer	Paper Feed:	2 to 4 inches per second
	Printing Rate:	2000 to 4000 characters per second
Datrac	Sample Rate:	approximately 25 kc

TABLE 1-3

TX-2 CHARACTERISTICS

MAGNETIC TAPE SYSTEM						
General	3 information channels, 2 tracks per channel, 400 bits per inch per channel 1 block mark channel 1 timing track (800 flux reversals per inch); recorded at fixed density, used to synchronize writing and to detect speed of tape					
Tape Speeds			Controlled Speeds			Full Bore
	Inches/sec:	35	60	150	300	320 - 1000
	Bits/sec (kc):	42	72	180	360	380 - 1200
	3 bit lines/sec (kc):	14	24	60	120	128 - 400
	Microsecond/9 bit byte:	214	125	50	25	23 - 7.5
Capacity	10^{10} bits in system 70×10^6 bits/reel 3/4-inch tape (now) (7200 ft. reel) 250×10^6 bits/reel 1-1/4-inch tape (future)					

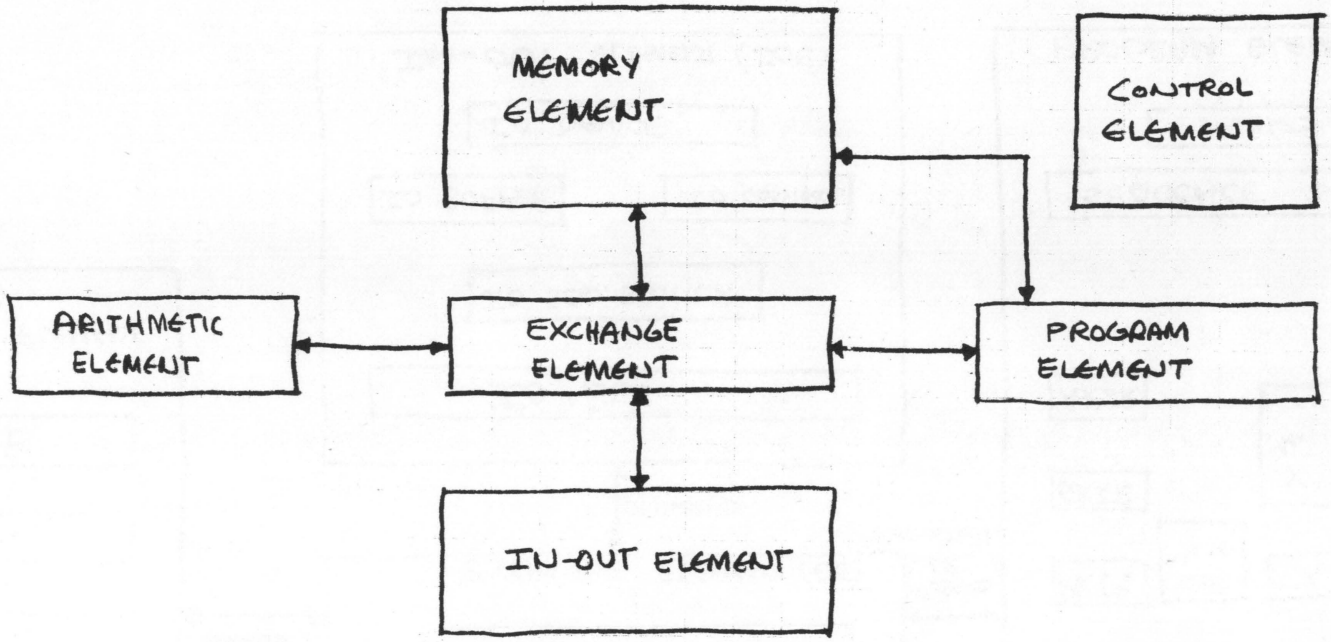


FIGURE 1-1. BASIC TX-2 ELEMENTS

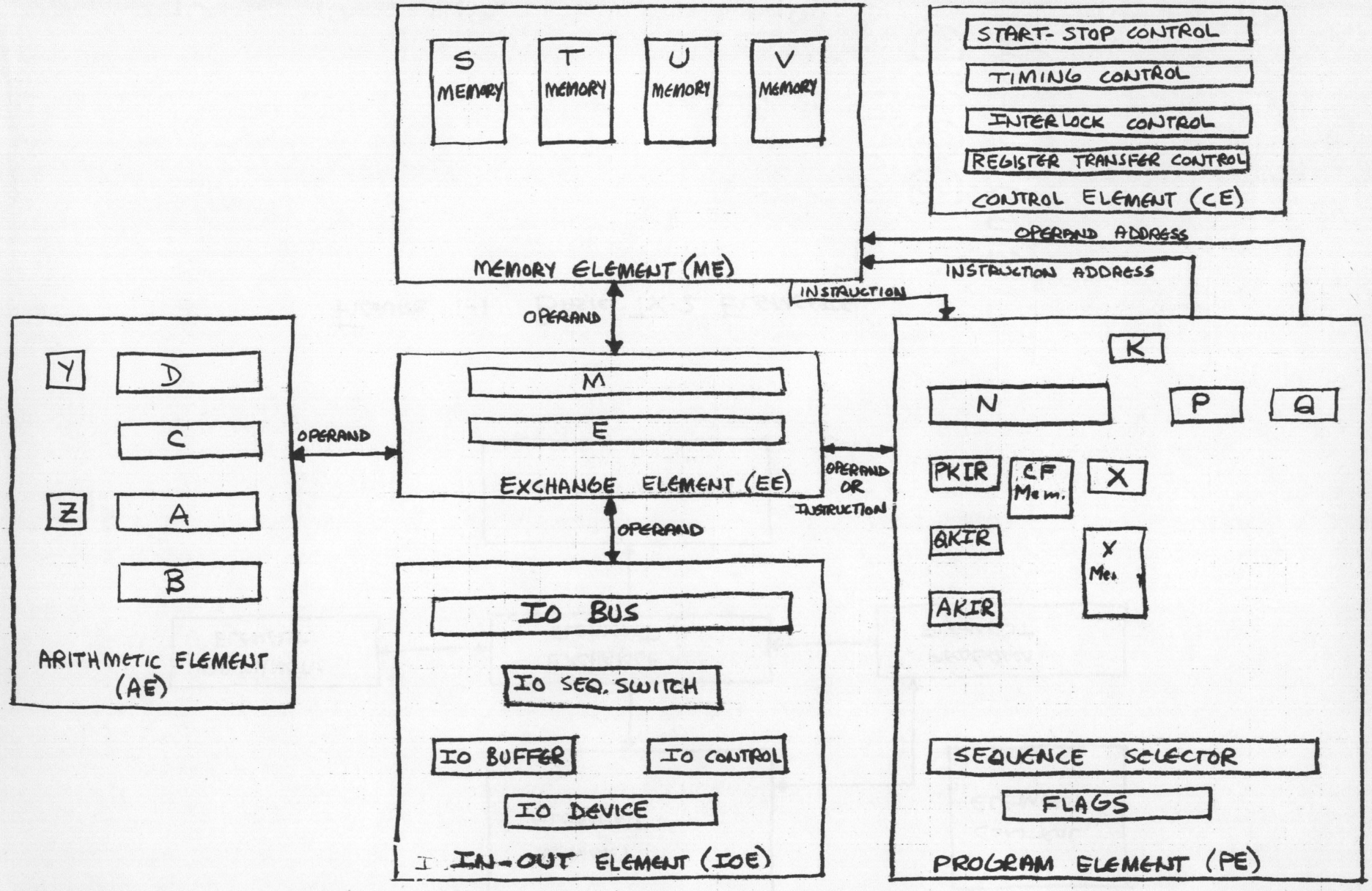
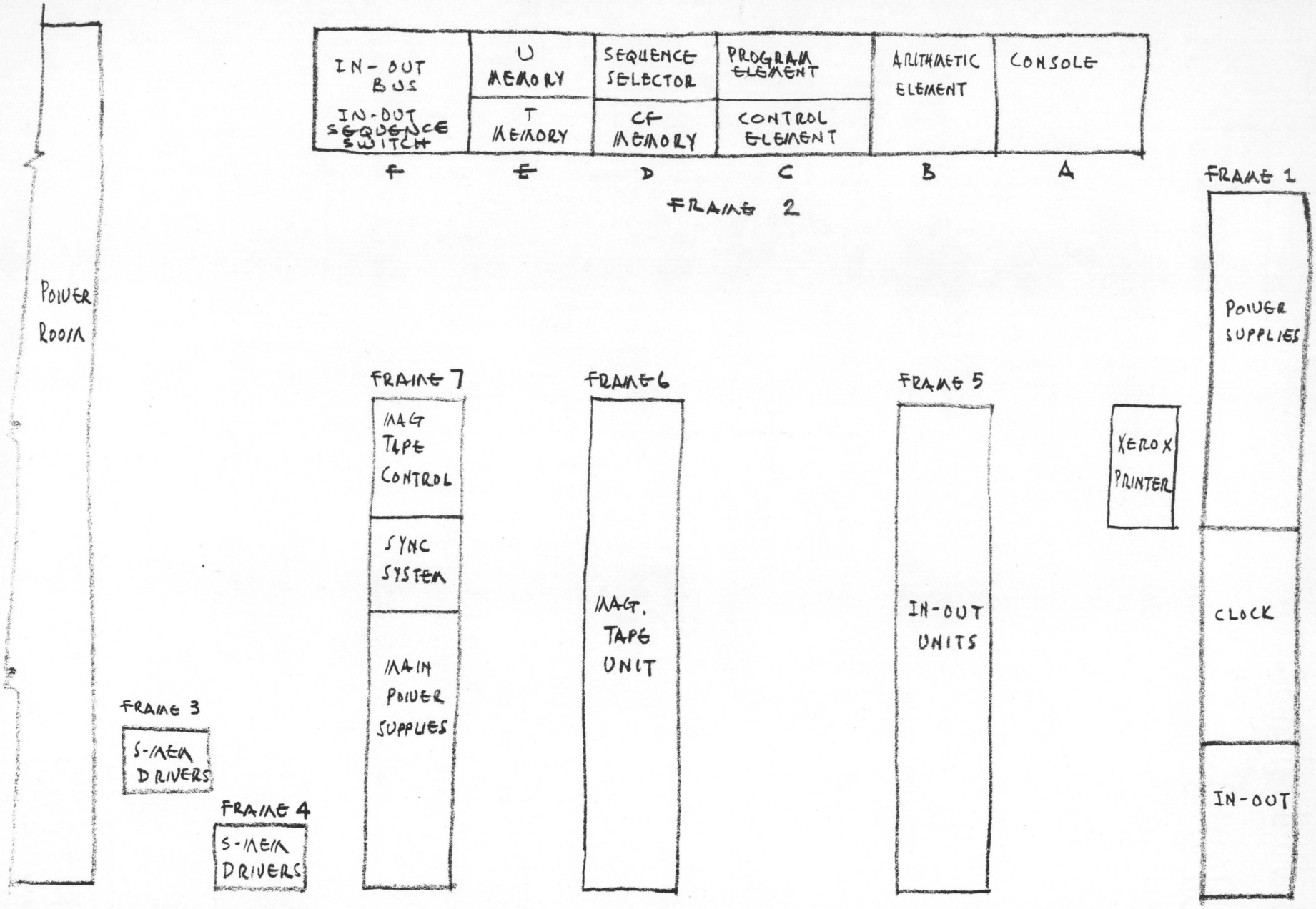


FIGURE 1-2 TX-2 SYSTEM BLOCK DIAGRAM, SHOWING PRINCIPAL REGISTERS AND TRANSFER PATHS



FLOOR PLAN - TX-2 COMPUTER ROOM (28 FT × 54 FT)

FIG 1-3

CHAPTER 2

FUNCTIONAL DESCRIPTION OF TX-2

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 2

FUNCTIONAL DESCRIPTION OF TX-2

2-1 INTRODUCTION

This chapter provides an introduction to the over-all operation of the computer. It will establish a useful perspective for reading the more detailed descriptions that appear in the succeeding chapters. It includes a brief functional description of five of the following six basic Elements that constitute the computer:

1) Memory Element
2) Exchange Element
3) Program Element
4) Arithmetic Element
5) In-Out Element
6) Control Element

The Control Element is not described in this chapter. It is sufficient to know that it is involved in all the activity that takes place in the other Elements.

The term "central computer" is used throughout the manual to refer to the computer as a whole, with the exception of the Memory Element and the In-Out Element. Thus the In-Out Element is described as "communicating with the central computer", etc.

The chapter begins by discussing the basic computer words, e.g., instruction words, operand words and deferred-address words. The more common interpretation of the bits in these words is established and the bit and quarter numbering scheme is described.

Since the basic cycle of the computer begins with an instruction word, the whole process by which an instruction word is obtained and interpreted is described. These processes primarily involve the Program Element. The subsequent activity that can occur as a result of interpreting the instruction word is then described. After this discussion, involving the Program Element, the chapter proceeds by discussing the other Element's in the computer. The basic processes that can occur in each Element are identified and described.

2-2 WORD STRUCTURE

2-2.1 GENERAL DESCRIPTION. The basic computer word is 36 bits long, plus a parity check bit and a word meta bit. The 36 bit word is divided into four 9-bit quarters.

The bit numbering scheme for the major flip-flop and memory registers is shown on Fig. 2-1. Note that the bits in each quarter, as well as the quarters themselves are ordered from right to left. When a double-number is used, the number to the left of the decimal point refers to the quarter in which the bit is found and the

number to the right of the decimal point refers to the specific bit within the quarter. Thus 2.6 refers to bit 6 in quarter 2.

As shown in Fig. 2-1 not all registers contain full words, nor do all full word registers contain a meta and/or a parity bit. Furthermore, the double-number scheme is not always used. For example, the $QKIR_{CF}$ register contains a 9-bit word plus a parity bit. These bits are numbered 1 to 10 from right to left with no reference to quarters.

2-2.2 INSTRUCTION WORD. The layout of an instruction word as it appears in the N register after being read out of memory is shown in Fig. 2-2(a). The content of N provides all the information for executing the instruction, including the information for obtaining an operand if one is required by the instruction. The content of N is actually disassembled and transferred to other registers for interpretation, but the discussion here will be limited to the content of the N register itself.

The instruction word in the N register can be broken up into groups of bits that have specific functions. Note that in nearly all instructions, all the bits which appear in the N register are interpreted, although not all the bits necessarily have the same functional interpretation in every instruction.

The base address is specified by the 18 Y bits (2.9 - 1.1). The final address is usually found by "indexing" the base address with the content of the X Memory index register selected by the six J bits (3.6 - 3.1), but is sometimes simply equal to the base address. The final address is usually the address of an operand or of the next instruction. It can also be the address of a deferred address.

Bit 2.9 is called the defer (*) bit.* When the defer bit is a ONE, the address is used to read a deferred address out of memory. The deferred address replaces the original address. When bit 2.9 is a ZERO, the address is used in the manner specified by the operation code.

The six OP bits (4.3 - 3.7) are used to specify the operation called for by the instruction word. There are currently 50 defined operation codes that can be specified by these six bits.

The five CF bits (4.8 - 4.4) are usually used to specify a computer configuration by specifying the address of a configuration word stored in the F Memory. This word is read out and its content used to restructure operand words. For example, the configuration information can be used to form two 18-bit subwords, one of which is inactive, from a 36 bit operand word.

* The defer bit is also frequently designated by a delta "Δ", as well as by an asterisk "*".

Bit 4.9 is called the hold (H) bit. The computer is designed to run with a variety of In-Out devices, for example a paper tape reader, punch, etc. Each of these devices requires its own computer program or program sequence. When the hold bit is a ONE, the computer can be forced to proceed from the current instruction to the next instruction in the same program sequence. For example, the computer can be forced to hold in the paper tape reader program sequence instead of changing to the punch sequence.

Bit 2.10 is called the "parity" bit and bit 4.10 is called the "word meta" bit. These bits have somewhat special purposes. The parity bit is used solely for checking for memory readout errors. The meta bit is ordinarily used as a kind of tag. By operating the computer with the "Trapping" program sequence turned "on", instruction words with meta bits set to ONE can be trapped and processed by the Trapping program sequence.

While the defer (*), operation (OP) and hold (H) bits are always interpreted as described above, the base address (Y), index (J) and configuration (CF) bits can be interpreted in quite different ways than those described. The interpretation is a function of the operation code being executed. The special interpretations will be discussed later.

2-2.3 DEFERRED ADDRESS WORD. This is a word that is used in the deferred addressing process described later in the chapter.

The layout of a deferred address word as it appears in the N register after being read out of memory is shown in Fig. 2-2(b). Bits 4.9 - 3.1 are not used. The function of the bits that are used is the same as the function of the corresponding bits in an instruction word.* Note that a deferred address can hence call for another deferred address when the defer bit is a ONE.

2-2.4 OPERAND WORD. When an instruction calls for an operand word, it is obtained from the Memory Element and placed in the M register. The layout of a memory operand word as it appears in the M register is shown in Fig. 2-3.

Depending on the operation and configuration specified by the instruction, the operand can be subjected to considerable manipulation as it is transferred through the computer. The configuration specifies which quarters of the operand word are to be used, and with which quarters of the central computer they are to be associated.

The parity bit is used to check the parity of the word as it appears in the M register, and the meta bit is used as a signal to the Trapping program sequence. Note that the meta bit of a word in the Memory Element can be altered only when the word is placed in the M register and even then it can only be altered by a particular instruction.

* The base address Y in a deferred address is always indexed.

2-3 PROGRAM ELEMENT

2-3.1 GENERAL DESCRIPTION. The primary functions of the Program Element are: (1) to determine what is the location in the Memory Element of the next instruction; and (2) to interpret the instruction when it is obtained and placed in the N register. Fig. 2-4 is a simplified block diagram of the Program Element.

The first function is determined by decisions made in the Sequence Selector. This unit determines what program sequence the next instruction will be taken from. During the execution of each instruction, the computer must constantly be deciding whether it will continue executing instructions in the current program sequence or change to a new program sequence. The logic for making these decisions is found in the Sequence Selector.

The function of interpreting the content of the N register requires a variety of registers, memories, coders, decoders and special circuits. Generally these can be grouped according to the bits in the N register they are interpreting. During the interpretation of a typical instruction word the components might be functionally grouped as follows:

<u>Interpreting J Bits</u>

J Decoder
X Memory
X Register

The J bits in the N register are decoded by the J Decoder to select a register in the X Memory. The X register serves as the memory buffer register.

<u>Interpreting Y Bits</u>

X Adder
Q Register

The X Adder can form the sum of the Y bits in the N register and the content of X register, or simply transmit the value of Y. This result can then be copied into the Q (or P or E) register.

<u>Interpreting OP Bits</u>

$PKIR_{OP}$
$QKIR_{OP}$
$AKIR_{OP}$ (during arithmetic instructions)

The OP bits are copied into the $PKIR_{OP}$ register and there interpreted. Further interpretation also can occur in the $QKIR_{OP}$ and $AKIR_{OP}$ registers.

Interpreting CF Bits

$PKIR_{CF}$
CF Decoder
F Memory
$QKIR_{CF}$ (and associated decoders)
$AKIR_{CF}$ (during arithmetic instructions)

The CF bits are copied into the $PKIR_{CF}$ register. The CF Decoder then selects a register in the F Memory. The content of the selected register is placed in the $QKIR_{CF}$ register and is interpreted there. Further interpretation also can occur in the $AKIR_{CF}$ register.

Interpreting H Bit

$PKIR_H$

The H bit is copied into the $PKIR_H$ flip-flop before being interpreted.

These distinctions are not rigid, however, since most of the components serve more than one function. Thus the X Memory, and the registers associated with it, are an integral part of the program sequence selection process, as well as being used in the interpretation of instruction words.

2-3.2 PROGRAM ELEMENT MEMORY SYSTEMS. The two memories in the Program Element are a 64-register 19-bit/register X Memory and a 32-register 10-bit/register F Memory. The X Memory holds the program counters used in the sequence selection process and the indices used to modify the base address of instruction and deferred address words. All of the 64 registers in the X Memory can be used as index registers; however, only 33 of them can be used to hold program counters. In addition to holding indices and program counters, the X Memory registers can also be used to store operands. Operands can be transferred from the X Memory to the Memory Element, and vice versa, over communication paths between the X Adder, Exchange Element and Memory Element. These paths also provide a means for loading an X register from a Memory Element register and for storing the content of an X register in a Memory Element register.

The F Memory holds configuration words. Paths between the $QKIR_{CF}$ register, Exchange Element and Memory Element provide a means for loading and storing F Memory registers.

Both memories are equipped with circuits which compute and check the parity of memory words. Also associated with each memory are a memory address decoder and a memory buffer. The decoder selects the memory register whose content is to be read into the memory buffer. During the memory write cycle, the content of the buffer register is written into the selected memory register.

The first register of both memories are made so as to always contain all ZEROES. Thus if register X_0 is selected for modification of the base address of an instruction, then effectively no modification occurs. Similarly, when register F_0 is selected for a configuration, then, as we shall see, only the normal 36-bit word computer configuration is realized.

2-3.3 SEQUENCE SELECTION. At some point just before the completion of an instruction word memory cycle, the Program Element must decide whether the next instruction is to be taken from the current program sequence or from some new program sequence. The decision is based on information from the Sequence Selector and on the "hold" and "dismiss" information found in the instruction being executed. This latter information is decoded from the $PKIR_H$, $PKIR_{CF}$ and $PKIR_{OP}$ registers, respectively.

Fig. 2-5 illustrates the parts of the computer involved in a sequence selection. The program sequences are numbered 0, 40, 41, ..., 77 (octal). These numbers correspond to the addresses of the registers in the X Memory which ordinarily store the program counter associated with each program sequence. Only the program counter of the current program sequence is not held in the X Memory; instead, it is held in the P register and its sequence number in the K register. As we shall see, the program sequences have a priority relationship among themselves.

The number of the current program sequence is stored in the K register. The Sequence Selector is informed of this number via the K Decoder. Another program sequence can request attention via the Sequence Selector if: (1) the flag of this new program sequence is up and requesting attention; (2) no other higher priority program sequence has its flag up; and (3) the current instruction is dismissing or the current program sequence has a lower priority number than the new sequence and the hold bit of the current instruction is in the "not hold" state. When these conditions are fulfilled, the computer will stop executing instructions in the current program sequence and start executing instructions in the new program sequence. This is called "a change of program sequence". The Sequence Selector provides the number of the new program sequence via the J Coder.

Fig. 2-6 illustrates schematically the procedure followed during the execution of each instruction which determines whether the next instruction will be from the current program sequence or from some other sequence. Note that this figure does not show <u>all</u> the details of the procedure; these will be covered later.

2-3.3.1 WAIT CYCLE. Sometimes the computer may be unable to proceed executing instructions in any program sequence. In such instances the computer goes through a wait cycle lasting 1.6 microseconds and then again tries to continue in some program sequence. The computer will repeat wait cycles until conditions are present for proceeding in some program sequence.

2-3.3.2 CHANGE OF SEQUENCE CYCLE. When the computer is able to make a change of sequence the number of the new program counter is placed in the J bits of the N register from the J Coder (Fig. 2-5). The new program counter is then obtained from the J-th register (X_J) in the X Memory and placed in the X register. The content of the P register and the X register are interchanged by transferring the content of P directly to the X register and, at the same time, copying the content of the X register to P via the X Adder.

Note that since the X Adder forms the sum of the content of the X register and the Y bits of the N register, the Y bits are ordinarily made all ZEROES during a change of sequence. However, if the number supplied by the J Coder is 00, then the content of X_J (i.e., of X_0) will always be ZEROES. In this case, the content of the special toggle switch register TSP is placed in the Y bits of N instead of ZEROES. This special register provides the programmer with a means of manually specifying the initial value of one particular program counter.

The content of the K register (K has been holding the old program counter number) is also interchanged with the J bits of the N register. After this, the old program counter, now in the X register, can be stored in its proper location (X_J) in the X Memory. Note that the J bits of the N register first contain the number supplied by the J Coder while the new program counter is being obtained from the X Memory, and afterwards contain the number from the K register while the old program counter is being stored in the X Memory. The state of the Sequence Selector changes to conform to this change of sequence by virtue of the change of the content of the K register. Certain information about the change of sequence is remembered in the E register of the Exchange Element in order to allow the programmer to analyze the sequence change.

2-3.3.3 PROGRAM SEQUENCE PRIORITY. The priority relationship of the program sequences is determined by the Priority Patch Panel plugboard. The programmer can arbitrarily specify various priority relationships. However, the plugboard is ordinarily connected with the order of priorities shown in Fig. 2-7. The names associated with the program sequences are also shown in the figure.

2-3.4 INTERPRETATION OF INSTRUCTION WORD DEFER AND OP BITS FOR BASIC SEQUENCE CYCLES. The address of the next instruction in the current program sequence is located in the P register. This is the register in the Program Element that addresses (selects) the register in the Memory Element whose content is read into the N register and interpreted as an instruction word. (See Fig. 2-8.) Note that since a read-out of the Memory Element is usually destructive, the word read out, in this case the

content of N, is parity checked and rewritten back into the Memory Element. Immediately after an instruction word is read out of memory (and rewritten), the content of the P register is increased (indexed) by one and the contents of the H, CF and OP bits are placed in the $PKIR_H$, $PKIR_{CF}$ and $PKIR_{OP}$ registers. This whole process is called the instruction word cycle.

2-3.4.1 DEFERRED ADDRESS WORD CYCLE. (See Fig. 2-8.) After an instruction word has been read into the N register, the value of its defer bit (*) is sensed. If the defer bit is a ONE, the instruction is called for a deferred address word. This base address Y is then placed in the Q register of the Program Element, and the content of the J bits is (temporarily) stored in $QKIR_{CF}$.

A deferred address word is then read out of the Memory Element, using the memory address in Q, into the N register. Only the base address (Y) and index (J) bits of this deferred address word will be interpreted by the Program Element. If the defer bit (*) in the base address is a ONE, the address of another deferred address word will be computed, by indexing the base address, and placed in Q. The process repeats until finally a deferred address word is read out whose defer bit (*) is ZERO. The indexed base address is again computed, but this time it replaces the Y bits in the N register. The original value of the J bits in the instruction word is then restored and the instruction is interpreted. Note that the interpretation is made with the hold, configuration and operation code information contained in the original instruction word that initiated the deferred addressing since this information was stored in the $PKIR_H$, $PKIR_{CF}$ and $PKIR_{OP}$ registers, respectively, all during the deferred addressing process. The purpose of the deferred addressing process is to compute a new set of Y bits that may be used with the original hold, configuration and operation code information.*

2-3.4.2 OPERAND ADDRESS CYCLE. When the defer bit is ZERO and the operation code calls for an operand, the final address is placed in the Q register. In this case, an operand is read into the M register from the register in the Memory Element addressed by the Q register. (See Fig. 2-9.) While the operand word is in the Exchange Element, it is altered according to the configuration specified by the instruction word. The configured operand can then be transferred to the Arithmetic Element, In-Out Element or Program Element. Or it can be simply kept in the Exchange Element for further processing.

* In effect, each new deferred address cycle simply substitutes new Y bits for the previous set, and then interprets the new defer bit.

If a "load" type instruction is executed, the original operand read out of memory will be rewritten back in memory. If a "store" type instruction is involved, the result of the operation is written back into memory in place of the original operand read out of memory.

2-3.4.3 JUMP CYCLE. When the defer bit is ZERO and the operation code specifies a jump instruction, the final address is placed in the P register if the jump conditions are satisfied. (See Fig. 2-8.) After either a jump cycle or an operand cycle, unless a change of sequence or wait cycle occurs, P addresses the Memory Element for another instruction word which is then read into the N register.

2-3.5 GENERAL INSTRUCTION WORD INTERPRETATION.

2-3.5.1 J AND Y BITS INTERPRETATION. Indexable type instructions add the content of an index register (X_J) to the base address Y of the instruction word. The sum is called the effective address, r. (See Fig. 2-4.) With this indexing system, it is possible to alter the address section of instructions before they are performed, without changing the instruction word in memory.

The selection of a particular index register is accomplished by decoding the six J bits in the N register instruction word. The content of the selected index register, X_J, is read into the X register, and then rewritten from the X register back into the specified index register of the X Memory. By means of the X Adder, the content of X_J is added to the base address, Y, to form the effective address, r.

It should be noted that the X Adder is part of the Program Element and that any additions performed here are not related to arithmetic operations performed in the Arithmetic Element. It should also be noted that index register X_0 is permanently wired so as to appear to contain ZEROES and is normally used when the base address of an indexable type instruction is to remain unchanged. In this case the effective address r is equal to the value of the Y bits.

2-3.5.2 OP BIT INTERPRETATION. The operation code of the instruction word is specified by the six OP bits of the N register (see Fig. 2-10). These six bits give a coded representation of the $2^6 = 64$ possible operations. Interpretation of the operation code is accomplished by means of three 6-bit registers designated $PKIR_{OP}$, $QKIR_{OP}$ and $AKIR_{OP}$.

The $PKIR_{OP}$ register is interpreted to determine the kind of instruction to be performed. Execution of the instruction may involve copying $PKIR_{OP}$ into $QKIR_{OP}$, and perhaps $QKIR_{OP}$ into $AKIR_{OP}$. Each of these OP registers is interpreted by two kinds of decoders: an "OP Decoder", which resolves the particular operation code to be performed; and an "OP class decoder", which determines certain class properties of the operation code. $PKIR_{OP}$ and $QKIR_{OP}$ are used by the whole computer, whereas $AKIR_{OP}$ is used primarily in the Arithmetic Element.

2-3.5.3 CF BIT INTERPRETATION. In configurable instructions, the five CF bits are copied into the $PKIR_{CF}$ register. (See Fig. 2-11.) They are then decoded and used to select a particular register in the F Memory. The content of the selected register is read into the F Memory buffer register, $QKIR_{CF}$, where it is interpreted.

Two of the bits in the $QKIR_{CF}$ register specify the form ("coupling" or "fracture") of the computer during the execution of the instruction. This subword form can be one 36-bit subword, two 18-bit subwords, one 27-bit subword and one 9-bit subword, or four 9-bit subwords (see Fig. 2-12). These subwords can be formed simply by coupling together the quarters of the data registers in various combinations during data processing.

Four bits in the $QKIR_{CF}$ register are used to specify the "activity" of the subwords. Each one of these four bits corresponds to a quarter of the data registers. When one of these bits is ZERO, then the associated quarter is "active"; when one of these bits is ONE, then the associated quarter is "inactive" or "latent".

Since subwords consisting of more than one quarter have more than one activity bit associated with them, it is possible to have partially active subwords. Depending upon the kind of instruction being executed and the direction of information flow (to or from the Memory Element), another process called "sign extension" occurs. In sign extension, the sign bits of active quarters are extended left to fill adjacent inactive quarters within subwords. This occurs in the Exchange Element as information flows from the Memory Element.

Finally, the remaining three bits in $QKIR_{CF}$ cause a "permutation" of the quarters of the operand words as they are passed through the Exchange Element. Only 8 of the 24 possible permutations are realized.

The interpretation of the CF bits in any instruction word is carried out after the N_{CF} bits are copied into the $PKIR_{CF}$ register. This permits deferred address cycles to occur, after an instruction word has been read out of the Memory Element and before the instruction is interpreted, without losing the value of the specified CF bits.

Some configurable-type instructions which use the Arithmetic Element also make use of a further configuration register, $AKIR_{CF}$. This register is supplied only with coupling and activity information from $QKIR_{CF}$, since no permutations are performed in the Arithmetic Element. In addition, all subwords used in the Arithmetic Element are forced to be fully active or completely inactive. Thus, an 18-bit subword originally specified with only one quarter active will appear to the Arithmetic Element to have both quarters active.

2-4 MEMORY ELEMENT

2-4.1 GENERAL DESCRIPTION. The Memory Element contains four physically separate memories. Each memory is a complete unit containing all the circuitry needed for the operation of the unit except for the memory address and memory buffer registers. The P and Q registers in the Program Element serve as memory address registers, and the N register, in the Program Element, and the M register, in the Exchange Element, serve as memory buffer registers.

The basic organization of the Memory Element and the information paths to and from the Memory Element are illustrated in Fig. 2-13. For simplicity the connections to the Control Element and the read-write control logic are omitted. The V Memory is also shown as a single unit.

The P register is used to specify the address in the Memory Element of instruction words only. Such words are read out of the selected memory and strobed into the N register. They are then rewritten (if the read out was destructive) back into the memory register. The Q register specifies the address either of deferred address words or of operand words. Deferred address words are strobed into the N register, and operand words into the M register.

2-4.2 MEMORIES. The four memories in the Memory Element are called the S Memory, T Memory, U Memory and V Memory. The characteristics of each of these memories is given in Table 2-1.

The S Memory is a 65,536 register magnetic core memory which uses magnetic core switches and vacuum tube drivers to select the register in the memory specified by the memory address register. As a result the access time to this memory is rather long compared to the other memories.

The T Memory is a 4096 register magnetic core memory. All of its circuitry uses transistors.

The U Memory has not yet been built, but will essentially be a copy of the T Memory.

The V Memory contains all the miscellaneous storage registers in the computer which can be addressed by a programmer. One part of this memory, called the V_{FF} memory, consists of the four 36-bit flip-flop registers in the Arithmetic Element and the E register in the Exchange Element. The transfer paths used when these registers are selected differ from the ones normally used. In the case of V_{FF}, information is transferred from or through the E register to the M and N registers.

The remainder of the V Memory, called the $V_{\overline{FF}}$ Memory, consists of various "fixed" registers, such as toggle switches and plugboards. These registers are fixed in the sense that a programmer cannot change the content of any of these registers by using "store" type instructions to store the content of central computer registers in them. Usually, however, there are other methods for changing the content of these registers, as for instance by manually changing the position of toggle switches in the toggle switch registers. This $V_{\overline{FF}}$ Memory contains a shaft encoder register. The content of this register is changed by rotating 4 shafts whose angular positions are each digitally encoded as 9 bit numbers. The V_{FF} Memory also includes a 36-bit real time clock which counts at a 100 kilocycle - per-second rate.

Each word in the S, T and U memories contains, in addition to the 36-bit "word" used by the central computer, a parity check bit. When a word is written into memory this bit is always made to have a value that will make the parity of the entire word odd.

Each word in the S, T, U and $V_{\overline{FF}}$ memories also contains a meta bit. The meta and parity bits are not included in the normal memory word interpretation process. This bit is instead used to tag, or mark, the word in which it is found. It is used either by the Trapping Sequence or by the one instruction (SKM) which can alter this bit.

Thus, all the memory registers in the S, T and U memories have 38 bits; those in the $V_{\overline{FF}}$ Memory have 37 bits; and those in the V_{FF} Memory have 36 bits.

2-4.3 MEMORY REGISTER SELECTION. The process of selecting a memory register and reading its content into a buffer register is initiated by the Control Element. The Control Element decides that a certain kind of word in memory is required and then starts the necessary read-write memory cycle to obtain it.

The memory register selection occurs as follows. The Memory Address Selector examines the content of either the P or Q memory address register, depending on the kind of memory word cycle called for. It examines bits of the address register successively from left to right. The first bits examined determine in which of the four memories contains the word and the remaining bits determine the register in this memory.

The Memory Address Selector routes the memory register selection bits in P or Q to the address decoder of the selected memory. It also serves to control the duration of the read-write cycle, since this depends on the particular memory selected. This unit is also used to route the content of the selected memory register into the desired memory buffer (M or N), during the memory read-out; and back through the inhibit selector to the memory register, during the memory rewrite.

The details of the memory register selection process are further described in Chapter 4 and 12.

2-4.4 MEMORY OVERLAPPING. Since there are two memory address registers and two memory buffer registers, in addition to the four memories in the Memory Element, it is sometimes possible to perform two memory read-write cycles simultaneously.

During an operand word memory cycle the Q register selects a memory register whose content is strobed into the M register. The memory cycle is completed when the content of the M register is rewritten back into the selected memory register. After such an operand cycle the computer will usually perform an instruction word memory cycle. (The operand cycle can be followed by a change of sequence cycle.) If an instruction cycle does follow, then the P register holds the address of the memory register containing the next instruction. Since the N register is usually not used during operand memory cycles, the computer will attempt to perform this instruction word memory read-write cycle while the operand cycle is still being performed.

A variety of conditions can inhibit the instruction word memory cycle until the operand cycle is finished. While we shall not be concerned with most of them here, one condition of note is whether the desired instruction word is in the same memory as the previous operand word selected by the Q register. If this is not the case, i.e., if Q selects one memory to obtain an operand word and P selects a different memory to obtain an instruction word, then, assuming all the other miscellaneous conditions are fulfilled, the two memory cycles will proceed simultaneously. Note that there is no restriction on which memories the operand and instruction word are stored in, except that, if overlap is allowed to occur, they must be stored in different memories. Note also that the instruction word cycle, if it uses the T Memory for example, can finish before the operand word cycle if the operand cycle uses the S Memory. This can happen, even though the instruction word cycle begins after the operand word cycle, because the T Memory read-write cycle time is much shorter than the S Memory cycle time.

2-4.5 MEMORY SPEED. The times listed in Table 2-1 for the memory cycle time are the minimum times only. These times usually occur when an instruction or deferred address word is obtained from the memories. If an operand word is obtained from the memory, the cycle time can be up to two or three microseconds longer when the operand word in the memory register is changed by the instruction. These timing situations are covered in Chapter 9.

2-5 EXCHANGE ELEMENT

2-5.1 GENERAL DESCRIPTION. Nearly all information transmitted to the two memory buffer registers M and N, except information coming directly from the Memory Element, passes through the E register in the Exchange Element. Also, nearly all information transmitted to the Program, Arithmetic and In-Out Elements passes through the E register. This register thus serves as a bus for most of the information transfers in the central computer. Fig. 2-14 illustrates the central position of the Exchange Element in the computer.

Information passing through the Exchange Element can be transformed in various ways. Many of these transformations are controlled by a configuration specified by the instruction which causes the transfer. The information for configuration control is obtained from the F Memory, as described earlier in the chapter. Transformations can also occur in the Exchange Element as part of the inherent execution logic of an instruction.

These transformations are realized for the most part by providing each quarter of each register in the Exchange Element with its own logical control. This is indicated, for example, in Fig. 2-15 by the separate paths between each pair of corresponding quarters of the E and M registers. Each quarter of the E register can also be separately cleared and/or complemented. In addition, each quarter of the E register is connected to every other quarter of the E register by permutation paths, as shown in Fig. 2-15. Each of these paths is also separately controlled by the computer logic.

This control of the paths among the quarters of the registers within the Exchange Element is essential for the realization of much of the power and flexibility of TX-2 instructions.

2-5.2 OPERAND CONFIGURATION IN THE EXCHANGE ELEMENT

2-5.2.1 PERMUTATION. Fig. 2-11 illustrated the decoding of a configuration word in the $QKIR_{CF}$ register during the execution of instructions which use a configuration.

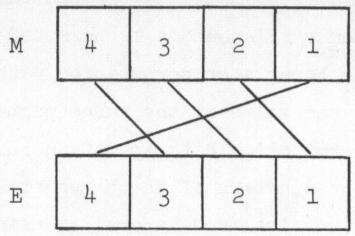
As an operand word is transmitted through the Exchange Element on its way to or from the Memory Element it is permuted in the E register. This permutation uses the E register permutation paths. The form of the permutation is determined by the value of the three permutation bits in the $QKIR_{CF}$ register. The permutation bits specify a route between the quarters of the M register and the quarters of the E register. The route is traversed by a combination of a vertical and lateral transfer as shown below. Since the permutation actually occurs in the E register one permutation must occur when a word is transmitted from the Memory Element through the Exchange Element to other parts of the computer, and the <u>inverse</u> permutation when a word is transmitted in the reverse direction.

In Fig. 2-16 the permutations are shown which can occur in the E register while an operand word is being transmitted from the Memory Element to the central computer. The graphic notation illustrates the "effective" path from the M register to the E register. These paths are realized by transferring the operand word from M to E and then permuting the word in E. (M is always assumed to be at the upper end of the arrows in such a graph, and E at the lower end, regardless of the direction of the arrows.) The recombination of the quarters of an operand word in the E register after the permutations is also shown. The quarter arrangement shown is that found after a memory operand word has been brought into a register (e.g., the A register) in the central computer.

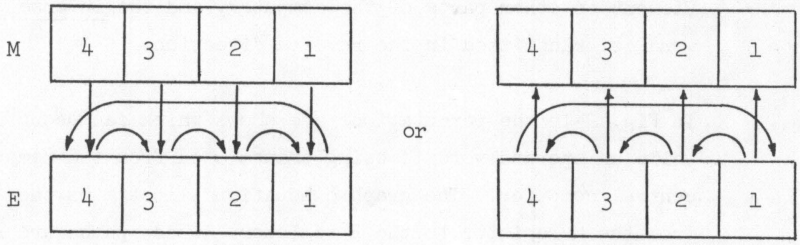
In Fig. 2-17 the inverse permutations are shown which can occur in the E register while an operand word is being transmitted from the central computer to the Memory Element. In this case the inverse permutations are graphically represented by arrows from E to M. The recombinations of central computer register quarters in a Memory Element register are also listed.

Note that for each permutation listed in Fig. 2-16, there is a corresponding inverse permutation listed in Fig. 2-17. Note also that some permutations are their own inverses, i.e., permutations 000, 010, 100 and 101. This means that such permutations and their inverses are realized by the same set of paths among the quarters of the E register.

The graphic notation portrays the effective path between the quarters of the E register and those of the M register. The value of this notation is that it is independent of the permutation of information that is actually occurring. Thus the graphic notation (with one quarter emphasized)



can be used to represent either of the following two actual permutation processes:



DIRECT PERMUTATION INVERSE PERMUTATION

Hence the graphic notation can be used to simply represent the permutation specified by the configuration without regard for the kind of instruction being executed.

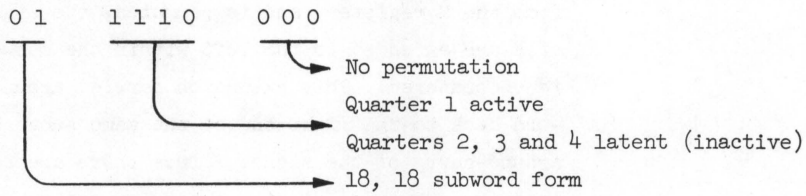
2-5.2.2 ACTIVITY. The four activity or latency bits in the configuration word specify the quarters of the registers in the central computer which shall be active during the execution of an instruction. Each bit is uniquely associated with a quarter. Fig. 2-18 shows the sixteen possibilities that can be realized. These activity bits act as a mask on the quarter of the register, permitting information to be transmitted through the quarter of the E register only if the corresponding activity bit is a ZERO. Note that a quarter is latent, i.e., inactive, when the corresponding activity bit is a ONE.

The graphic notation used on Fig. 2-18 indicates the latency of a quarter by removing the arrowhead from the arrow. (In Fig. 2-19 only permutation 000 is illustrated.)

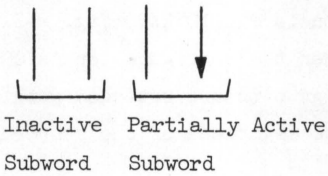
2-5.2.3 SUBWORD FORM. As described earlier, the left two configuration word bits specify the subword form of the operands in the central computer registers. These subword forms are illustrated in Fig. 2-19. The graphic notation used in the figure makes use of an under-bracket to show the quarter groupings that form the subwords.

The subword forms primarily influence the manner in which arithmetic operations are performed. Thus the carry process that occurs in addition would be influenced by the subword form, since carries would not occur between quarters that were not in the same subword.

2-5.2.4 PARTIALLY ACTIVE SUBWORDS. These are subwords in which one or more of the quarters (but not all) are made inactive by the configuration specified by the instruction. For example, a programmer can specify a configuration which has the binary value,

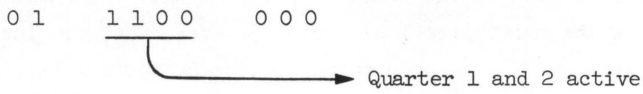


The graphic notation for this configuration would look as follows:



This notation indicates that two 18-bit subwords are formed in the central computer register, but that only quarter 1 is active. It is convenient to have the central computer work at all times with fully active subwords. This is realized automatically in the computer by "activity extension" and "sign extension". This allows the programmer to specify any of the possible activity and subword form combinations without worrying about how arithmetic operations are performed upon partially active subwords. However, use is made of the fact that a subword is only partially active while the subword is in the E register.

Activity Extension. The computer "extends" the activity of partially active subwords to make the whole subword active. In the example the effect of activity extension is, as far as subword form is concerned, the same as if the programmer had specified the configuration



Note, however, that there is now ambiguity about the content of the quarters made active by "activity extension".

Sign Extension. This process extends the sign bit of the operand in the active quarters of partially active subwords into the inactive quarters. This gives a meaning to the inactive quarters of a subword, to be used in arithmetic operations, when the subword is made fully active by activity extension.

After an operand arrives in the E register from the Memory Element (i.e., from the M register) and is permuted, the sign bits of the active quarters of E are extended to the left within the subwords until an active quarter is encountered. This extension carries around from the left end of a subword back to the right end of the same subword, i.e., there can be an end-around-carry of the signs. Since there are no active quarters in wholly inactive subwords, no sign extension occurs in such subwords.

Fig. 2-20 illustrates the way in which the sign would be extended if the configuration is 011110XXX. Fig. 2-21 shows the more complex situation resulting when the configuration is 000101XXX. In this latter case two different sign bits are extended within the same subword, one of them around to the right end of the subword.

It should be realized that activity and sign extension occur in the E register only when an operand is being brought from memory to the central computer. Since there are no possible interquarter transfers of information in a Memory Element register, the subword form is ignored when an operand is transmitted from the central computer to the Memory Element. Only the permutation and activity bits of the configuration are used in this case.

2-6 ARITHMETIC ELEMENT

2-6.1 GENERAL DESCRIPTION. Most of the arithmetic and logical operations in TX-2 are carried out in the Arithmetic Element. (Some of these operations can also be performed in the Exchange Element and the Program Element.) Since many of these operations are complex and time consuming, the Arithmetic Element is designed to operate independently of the rest of the computer once it has started performing some operation. Thus, a multiplication can be executed in the Arithmetic Element while the rest of the computer proceeds with the execution of the instructions that follow the multiplication. Note, however, that the instructions that follow are inhibited if they also require the use of the Arithmetic Element.

2-6.2 ARITHMETIC ELEMENT REGISTERS. Fig. 2-22 illustrates the registers and information transfer paths in the Arithmetic Element.

The A register serves as the accumulator, i.e., this register usually contains one of the operands involved in an arithmetic operation (the other operand comes from the Memory Element via the Exchange Element), and the result of the operation is usually left in the A register. The B register usually serves as an extension on the right end of A register. It is used this way during multiplication, for example, when a double register length product is formed. The primary purpose of the C register is to hold the partial carries which are generated when an addition is performed. The D register usually holds the operand brought from memory.

The content of any of these registers can be stored in a memory register, or be replaced by the content of a memory register. The four registers are also addressable as part of the V_{FF} memory in the Memory Element. The variety of means of access to these registers provides considerable programming flexibility. Note, however, that all communication with these registers from outside the Arithmetic Element is through the E register.

Each of the four registers is divided into four 9-bit quarters. A Z overflow flip-flop is associated with each quarter of the A register. These overflow flip-flops are used to remember whether an arithmetic overflow occurred during a previous arithmetic instruction (e.g., during an addition). The Z flip-flops are also used for sign control during some instructions.

A Y flip-flop is associated with each sign quarter of the D register. These four Y flip-flops are altered only when an operand is placed in the D register from the E register. They remember the original sign of the memory operand word placed in D after the content of D has been altered during the execution of an instruction.

2-6.3 INFORMATION PATHS IN THE ARITHMETIC ELEMENT. The only simple register transfer paths within the Arithmetic Element are between the A and B registers. These are "jam" transfer paths which copy the content of one register into the other.

The contents of the A and B registers can be rotated either to the left or to the right. This rotation can also occur when the B register acts as an extension of the A register, i.e., the content of the AB register can be rotated to the left or right. These shift paths are illustrated in Fig. 2-22.

The other paths in the Arithmetic Element involve transformations of the information being transferred. The exclusive OR of the contents of the D and A registers can replace the content of the A register. The result left in the A register by this operation is the "partial sum" of the contents of the two registers.

The logical product of the A and D registers can be placed in the C register. This operation forms the "partial carry" of the contents of the A and D registers.

Both the partial sum and the partial carry are formed simultaneously during an addition in the Arithmetic Element. The "complete sum" of the original contents of A and D is then formed in the A register by forming the "complete carry". The complete carry circuit forms the complete sum by combining the partial sum and partial carry in the A and C registers and placing the result in A.

There are other transformations in the Arithmetic Element that involve a partial sum and partial carry. For example, these quantities are combined in the partial carry and shift right logic called "multiply step". In "multiply step" the partial carries in the C register are carried only one bit position to the left in both A and C. The entire result, in both the A and C registers, is then shifted one place to the right. The multiply step transformation is used to speed up the multiplication algorithm. It is described in detail in Chapter 14.

The quarters of the D register can also act as counters. There are circuits which add ONE to the content of each quarter of the D register each time certain other operations are performed in the Arithmetic Element.

2-6.4 OPERATIONS IN THE ARITHMETIC ELEMENT. The operations which can be performed in the Arithmetic Element are:

Addition. The memory operand is placed in the D register and the partial sum and carries are formed. The complete carry is then placed in the A register, forming the desired sum. An overflow, if it occurs, is simultaneously placed in Z.

Subtraction. This operation is identical to addition except that the memory operand in D is complemented before the addition occurs.

Multiplication. The double length product of the memory operand, which is placed in D, and the original content of A is placed in the AB register. The product is formed by first placing the content of A in B and clearing the A register. A multiplication cycle is then repeated as many times as there are bits in the operand subword. The cycle consists of adding the content of D to the content of A if the least significant bit in B is a ONE. The content of AB is then shifted to the right one place. The addition consists of forming the partial sum and partial carry in A and C, respectively, and then doing a multiply step. A shift right of one place in A, B and C occurs when the multiply step is performed, and the partial carries are reduced so that further partial sums can be formed. After a sufficient number of repetitions of the cycle the complete carry is formed in A and the result in AB is then the desired product.

Division. The operand from memory is divided into the content of the AB register. This process is the inverse of multiplication and leaves the quotient in A and the remainder, if any, in B.

The cycle used here involves forming the difference between the contents of D and A in A, and then shifting the content of AB left one place. The sign bits of each difference formed in A are shifted into B to form the quotient. The content of D will be complemented, if necessary, at the beginning of each cycle so that it always differs in sign from the content of A. An addition is then performed.

If an overflow occurs initially the Z flip-flops are set. After the last cycle the remainder is in the A register. At the end of the instruction the contents of A and B are interchanged.

Shift. The content of A, AB (including the overflow in Z) or B can be arithmetically shifted either to the left or to the right. The number of places shifted is determined by the memory operand word placed in D. The count circuit on D is indexed once for each shift.

Cycle. This operation is identical to Shift, except that a pure rotation of the content of the selected register occurs, and that the overflow is not involved.

Normalize. The content of A or AB (including the overflow in Z) is arithmetically shifted until the sign bit in A and the bit to the right of the sign bit differ. The number of shifts to the right (left) which occur is added to (subtracted from) the memory operand placed in D.

Tally. The number of bits which are ONES in a memory operand word placed in the A register is added to the original content of D.

Logical Operations. The logical "inclusive OR", "exclusive OR", and "AND" of a memory operand and the original content of A can be formed in A by these operations.

These are the basic operations which can be performed in the Arithmetic Element. Variations on the simple process of loading and storing the contents of registers in the Arithmetic Element can also be performed. These operations are covered in Chapter 16.

2-6.5 CONFIGURATION IN THE ARITHMETIC ELEMENT. Subword form and activity in the Arithmetic Element, which constitute the configuration of the Arithmetic Element, are determined by the content of the $AKIR_{CF}$ register; just as configuration in the Exchange Element is determined by the content of the $QKIR_{CF}$ register. By having configuration

($AKIR_{CF}$) and operation code ($AKIR_{OP}$) registers of its own, the Arithmetic Element becomes independent of the $QKIR_{CF}$ and $QKIR_{OP}$ register. In this way an Arithmetic Element instruction can continue to be performed even though a new instruction is begun which fills $QKIR_{CF}$ and $QKIR_{OP}$ with new configuration and operation information, respectively.

However the activity bits in $AKIR_{CF}$ do not represent mere copies of the activity bits in $QKIR_{CF}$. Instead $AKIR_{CF}$ is set up so that activity is extended in the Arithmetic Element by the process described earlier. In this way subwords in the Arithmetic Element are made either wholly active or wholly inactive.

Note that the permutation information contained in the configuration word is not used by the Arithmetic Element, but only influences the memory operand as it passes through the Exchange Element to the Arithmetic Element. Note also, that the sign extension process is completed in the Exchange Element, before the memory operand is transferred into the Arithmetic Element.

2-6.6 QUARTERED PATHS IN THE ARITHMETIC ELEMENT. All the registers in the Arithmetic Element and also all the lateral information transfer paths in it are quartered. These lateral transfer paths are the shift paths in, and between, the A and B registers, and the carry paths. The quarters of the shift paths can transmit information either to the left or right into each quarter of the A and B registers. The quarters of carry circuit can transmit information only to the left into each quarter of A.

The subword form specifies the connection between the quartered segments of these paths. The actual connections are realized by coupling units, as illustrated in Fig. 2-23. Fig. 2-24 illustrates how the complete shift and carry paths can be formed by connecting the outputs of the quarters of the registers (or carry circuits) in various ways. The subword form specified by the configuration bits determine which one of the several inputs to each coupling unit actually is transmitted through the unit. Since the subword forms are limited to the ones illustrated in Fig. 2-19, not all conceivable input connections to coupling units are realized. For example, a carry coupling unit receives an input either from the quarter immediately to the right, or from the quarter farthest to the left in the same subword.

It should be realized that these coupling units in the shift and carry circuits are the sole means used to realize the variety of subword forms during the execution of Arithmetic Element instructions. Fig. 2-25 illustrates these subword forms as they are reflected in the apparent structure of the Arithmetic Element register. A programmer can effectively use several Arithmetic Elements simultaneously when he specifies a configuration with subwords less than 36 bits in length. Fig. 2-25 shows these multiple Arithmetic Elements and the corresponding operand word structure. The activity bits of course also give the programmer the ability to control just which

of these Arithmetic Elements are actually used during the execution of an instruction. For example, one, two, three or four 9-bit additions can be simultaneously performed in any of the 9-bit Arithmetic Elements illustrated.

2-7 IN-OUT ELEMENT

2-7.1 GENERAL DESCRIPTION. The principal paths for transmitting data into or out from TX-2 are in the In-Out Element. These paths are routed over an In-Out Bus. The bus effectively connects a variety of input and output devices with the E register in the Exchange Element. The bus also transmits signals which enable the central computer to control the operation of the In-Out devices. Included in these control signals are central computer clock pulses. These clock pulses are used to synchronize signals generated by the In-Out device with operations in the central computer.

Each In-Out device has its own program, called a program sequence, stored in the Memory Element. As described earlier, there are 33 different program counters stored in the X Memory in the Program Element. The content of each of these program counter registers, when transferred to the P register, addresses its own program sequence in the Memory Element. Instructions can be executed from only one program sequence at a time, i.e., only one program counter can actually be in use at any given time. However, certain instructions in a program can cause a change from one program sequence to another.

Thus, each In-Out device is uniquely associated with one of the 33 program counters. The number of the associated program counter is assigned to the In-Out device. Thus there can exist at most 33 In-Out devices. As shown in Fig. 2-7, there are currently about 16 such devices. The four highest priority program counters also have a (special) relationship with the In-Out Element, so that about 20 of the program counters are associated with the In-Out Element.

These 20 program counters are distinguished by the fact that the FLAG flip-flop for each program counter can be set (i.e., "raised") when the associated In-Out unit generates a raise flag signal. This signal indicates that, for some reason, the In-Out unit requires the corresponding program sequence to be performed by the computer. The computer does this as soon as this sequence becomes the highest priority sequence with a FLAG raised.

As stated before, there are basically two different situations which can initiate a change of program sequence. Either the program sequence, which is currently being performed, can initiate the change, or the In-Out unit associated with one of the 20 program counters can do this.

2-7.2 STRUCTURE OF THE IN-OUT ELEMENT. Fig. 2-26 is a simplified block diagram of the In-Out Element. It illustrates the structure of the In-Out Element and the connections between it and the central computer.

Each In-Out unit is divided into a number of packages. These packages include:

1) The In-Out device itself
2) A control box for the device
3) A sequence switch

Only the sequence switch is connected to the In-Out Bus.

2-7.3 IN-OUT BUS. At any given time data can be transmitted over the In-Out Bus between the E register and only one In-Out unit. These data transfers occur only during the TSD (TranSfer Data) instructions. The particular connected unit is the one associated with the current program sequence. It is selected by the decoder on the K register in the Program Element. The output of the K Decoder is sent out over the In-Out Bus to the In-Out units and only the K-th unit is allowed to transmit data over the bus.

The computer can also control the operation of In-Out units. This control occurs only during the IOS (In-Out Select) instruction. It is realized by transmitting the Y bits in the N register (this is a case of the Y bits being used for a special purpose) over the In-Out Bus to the specified In-Out unit. In this case the particular unit is selected by the decoder on the J bits in the N register. The output of the N_J Decoder is sent out over the In-Out Bus to the In-Out units and only the J-th unit receives the Y bits.

Synchronization and alarm control signals are also transmitted over the In-Out Bus between the Control Element of the central computer and the In-Out units.

2-7.4 SEQUENCE SWITCHES. The actual connection between an In-Out unit and the In-Out Bus is realized in the sequence switch of the unit. The data transfers are gated in the sequence switch by the K Decoder outputs. The Y bits from the N register are gated by the N_J Decoder outputs.

The sequence switches isolate the In-Out units from the In-Out Bus so that at most one unit is connected to a given set of information lines in the bus at one time.

Some units, particularly the ones for the special program sequences, consist only of sequence switches. (See Fig. 2-26.) These units are:

The Startover Sequence Switch (0) transmits raise flag signals to $FLAG_0$ whenever the Startover button on the control console is pressed. The associated program sequence is usually used to initially start the computer.

The Computer Alarm Sequence Switch (40) will raise $FLAG_{40}$ whenever a selected central computer alarm occurs.

The In-Out Alarm Sequence Switch (41) will raise $FLAG_{41}$ whenever an In-Out unit alarm occurs.

The Trapping Sequence Switch (42) will raise $FLAG_{42}$ whenever a selected set meta bit occurs in the M or N register.

All of these sequence switches contain some control logic which is governed by the IOS instruction. However, these special sequence switches are rather simple when compared with the sequence switches for the In-Out devices.

2-7.5 IN-OUT CONTROL BOXES. The control boxes control the operation of the In-Out devices. They also contain the In-Out buffers which hold the data flowing between the In-Out devices and the E register.

The Y bits in an IOS instruction are copied into control flip-flops in the control box of the In-Out unit selected by the IOS, and the contents of these flip-flops then determine the mode of operation of the In-Out devices. There are usually also some manual controls in a control box which allow the computer operator to influence the operation of the In-Out device.

The buffer register holds one character of the data transmitted to or from the device. The central computer can change or read this character in a buffer only by performing a TSD instruction in the program sequence associated with the device.

2-7.6 IN-OUT DEVICES. An In-Out device is some electrical, mechanical or optical, etc. device which either samples some external signal source or reads some data record and converts these inputs to characters of digital data for input to the computer, or, conversely, converts a character of data to an external signal or record.

Fig. 2-7 listed the names of the In-Out devices associated with some of the program sequences. A brief description of these devices follows:

Magnetic Tape (46). (Currently not installed.) This is the only input and output device. 9-bit data characters can be recorded or read while the magnetic tape is traveling in either the forward or reverse direction.

Miscellaneous Inputs (47). This device simply raises $FLAG_{47}$ whenever some selected external source generates a pulse.

Datrac (50). This device digitizes an external analogue signal. 11-bit data characters are formed by this process and transmitted to the computer.

Xerox (51). A high speed printer which can print 88 different 8-bit characters in two different sizes.

PETR (52). A photo electric paper tape reader. It reads six bit characters from punched paper tape.

Interval Timer (54). An 18-bit timer which can be reset by the computer. When the timer counts down to zero a pulse is generated which can either raise $FLAG_{54}$ or be sent to some other device.

Light Pen (55). A photo electric sensing device which raises $FLAG_{55}$ when the pen is held over a point displayed on a cathode ray tube (CRT) display (see below).

Display No. 2 (56). (Currently not installed.) A CRT point display tube similar to Display No. 1.

Display No. 1 (60). A CRT point display tube which can intensify any point in a 1024 X 1024 raster.

Random Number Generator (61). This unit generates nine bit numbers with the properties of a sequence random number. It uses a radioactive source.

Punch No. 2 (62). (Currently not installed.) This unit is similar to Punch No. 1.

Punch No. 1 (63). This unit can punch six bit characters with or without a seventh hole in paper tape.

Lincoln Writer Input No. 1 (65). This unit can read six bit characters into the computer. These characters can be generated either by a keyboard or a paper tape reader.

Lincoln Writer Output No. 1 (66). This unit can print six bit characters using a typewriter. The unit can also punch the characters on paper tape.

Lincoln Writer Input No. 2 (71). This unit is identical to Lincoln Writer Input No. 1.

Lincoln Writer Output No. 2 (72). This unit is identical to Lincoln Writer Output No. 1.

Plotter (74). A two-coordinate line plotter. The motion of the pen is controlled by specifying the coordinate position of the pen.

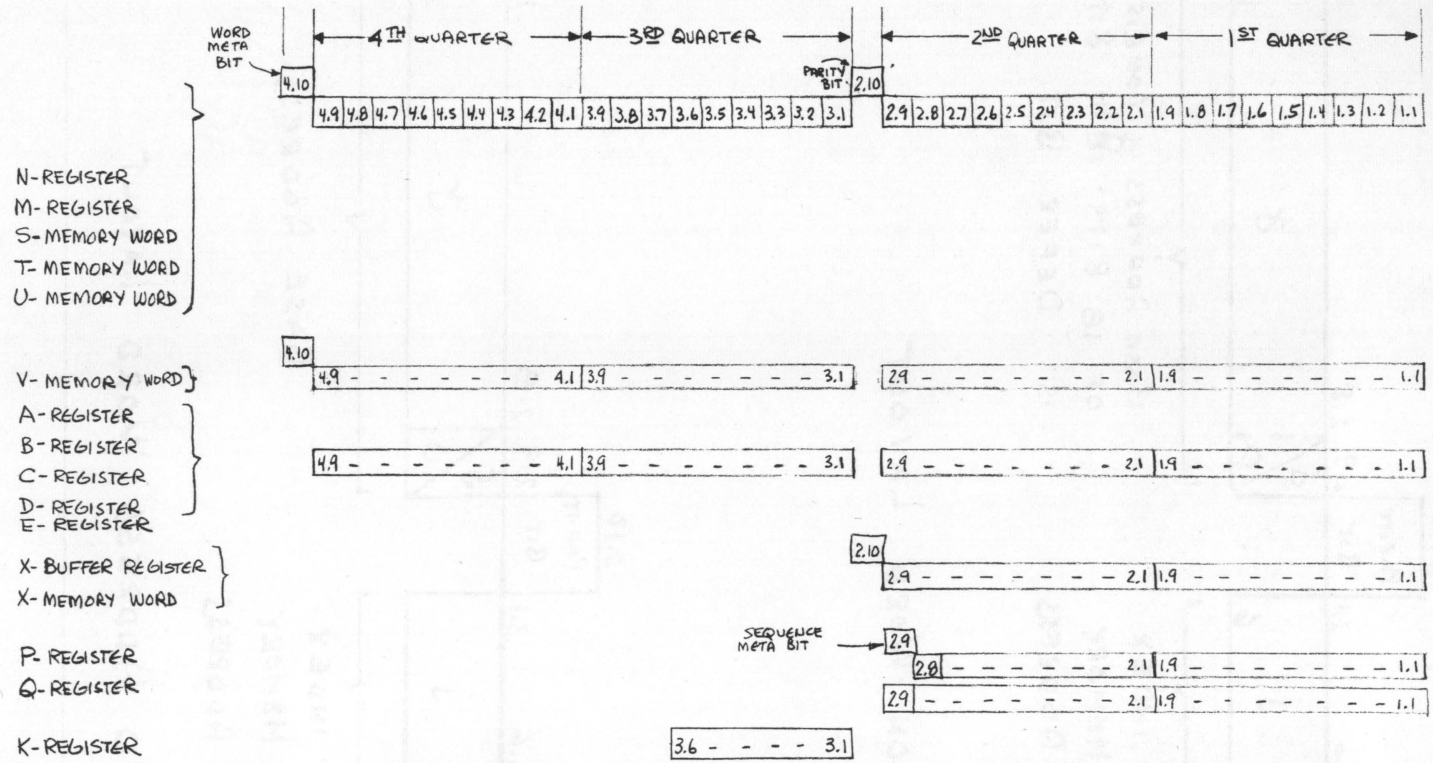
Miscellaneous Outputs (75). This device simply holds the value of the Y bits of an IOS. This information can then be used to control an arbitrary external device.

MEMORY	TYPE	ACCESS TIME (μsec)	CYCLE TIME (μsec)	REGISTERS	BIT WORD LENGTH
S	MAGNETIC CORE	4.0	6.4	65,536	38*
T	MAGNETIC CORE	2.0	4.4	4096	38*
U	MAGNETIC CORE	**	**	4096	38*
$V_{\overline{FF}}$	PLUGBOARD REGISTERS			32	37
$V_{\overline{FF}}$	TOGGLE-SWITCH REGISTERS			16	37
$V_{\overline{FF}}$	REAL-TIME CLOCK REGISTER			1	36
$V_{\overline{FF}}$	SHAFT ENCODERS REGISTER			1	36
V_{FF}	A REGISTER			1	36
V_{FF}	B REGISTER			1	36
V_{FF}	C REGISTER			1	36
V_{FF}	D REGISTER			1	36
V_{FF}	E REGISTER			1	36

* INCLUDES PARITY BIT

** CURRENTLY UNDEFINED

TABLE 2-1 MEMORY ELEMENT REGISTER



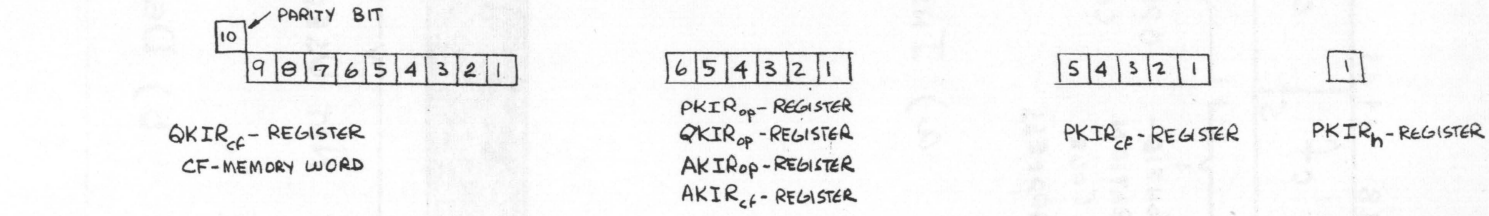
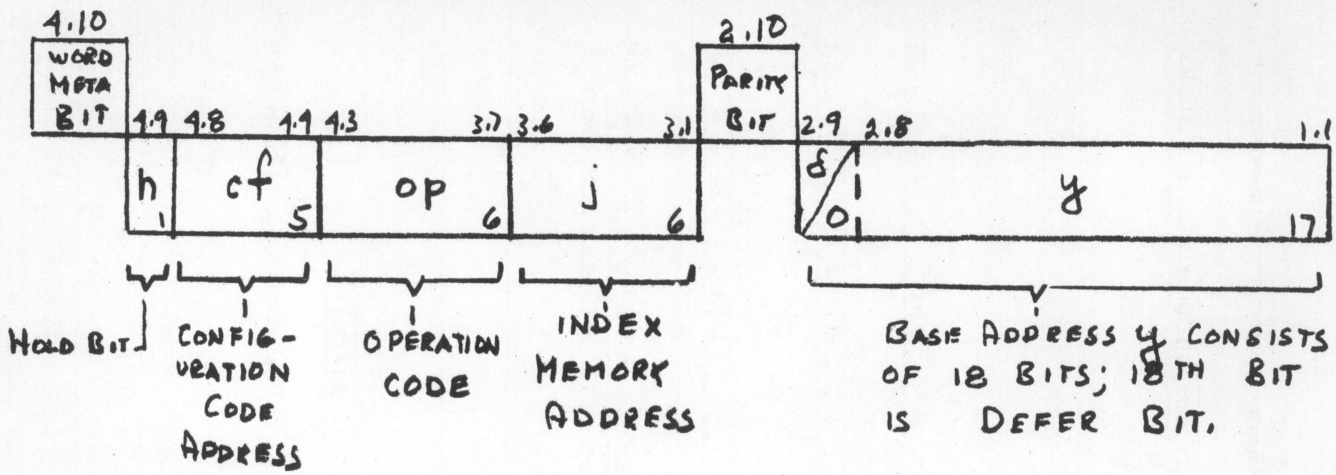
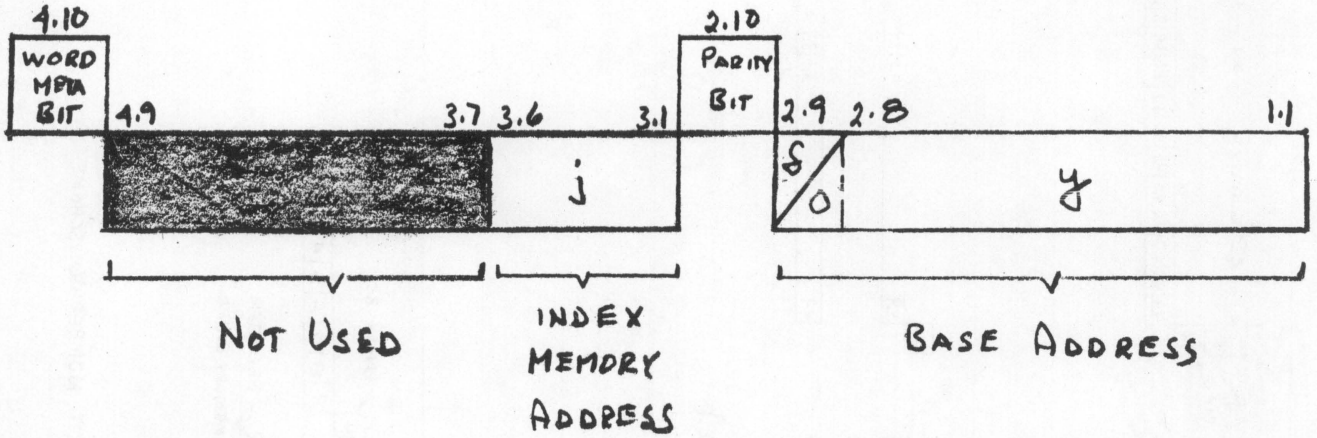


FIG. 2-1 . BIT NUMBERING SCHEME



a) INSTRUCTION WORD LAYOUT



b) DEFERRED ADDRESS WORD LAYOUT

FIG. 2-2 N REGISTER INSTRUCTION AND DEFERRED ADDRESS WORD LAYOUT

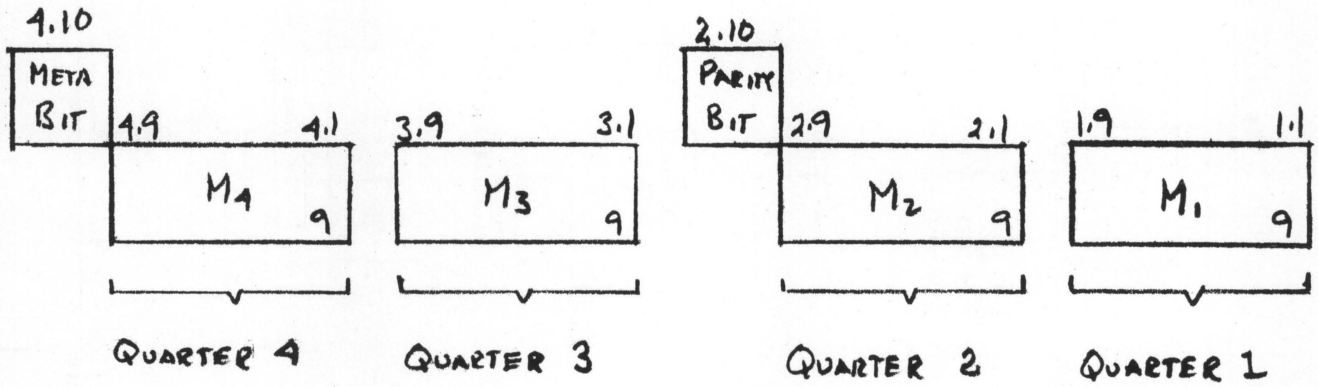


FIG 2.3 M REGISTER OPERAND WORD LAYOUT

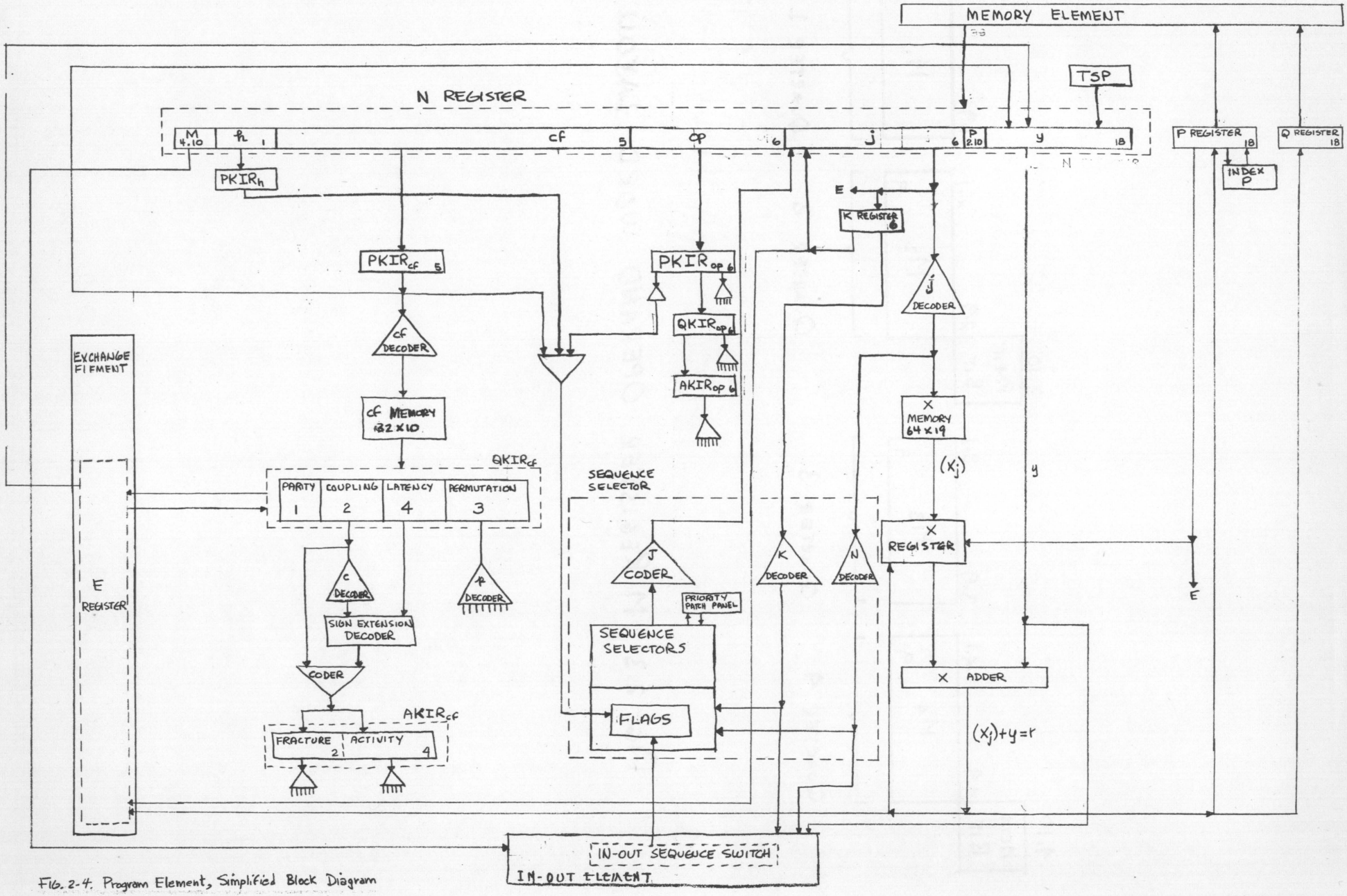


FIG. 2-4. Program Element, Simplified Block Diagram

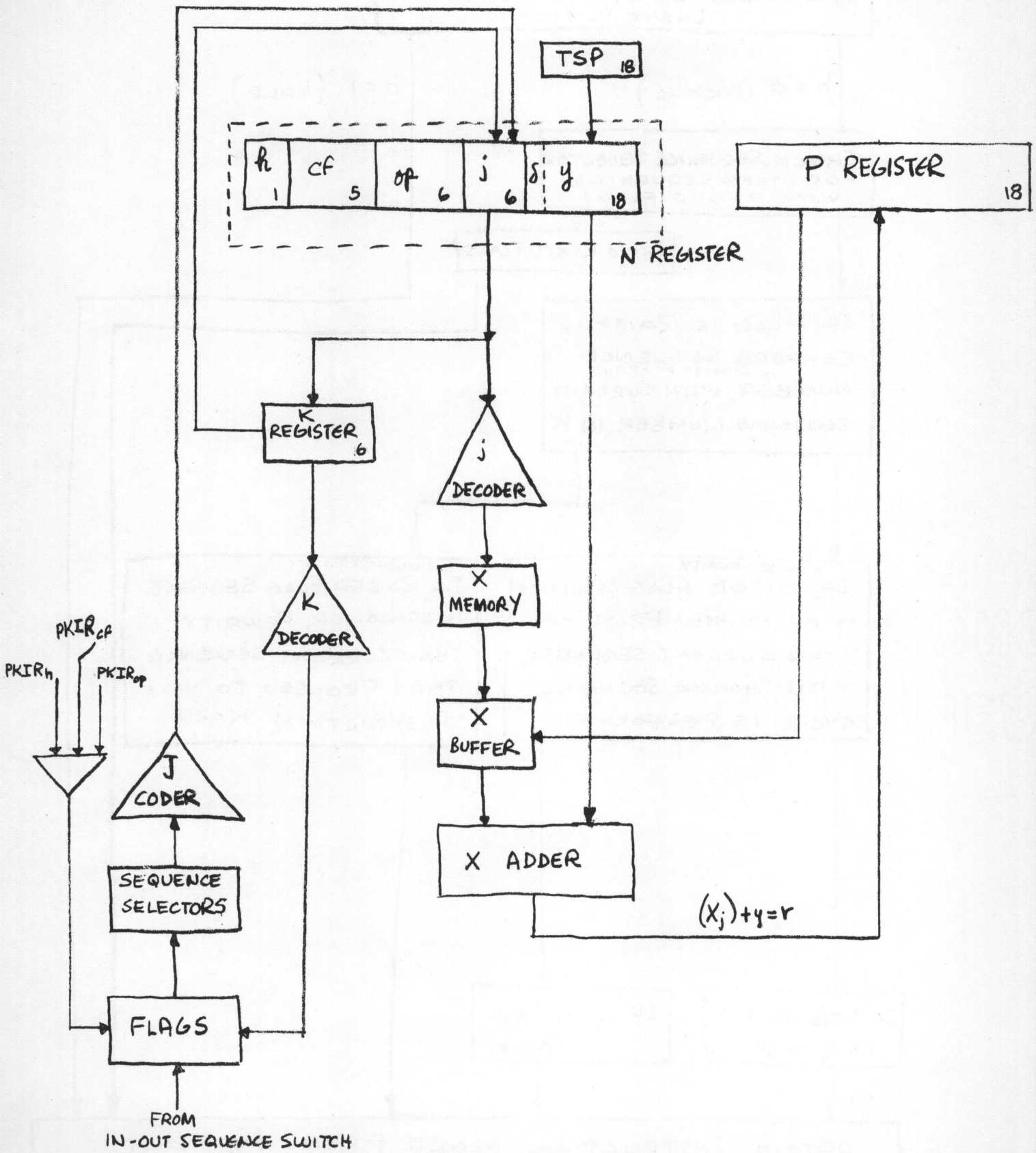
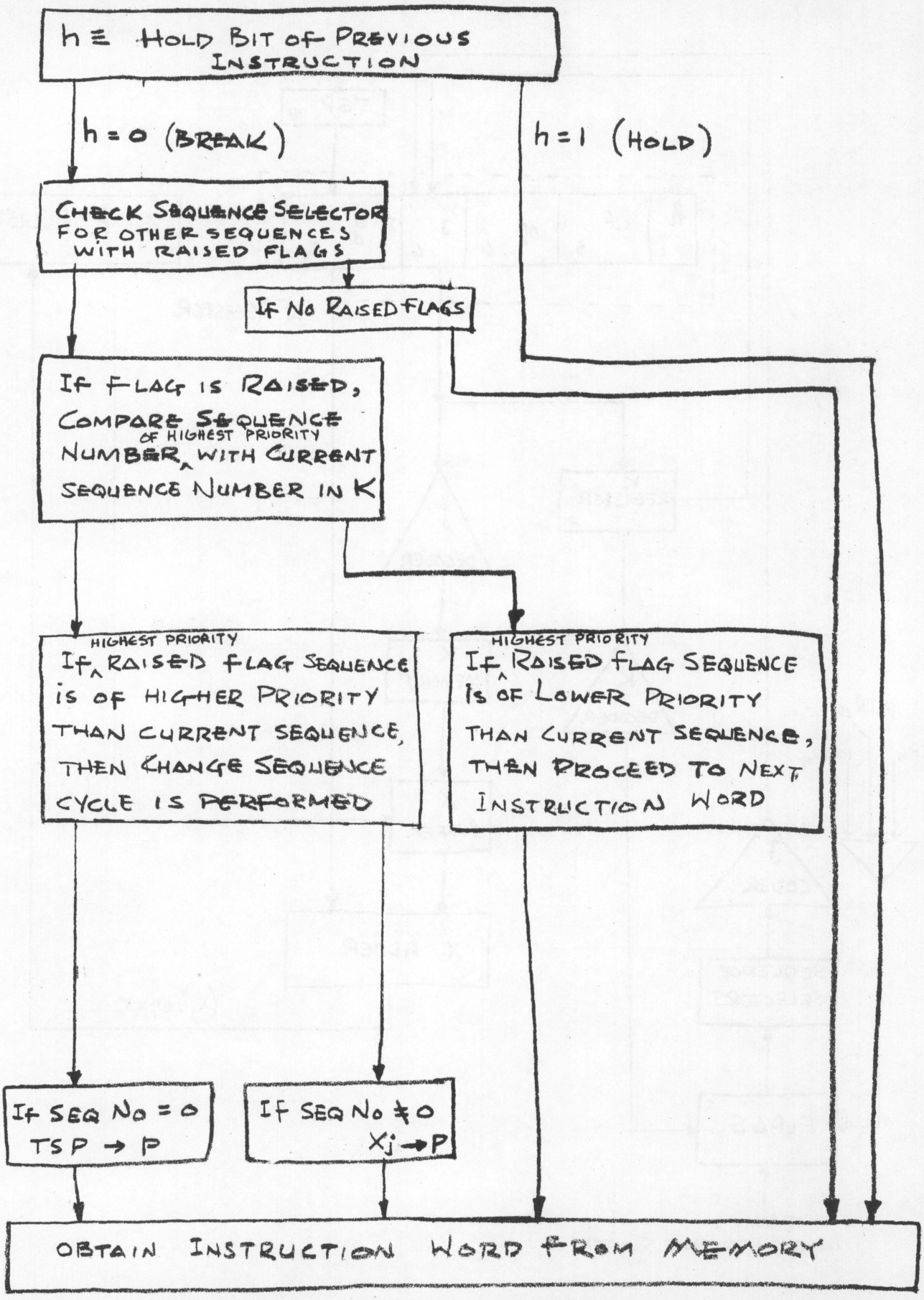


FIG. 2-5 TRANSFER PATHS DURING SEQUENCE SELECTION AND CHANGE OF SEQUENCE



SEQUENCE DETERMINATION & HOLD CYCLE

FIG 2-6

FIG. 2-7 PROGRAM SEQUENCE PRIORITY LIST

PROGRAM SEQUENCE NO.	SEQUENCE NAMES	COMMENT
00	STARTOVER	SPECIAL PROGRAM SEQUENCES
40	- -	
41	IN-OUT ALARM	
42	TRAPPING SEQUENCE	
43	- -	PROGRAM SEQUENCES ASSOCIATED WITH IN-OUT DEVICES
44	- -	
45	- -	
46	MAGNETIC TAPE	
47	MISCELLANEOUS INPUTS	
50	DATRAC	
51	XEROX	
52	PETR	
53	- -	
54	INTERVAL TIMER	
55	LITE PEN	
56	DISPLAY NO. 2	
57	- -	
60	DISPLAY NO. 1	
61	RANDOM NUMBER GENERATOR	
62	PUNCH NO. 2	
63	PUNCH NO. 1	
64	- -	
65	LINCOLN WRITER INPUT NO. 1	
66	LINCOLN WRITER OUTPUT NO. 1	
67	- -	
70	- -	
71	LINCOLN WRITER INPUT NO. 2	
72	LINCOLN WRITER OUTPUT NO. 2	
73	- -	
74	PLOTTER	
75	MISCELLANEOUS OUTPUTS	
76	- -	
77	- -	

HIGH PRIORITY ↑

LOW PRIORITY ↓

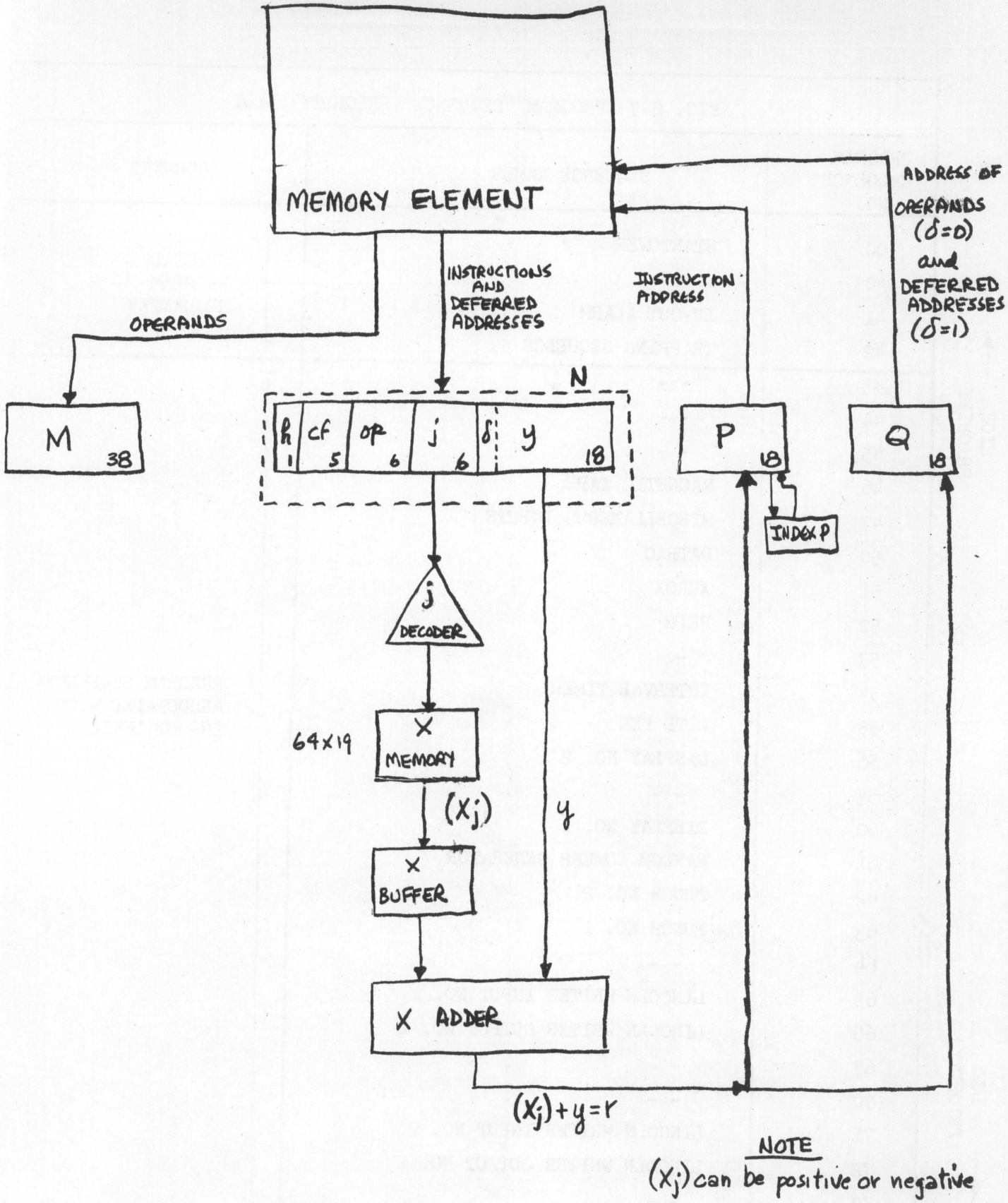


FIG. 2-8 Transfer paths for instruction and deferred address cycles

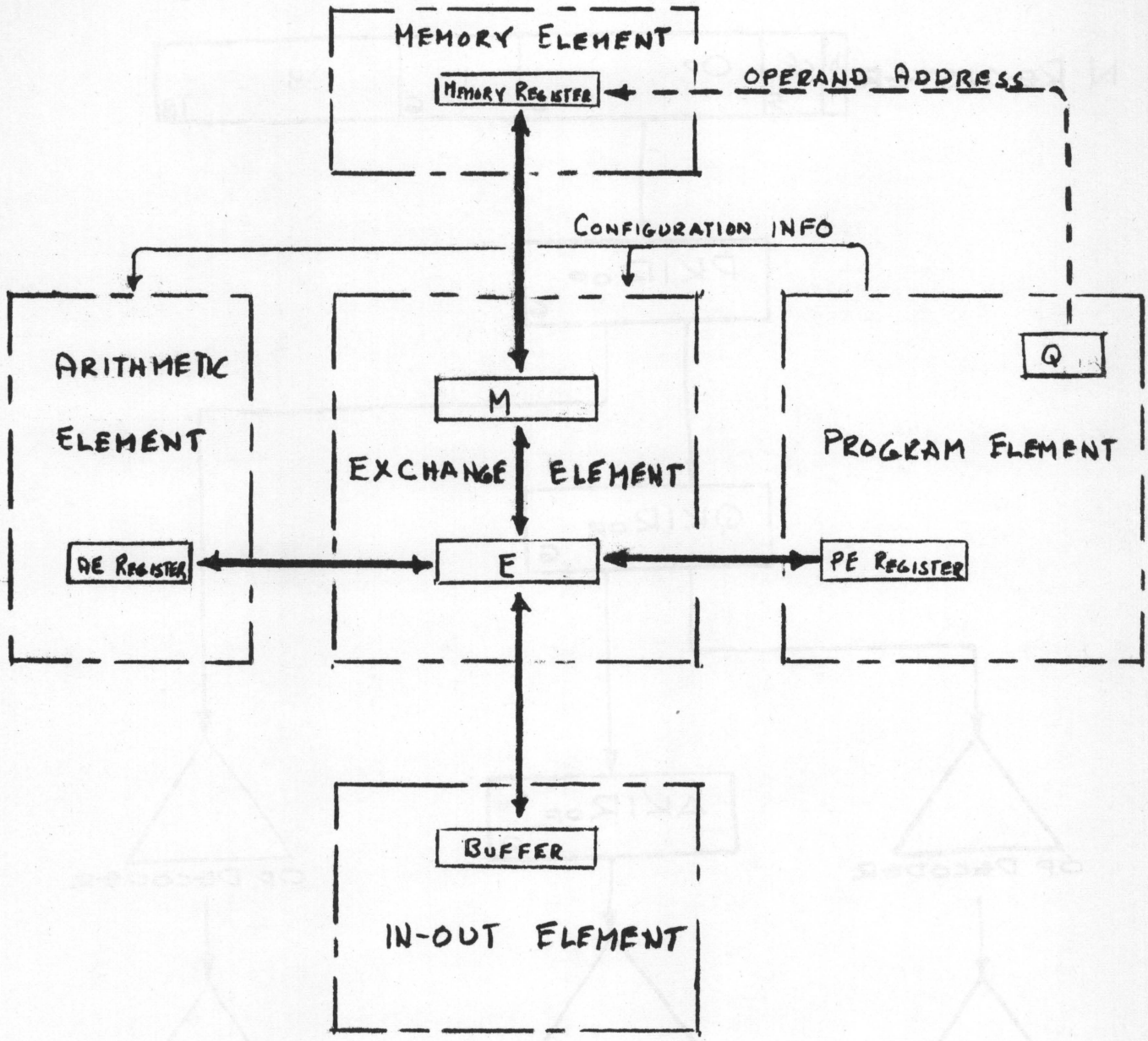
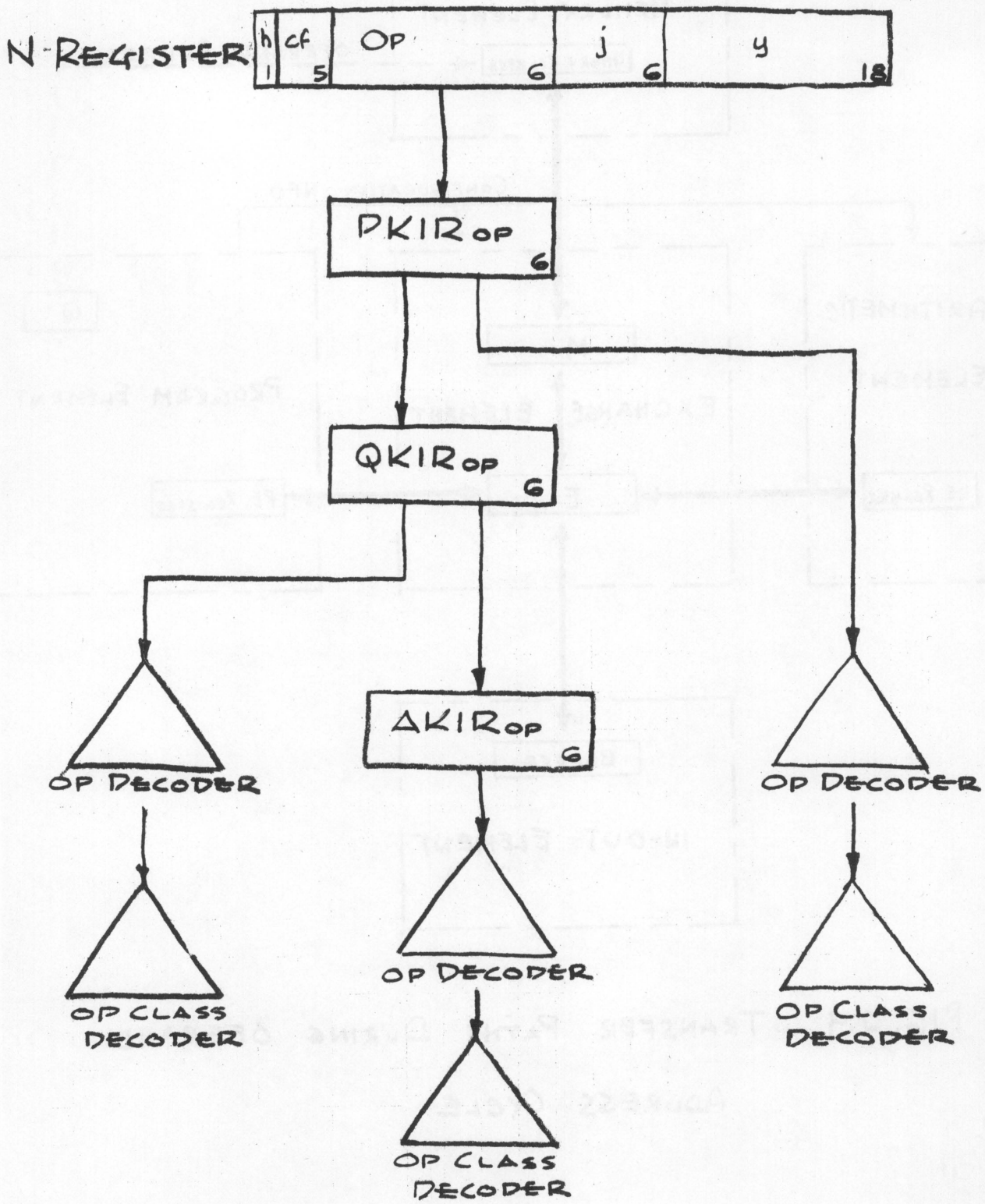
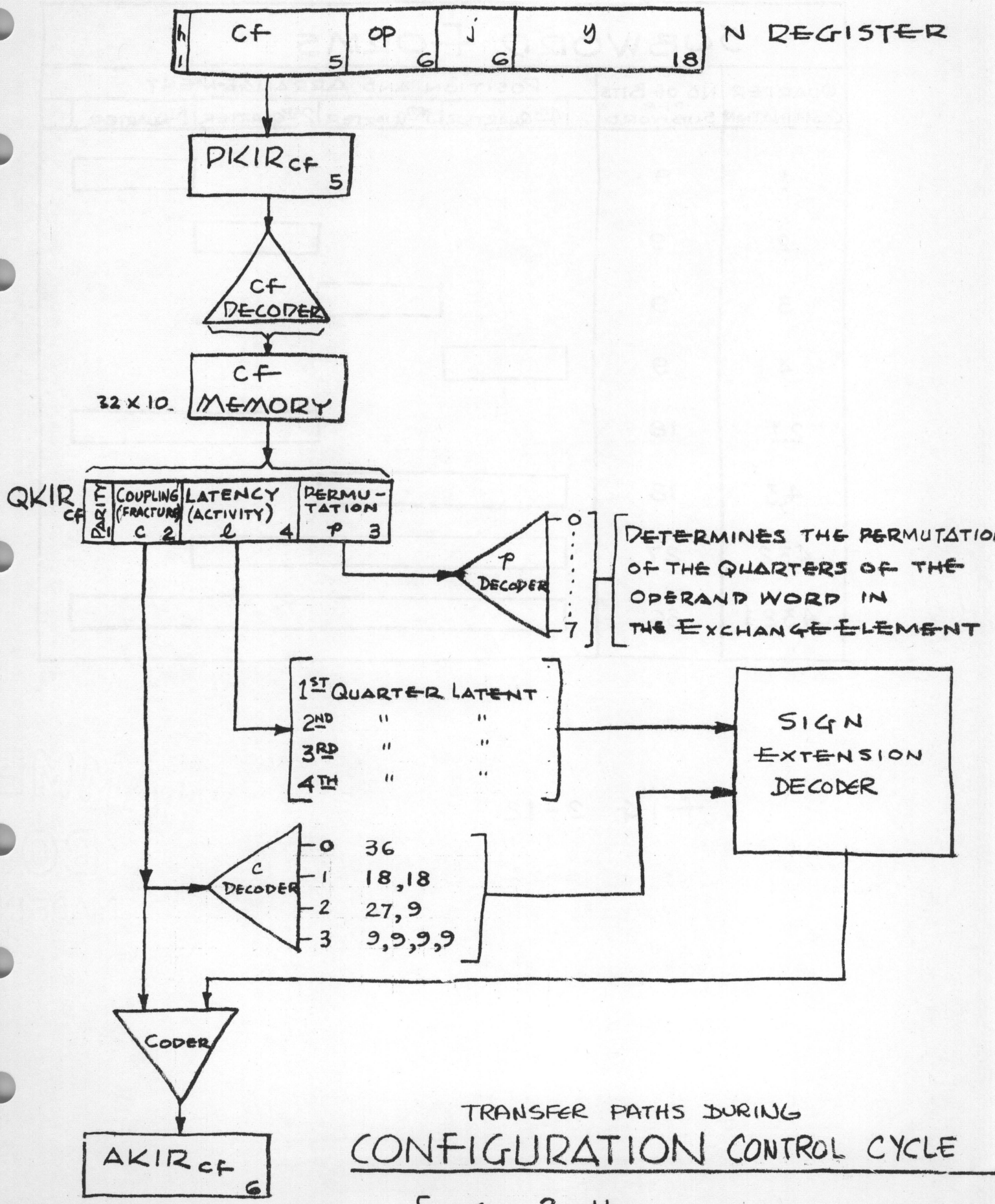


FIG. 2.9 TRANSFER PATHS DURING OPERAND ADDRESS CYCLE



OP CODE BLOCK DIAGRAM

FIG 2-10



TRANSFER PATHS DURING

CONFIGURATION CONTROL CYCLE

FIG 2-11

SUBWORD FORMS

QUARTER COMBINATION	NO OF BITS PER SUBWORD	POSITION AND ARRANGEMENT			
		4TH QUARTER	3RD QUARTER	2ND QUARTER	1ST QUARTER
1	9				▭
2	9			▭	
3	9		▭		
4	9	▭			
2,1	18			▭	▭
4,3	18	▭	▭		
4,3,2	27	▭	▭	▭	
4,3,2,1	36	▭	▭	▭	▭

FIG 2-12

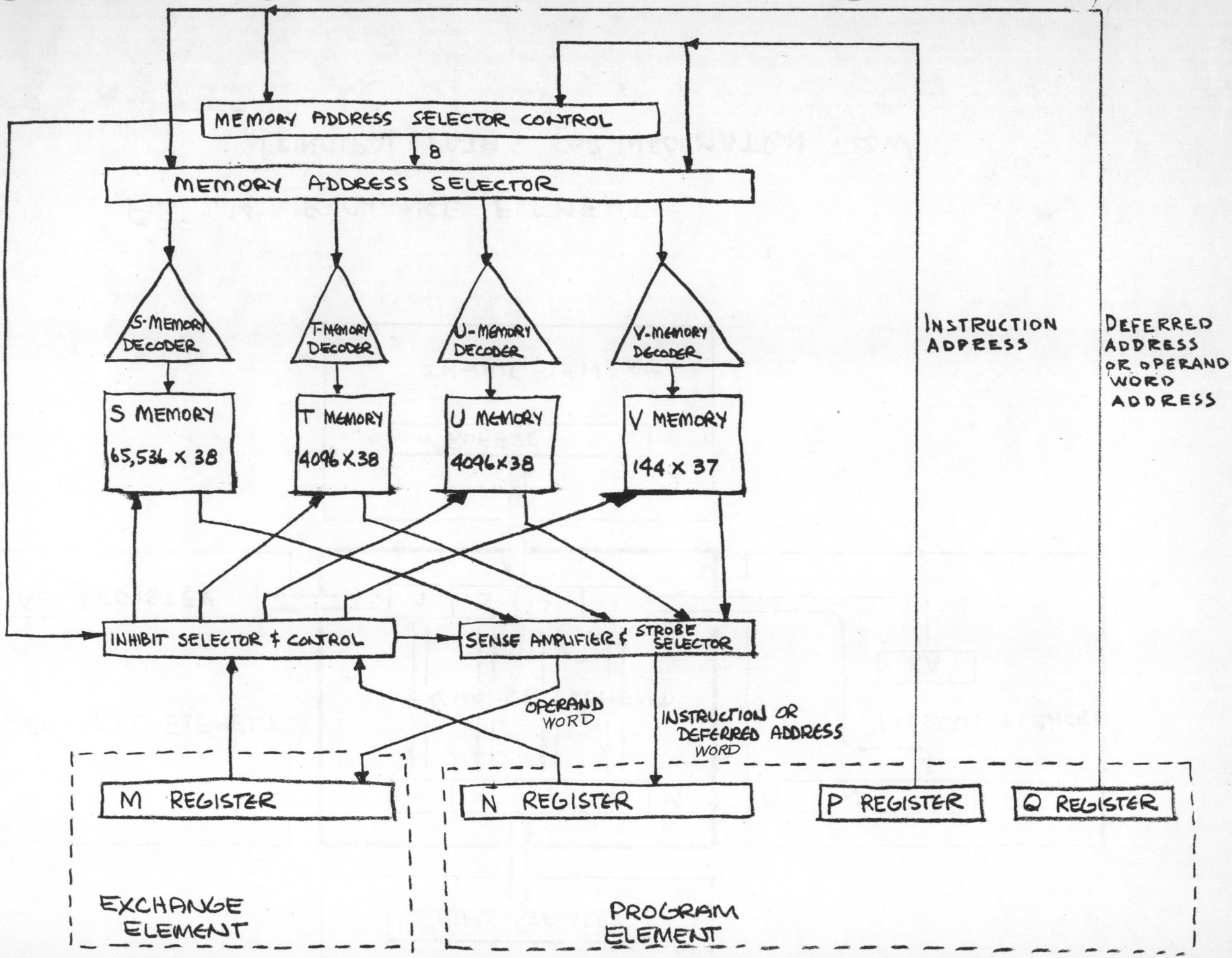


FIG. 2-13. MEMORY ELEMENT, SIMPLIFIED BLOCK DIAGRAM

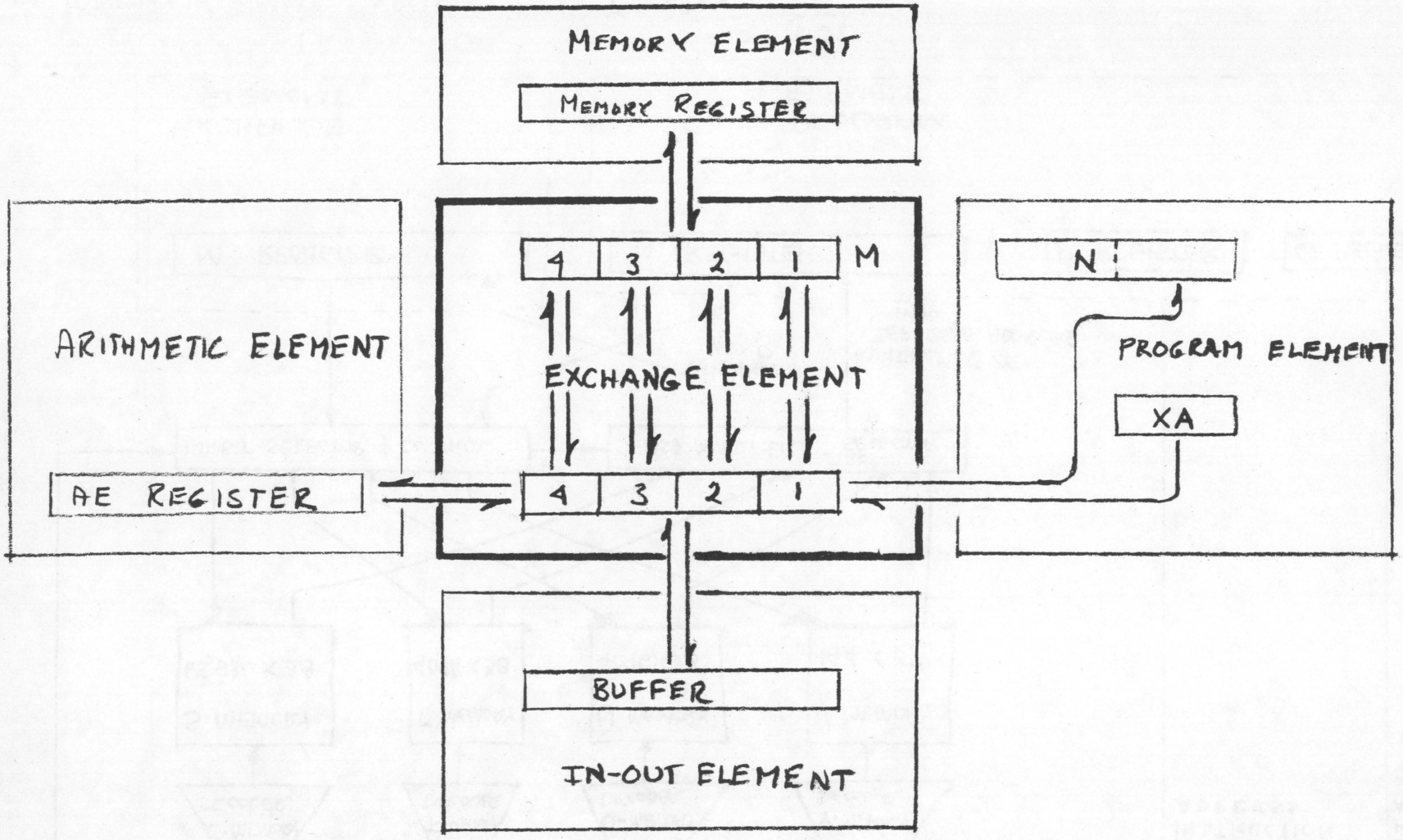


FIG. 2.14 EXCHANGE ELEMENT

PRINCIPAL PATHS FOR INFORMATION FLOW

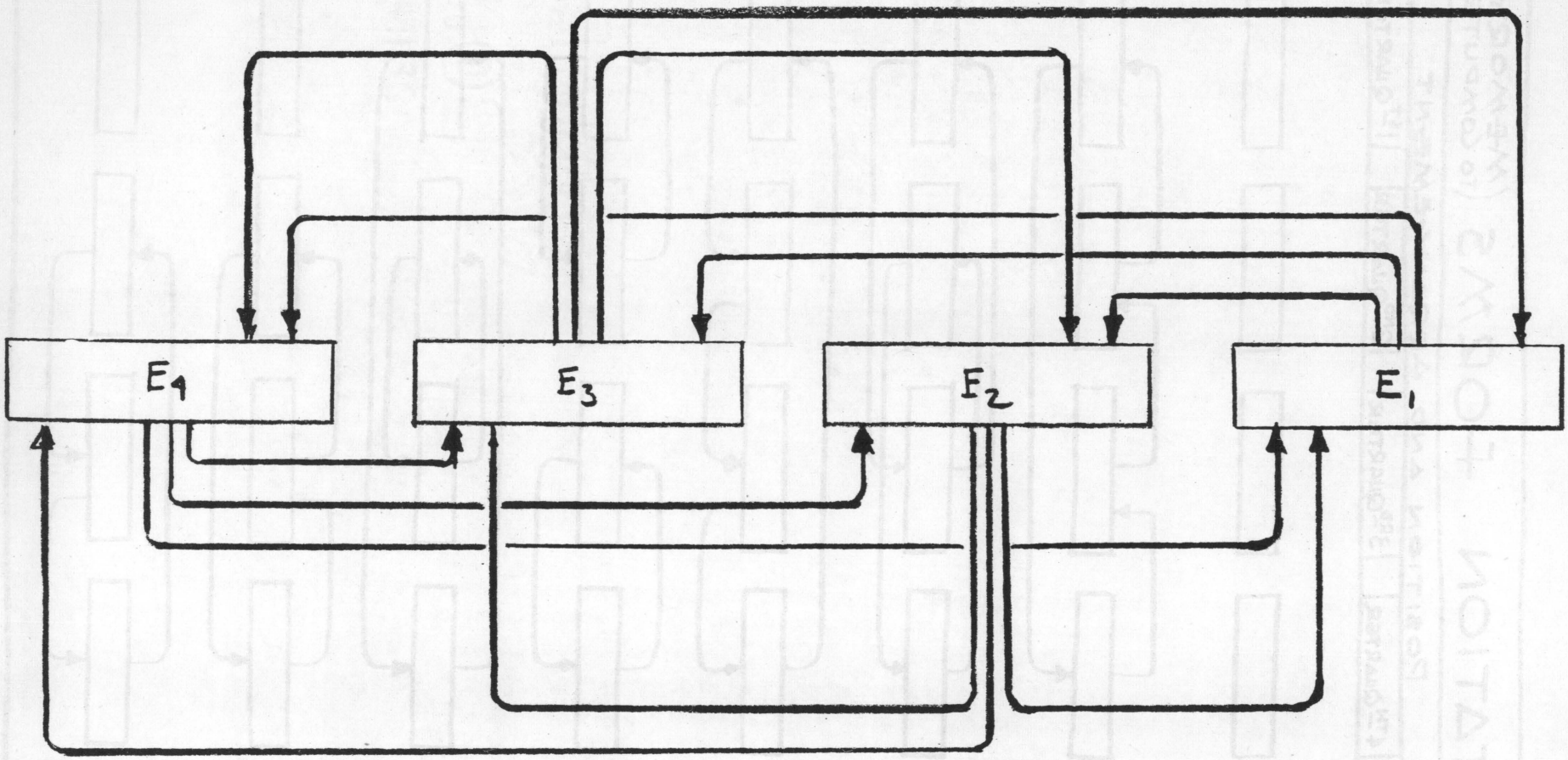


FIG. 2.15 E REGISTER PERMUTATION PATHS

PERMUTATION FORMS (MEMORY TO COMPUTER)

QKIRef 3,31	QUARTER COMBINATION	GRAPHIC NOTATION	POSITION AND ARRANGEMENT			
			4TH QUARTER	3RD QUARTER	2ND QUARTER	1ST QUARTER
000	4321	4321 / 4321				
001	1432	4321 / 1432				
010	2143	4321 / 2143				
011	3214	4321 / 3214				
100	3412	4321 / 3412				
101	1234	4321 / 1234				
110	2431	4321 / 2431				
111	3241	4321 / 3241				

FIG 2-16

INVERSE PERMUTATION FORMS
(COMPUTER TO MEMORY)

$QKIR_{(F_3,F_4)}$	QUARTER COMBINATION	GRAPHIC NOTATION	POSITION AND ARRANGEMENT: 4TH QUARTER	3RD QUARTER	2ND QUARTER	1ST QUARTER
000	4321	4 3 2 1 / 4 3 2 1	4	3	2	1
001	3214	3 2 1 4 / 4 3 2 1	1	4	3	2
010	2143	2 1 4 3 / 4 3 2 1	2	1	4	3
011	1432	1 4 3 2 / 4 3 2 1	3	2	1	4
100	3412	3 4 1 2 / 4 3 2 1	3	4	1	2
101	1234	1 2 3 4 / 4 3 2 1	1	2	3	4
110	3241	3 2 4 1 / 4 3 2 1	2	4	3	1
111	2431	2 4 3 1 / 4 3 2 1	3	2	4	1

FIG 2-17

ACTIVITY FORMS

QKIR CF 3,6,61	ACTIVE QUARTERS	GRAPHIC NOTATION	POSITION: 4TH QUARTER	3RD QUARTER	2ND QUARTER	1ST QUARTER
0000	4321	↓↓↓↓				
0001	432	↓↓↓|				
0010	431	↓↓|↓				
0011	43	↓↓||				
0100	421	↓|↓↓				
0101	42	↓|↓|				
0110	41	↓||↓				
0111	4	↓|||				
1000	321	|↓↓↓				
1001	32	|↓↓|				
1010	31	|↓|↓				
1011	3	|↓||				
1100	21	||↓↓				
1101	2	||↓|				
1110	1	|||↓				
1111	—	||||				

LEGEND:

ACTIVE

INACTIVE

FIG 2-18

SUBWORD FORM		
$QKIR_{CF_{9,8}}$	GRAPHIC NOTATION	SUBWORD POSITION
0 0		36
0 1		18 18
1 0		27 9
1 1		9 9 9 9

FIG. 2-19 SUBWORD FORMS

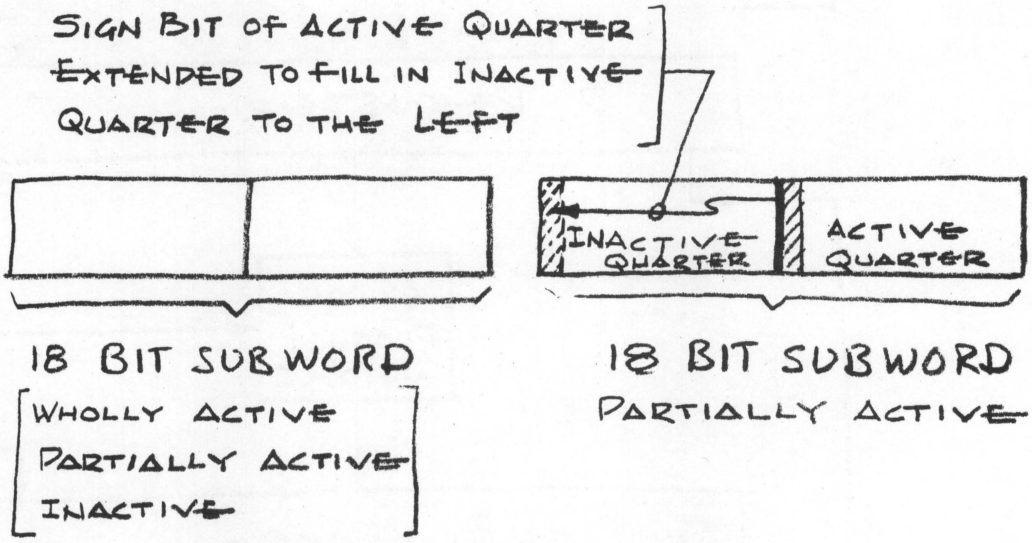


FIG 2-20

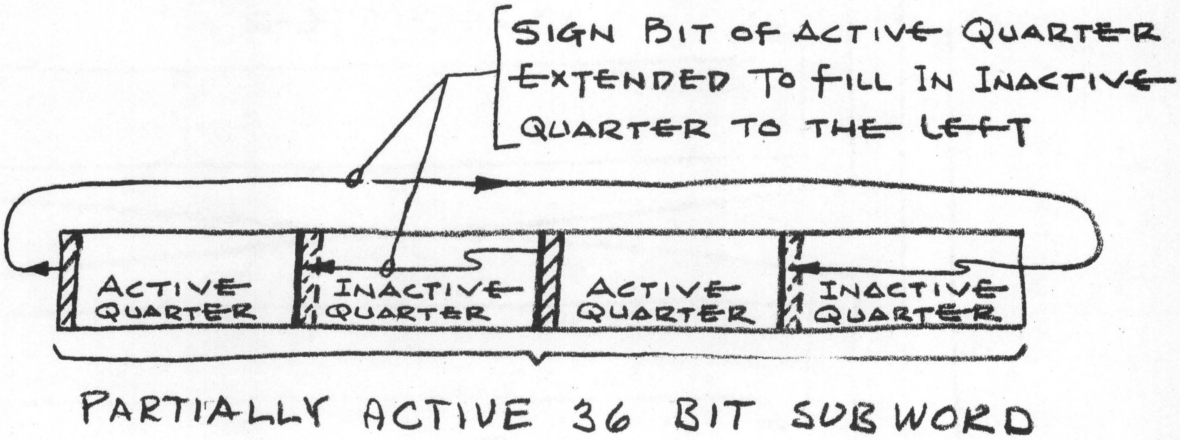
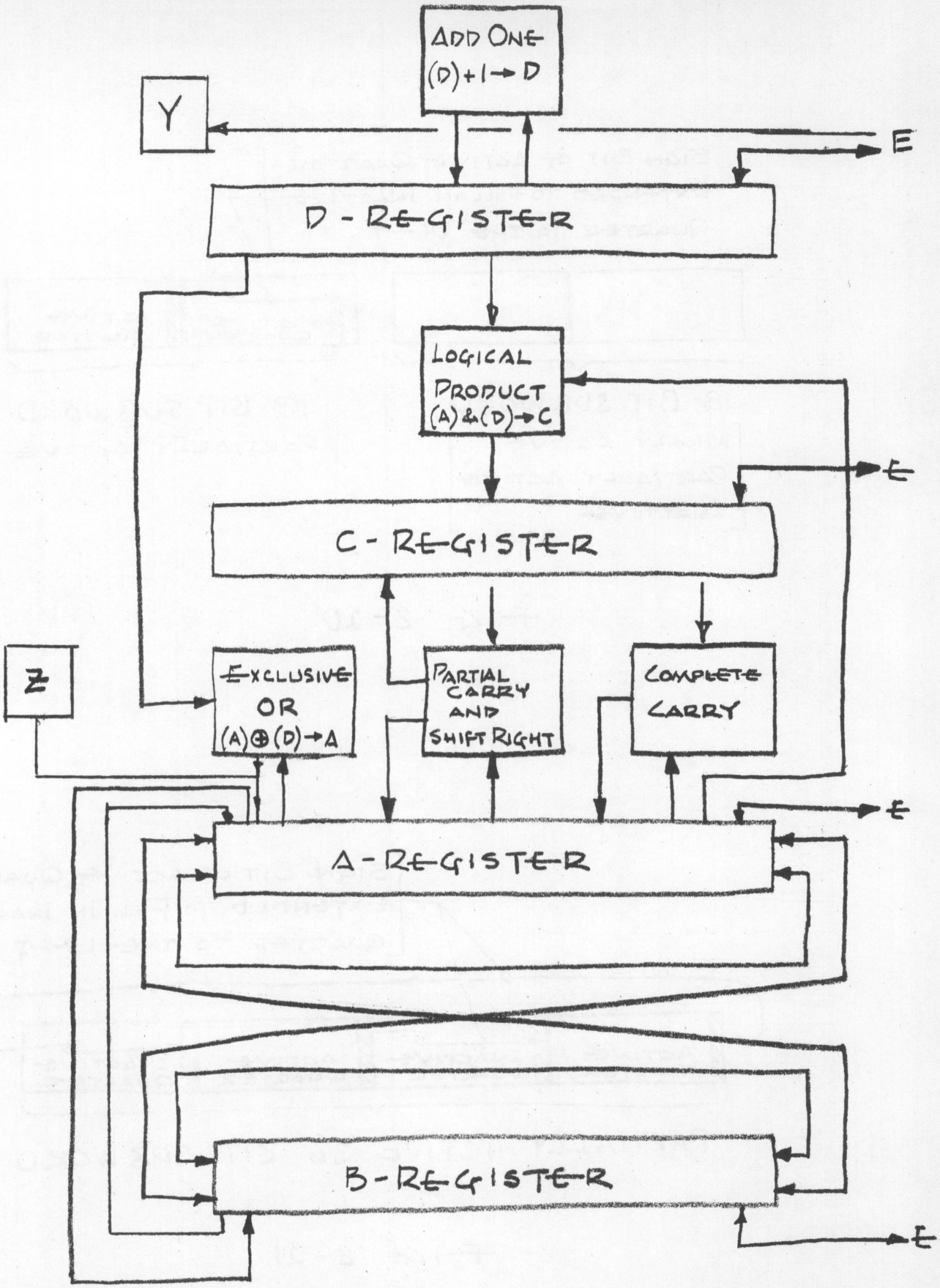


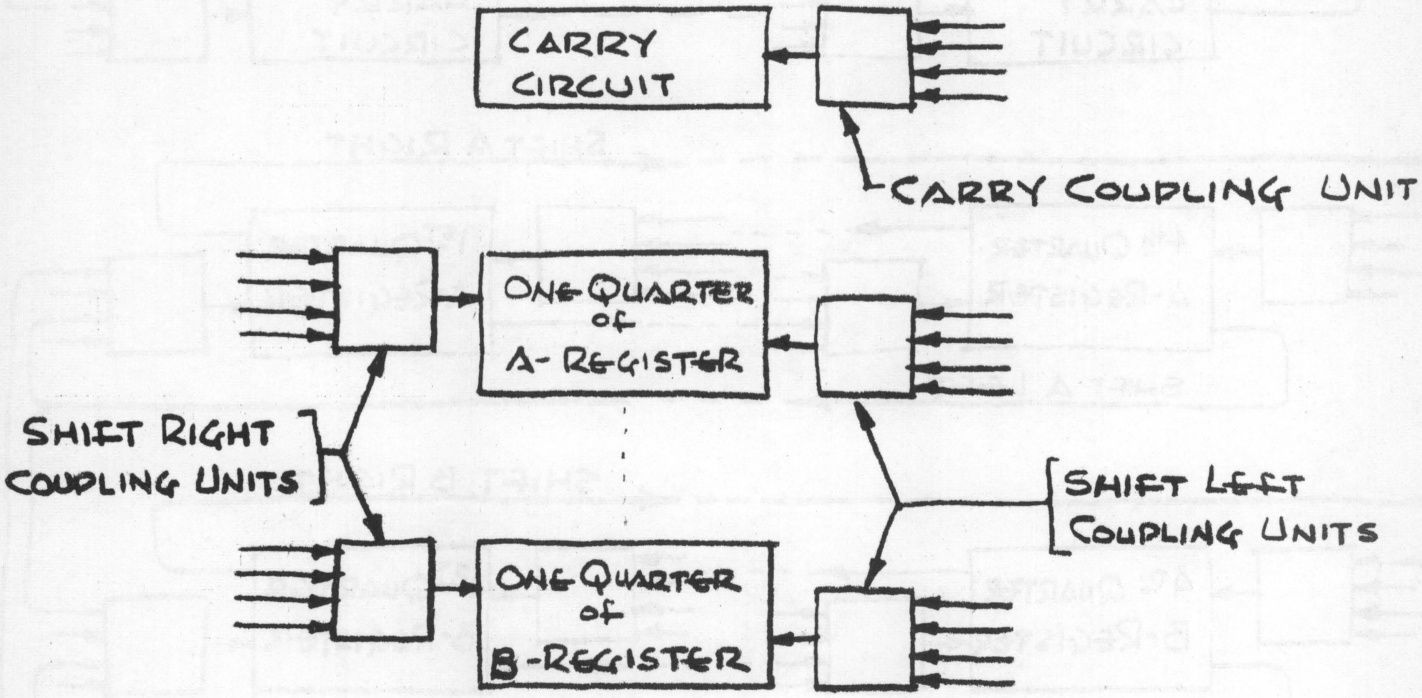
FIG 2-21

SIGN EXTENSION



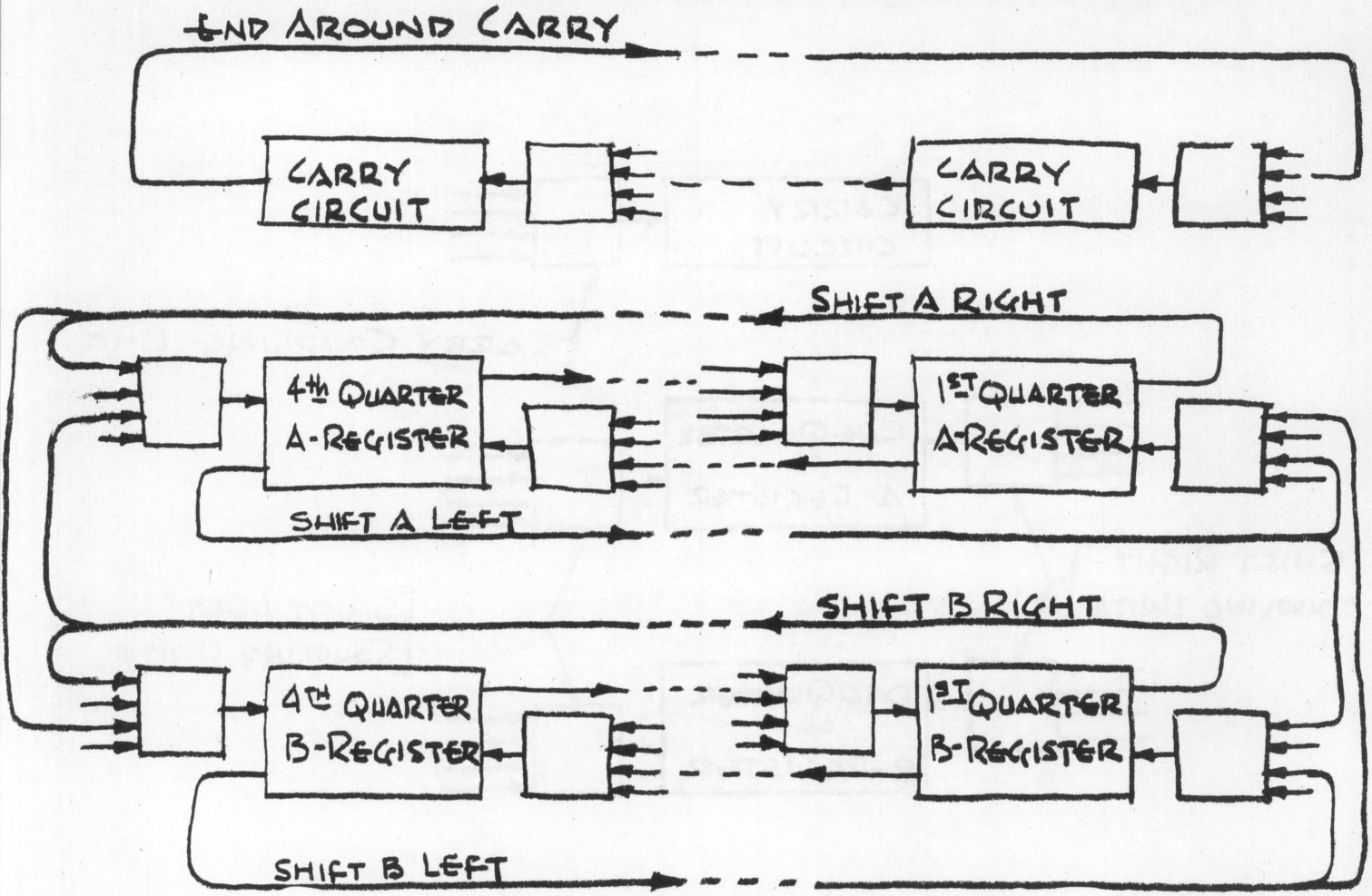
ARITHMETIC ELEMENT TRANSFER PATHS

FIG 2-22



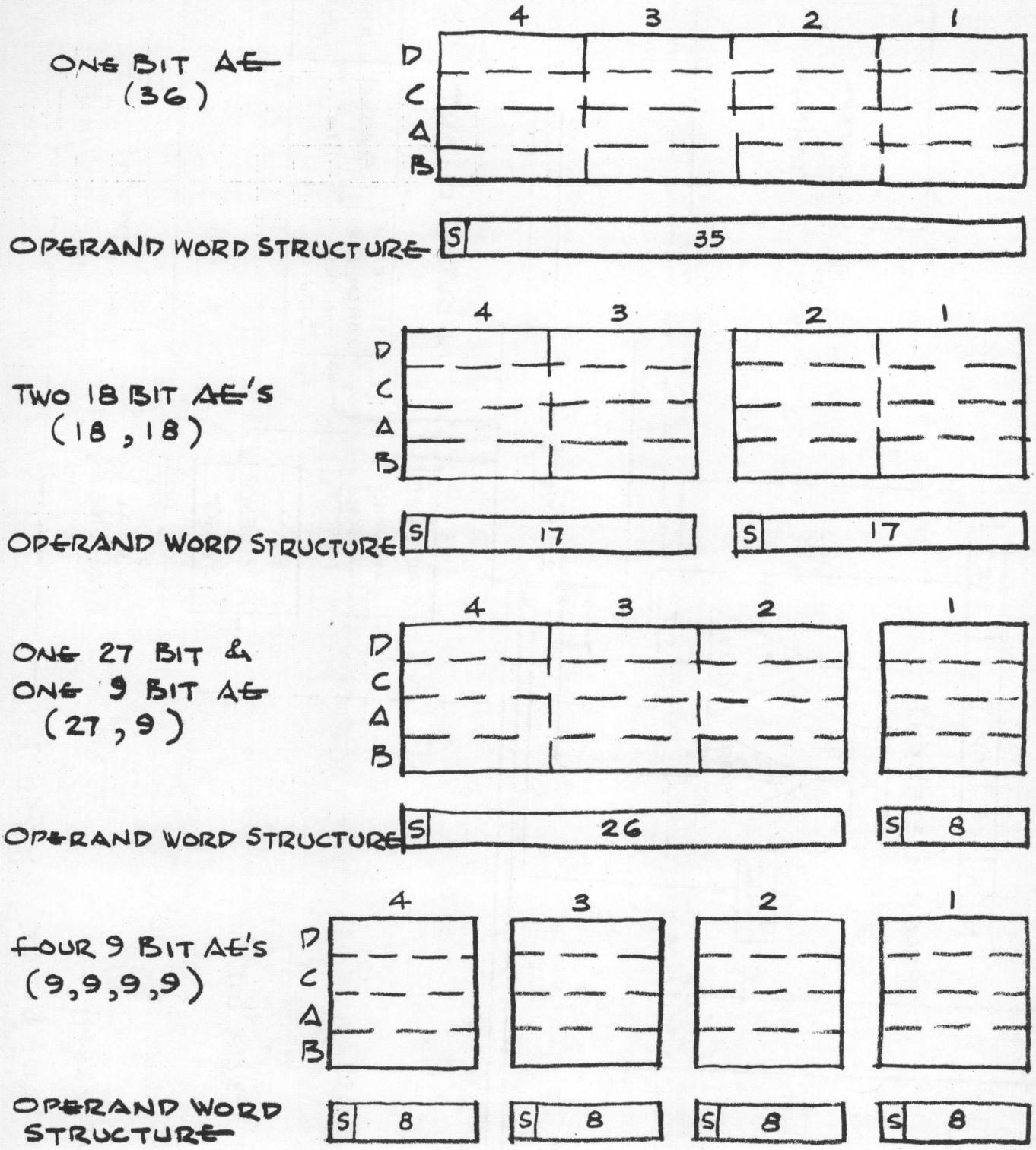
QUARTER COUPLING UNIT BLOCK DIAGRAM

FIG 2-23



QUARTER COUPLING CONNECTIONS

FIG 2-24



COUPLING UNIT CONNECTION FORMS

FIG 2-25

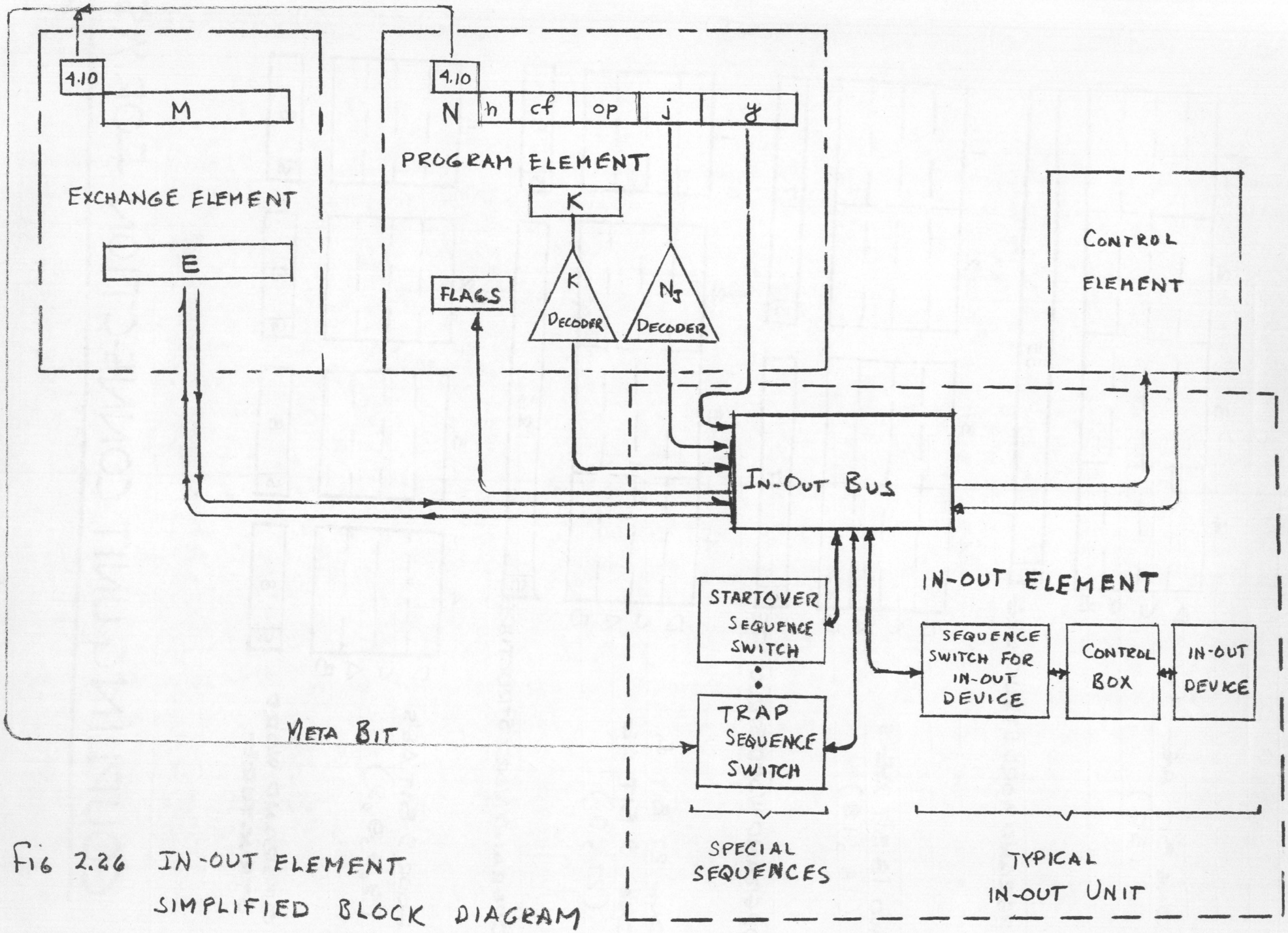


Fig 2.26 IN-OUT ELEMENT SIMPLIFIED BLOCK DIAGRAM

CHAPTER 3

CIRCUIT LOGIC ELEMENTS

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 3
CIRCUIT LOGIC ELEMENTS

3-1 INTRODUCTION

TX-2 has a modular construction, where the modules are plug-in packages, each containing several transistor circuits. In order to understand the details of the logic of the computer, it is necessary to understand the operation of and connections among the individual circuits and the techniques for physically realizing logic functions.

This chapter describes the logical characteristics of the TX-2 circuits. These characteristics are discussed only in sufficient detail to make the logical function apparent or to indicate certain operating limitations. A complete circuit analysis of each of the TX-2 plug-in packages is contained in the TX-2 Circuits Handbook.

It is assumed that the reader is familiar with symbolic logic as applied to computers and is able to interpret truth tables. Fig. 3-1 defines the logical AND, inclusive OR, and exclusive OR functions, as well as the complement function. The more common circuit conventions and symbols found on TX-2 block schematics are shown in Fig. 3-2.

3-2 SINGLE-TRANSISTOR LOGIC ELEMENTS

3-2.1 INTRODUCTION. The two basic transistor circuits used throughout the computer are: the saturated inverter and the emitter follower. Both circuits are used extensively for logic nets in which the inputs are levels. The inverter is also used for pulse gating and mixing. In these nets, some inputs are levels and others are pulses. The circuit diagrams for the inverter and emitter follower are shown in Fig. 3-3.

3-2.2 NOTATION. In level logic nets, all single-transistor logic elements are drawn as a rectangle with the emitter on the top and the collector on the bottom. Inverters can be distinguished from emitter followers by usage; that is, the emitter follower collector is always connected directly to a -3 volt supply, and the inverter emitter is always returned to ground either directly or through the collectors of other inverters. The diamond on a small side of the rectangle indicates a level input and is solid for -3 volts and hollow for ground. These same diamond conventions are used for the output.

In pulse nets, all inverters are drawn opposite to the above, i.e., with the emitter on the bottom and the collector on the top. Level inputs to these nets are shown by diamonds as in level nets. However, pulse inputs (which are always negative) are shown as solid arrows, and the positive output pulses are shown as hollow arrows.

3-3 LEVEL LOGIC CIRCUITS

3-3.1 EMITTER FOLLOWERS. In this circuit, Fig. 3-3(a), the collector is tied to a -3 volt supply (indicated by a solid circle) and the emitter, which is the output terminal, is connected through a load resistor R_L to a +10 volt supply (indicated by hollow square). This circuit is somewhat similar to the vacuum-tube cathode follower. Whenever the input is more negative (say 0.3 volt) than the output, more negative base current is drawn. The transistor amplifies this current and a negative emitter current many times larger flows in the emitter lead. This current tends to make the output go negative because of the voltage drop developed across R_L. The output (emitter) tends to follow the input, hence the terminology "emitter follower".

To summarize: In the emitter follower, a ground input results in a ground output, and a -3 volt input results in a -3 volt output.

3-3.2 INVERTERS. In this circuit, Fig. 3-3(b), the emitter is tied to ground (indicated by a hollow circle) and the collector is tied through a load resistor R_L to a -10 volt supply (indicated by solid square). When the input is at -3 volts, negative base current is drawn, and the transistor acts as a short circuit. This has the effect of connecting the output to ground. When the input is at ground, the transistor acts as a high impedance. In this case, the current through load register R_L makes the output drop from ground to -3 volts or lower.

To summarize: In inverters, a ground input results in a -3 volt output, and a -3 volt input results in a ground output. Hence the terminology "inverter".

It should be noted that resistors R_1 and R_2 and capacitor C are contained in the symbolic representation of the circuit. Hence, the left half of Fig. 3-3(b) is drawn with the implicit understanding that the circuitry shown on the base of the transistor in the right half of Fig. 3-3(b) is present. Input resistance R_1 and positive bias resistance R_2 provide tolerance to noise, and to signal-voltage and transistor-parameter variations. Capacitor C improves the circuit transition time.

3-3.3 APPLICATION TO LEVEL LOGIC NETS. The basic single-transistor logic elements, namely emitter followers and inverters, can be interconnected to perform logical functions on input signals as shown in Fig. 3-4. The emitters of two or more emitter followers connected in parallel will assume the voltage of the most negative input. See Fig. 3-4(a). In no case may emitter followers be put in a series connection.

In the case of the inverter, two basic types of connection are possible as shown in Figs. 3-4(b) and (c). For the series connection, the output will remain negative unless both input A and input B are made negative. In this case, the two series-connected short circuits will ground the output; in all other cases, the output will be negative. For the parallel connection, the output will be grounded if either input A or input B is made negative. These two basic types of connection can be combined to produce more complicated nets.

3-3.4 TRANSIENT CHARACTERISTICS. The dynamic characteristics of emitter followers and inverters are fundamental limitations on the speed at which the computer can operate. Two factors which must be considered are "transition time" and "saturation".

a) Transition Time: Logically speaking, all the computer level signals are nominally 0 volts (ground) or -3 volts, although they may vary widely from these values. Ideally, the transistor circuit element should reproduce or invert the signal transition appearing at its input instantaneously. In practice, this ideal situation is compromised both by the transistor and by the external circuitry tied to the transistor.

Because of hole storage time in the transistor, a finite time delay (dead time) occurs between the time an input changes and the time the output responds. In addition to this delay time, the transient properties of the transistor and the external loading circuitry result in a finite rise time or fall time. The sum of the delay time and the rise or fall time is called the "transition time". Fig. 3-5 shows time plots of a typical transistor circuit.

b) Saturation: It is important that circuits be insensitive to wide voltage and transistor-parameter variations; that is, while the nominal signal voltages are 0 and -3 volts, actually they may lie in the two bands of +0.5 to -0.4 volts and -2.4 to -5 volts. Fig. 3-6 shows how operating an inverter circuit in its "saturated" region minimizes the effects of these variations. For this, and other reasons, the transistors are normally saturated.

3-3.5 TRANSISTOR TYPES. Two basic PNP types of transistors are used in the TX-2 computer. The high-speed logic circuitry uses a surface-barrier transistor usually referred to as an L-5122, but similar to the 2N240. If high current gain at high current levels is required, a micro-alloy transistor (L-5134, similar to 2N393) is used. Other PNP and some NPN transistors are used for special applications.

3-3.6 CASCADING LIMITATIONS. As indicated previously, it is important that the transistors operate in their saturated region. It is also important that the cumulative time lag through circuit stages be kept within certain limits. These two considerations result in rules of combination for series and parallel arrangements of emitter followers and inverters. These are known as "fan in" rules. The same considerations limit the types and number of circuits that can be loaded on a driving stage and result in so-called "fan out" rules.

Two classes of emitter followers are used: saturated and non-saturated. Saturated emitter followers are those driven by inverters. In this case, the input voltage will go more negative than the -3 volt supply, causing the emitter follower to saturate and thereby normally produce a -2.9 volt output signal. The outputs of flip-flops and the cascode to be described later are special cases of the saturated

emitter follower using micro-alloy transistors. Nonsaturated emitter followers are those driven by saturated emitter followers. In this case, the output voltage is a function of the input and is normally only -2.6 volts. However, this will produce adequate drive on all following inverter circuits. In no case may emitter followers be driven by nonsaturated emitter followers.

To summarize: saturated emitter followers are those driven by inverters; nonsaturated emitter followers are those driven by saturated emitter followers, flip-flops, or cascodes. A nonsaturated emitter follower may not drive other emitter followers, only inverters.

There are also limitations on the inverter. The ground input level to an inverter in a level logic net must be no more negative than -0.4 volt; hence, no more than two inverters may be placed in series. No firm limitation may be placed on the number of parallel inverters, although it is usually limited to eight. Special considerations, involved in the case of series-parallel nets, will be covered later.

The driving ability of emitter followers and the load on their drivers are not well defined. Lightly loaded emitter followers impose a load on their drivers equal to approximately 2/3 of a "standard load", which is defined as the load presented by an inverter. This increases to one "standard load" when the emitter follower is driving its full load of four "standard loads".

The load presented by an inverter to its driver is one "standard load" and amounts to about 1.2 ma. The driving ability of an inverter is limited to two "standard loads". Hence, an inverter may drive two other inverters or three emitter followers.

3-3.7 CASCODES. Fig. 3-7 shows a typical cascode. This circuit can be used to provide logical inversion. The cascode has the additional properties of fast rise and fall times, and the ability to supply a large amount of current in both the ground and -3 volt states. For these reasons, it is used as a power amplifier. When used for driving signals over coaxial cable, a terminating resistor is employed to match the output impedance of the cascode to the cable impedance.

The cascode input signal is applied simultaneously to the base circuits of transistors Q_1 and Q_2. Therefore, the inputs to Q_2 and Q_3 are always opposite in phase, so that in the steady state only one transistor is conducting. Transistor Q_3 acts as an emitter follower and provides the driving current in the -3 volt stage, quickly pulling the output down to -3 volts. Q_2 acts as an inverter and provides the current in the ground state, quickly pulling the output up to ground. Thus, the cascode circuit exploits the fast fall time of the emitter follower and the fast rise time of the inverter, and makes the total power available to the load since none is dissipated in load resistances.

The cascode circuit is capable of driving twelve "standard loads" or one series-terminated 93-ohm coaxial cable and four other "standard loads". A series-terminated coaxial cable can drive three inverter loads. Emitter followers may not be used as loads on a series-terminated coaxial cable. The cascode input presents two "standard loads" to its driver.

3-4 LEVEL LOGIC NETS

3-4.1 GENERAL. This section investigates the logical possibilities that exist for mixing levels by means of two or more transistors. All of these circuits will be either emitter followers or inverters.

3-4.2 TWO TRANSISTOR LEVEL LOGIC. Fig. 3-8 shows how logical AND and OR type circuits can be realized using emitter followers or inverters in series or in parallel. Note that Fig. 3-8(a), (c), and (e) are all AND circuits with two inputs. If the circuits are considered as black boxes, the only differences are the voltages used to represent the truth state of the input and output variables. Thus, the truth and false statements given in the respective truth tables are identical; only the corresponding voltages are different. Three possibilities exist: (1) ground inputs produce a ground output; (2) ground inputs produce a -3 volt output; and (3) -3 volt inputs produce a ground output. There is no AND circuit which combines -3 volt inputs and gives a -3 volt output.

Figs. 3-8(b), (d), and (f), in comparison, are inclusive OR circuits with two inputs. All the statements made about the AND circuits are also true for the OR circuits. However, the voltages used to represent the truth states of the OR circuits are the complements of the voltages used to represent the truth states of the AND circuits. In Fig. 3-8(a), for example, both A <u>and</u> B must be at ground in order that C be at ground. Therefore, the circuit is an AND circuit. In Fig. 3-8(b), C will be at -3 volts if either A <u>or</u> B (or both) is at -3 volts. Therefore, the circuit is an inclusive OR circuit.

3-4.3 THREE-TRANSISTOR LEVEL LOGIC. Fig. 3-9 shows how three-transistor circuits may be used to express more complex relationships than are possible with two-transistor circuits, that is, logical AND's and OR's appear together in the same expression. These circuits may be considered as simple elaborations of the two-transistor logic nets shown in Fig. 3-8. Thus, the A + B circuit shown in Fig. 3-8(d) becomes the core of the $(A + B) \cdot C$ circuit shown in Fig. 3-9(f).

The circuits on the right side of Fig. 3-9 are the same as those on the left side, except that the corresponding inputs and outputs are complements of one another. Complementing the voltages has the effect of changing all AND's to OR's, and all OR's to AND's, in the logical expressions.

3-4.4 FOUR-TRANSISTOR LEVEL LOGIC. The use of four transistors provides even more combination possibilities, as shown in Fig. 3-10. The principles of operation are similar to those for two- and three-transistor circuits.

3-4.5 LARGER NETS. Still larger nets of emitter followers and inverters are useful in the computer. Fig. 3-11 shows some of the larger inverter nets with their logic function for -3 volt inputs and ground outputs.

Circuit limitations determine the extent to which these nets can be increased. There is a limit on the total number of inputs to non-ground elements in series-parallel or parallel-series nets. Such nets contain transistors with emitters not tied directly to ground. For example, inputs F and G, D and F, and D, G, and $\overline{G}$ in Fig. 3-11(a), (b), and (c) respectively are of this type. This rule limits the total number of these inputs to two. Mutually exclusive inputs are considered as one; thus, G and $\overline{G}$ in Fig. 3-11(c) may be counted as only one input.

3-5 PULSE LOGIC CIRCUITS

3-5.1 GENERAL. In addition to static signal levels, pulses are also used in the computer. These pulses are used for timing control and therefore are precisely spaced in time. The times at which level transitions occur are determined by pulses.

3-5.2 POLARITY. Pulses are described as either negative (going) or positive (going) depending on whether they have ground as a reference and extend to -3 volts, or have -3 volts as a reference and extend to ground.

3-5.3 REPETITION FREQUENCY. The TX-2 computer uses a vacuum-tube pulse-generator to produce pulses at a 2.5 megacycle basic clock rate; that is, one pulse every 0.4 microsecond. As shown in Fig. 3-12, two pulse outputs are available from the 2.5 megacycle clock. One pulse stream, the β phase, is delayed 0.2 microsecond from the other pulse stream, the α phase. The α and β pulses differ only in their reference to some arbitrary zero starting time. By means of this system, events can occur every 0.2 microsecond despite the fact that the clock frequency is only 2.5 megacycles on each pulse line.

3-5.4 PULSE SOURCE. Fig. 3-13 shows the α and β pulse-generator system. Each clock pulse output phase is fed to a shaper and then to a shaper amplifier which drives ten vacuum-tube driver amplifiers. The output of these amplifiers is a 40 volt positive pulse which, in turn, drives five buffer amplifiers. Negative, 30 volt, 0.1 microsecond pulses are transferred from each buffer amplifier to the computer frame over separate coaxial cables. At the end of each cable, a non-inverting 10:1 pulse transformer provides -3 volt clock pulses for the logical circuitry.

3-5.5 PULSE NOTATION. Since diamonds are used to indicate levels, arrows are used to indicate pulses (see Fig. 3-2). A hollow arrow indicates a positive pulse, and a solid arrow indicates a negative pulse. (The solid arrow used for negative pulses should not be confused with similar-type arrows used on block diagrams to indicate signal paths.)

3-5.6 PULSE GATING. In logic nets using both pulses and levels, levels are used to gate pulses. The inputs to these nets are one or more levels and a single pulse line input. The output is always a pulse (usually of the opposite polarity to the input pulse). When the input level logic is satisfied, an input pulse will cause a pulse output. This process is sometimes described as the input pulse "sampling" the input level logic. The need for logic nets with both pulse and level inputs will become apparent in the sections that follow.

3-5.7 SINGLE-STAGE PULSE GATES. Fig. 3-14(a) shows an inverter with a negative pulse input and a positive pulse output. If the emitter is tied to ground through a level logic circuit, the situation in Fig. 3-14(b) results. In this case, when the emitter is at -3 volts, the transistor will not conduct regardless of the pulse on the base circuit. However, when the emitter is at ground, a negative pulse on the base produces a positive pulse at the collector. This ground level to the emitter may come from a flip-flop or another inverter. In Fig. 3-14(c), two transistors are used, with a negative pulse and a -3 volt level input producing a positive pulse output. Fig. 3-14(d) shows the same circuit used for mixing a negative pulse and two levels of opposite sense. Note, in Fig. 3-14, that all the circuits form logical AND gates; that is, all the indicated input signals must be present in order to produce a pulse output.

The inverter circuits used to make up pulse gates are standard inverters. Up to three inverters may normally be put in series, with the pulse being applied to the output transistor. The level inputs present one "standard load" to their driver. The pulse input presents one "pulse load" to its driver when its emitter is grounded, and approximately 1/3 "pulse load" when gated off.

3-5.8 REGISTER DRIVERS. In order that the single-stage gating circuits just discussed operate properly, it is necessary that the pulse inputs be of proper magnitude and shape. To accomplish this, the outputs of special circuits called "register drivers" are used as pulse inputs to the single-stage gates. (See Fig. 3-15)

If the two register-driver level inputs are at ground, the inverter supplies current to the bases of Q_3 and Q_4. These transistors are saturated emitter followers whose collector supply voltage is made up of clock pulses from the distribution system of Fig. 3-13. When saturated, these transistors short the input pulse through to the output terminal. The circuit is able to drive twelve "pulse inputs" plus a 100-ohm line terminator.

Both level inputs to the register driver must be at ground before a pulse output can occur. The clock pulses determine precisely when a pulse will occur at the output of the register driver. The level inputs to the register driver each present two "standard loads" to their driving source.

3-6 FLIP-FLOPS

3-6.1 GENERAL. Flip-flops are memory or data-storage devices. The basic activity of the computer involves transferring data into and out of these storage devices. The inputs to the flip-flops are positive pulses and the outputs are levels. Like any memory element, the flip-flop is a two-state (bistable) device. These states are called the ZERO state and the ONE state. The flip-flop will remain indefinitely in one of these states until pulsed into the opposite state.

3-6.2 OPERATION. There are three inputs to the flip-flop (see Fig. 3-16). A "set to zero" input pulse places the flip-flop in the ZERO state. This is true regardless of the state the flip-flop was in prior to the pulse. Likewise, a "set to one" input pulse places the flip-flop in the ONE state regardless of the prior flip-flop state. A "set to complement" pulse places the flip-flop in the opposite state from that which it was in prior to the pulse. Note that the inputs are always at -3 volts until an input pulse occurs. Pulses are barred by logical design from occurring on more than one input at a time, or closer together than 0.2 microseconds, because of the ambiguity involved.

3-6.3 NOTATION. While the operation of flip-flops is straightforward, there are certain subtleties in the labeling of the outputs that are important to grasp. The flip-flop is a two-state device and may be in either the ONE state or the ZERO state. The flip-flop states themselves can be represented by variables, namely FF^0 and FF^1 FF^0 is the name of the variable representing the ZERO state, and FF^1 is the name of the variable representing the ONE state. The variable is TRUE if the flip-flop is in the state represented by the variable, and is FALSE if the flip-flop is not in the state represented by the variable (that is, in the opposite state). Thus, FF^1 is TRUE if the flip-flop is in the ONE state, and FALSE if the flip-flop is in the ZERO state.

Note that nothing has been said thus far about the output voltage levels used to represent the state of the flip-flop. A flip-flop has two output wires which have opposite voltages on them: that is, when one is at ground, the other is at -3 volts. These wires are called the "0 wire" and the "1 wire". By definition, the "0 wire" has -3 volts on it when the flip-flop is in the ZERO state, and the "1 wire" has -3 volts on it when the flip-flop is in the ONE state.

Fig. 3-16 summarizes the operation of the flip-flop. If the flip-flop is first "set to ZERO", then "set to ONE", and then "set to complement", the sequence of voltages given in the variable truth table will appear on the 0 and 1 wires. Whether

the wire is in the TRUE or FALSE state of the variable depends on the agreement (or lack of agreement) between the state of the flip-flop and the state represented by the variable.

3-6.4 INTERNAL CIRCUITRY. Fig. 3-17 shows the internal circuitry of the flip-flop. The heart of the circuit is a transistor variation of the familiar Eccles-Jordan multivibrator. In this case, the gating transistors external to the flip-flop form inverters with resistors in the flip-flop themselves. The outputs of the inverters provide the trigger inputs to the two-state circuitry. The cascodes provide a low-impedance driving source and decouple the flip-flop from its load. Note that the two cascodes are cross-tied. In all other respects, the cascodes are similar to those described previously.

3-6.5 EXTERNAL INPUT CIRCUITRY. Fig. 3-18 shows the typical input circuitry external to the flip-flop. The flip-flop is pulsed by ANDing a "when" and "what" type signal at the input gate. The "when" pulse signal determines precisely when in time the flip-flop is going to be pulsed (assuming that the "what" circuitry indicates a ground level is available to be sampled). The "what" level signal determines the state to which the flip-flop will be set. For example, if the emitter on the external "set to ONE" gate is at ground, the flip-flop will be pulsed to the ONE state. Data is shifted through the computer along the "what" lines. Data shift time control occurs along the "when" lines.

3-6.6 PULSE AND LEVEL TIME RELATIONSHIP. Fig. 3-19 shows the time relationship of the pulse and level transitions that might occur in Fig. 3-18. Assume that at time t = 0, the output of logic net C1 is at ground and that the outputs of logic nets C2 and C3 are at -3 volts. Clock pulses of the shape shown in the Fig. 3-19 arrive at the input to the register driver at 0.4-microsecond intervals. The output of the register driver, however, will remain at a constant voltage that is slightly positive of ground level.

At approximately t = 0.15 microsecond, the output of logic net C2 rises toward ground. The total rise time depends on the circuitry of logic net C2 and typically might be 40 millimicroseconds. As the output of logic net C2 approaches ground, the output of the register driver will begin to follow the input clock pulses. A small amount of attenuation and lag occur through the register driver, but can be neglected for all practical purposes. No input gating occurs at this time because the output of logic net C3 is at -3 volts. At approximately t = 0.7 microsecond, the output of logic net C3 rises toward ground. Generally, the rise time will be shorter for logic net C3 than for logic net C2.

The register driver pulse occurring at t = 0.8 microsecond will now sample the ground level from logic net C3. The output of the gate will be at -4 volts up until the time it is pulsed. The gate output pulse never actually reaches ground, but nevertheless triggers the flip-flop. The positive input pulse to the flip-flop now

reverses the polarity of the voltage on the 0 and 1 wires coming out of the flip-flop. The delay time for the flip-flop is about equal to the width of the clock pulses; that is, 0.1 microsecond.

Several interesting and very important observations can now be made. First, the state of the flip-flop did not start to change until the clock pulse that caused the change was over. This means that the same clock pulse that was used to change the state of the flip-flop could also have been used to sample the output of the flip-flop before its state changed. On the other hand, the flip-flop state changed quickly enough to allow a clock pulse to complement the state of the flip-flop again. Thus, the flip-flop can change states at a 5-megacycle rate (once every 0.2 microsecond) even though the pulses from any other register driver occur at a maximum 2.5-megacycle rate (once every 0.4 microsecond).

3-7 FLIP-FLOP REGISTERS

3-7.1 GENERAL. A flip-flop register comprises one or more flip-flops and assumes an identity of its own. Flip-flops contain a bit of information and registers contain a word. When one speaks of a 36-bit flip-flop register, he means a register composed of 36 flip-flops.

All the normal things that are done to flip-flops, such as "setting to ZERO", "setting to ONE", and "complementing", are also done to registers. Sometimes the terminology is different. For example, one commonly speaks of "clearing" a register rather than "setting to ZERO", but the idea is the same.

3-7.2 REGISTER TRANSFERS. Register transfers will be treated at the level of data transfers into the individual flip-flops that make up the registers. Fig. 3-20 shows how ZEROS and ONES are gated into the flip-flop by transfer pulses.

3-7.2.1 JAMMING. Data transfer between flip-flops may be accomplished by several different techniques. One technique, called "jam transfer", is illustrated in Fig. 3-21. In this case, the bit of data stored in flip-flop FF_1 is transferred into flip-flop FF_2 by a single transfer pulse. Symbolically, this transfer is written

$$FF_1 \xrightarrow{j} FF_2$$

This transfer is interpreted as "jamming the contents of FF_1 into FF_2".

Two types of jamming circuitry are used (see Fig. 3-21). One type uses an emitter fed input gate. This can be done only when FF_1 is physically close to FF_2 so that the ground in FF_1 is in close proximity to the emitter of the gate. In this case, only one transistor is required for each input. The other jamming circuit uses a base fed input gate and requires two transistors on each input. In this case, FF_1 and FF_2 may be physically distant from one another.

An obvious, but sometimes overlooked fact, is that the data transfer has no effect on the state of the register from which the data is transferred.

3-7.2.2 O'S, 1'S TRANSFER. In this technique, the circuitry is similar to that used in jamming but the transfer pulse for the ZEROS comes from a different register driver than the transfer pulse for the ONES. (See Fig. 3-22.) It is now possible to effect a register transfer by first clearing (that is, "setting to ZERO") all the flip-flops in a register and then transferring ONES. One clear gate on each bit (requiring one transistor per bit) may be used to clear the register before ONES are transferred from several different flip-flops. In this technique, fewer transistors are required than in jamming, but a time penalty is imposed in that two succeeding time pulses are required -- one for clearing and one for setting ONES. It is also obvious that logical transfers requiring separate control of the ZERO and ONE inputs are now possible.

3-7.2.3 INTERNAL REGISTER TRANSFERS. There are several types of internal register transfers which are all "jam transfers" between bits in a register. One of these is called "shifting" and involves transferring individual bits from one flip-flop to either the flip-flop on the immediate right or on the immediate left; hence the names "shift right" and "shift left". Arithmetic Element registers in TX-2 are often considered as closed rings, in which case the bits shifted off the right or left end of a register are jammed into the opposite end of the register. Under certain other conditions, the registers may be broken into several smaller rings in which case the bits are shifted around these rings.

Another type of internal register transfers is called "permuting" and consists of interchanging subwords of the register. This is done by jamming the bits of each subword into the corresponding bits of its permuted subword.

3-7.3 LOGICAL TRANSFERS. This operation involves the logical combination of two registers. The result is either stored in one of the two registers or in a third register.

3-7.3.1 PARTIAL ADD (EXCLUSIVE OR). Fig. 3-23 shows the truth table for a partial add operation and indicates how it is performed. Note that the state of FF_2 is changed only when FF_1 is in the ONE state. When FF_1 is in the ZERO state, FF_2 is unaffected. The logic is performed by means of the complement input and is executed by a single pulse.

3-7.3.2 LOGICAL SUM (INCLUSIVE OR). In this case, two registers are combined and the result stored in a third register. (See Fig. 3-24.) With this circuit, the operation requires two successive time pulses. On the first time pulse, flip-flop FF_3 is cleared. The logical sum pulse then performs the logical transfer itself. Note that an OR inverter circuit is used, so that when

either FF_1 or FF_2 is in the ONE state (FF_1^1 or FF_2^1), FF_3 is set to ONE. When both FF_1 and FF_2 are in the ZERO state, the output of the inverter is tied to ground and the logical sum pulse has no effect on FF_3. Note also that the 1's transfer of Fig. 3-22 is performing the logical sum of FF_1 and FF_2 but with the results stored in FF_2.

3-7.3.3 PARTIAL ADD TRANSFER (THREE REGISTERS, TWO PULSES). In this case, the result of the partial addition of two registers is stored in a third register. (See Fig. 3-25.) It is first necessary to clear the register in which the result is to be stored. This is accomplished by a clear pulse. The "exclusive or" logic itself is performed by the cross-tied four-transistor inverter circuit. A -3 volt output from the inverter occurs when both top transistors or both bottom transistors have their inputs grounded. Note that this occurs in only two of the four possible state combinations of FF_2 and FF_1.

3-7.3.4 PARTIAL ADD TRANSFER (THREE REGISTERS, ONE PULSE). In this case, the result that was accomplished with two time pulses in Fig. 3-25 is obtained with one time pulse. This increase in speed is obtained at the expense of additional transistor circuitry. (See Fig. 3-26.) Note that an input pulse is applied to FF_3 in all four possible cases. Two of the cases effect the "set to one" input, and two of the cases effect the "set to zero" input.

3-8 REGISTER DECODERS

If a register contains n flip-flops, the register as a whole may be in any one of 2^n states. For example, a 3-bit register may be in any one of $2^3 = 8$ states. Each of these register states may have a name, and it may be necessary to recognize and indicate each of these states.

Fig. 3-27 shows a typical "register decoder" in which the decoding networks are simple ANDing emitter follower circuits. There is an output wire for each state of the register and the wire is labelled according to the name of the particular register state. All the output wires will be at -3 volts, except the wire that represents the present register state. This wire will be at ground.

Note that the register states may have completely arbitrary names. The figure shows the states named sequentially (both numerically and alphabetically) according to the state of the register, which is represented as a binary number.

3-9 COUNTER AND TIME LEVEL DECODERS

3-9.1 GENERAL. A counter is a register with a count circuit as an input. A time level decoder is an ordinary register decoder.

3-9.2 COUNTERS. Fig. 3-28(a) shows a typical counter. The principle of operation is that each count pulse indexes the counter by one. When all the flip-flops contain ONES, the next count pulse resets all the flip-flops to ZERO. Each of the states of the register is given a time name, such as K^0, K^1, K^2, K^3, etc. Fig. 3-28(b) shows a variation on the simple counter just described. In this case, K_1, K_2, and K_3 form the indexing counter. K_4 is independently controlled and effectively determines how the indexing counter state should be interpreted. This is clear from the accompanying table. Note that both counters in Fig. 3-28 contain four flip-flops and are capable of registering $2^4 = 16$ different states.

3-9.3 TIME LEVEL DECODERS. Fig. 3-29 shows how the state of the counter is decoded. The pulses for the α counter come from register drivers with α clock pulses. Note that the inputs to the counter are time pulses and that the outputs of the time level decoder are levels representing time states.

Actually, it is convenient to have β time levels as well as α time levels. The β time levels occur just 0.2 microsecond after the corresponding α time levels. This is accomplished by gating the α counter register into a β counter register with β pulses. Thus, the α register contents appear in the β register, but delayed 0.2 microsecond from the time they appeared in the α register. The β register is then decoded by a β time level decoder.

Each time state (0α, 1α, etc.) has two complementary output wires, so that the counter state can be represented by either a ground level or a -3 volt level.

3-10 REGISTER DRIVER CONTROL NETS

A somewhat special type of logical notation is used when discussing register drivers. In some cases, it is more advantageous to indicate the logic that inhibits pulses from the register driver rather than the logic that permits pulses from the register driver. This is shown in Fig. 3-30. In order that a pulse appear at the output of the register driver, it is necessary that both register driver level inputs be at ground. Fig. 3-30(a) shows a typical permissive type of register driver control circuit. When either A or B is -3 volts, both inputs to the register driver are at ground and a pulse appears at the output of the register driver. On the other hand, when both inputs to the logic net are at ground, no pulse appears at the output of the register driver.

Fig. 3-30(b) shows a typical inhibitory type of register driver control circuit. First look at the inputs and output of the register driver. They indicate that, when either input is at -3 volts, no pulse appears at the output of the register driver that will transfer the contents of flip-flop FF_1 into FF_2. One of these register driver inputs will be at -3 volts when A and B are not present and C is present in one of the logic nets, or when D and E are present in the other logic net.

The question now arises as to when a transfer pulse will appear at the output of the register driver. It will occur when both register driver inputs are at ground. This in turn will occur when A or B is present and C is not present in one logic net, and D or E is not present in the other logic net. By "present", it is meant that the voltage on the wire is the same as the voltage used to represent the truth state of the uncomplemented variable.

3-11 OTHER COMPUTER CIRCUITRY

3-11.1 GENERAL. The other miscellaneous special-purpose circuitry used in the computer falls into two categories: (1) high-speed circuitry generally used in the central computer; and (2) low-speed circuitry generally used in the In-Out Element. The pulses used in the low-speed circuitry tend to look more like the levels used in the high-speed circuitry. That is, instead of being 0.1-microsecond wide, the pulses are approximately 0.4-microsecond wide.

3-11.2 SYNCHRONIZER. The information from pushbutton controls on the console must be synchronized with the basic clock-rate activity in the central computer. The circuit for doing this is shown in Fig. 3-31. A similar low-speed type synchronizer is used in the In-Out Element to synchronize information from the external world.

The logical operation of the synchronizer is as follows: An asynchronous pulse appears at the "set to one" input of FF_1. A -3 volt level will appear asynchronously at the output of FF_1. Since FF_2 is in the ZERO state at the time FF_1 is set to ONE, the synchronous clock pulse can trigger the "set to one" pulse of FF_2 synchronously. When this clock pulse appears, FF_2 is in the ZERO state, so that the open gate on the "set to zero" input of FF_1 and FF_2 prevents the pulse from effecting these inputs. However, by the next clock pulse, FF_2 is in the ONE state. Both FF_1 and FF_2 will now be set to ZERO. The time relationship of these events is shown in Fig. 3-32.

Note, in Fig. 3-32, that the output on the 1 wire of FF_2 is a synchronous level exactly 0.4 microsecond wide and positioned with respect to the clock pulses. The asynchronous input pulses cannot appear closer together than 0.8 microsecond. Also note that the cycle is completed by both FF_1 and FF_2 being "set to zero" by the same clock pulse. Both flip-flops stay in the ZERO state until the next asynchronous pulse requiring synchronization appears.

3-11.3 PULSE DELAY LINE. Fig. 3-33 illustrates the characteristics of a typical pulse delay line. For the delay line shown, the input pulse can be delayed in twenty discrete 20-millimicrosecond intervals. Thus, the pulse appearing at output No. 1 occurs exactly 120 millimicroseconds after a pulse appeared at the input. Similarly, the pulse appearing at output No. 2 occurs exactly 400 millimicroseconds after a pulse occurs at the input. Pulse delay lines are used to solve some of the timing problems found in the memory systems. They are also used in other parts of the computer.

3-11.4 GATED PULSE AMPLIFIER. This is a two-stage transformer-coupled amplifier for 0.1-microsecond negative pulses. The output is capable of driving 10 bases and a 100-ohm termination. It is used to amplify the weak outputs of delay lines and for amplification purposes in some of the in-out circuitry.

3-11.5 VARIABLE DELAY UNIT. Fig. 3-34 shows a typical variable delay unit. A 0.1-microsecond negative pulse applied to the input provides a -3 volt output level for a period continuously variable from 0.3 microsecond to 2.2 seconds. The coarse ranges are set by means of switching capacitors, and a potentiometer provides the fine time setting. The end of the delay can be made to occur synchronously with a clock pulse by use of the added external inverter shown in Fig. 3-34. Fig. 3-35 shows time characteristics of the variable delay unit.

3-11.6 LOW-SPEED FLIP-FLOP AND CAPACITOR-DIODE GATE. The low-speed flip-flop is a simple transistor version of the Eccles-Jordan flip-flop with "set to zero" and "set to one" inputs applied directly to the base terminals of the transistors through capacitor-diode gates. (See Fig. 3-36.) If the flip-flop is in the ONE state, the "one" output is at -3 volts and inverter Q1 is saturated. If the base of Q1 is driven positive, thereby making its base current zero, then Q1 will become an open circuit and its output will turn on Q2, thus reversing the state of the flip-flop. Pulses which do this are formed, gated, and coupled into the base of the flip-flop by the capacitor-diode gate.

The input pulses to the capacitor-diode gate are wide positive pulses normally at -3 volts. The circuit forms a pulse from the rising edge of the input and applies it to the flip-flop if the gate input is at ground, but does not couple it if the gate is at -3 volts.

There are several limitations on the use of the low-speed flip-flop and capacitor-diode gate. First, the pulse input signal must not have an amplitude greater than 3.6 volts. Next, the pulse input to this circuit draws 2 ma. The gate input draws 1 ma and does not gate off the input pulse immediately, but only after 1.2 microseconds. Finally, the flip-flop has an uncontrolled delay which may be shorter than 0.1 microsecond and has a rise time of 0.2 microsecond or better. It is capable of driving five "standard loads". As a result of these limitations, the circuit is used only on in-out equipment where necessary operating speeds are less than 600 kilopulses per second.

3-11.7 0.3 MICROSECOND PULSE FORMER. This unit is essentially a low-speed flip-flop which has been modified so that it is stable in the ONE state only. When a pulse is applied to its input via a capacitor-diode gate, the unit goes to the opposite state for 0.3 microsecond, but then automatically falls back to its rest state. Operation is therefore similar to a one-shot multivibrator.

3-11.8 SCHMIDT TRIGGER. This is a static hysteresis-type circuit with input-output characteristics as shown in Fig. 3-37. The circuit provides an output between ground and -3 volts that is independent of the rise and fall time of the input. The output is inverted with respect to the input. The circuit triggers when a rising input reaches 0.9 volt (the output then drops to the -3 volt level in less than 0.15 microsecond). It also triggers when a falling input reaches -2.2 volts (the output then rises from -3 volts to ground in less than 0.1 microsecond).

3-11.9 INPUT MIXER AND OUTPUT DISTRIBUTOR. These two units provide complementary functions in the In-Out Element. The input mixer selects a particular in-out unit and routes one of its output signals to the central computer. Several mixers use a common amplifier to drive their output over coaxial cable to the E register. The output distributor, on the other hand, routes a signal from the central computer to a particular preselected in-out unit. The distributor drives the central computer signal over coaxial cable from the Sequence Switch to the In-Out Control Box.

A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

Logical AND

$A \cdot B$

(A and B)

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Inclusive OR

$A + B$

(A or B, or both)

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive OR

$A \oplus B$

(A or B, but not both)

A	$\bar{A}$
0	1
1	0

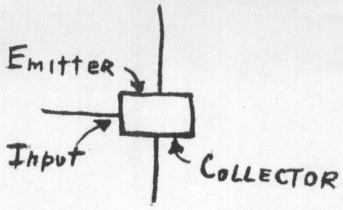
Complement

$\bar{A}$

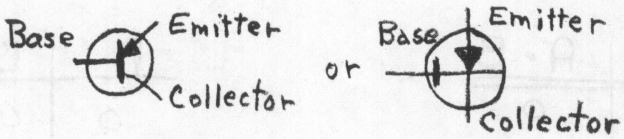
(Not A)

FIG. 3-1 TRUTH TABLES FOR COMMON LOGICAL FUNCTIONS

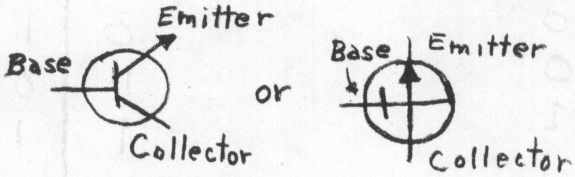
SINGLE-TRANSISTOR LOGIC ELEMENT



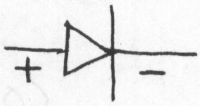
PNP Transistor



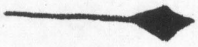
NPN Transistor

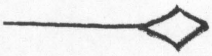


DIODE



LEVELS:

−3 VOLTS 

0 VOLTS (GRD) 

PULSES:

NEGATIVE 

POSITIVE 

SUPPLY VOLTAGES:

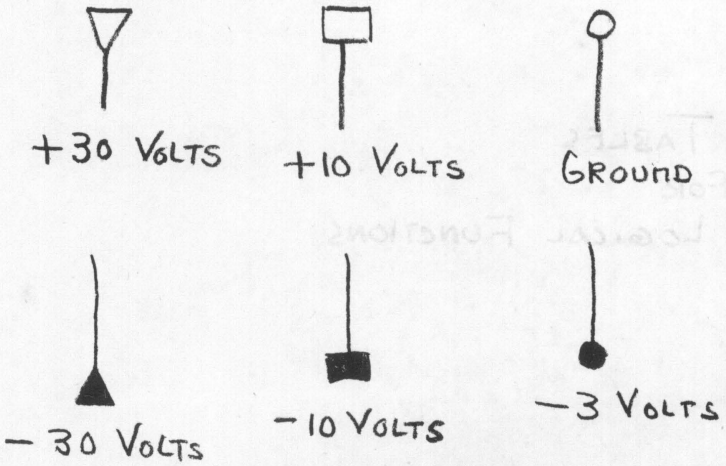
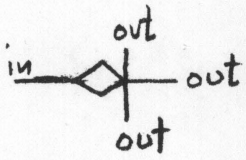


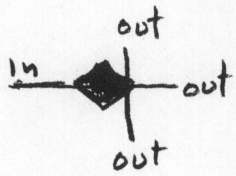
FIG. 3-2 TX-2 BLOCK SYMBOLS

CONNECTIONS

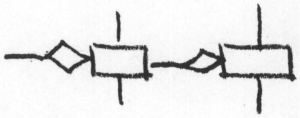
no connection



GRD LEVEL BRANCH



-3 volt BRANCH



is equivalent to

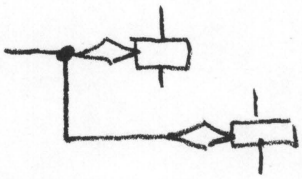
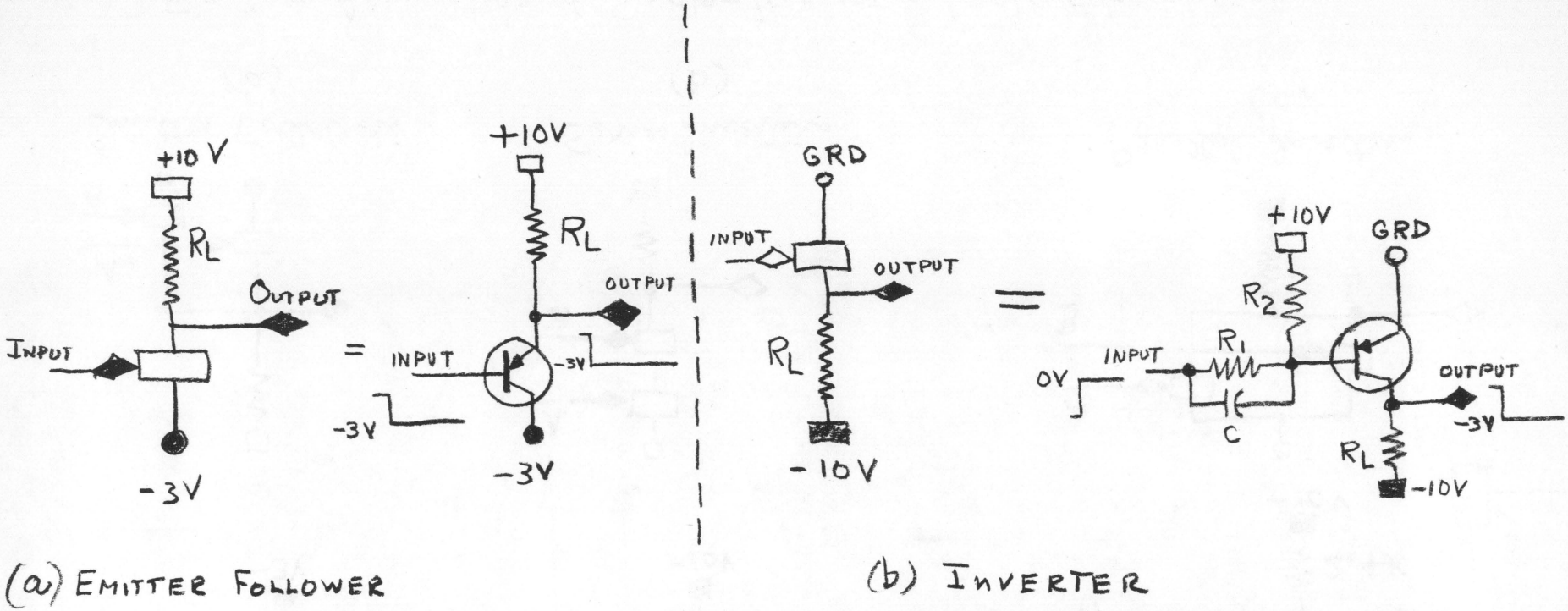


FIG. 3-2. TX-2 BLOCK SYMBOLS
(continued)



(a) EMITTER FOLLOWER

(b) INVERTER

FIG 3-3 BASIC EMITTER FOLLOWER AND INVERTER CIRCUITS

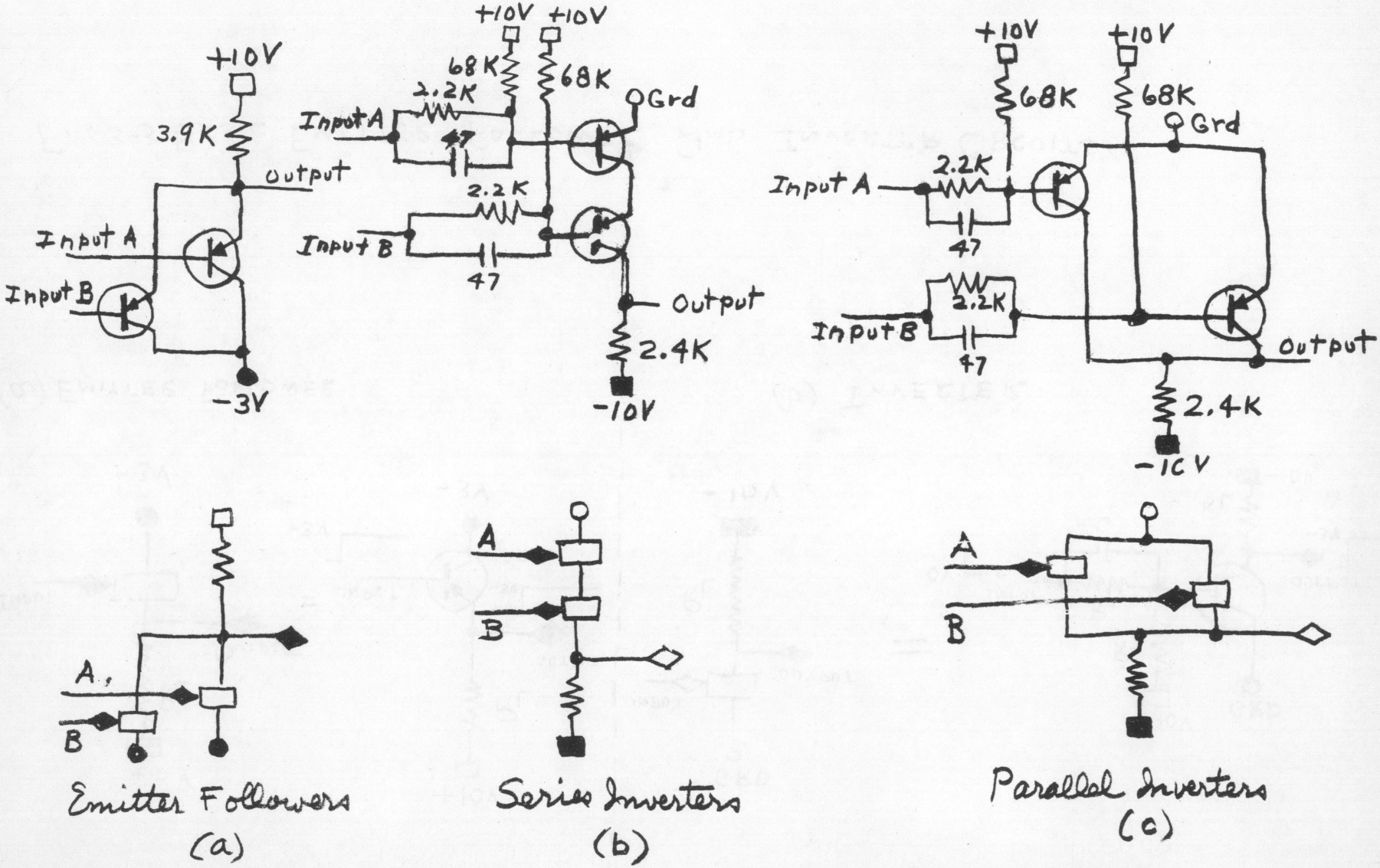
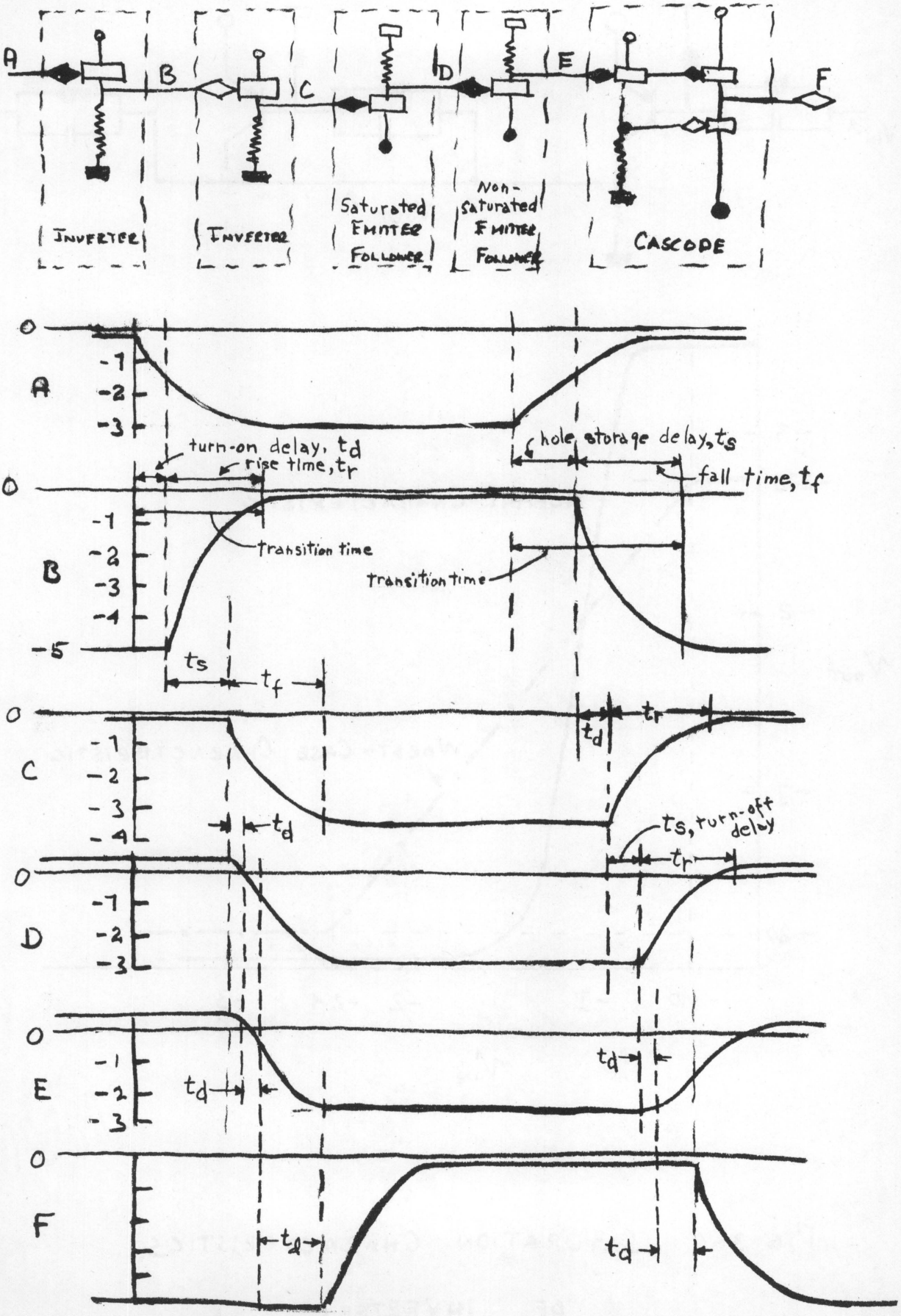
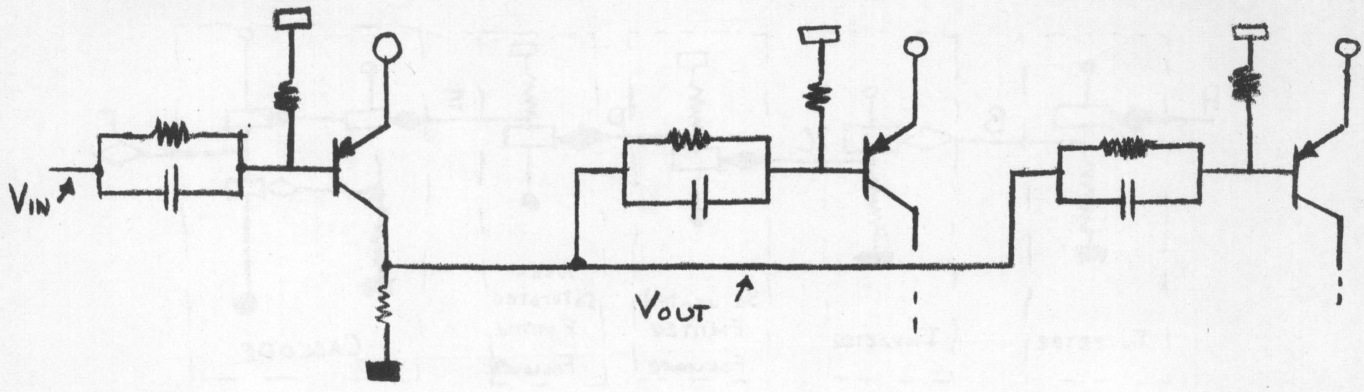


Fig. 3-4 Application of Single-Transistor Logic Elements to D-c Logic Nets

FIG. 3-5 TRANSISTOR CIRCUIT TIME CHARACTERISTICS





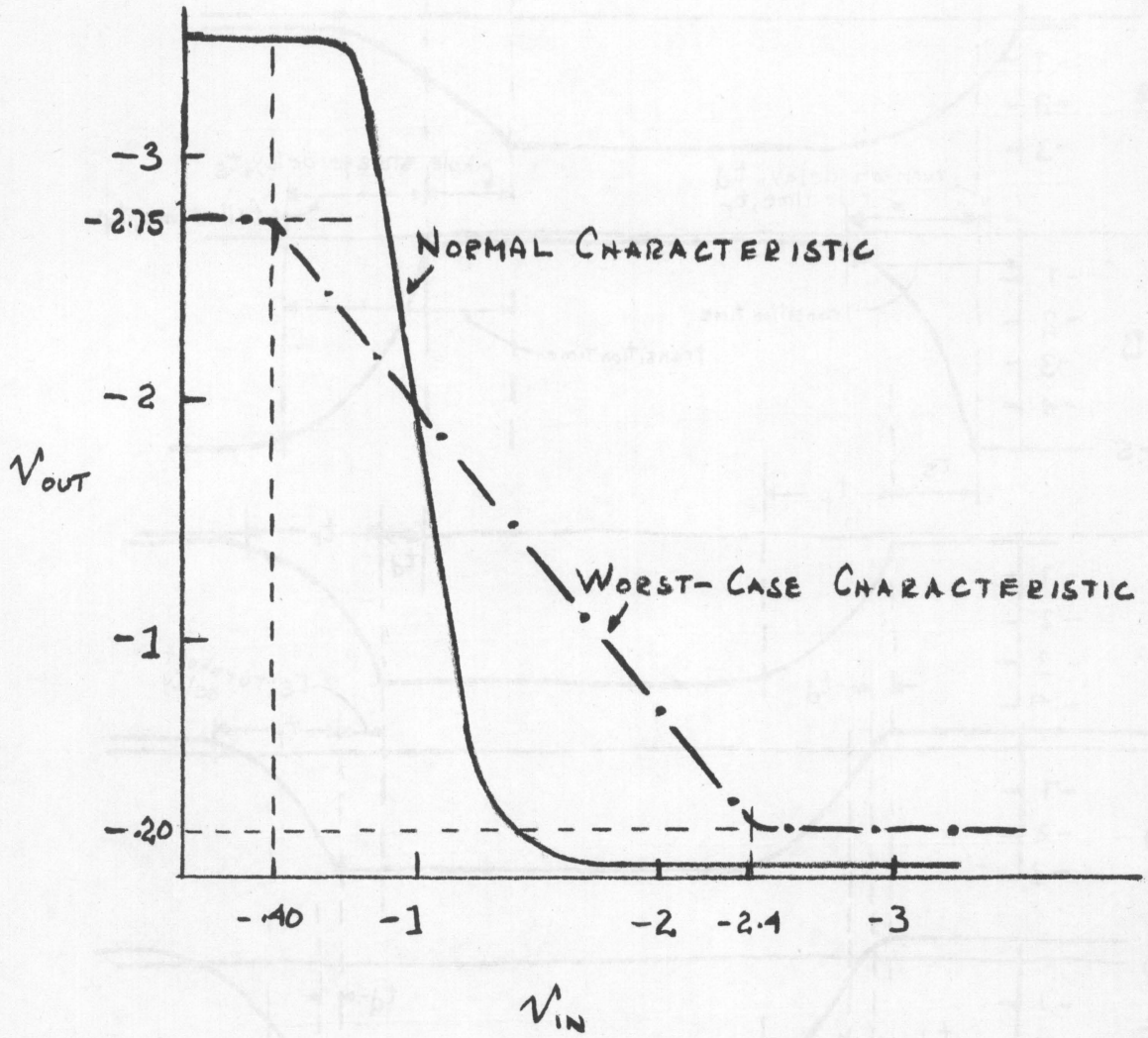


FIG 3-6 SATURATION CHARACTERISTICS OF INVERTERS

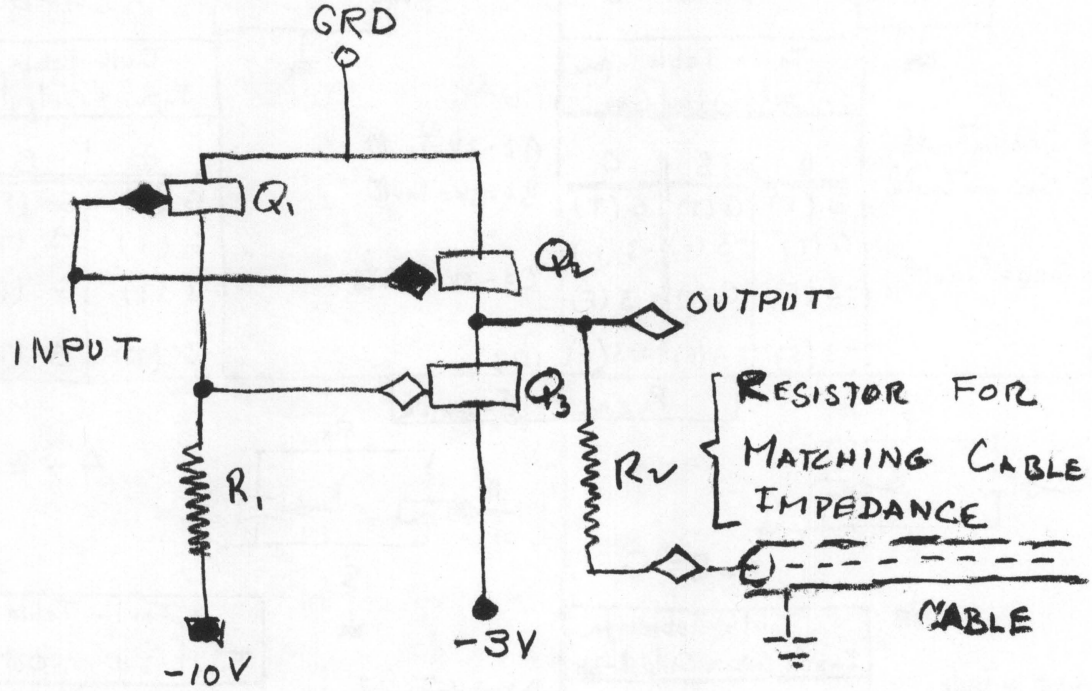
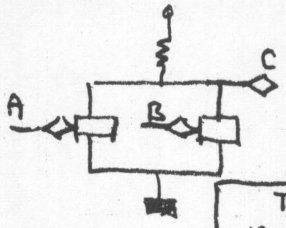


FIG. 3-7 CASCODE

PARALLEL EMITTER FOLLOWER



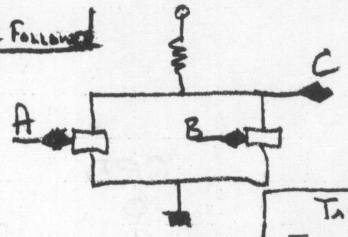
$A \cdot B = C$

A: GRD = Truth
B: GRD = Truth
C: GRD = Truth

Truth Table for Input + Output GRD

A	B	C
G (T)	G (T)	G (T)
G (T)	-3 (F)	-3 (F)
-3 (F)	G (T)	-3 (F)
-3 (F)	-3 (F)	-3 (F)

(a)



$A + B = C$

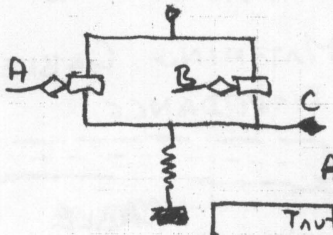
A: -3V = Truth
B: -3V = Truth
C: -3V = Truth

Truth Table for Input + Output - 3V

A	B	C
G (F)	G (F)	G (F)
G (F)	-3 (T)	-3 (T)
-3 (T)	G (F)	-3 (T)
-3 (T)	-3 (T)	-3 (T)

(b)

PARALLEL INVERTER



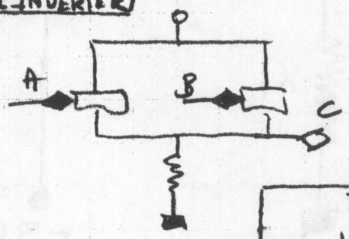
$A \cdot B = C$

A: GRD = Truth
B: GRD = Truth
C: -3V = Truth

Truth Table for Input GRD + Output -3V

A	B	C
G (T)	G (T)	-3 (T)
G (T)	-3 (F)	G (F)
-3 (F)	G (T)	G (F)
-3 (F)	-3 (F)	G (F)

(c)



$A + B = C$

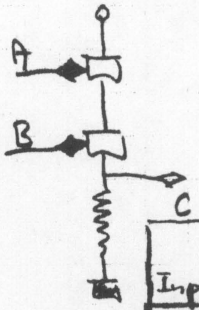
A: -3V = Truth
B: -3V = Truth
C: GRD = Truth

Truth Table for Input -3V + Output GRD

A	B	C
G (F)	G (F)	-3 (F)
G (F)	-3 (T)	G (T)
-3 (T)	G (F)	G (T)
-3 (T)	-3 (T)	G (T)

(d)

SERIES INVERTER



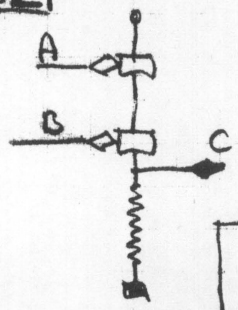
$A \cdot B = C$

A: -3V = Truth
B: -3V = Truth
C: GRD = Truth

Truth Table for Input -3.25V + Output GRD

A	B	C
G (F)	G (F)	-3 (F)
G (F)	-3 (F)	-3 (F)
-3 (T)	G (F)	-3 (F)
-3 (T)	-3 (T)	G (T)

(e)



$A + B = C$

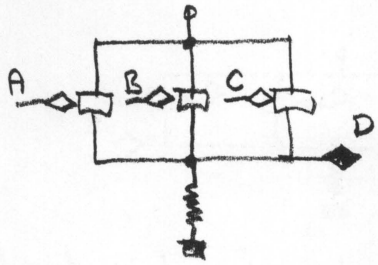
A: GRD = Truth
B: GRD = Truth
C: -3V = Truth

Truth Table for Input GRD + Output -3V

A	B	C
G (T)	G (T)	-3 (T)
G (T)	-3 (F)	-3 (T)
-3 (F)	G (T)	-3 (T)
-3 (F)	-3 (F)	G (F)

(f)

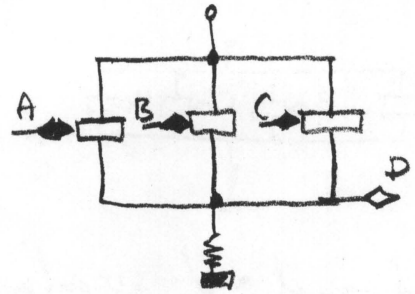
Fig 3-8 TWO-TRANSISTOR LOGIC



For gnd input and -3v output,

$$A \cdot B \cdot C = D$$

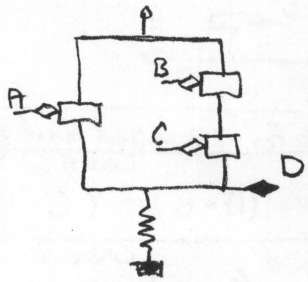
(a)



For -3v input and gnd output,

$$A + B + C = D$$

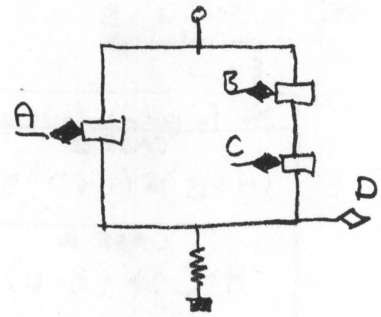
(b)



For gnd input and -3v output,

$$A \cdot (B + C) = D$$

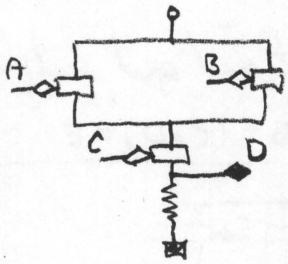
(c)



For -3v input and gnd output

$$A + (B \cdot C) = D$$

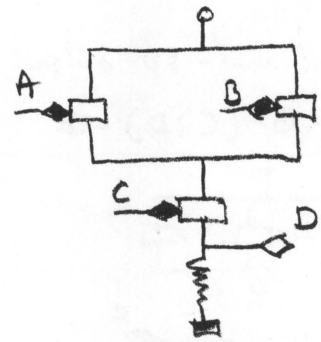
(d)



For gnd input and -3v output,

$$(A \cdot B) + C = D$$

(e)

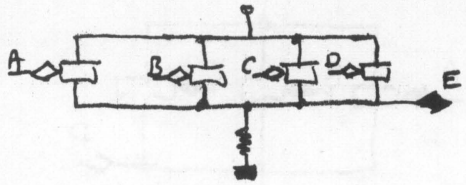


For -3v input and gnd output,

$$(A + B) \cdot C = D$$

(f)

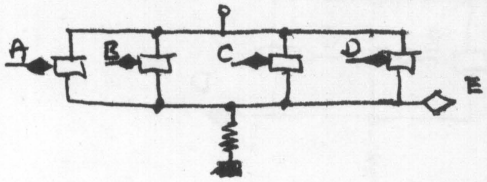
Fig. 3-9 THREE-TRANSISTOR LOGIC



For gnd input and −3v output,

$A \cdot B \cdot C \cdot D = E$

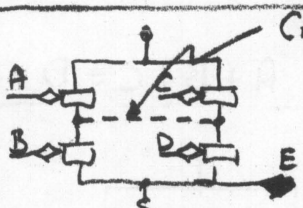
(a)



For −3v input and gnd output,

$A + B + C + D = E$

(b)



For gnd input + −3v output

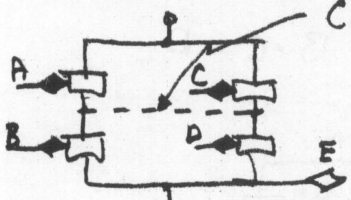
CASE 1

$(A+B) \cdot (C+D) = E$

CASE 2

$(A \cdot C) + (B \cdot D) = E$

(c)



For −3v input and gnd output

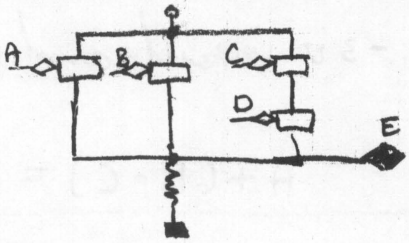
CASE 1

$(A \cdot B) + (C \cdot D) = E$

CASE 2

$(A+C) \cdot (B+D) = E$

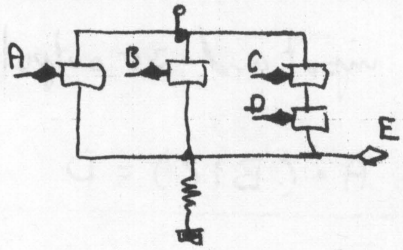
(d)



For gnd input and −3v output,

$A \cdot B \cdot (C+D) = E$

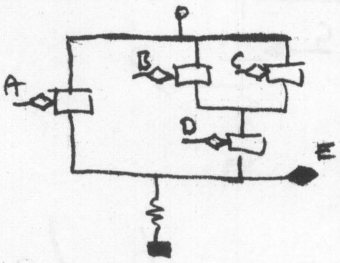
(e)



For −3v input and gnd output,

$A + B + (C \cdot D) = E$

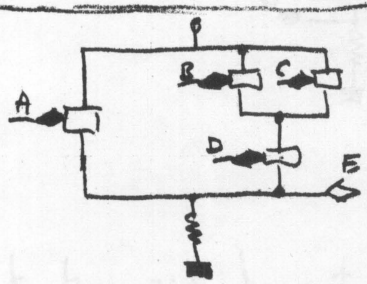
(f)



For gnd input and −3v output,

$A \cdot [(B \cdot C) + D] = E$

(g)

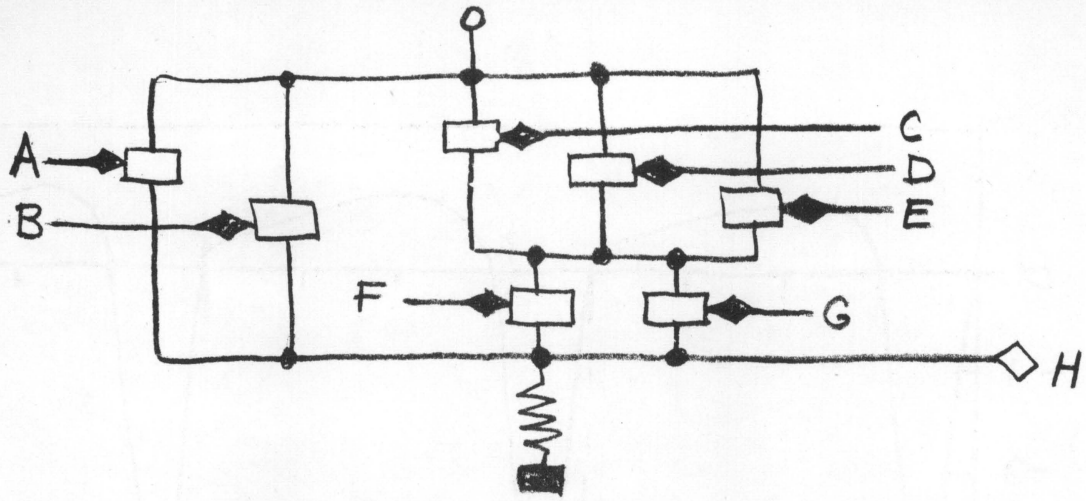


For −3v input and gnd output,

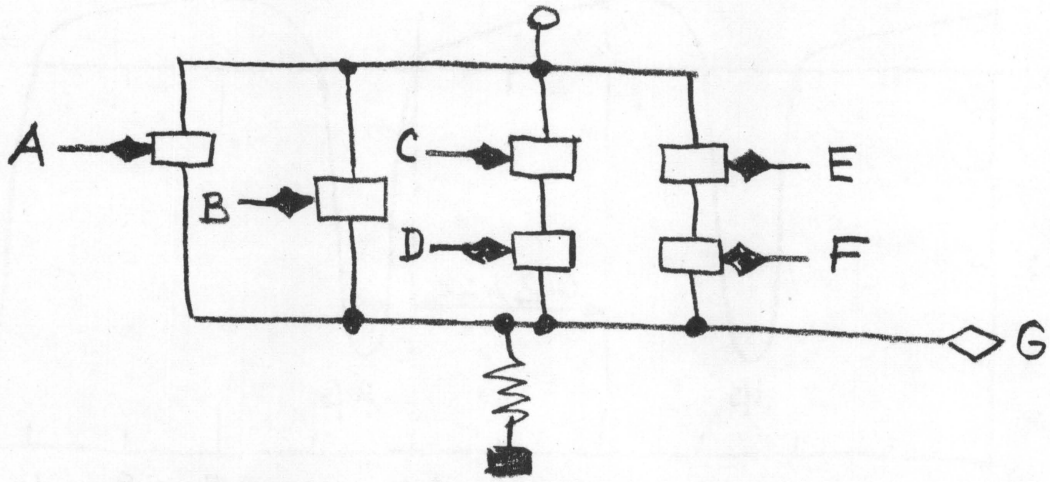
$A + [(B+C) \cdot D] = E$

(h)

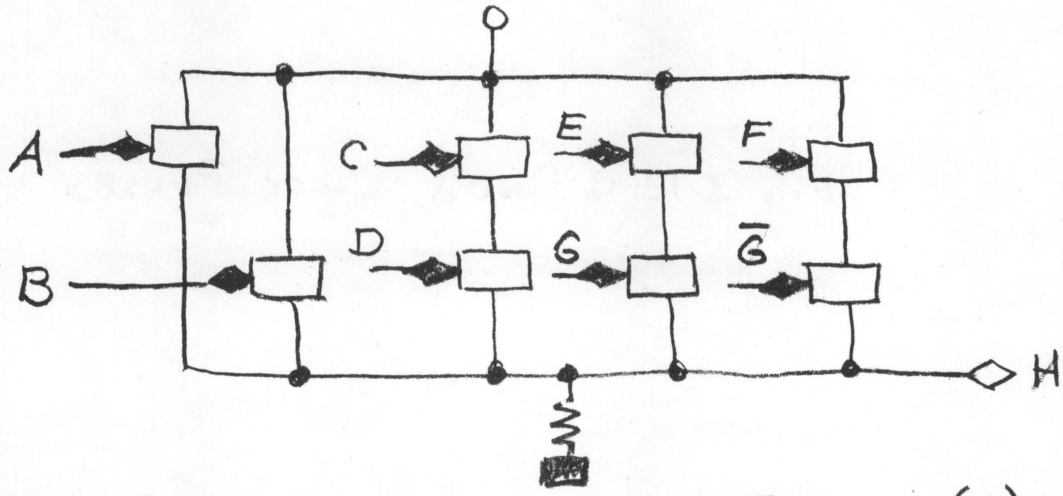
FIG. 3-10 FOUR-TRANSISTOR LOGIC



$$H = A + B + (C + D + E) \cdot (F + G) \qquad (a)$$



$$G = A + B + C \cdot D + E \cdot F \qquad (b)$$



$$H = A + B + C \cdot D + E \cdot G + F \cdot \overline{G} \qquad (c)$$

Fig 3-11 TYPICAL LARGER NETS

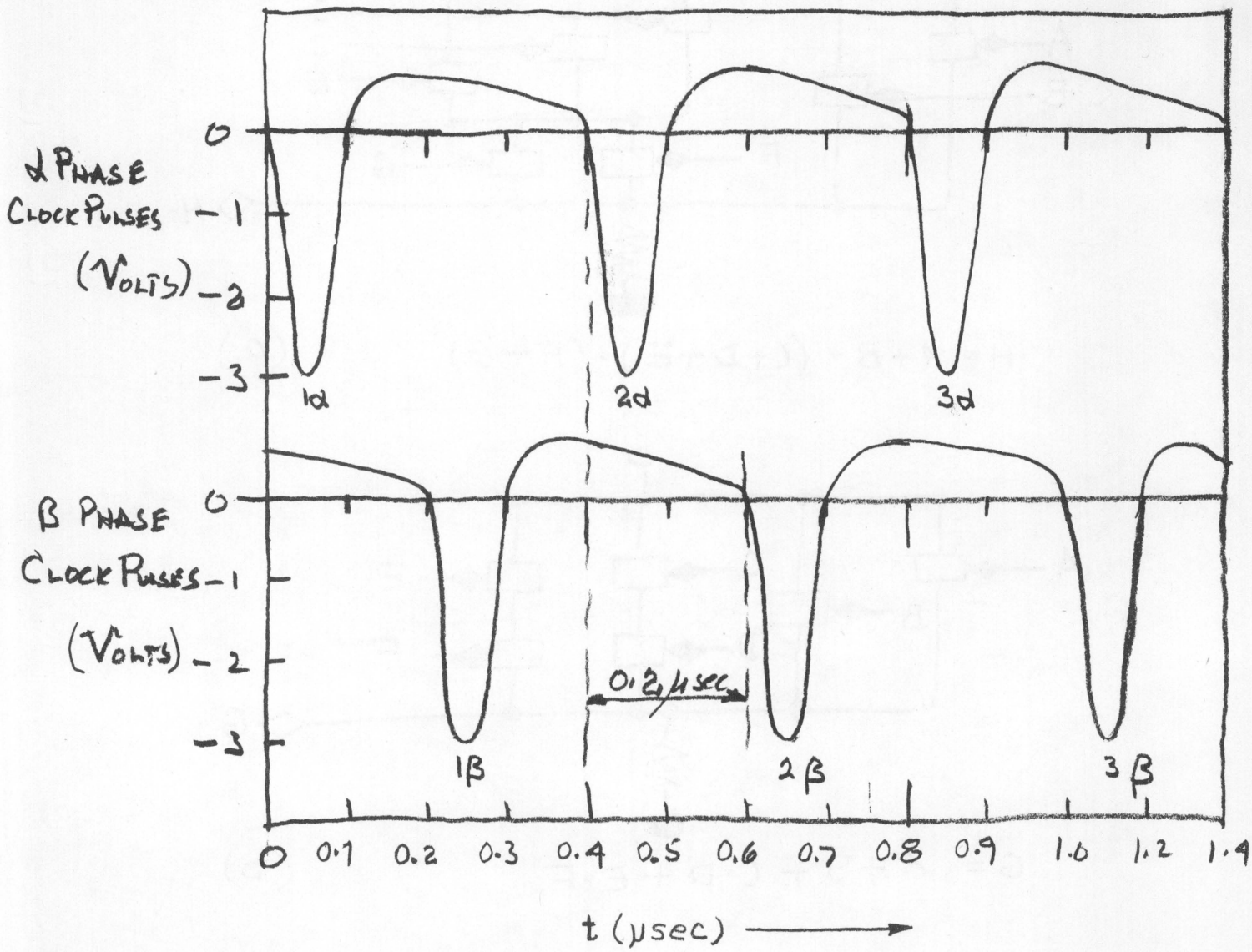


Fig 3-12 α And β Clock Pulses

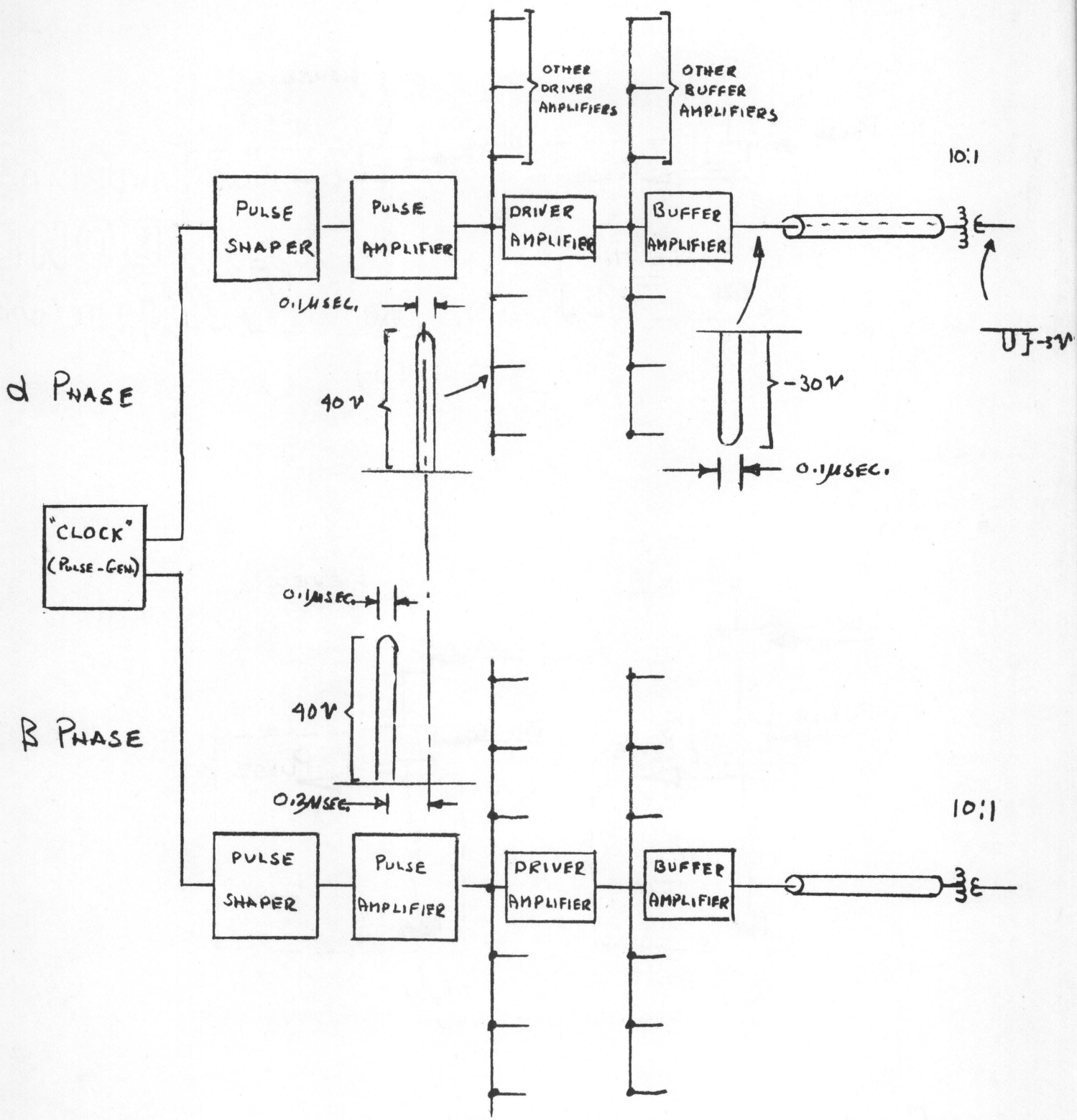


FIG. 3-13 α AND β PULSE - GENERATOR SYSTEM

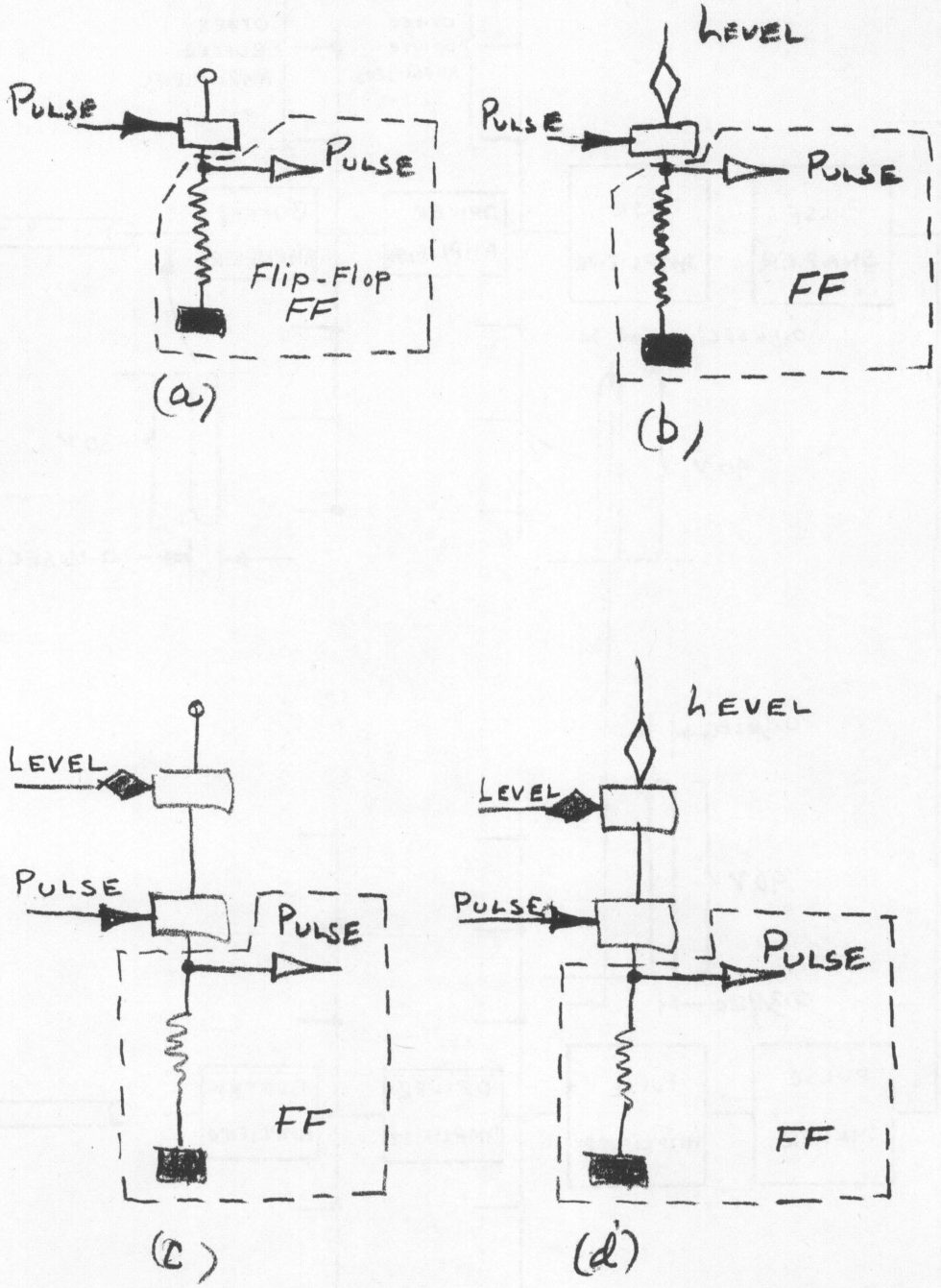
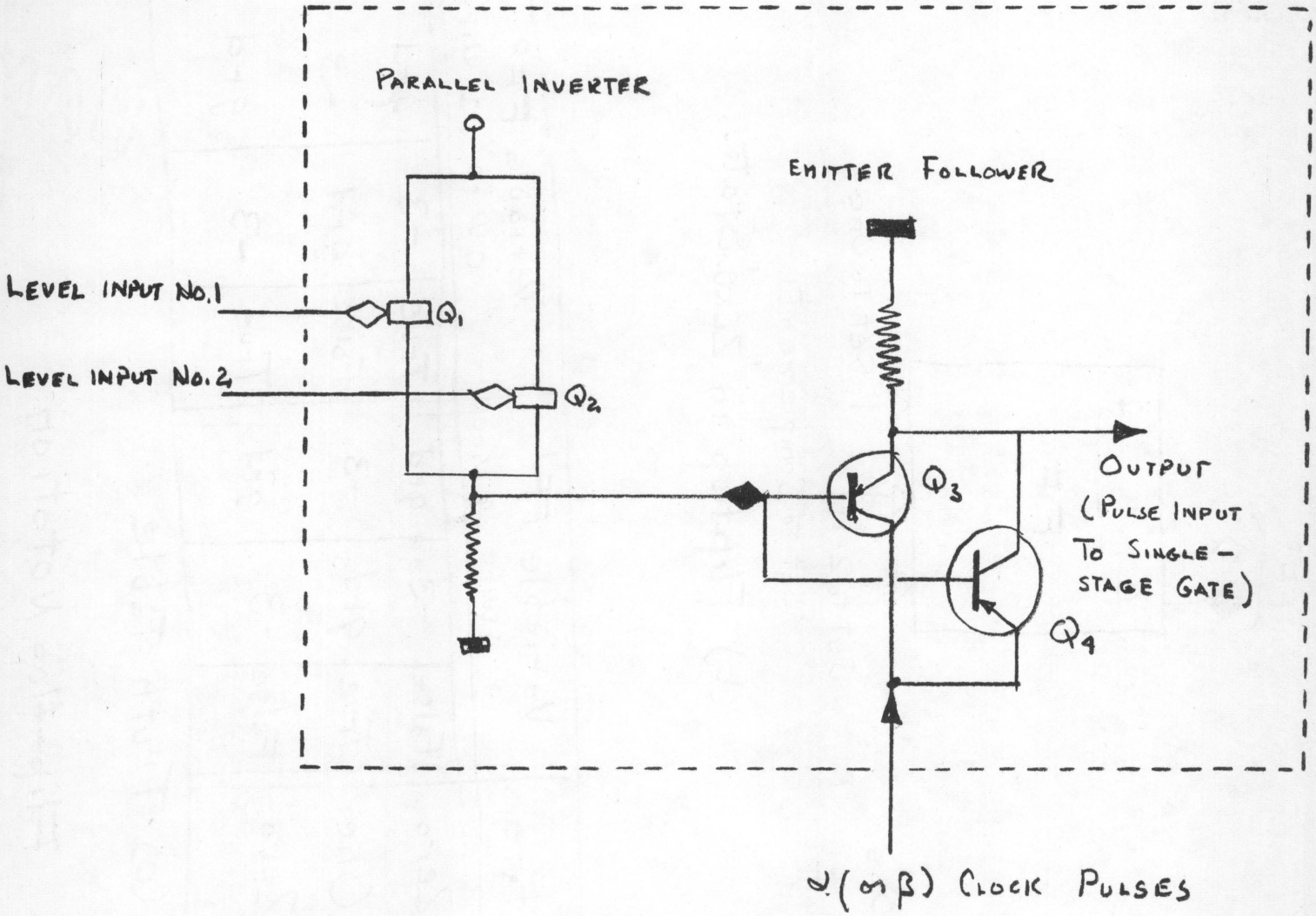
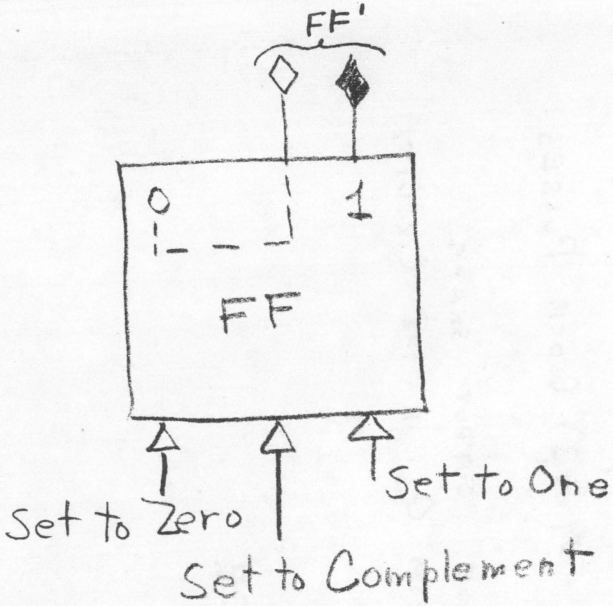


FIG. 3-14 SINGLE - STAGE PULSE GATES

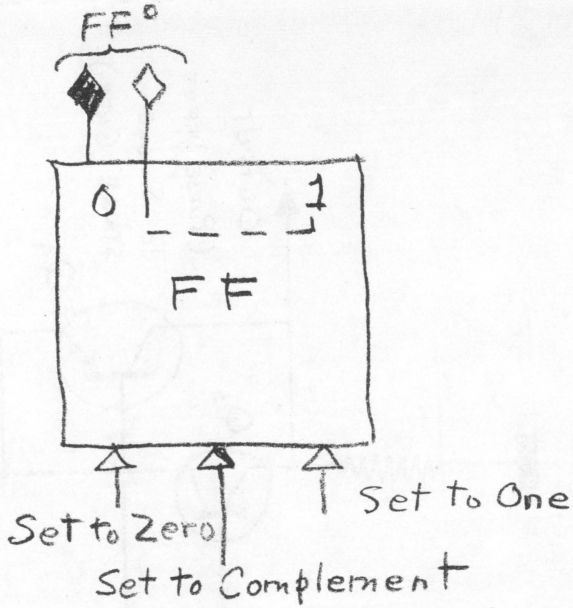


NOTE: TO IMPROVE THE PULSE OUTPUT SHAPE Q_4 IS INVERTED FROM Q_3 IN THE CIRCUIT.

FIG. 3-15 REGISTER DRIVER



(a) Flip-flop in "ONE" State



(b) Flip-flop in "ZERO" State

Input Sequence	Input	State	Variable FF' 0 wire	Variable FF' 1 wire		Variable FF° 0 wire	Variable FF° 1 wire	
t_n	Set to Zero	Zero	False	-3	grd	True	-3	grd
t_{n+1}	Set to One	One	True	grd	-3	False	grd	-3
t_{n+2}	Set to Complement	Zero	False	-3	grd	True	-3	grd

(c) Truth Table

Fig. 3-16 Flip-flop Notation

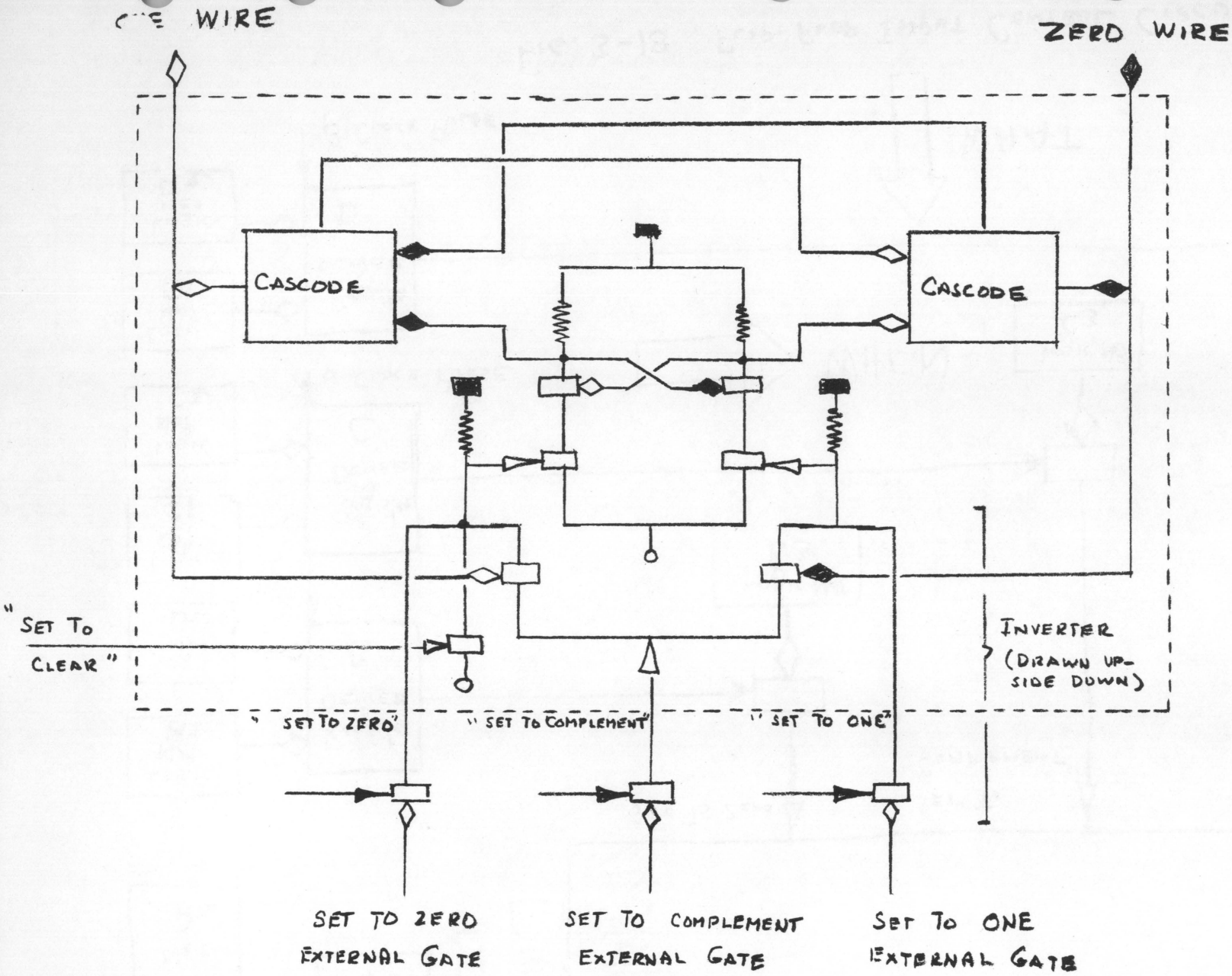


FIG. 3-17 BASIC FLIP-FLOP CIRCUIT

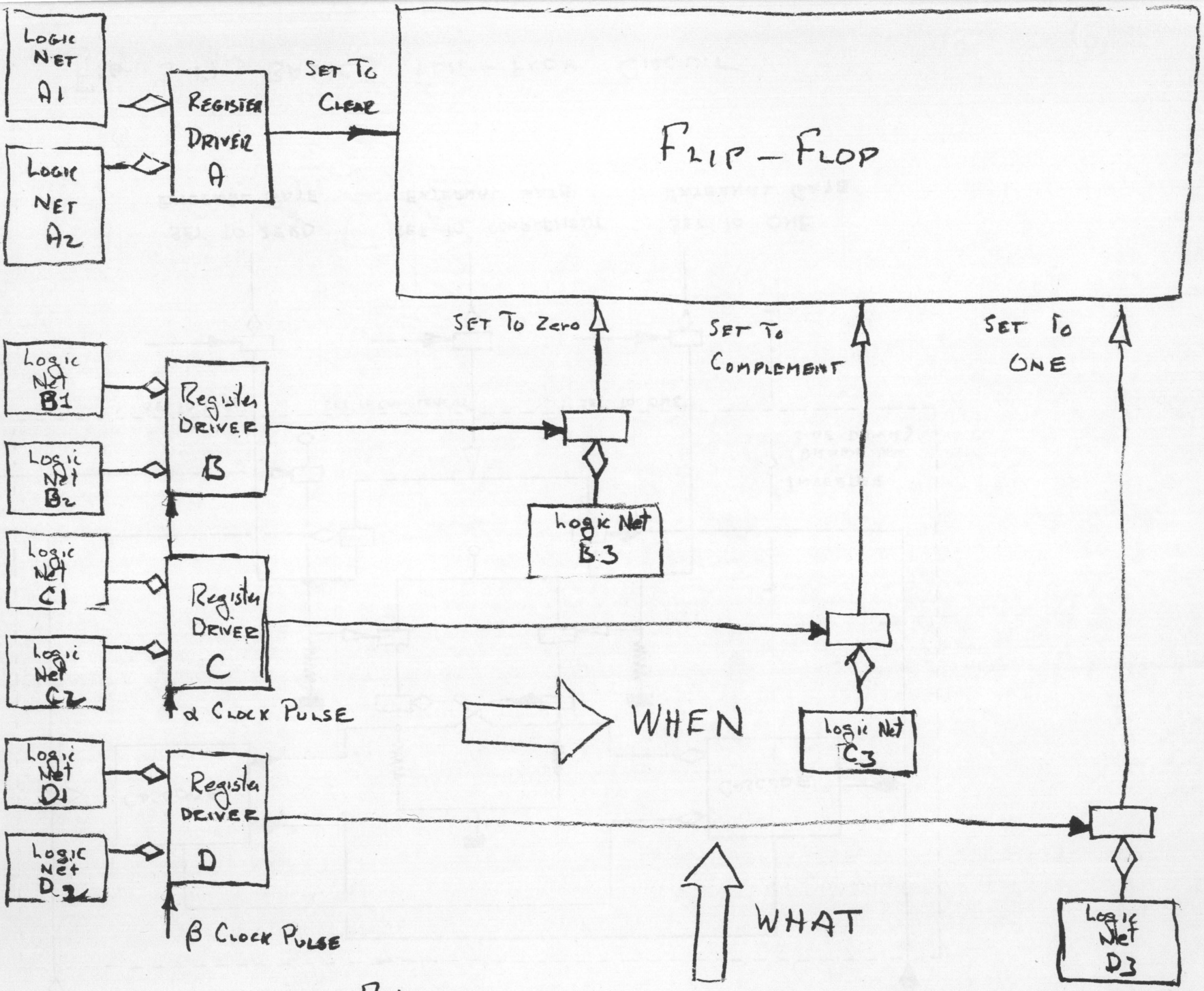
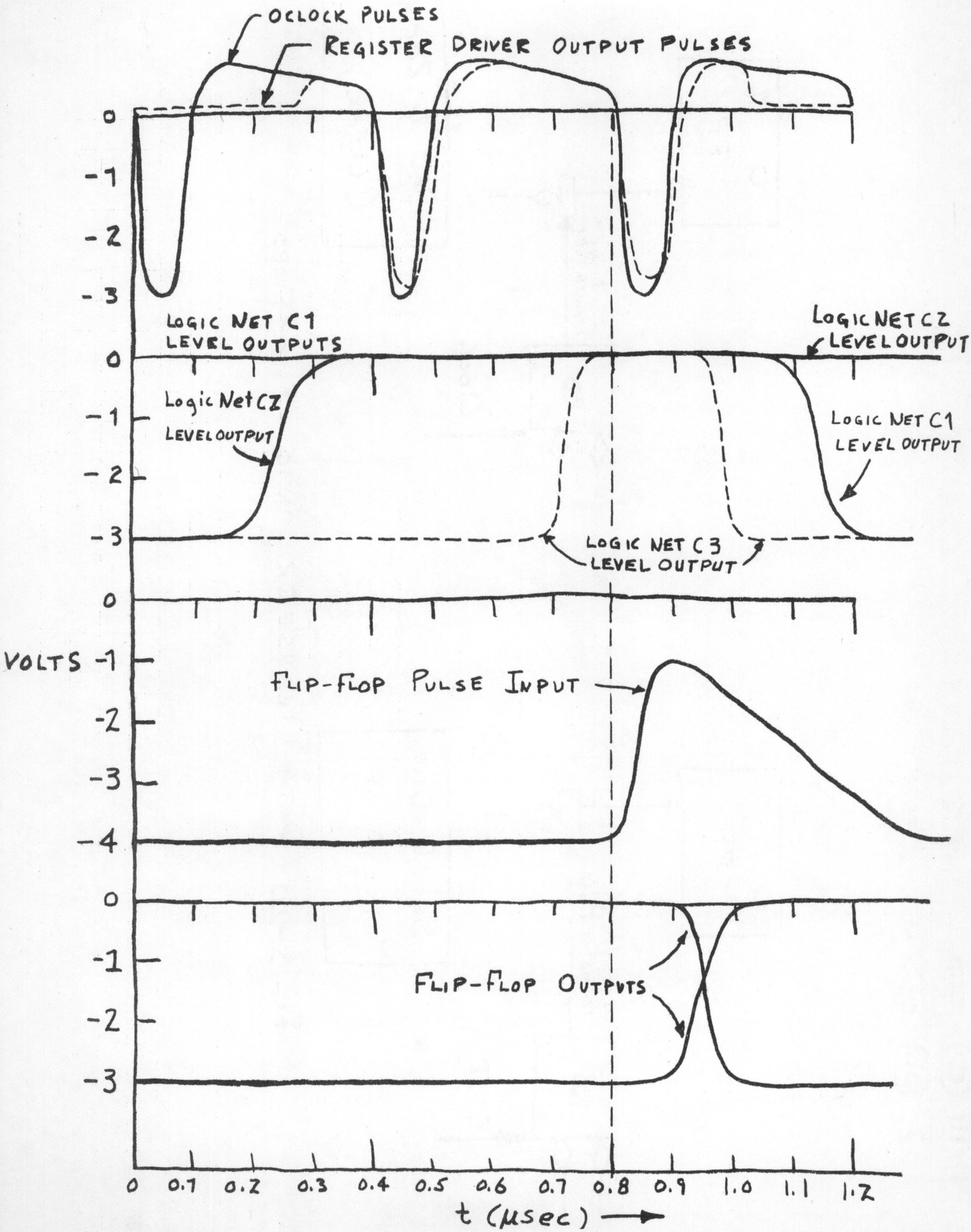


Fig. 3-18 Flip-Flop Input Control Circuitry

FIG 3-19 WAVEFORMS FOR FIG 3-18



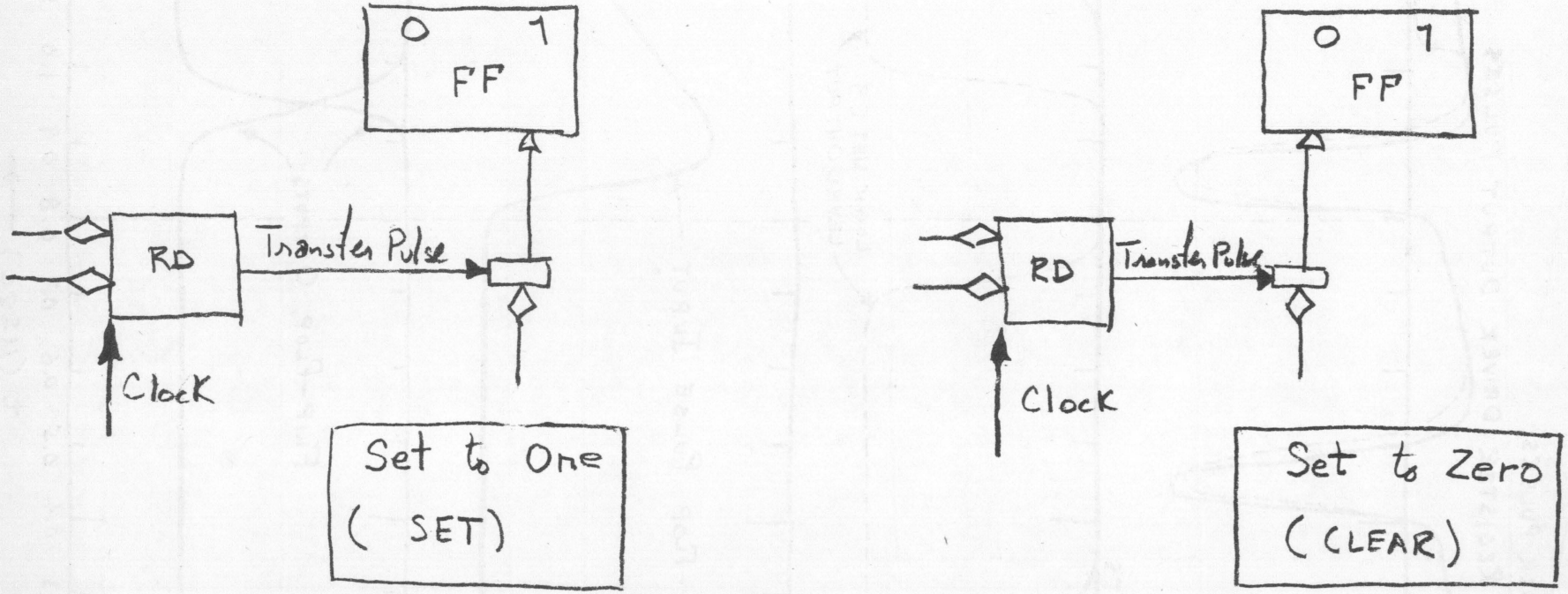
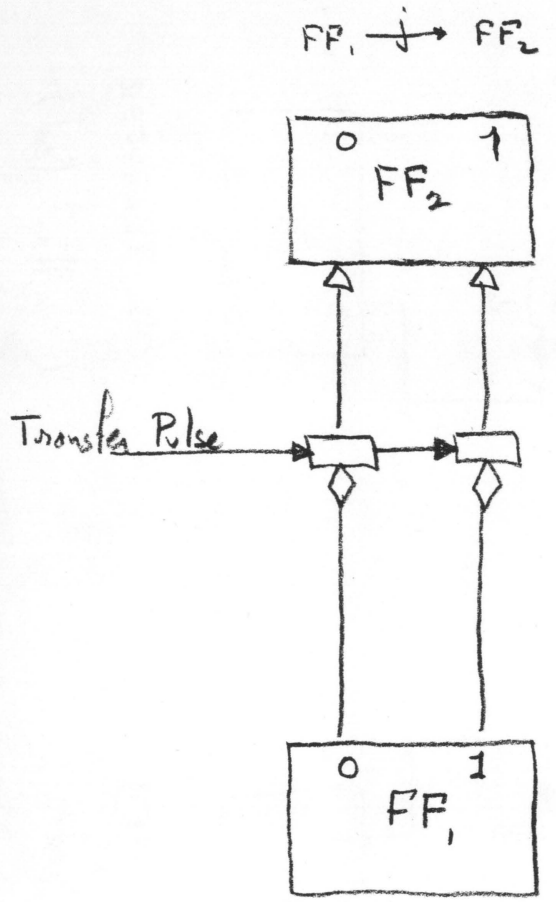
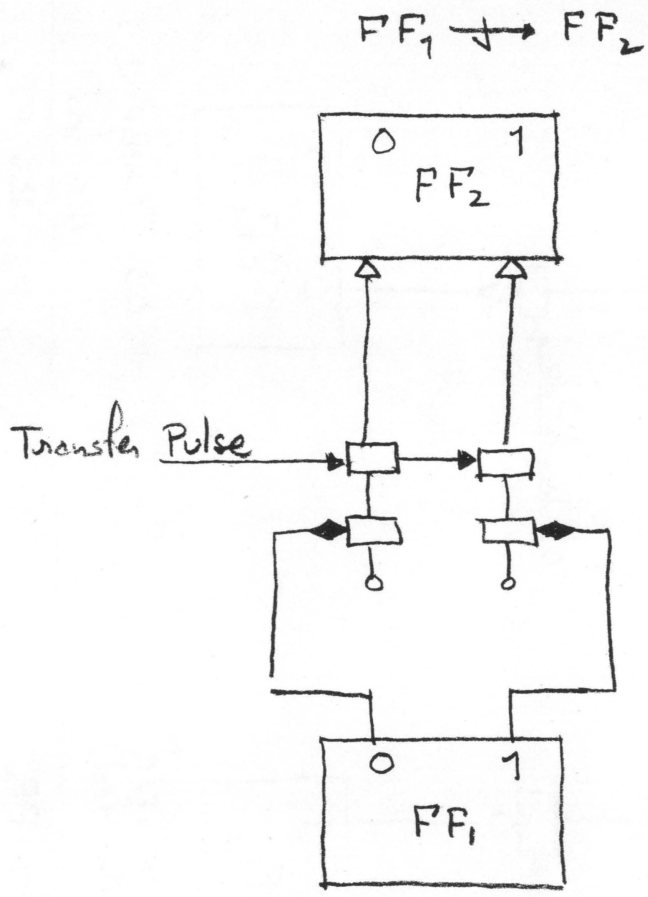


FIG. 3-20 DATA TRANSFER INTO FLIP-FLOPS



Emitter Fed



Base Fed

Fig. 3-21 JAM REGISTER TRANSFER

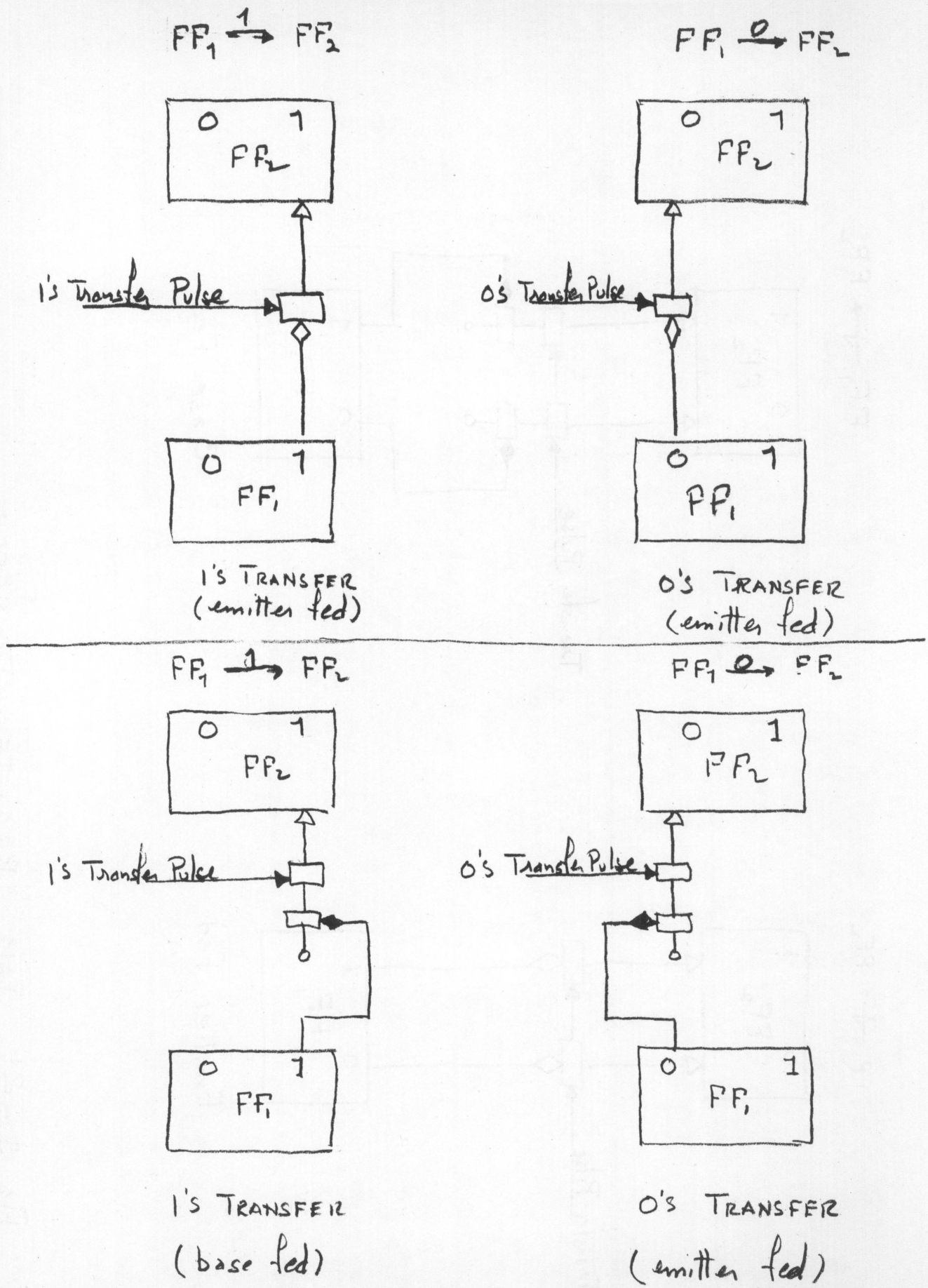
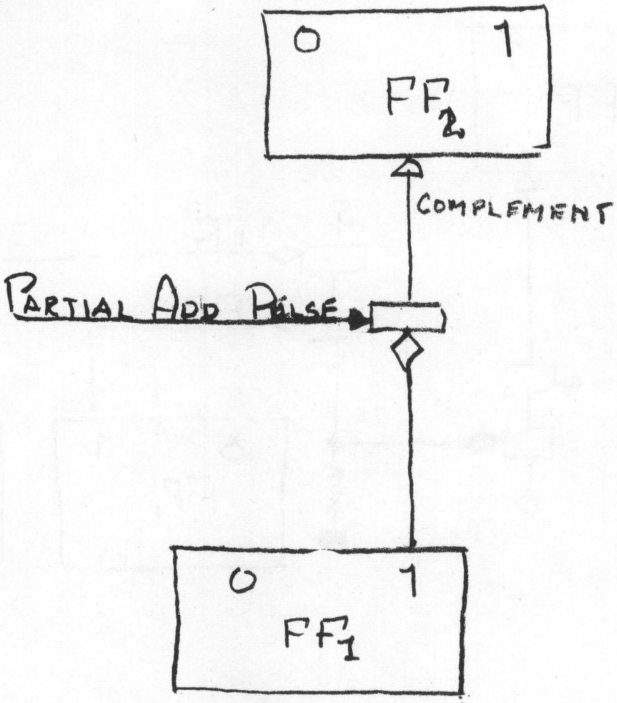


Fig. 3-22 | 0's AND 1's REGISTER TRANSFER

$FF_1 \oplus FF_2 \longrightarrow FF_2$

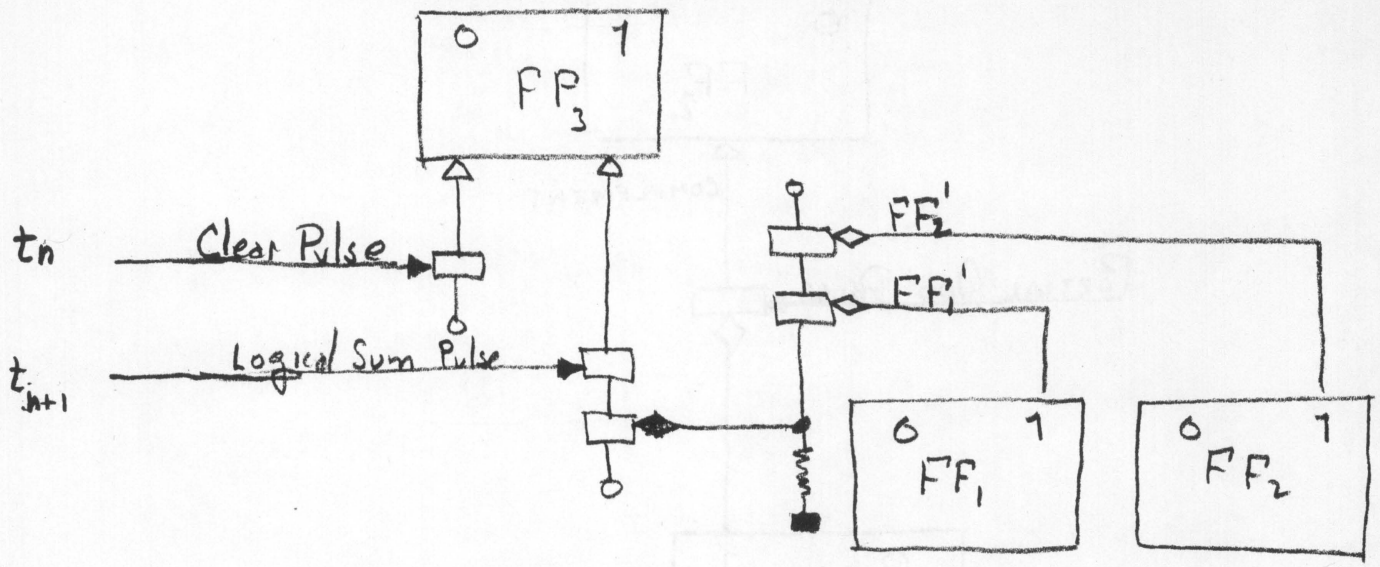


t_n		t_{n+1}
FF_1	FF_2	FF_2
0	0	0
0	1	1
1	0	1
1	1	0

← FF_2 CHANGES STATE

Fig. 3-23 PARTIAL ADD (EXCLUSIVE OR) REGISTER TRANSFER

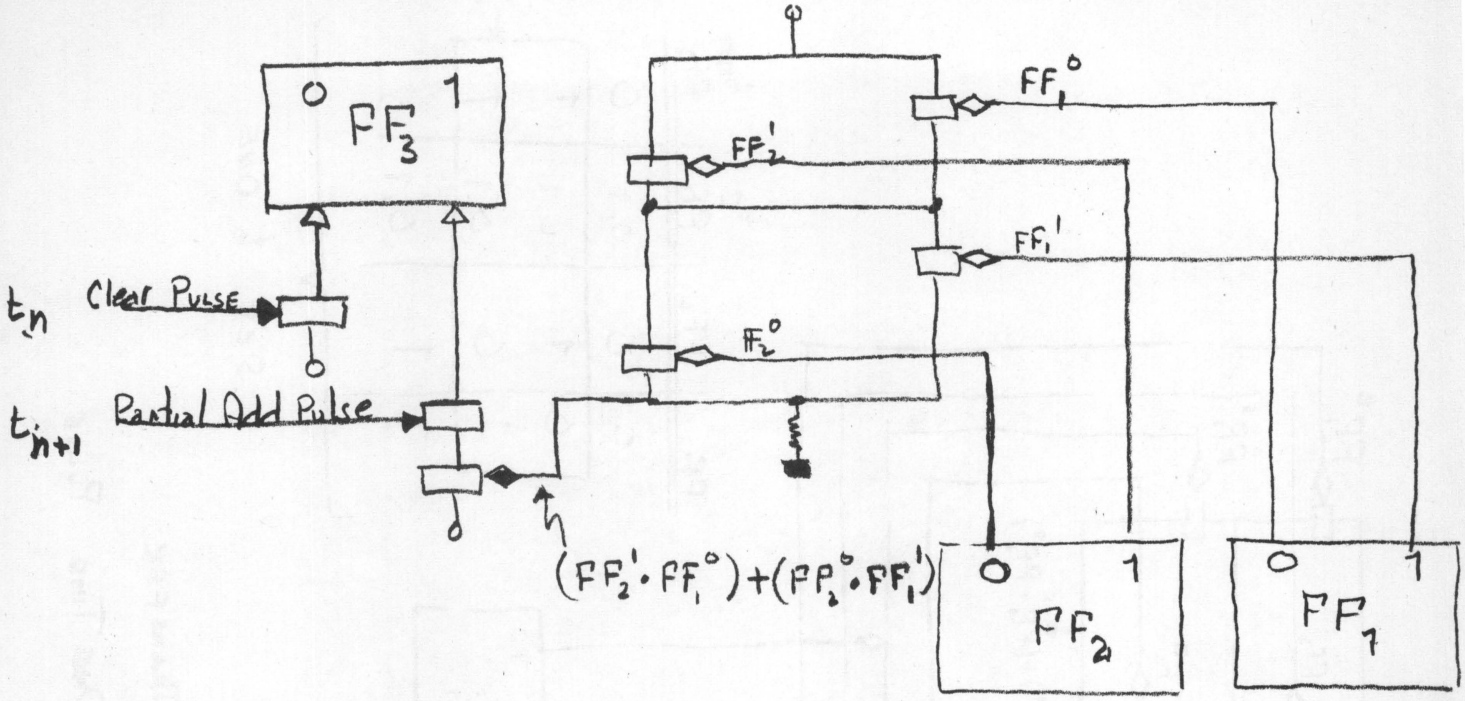
$FF_1 + FF_2 \longrightarrow FF_3$



FF_1	FF_2	t_n FF_3	t_{n+1} FF_3
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	1

← FF_3 set to "one"

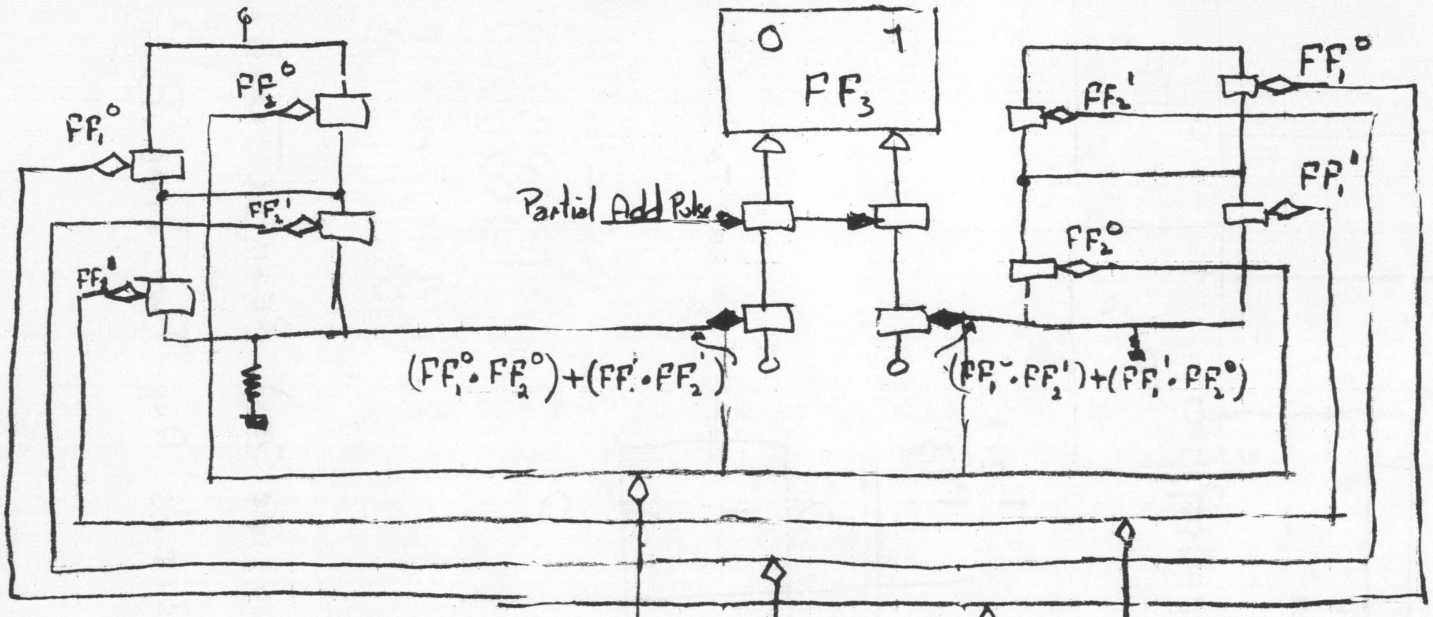
Fig. 3-24 Logical Sum (Inclusive Or), Register Transfer Involving Three Registers And Two Pulses



FF_1	FF_2	t_n FF_3	t_{n+1} FF_3
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0

FF_3 set to "One"

Fig 3-25 Partial Add (Exclusive Or) Register Transfer Involving Three Registers And Two Time Pulses



FF_1	FF_2	t_n FF_3	t_{n+1} FF_3
0	0	0,1	0
0	1	0,1	1
1	0	0,1	1
1	1	0,1	0

FF_3 Set to "Zero"

Set to ZERO

FF_1	FF_2	t_n FF_3	t_{n+1} FF_3
0	0	0,1	0
0	1	0,1	1
1	0	0,1	1
1	1	0,1	0

FF_3 Set to "One"

Set to ONE

FIG 3-26 PARTIAL ADD REGISTER TRANSFER INVOLVING THREE REGISTERS AND ONE TIME PULSE

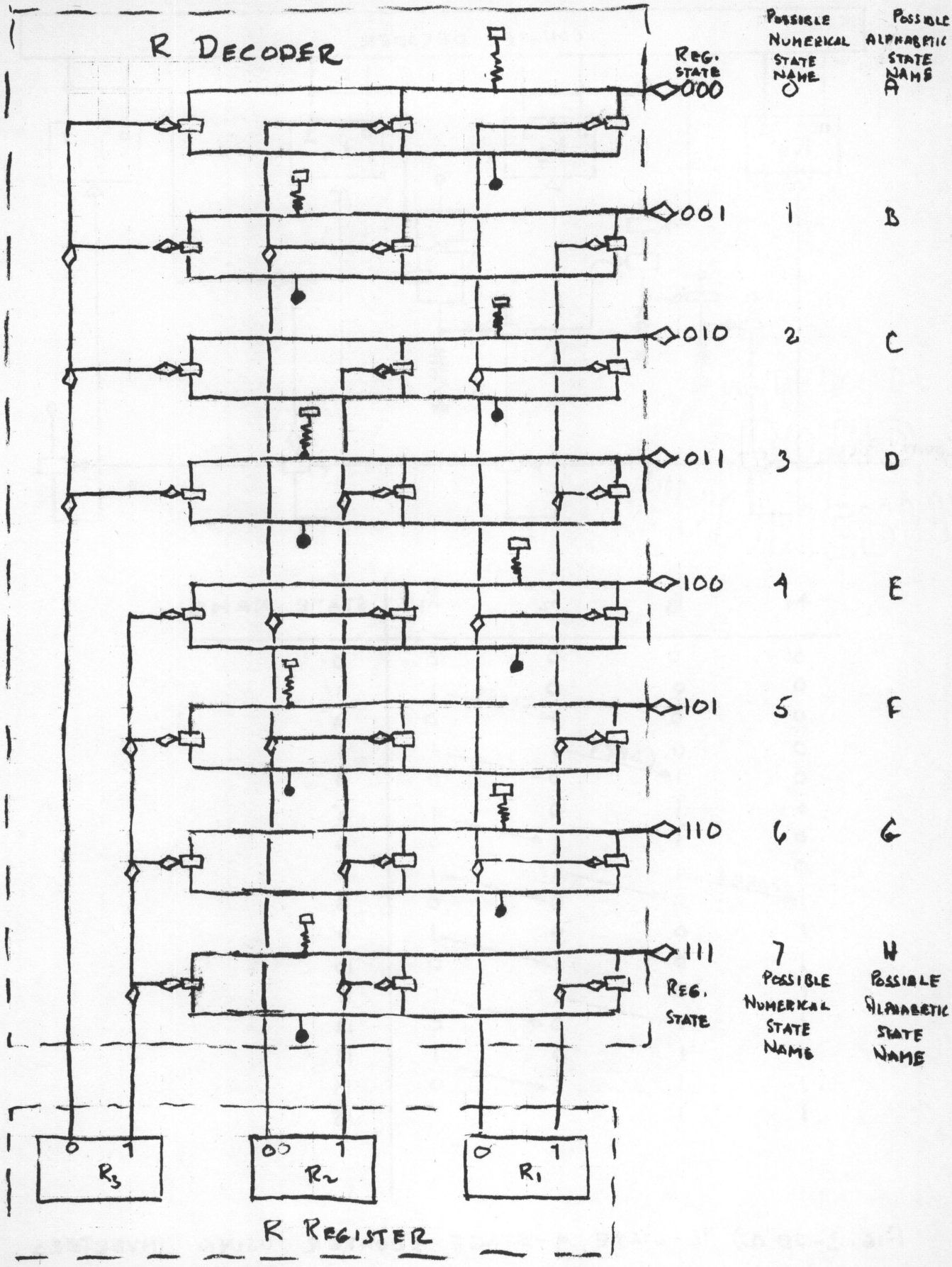
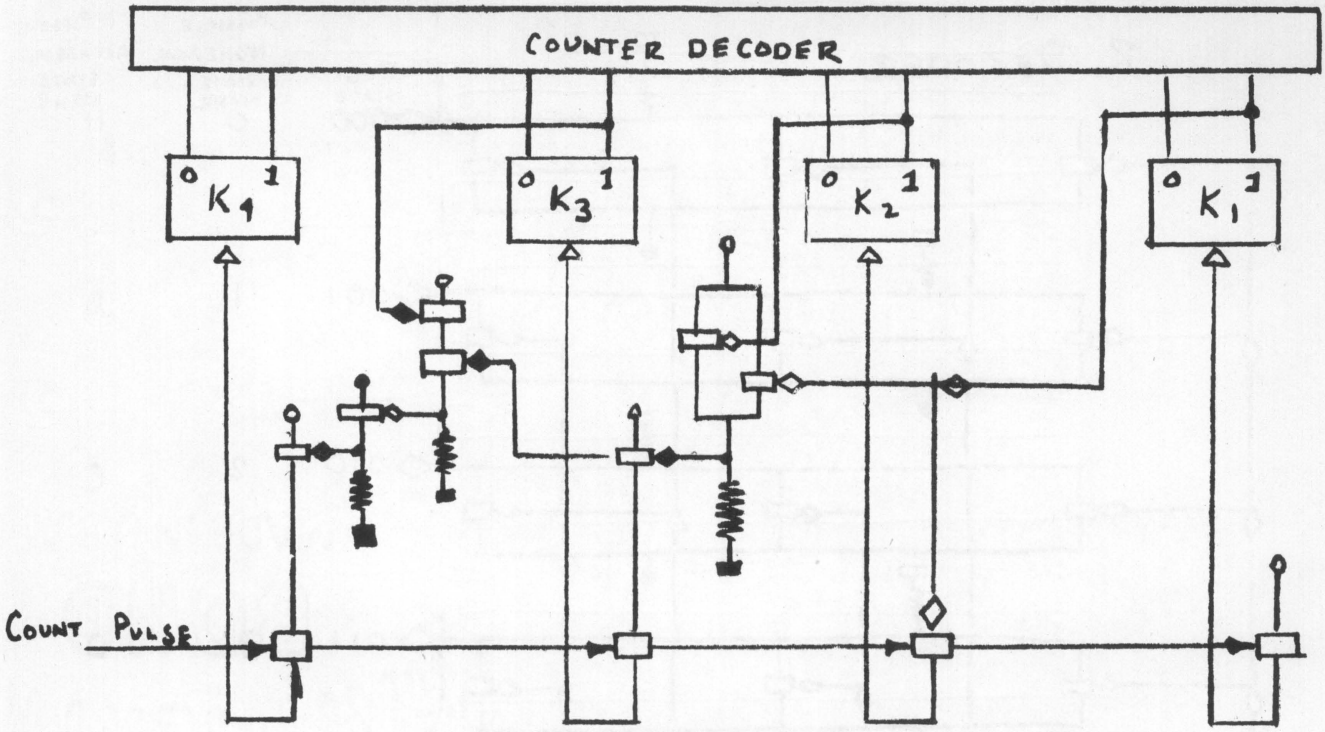


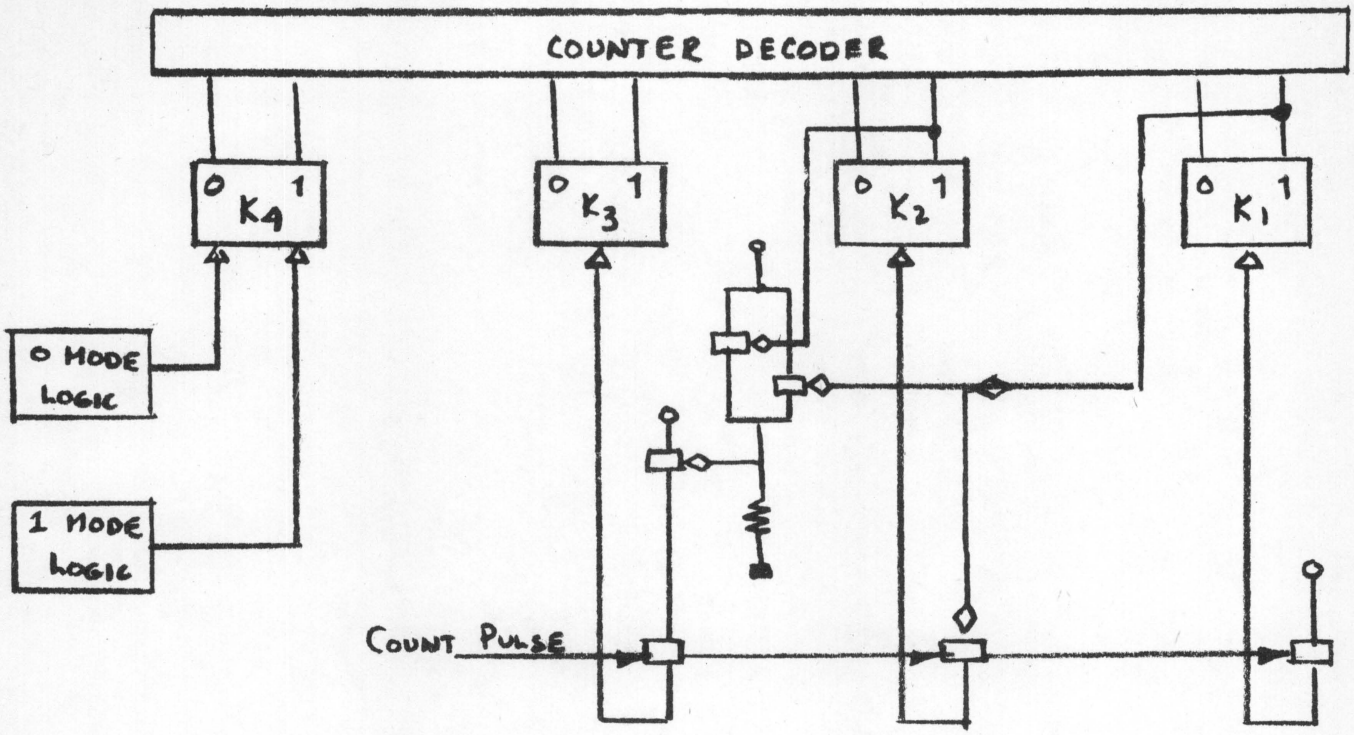
FIG. 3-27 REGISTER DECODING



M	K_3	K_2	K_1	STATE NAME
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

CARRY

FIG. 3-28 a) 16-STATE, 4-STAGE COUNTER USING INVERTERS



	K_4	K_3	K_2	K_1	STATE NAME
0 MODE	0	0	0	0	0
	0	0	0	1	1
	0	0	1	0	2
	0	0	1	1	3
	0	1	0	0	4
	0	1	0	1	5
	0	1	1	0	6
	0	1	1	1	7
1 MODE	1	0	0	0	8
	1	0	0	1	9
	1	0	1	0	10
	1	0	1	1	11
	1	1	0	0	12
	1	1	0	1	13
	1	1	1	0	14
	1	1	1	1	15

Fig. 3-28 b) 16 STATE, TWO-MODE, 3-STAGE COUNTERS USING INVERTERS

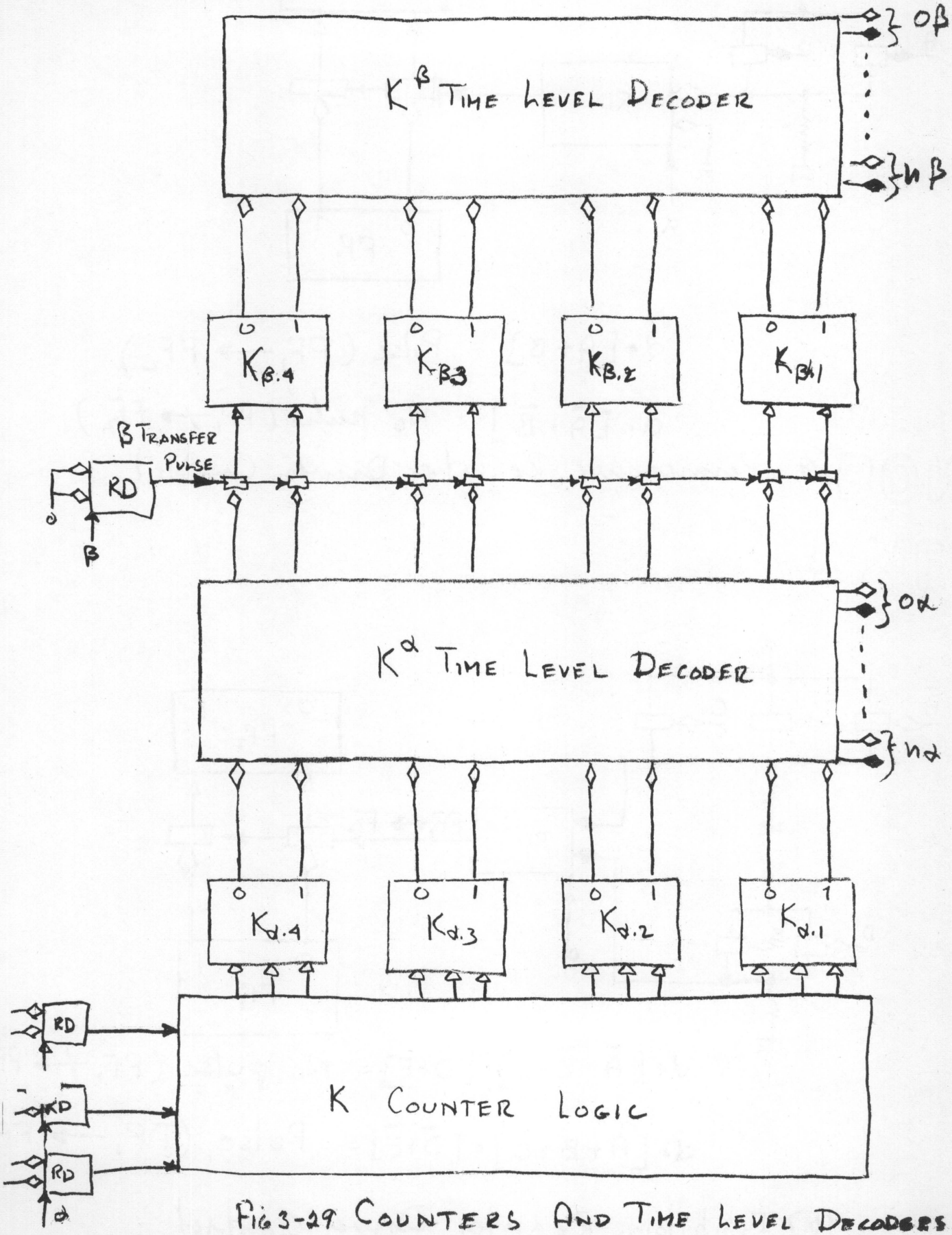
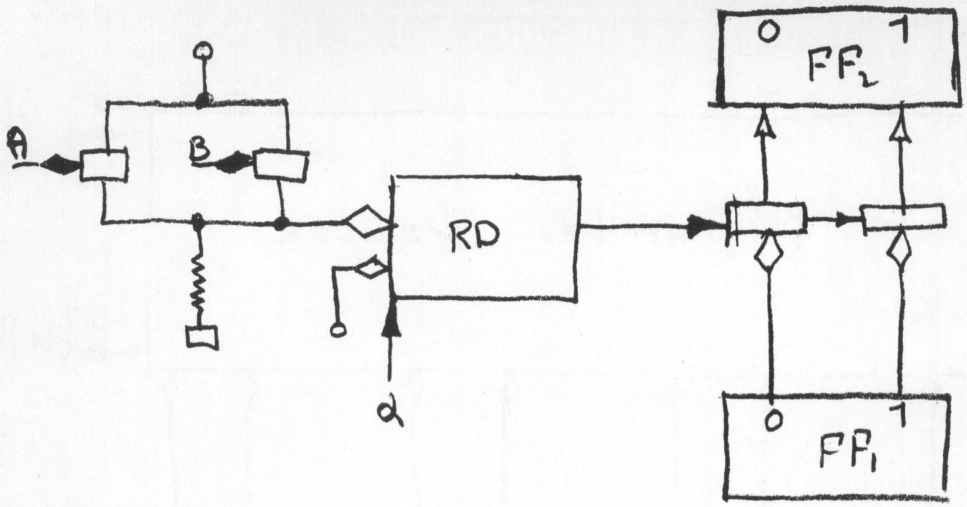


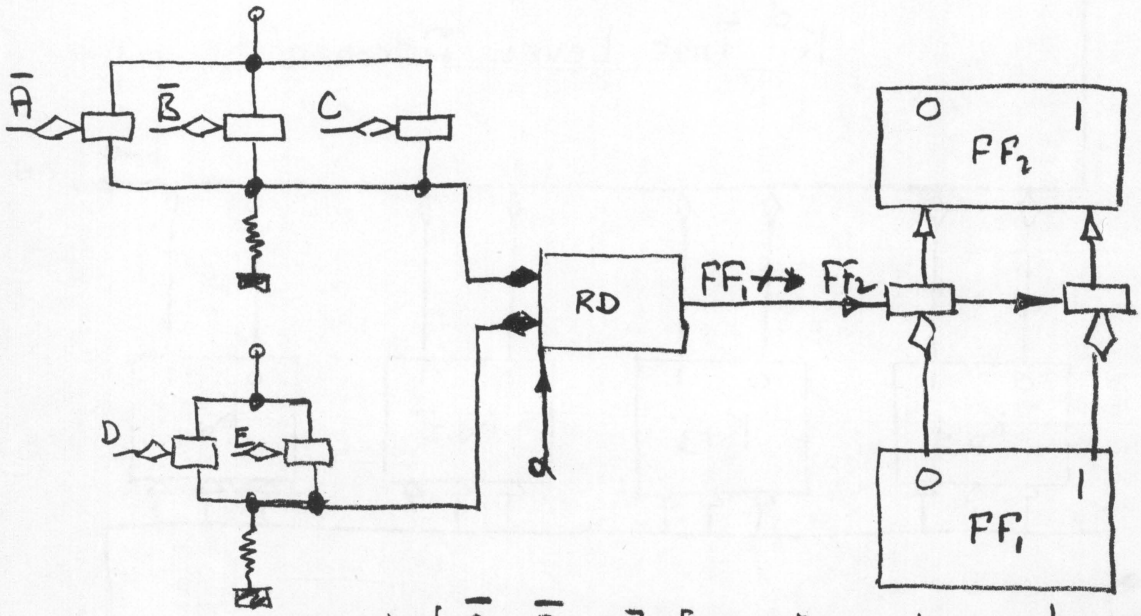
Fig 3-29 COUNTERS AND TIME LEVEL DECODERS



$$\alpha \cdot [A + B] = \text{Pulse}\ (FF_1 \longrightarrow FF_2)$$

$$\alpha \cdot [\bar{A} \cdot \bar{B}] = \text{No pulse}\ (FF_1 \not\longrightarrow FF_2)$$

(a) Permissive Register Driver Control



$$\alpha \cdot [\bar{A} \cdot \bar{B} \cdot C] + [D \cdot E] = \text{No pulse}\ (FF_1 \not\longrightarrow FF_2)$$

$$\alpha \cdot [A + B + \bar{C}] \cdot [\bar{D} + \bar{E}] = \text{Pulse}\ (FF_1 \longrightarrow FF_2)$$

(b) Inhibitory Register Driver Control

Fig. 3-30 REGISTER DRIVER CONTROL LOGIC

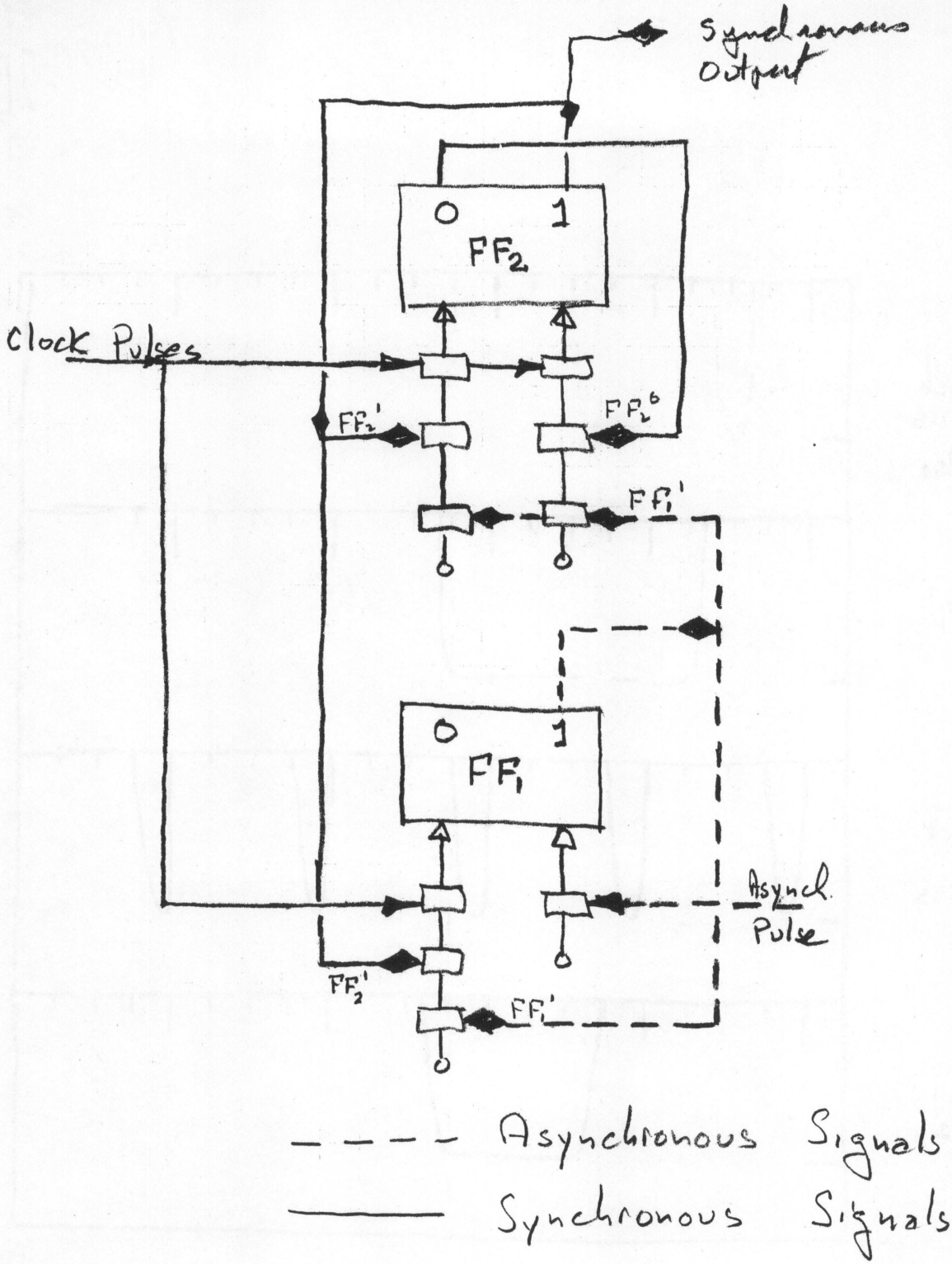


Fig. 3-31 HIGH-SPEED SYNCHRONIZER

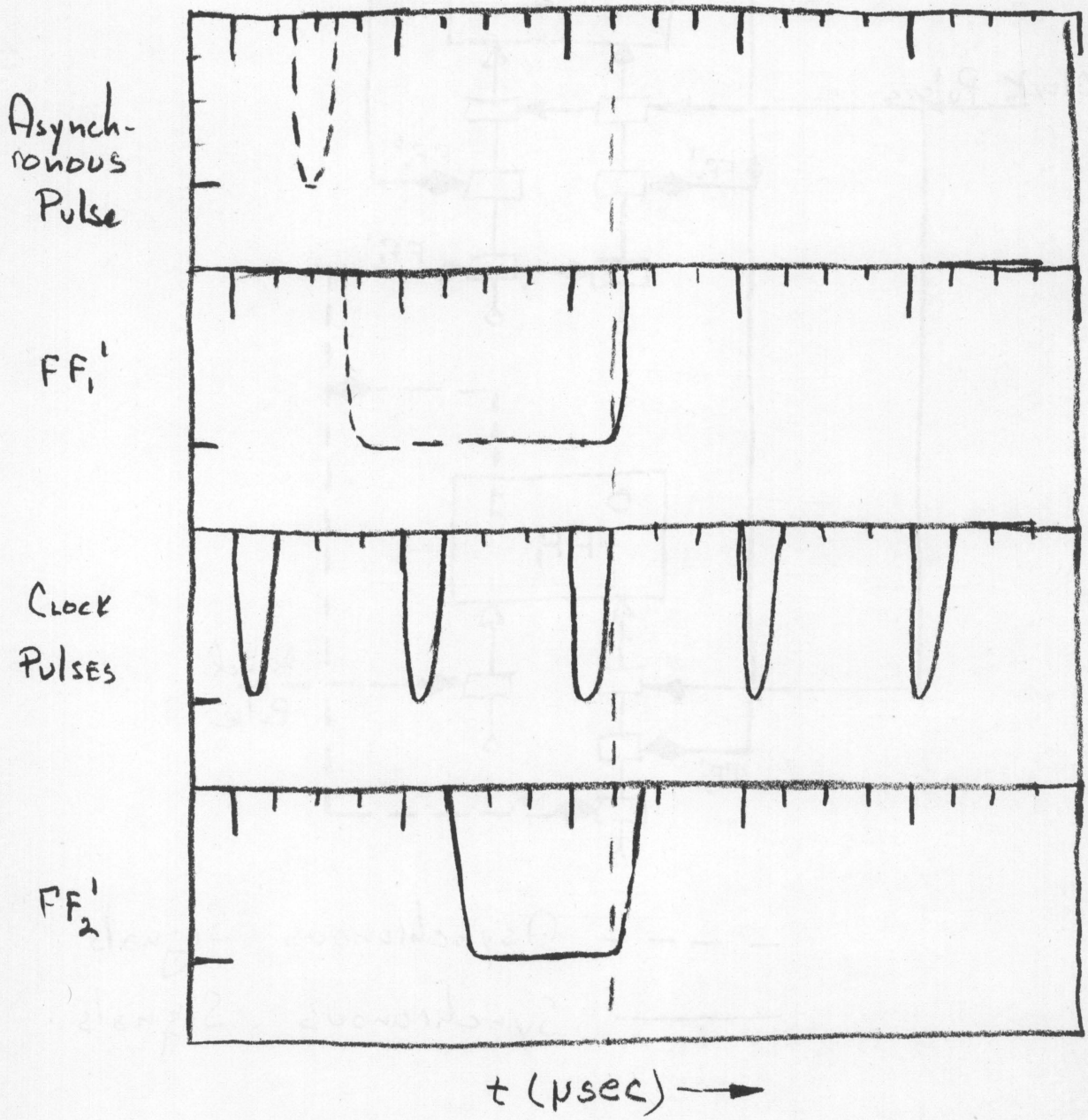


Fig 3-32 High-Speed Synchronizer Time Characteristics

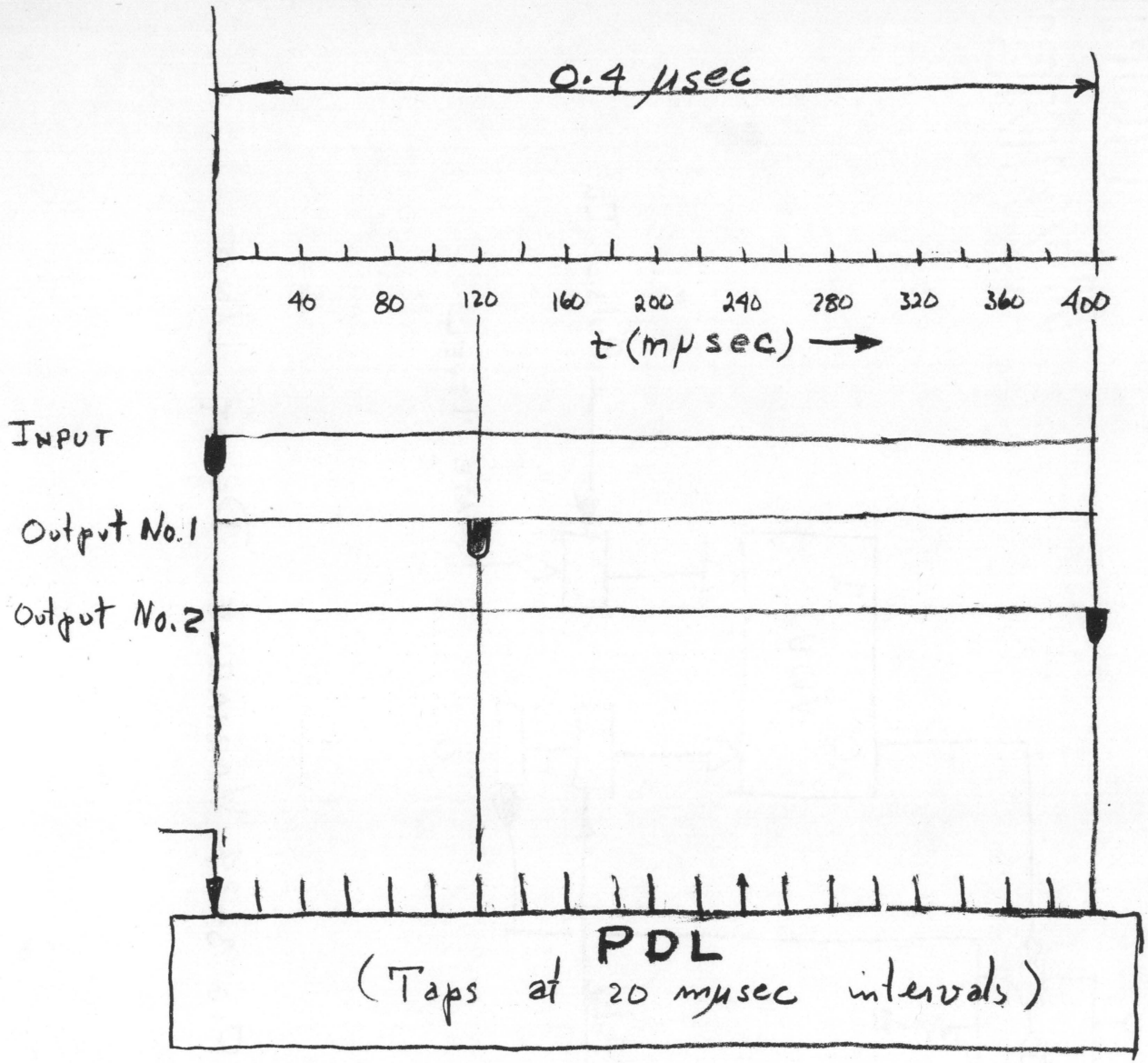


Fig 3-33 TYPICAL PULSE DELAY LINE

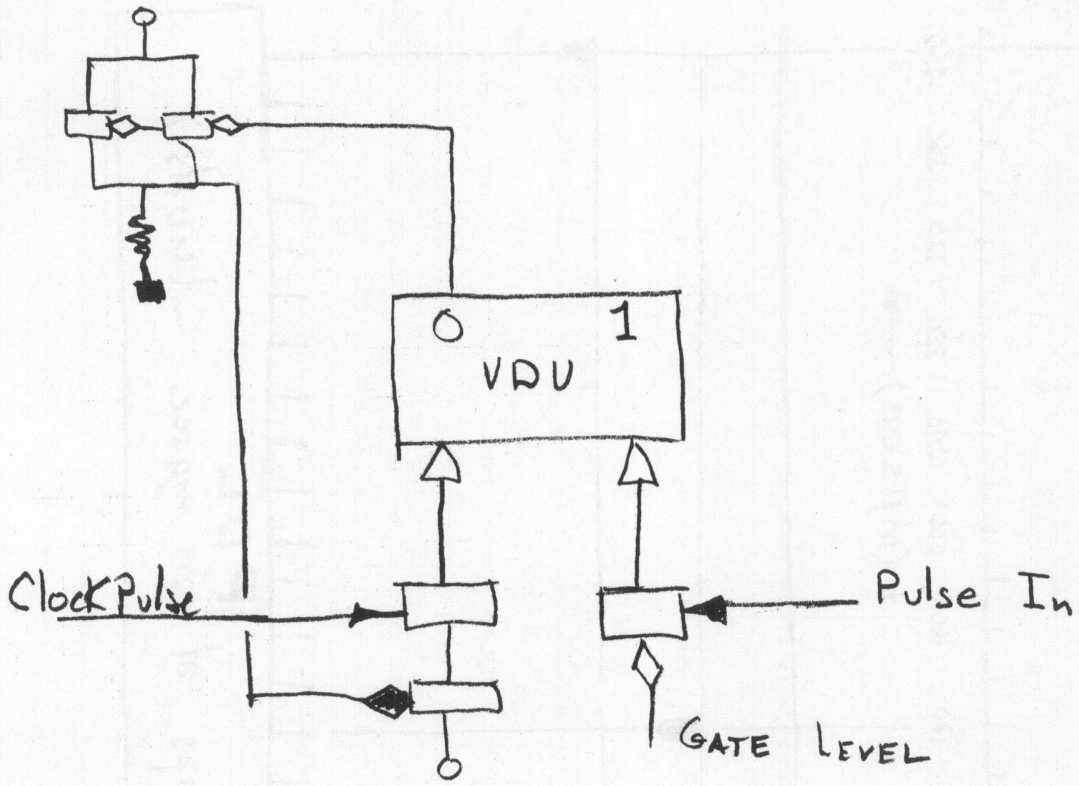
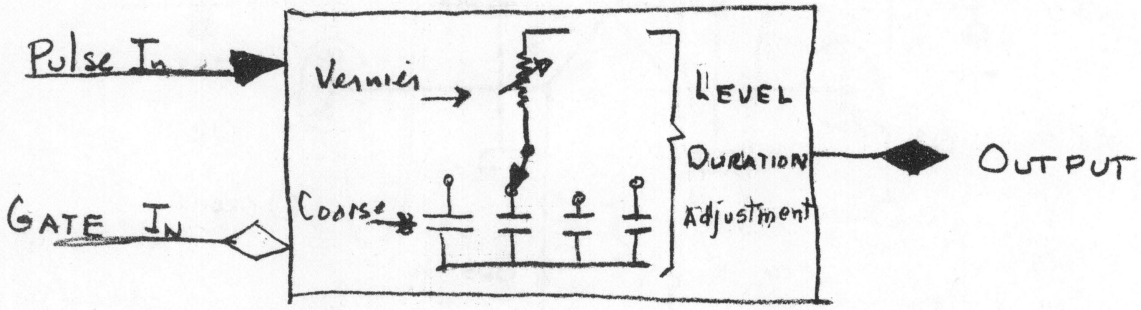
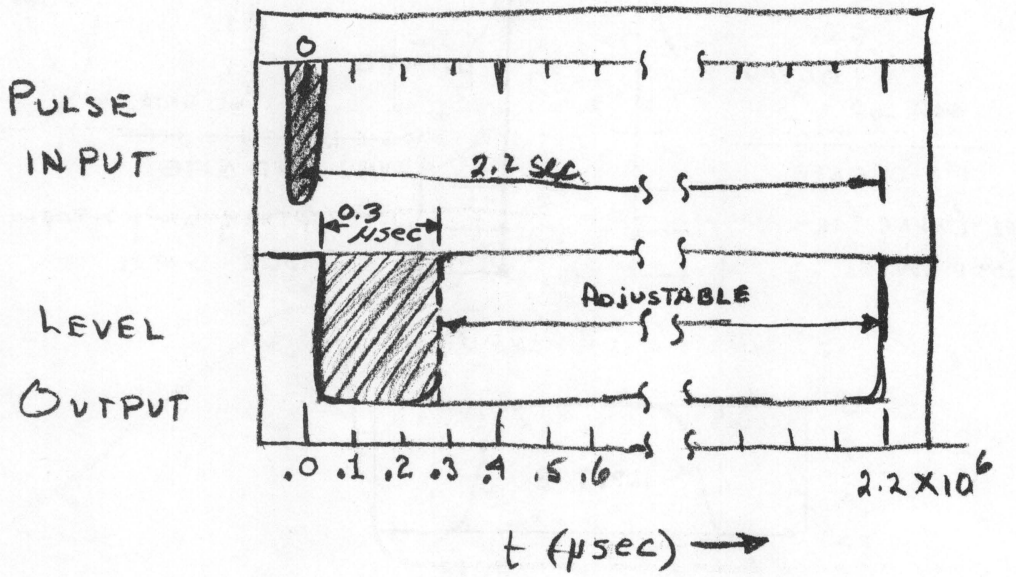


FIG 3-34 VARIABLE DELAY UNIT



VDU



Time Characteristics

Fig. 3-35 Characteristics of Variable Delay Unit

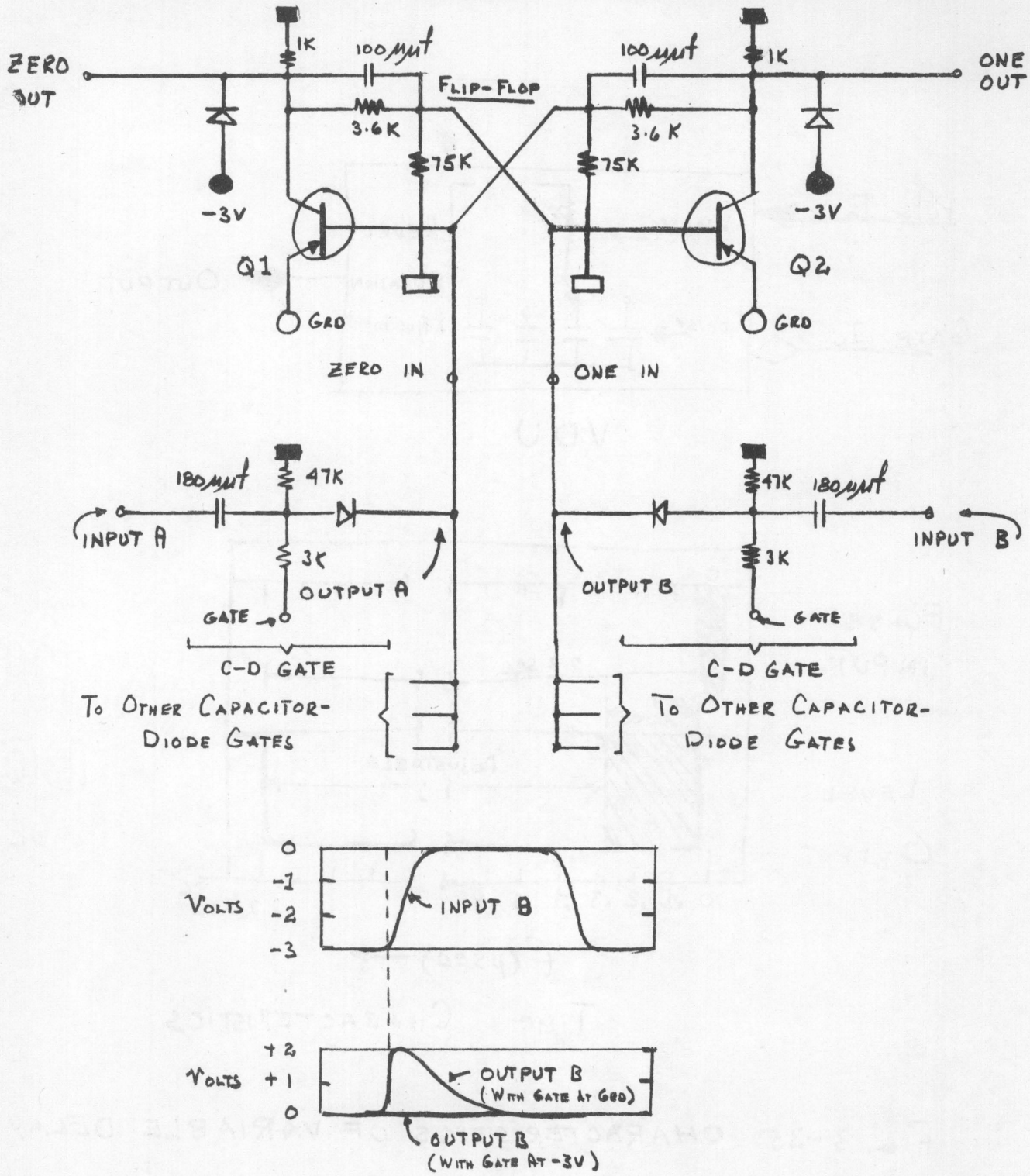
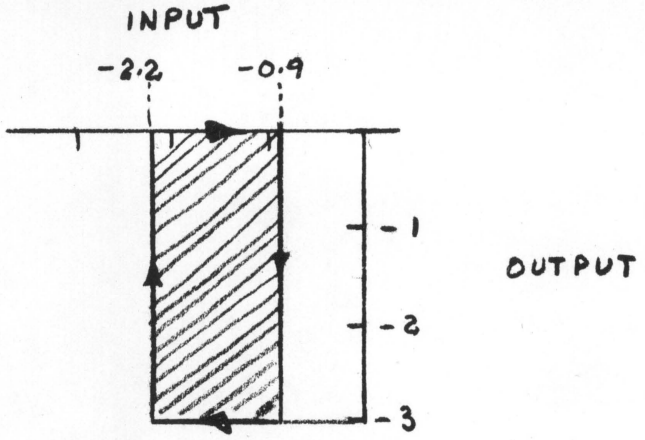
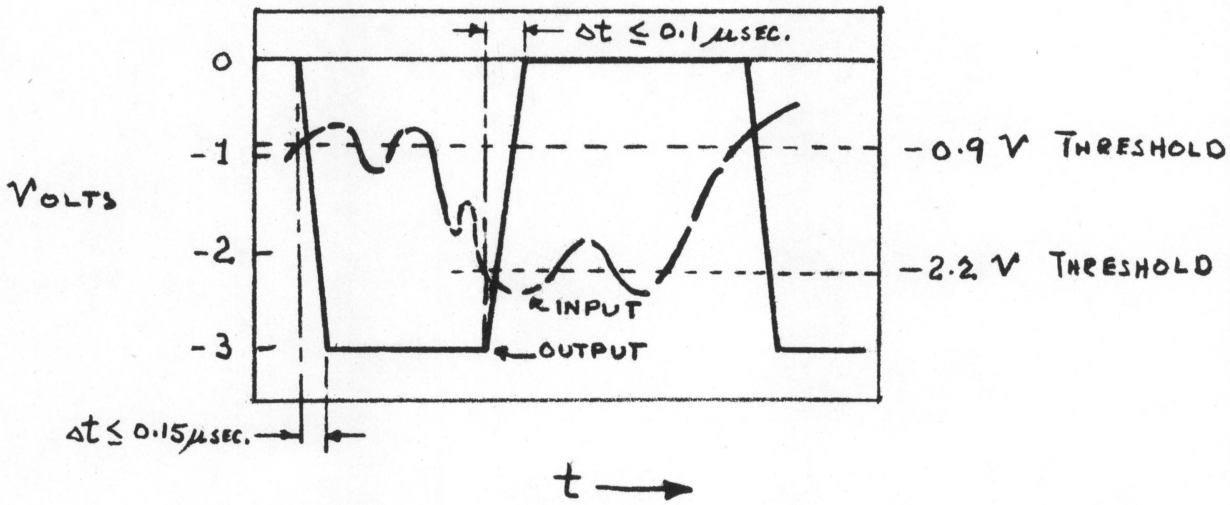


FIG. 3-36 LOW-SPEED FLIP-FLOP AND CAPACITOR-DIODE GATE



INPUT-OUTPUT STATIC CHARACTERISTICS



INPUT-OUTPUT DYNAMIC CHARACTERISTICS

FIG. 3-37 SCHMIDT TRIGGER CHARACTERISTICS

CHAPTER 4

MEMORIES

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 4

MEMORIES

4-1 INTRODUCTION

The computer has six independent memories designated by S, T, U, V, X and F. The function of each of these memories was described in Chapter 2. This chapter will describe the structure of the memory systems and explain the principle of operation of the cores used in the memories. The details of control for the S, T, U and V memories will be discussed in Chapter 11, and for the X and F memories in Chapter 12.

4-2 MEMORY ELEMENT

The major components of the S, T, U and V memories are shown in Fig. 4-1. They are: (1) the memory address selector, (2) the decoders for each memory, (3) the S, T, U and V memories, and (4) the read-write control for each memory.

4-3 MEMORY ADDRESS SELECTOR

The memory address selector uses the leftmost bits of both the P and Q registers to select the desired memory. The remaining bits in the P and Q registers determine the address of the word in the selected memory. The selection and address bit coding for the S, T, U and V memories is illustrated in Fig. 4-2. Defer bit 2.9 is not used during the memory cycle and, since it may be a ONE or a ZERO, is represented by "x". Fig. 4-2 illustrates the following points:

1) The selection and address bit coding is the same for both the P and Q registers.
2) When bit 2.8 of one of these registers is a ZERO, then that register will select S Memory. Similarly, when bit 2.8 is a ONE and bits 2.7 - 2.4 are ZEROS, the T Memory is selected; when bits 2.8 and 2.4 are ONES and bits 2.7 - 2.5 are ZEROS, the U Memory is selected; and finally, when bits 2.8 - 1.8 are ONES, the V Memory is selected. The selection of a particular register in V_{FF} or $V_{\overline{FF}}$ is determined by address bits 1.7 - 1.5. When bits 1.7 - 1.5 are all ZEROS, V_{FF} is selected; when any one of bits 1.7 - 1.5 is a ONE, $V_{\overline{FF}}$ is selected.
3) Not every combination of the 17 bits in P or Q actually selects a register in a memory.

4-4 S MEMORY

4-4.1 GENERAL DESCRIPTION. The S Memory is a high-speed, random-access coincident-current magnetic-core unit with a storage capacity of 65,536 38-bit words (registers). The bits in the word are read out in parallel. The access time, which is the time required to locate and read out a particular register in a memory, is 4.0 microseconds. The cycle time is 6.4 microseconds. (Cycle time is defined as the minimum time between successive read operations in the same memory when complete read-write cycles are performed.) Two coordinates are used to select a register during the read operation, and three coordinates are used for writing.

The major components that comprise the S Memory are shown in Fig. 4-3. They include the following: an address decoder and associated cable drivers, 8 switch driver current regulators, 32 switch driver input amplifiers, 64 switch drivers, two 16 X 16 magnetic-core switches, a 256 X 256 core memory array, 152 digit plane drivers, and 38 sensing amplifiers. The number of lines interconnecting the various components is indicated in Fig. 4-3.

4-4.2 CORE MEMORY ARRAY (See Fig. 4-4). The S Memory array is made up of thirty-eight 256 X 256 memory planes and contains 2,490,368 ferrite memory cores. The outside diameter of each core is 80 mils, the inside diameter 50 mils, and the thickness 22 mils. The cores are wired into 64 by 64 subassemblies. Each subassembly is a complete operating memory plane with its own sense and digit winding. Sixteen 64 by 64 subassemblies are assembled in a square array and connected together to form each 256 by 256 plane. Thirty-eight 256 by 256 planes are stacked one on top of another with their X and Y wires connected in parallel to form the complete memory array.

4-4.2.1 OPERATING PRINCIPLE. Consider an array of cores arranged in planes and oriented with respect to three orthogonal axes as shown in Fig. 4-5. Each core in the array lies at the intersection of a unique set of X, Y and Z co-ordinate planes. A wire runs through the cores in each plane. The sense windings which are in the Z plane are not shown.

The memory-core hysteresis loop is sketched in Fig. 4-6. The lower flux state by definition will constitute a ONE and the upper flux state a ZERO. During a read-out, enough positive current must be applied to switch the core from the lower to the upper state. Thus, when the core holds a ONE, reading causes a large flux change which produces a large voltage output. If, however, the core was already in the ZERO state, reading causes a very small flux change which in turn produces a small voltage output. (See Fig. 4-7)

The memory cycle consists in reading out the information contained in a register, and then writing information back into the same register (not necessarily the same information). It is important at this point to recognize two important facts:

1) The read-out destroys the information held in the core, i.e., all cores in the register are put in the ZERO state.
2) The read-out does not require any selection <u>among the digits</u>. The entire register receives the read-out excitation, whereas in writing, some cores in the register will be switched to the ONE state and others will be left, as they were, in the ZERO state - depending, of course, on the information to be stored in the register.

Two functions need explaining:

1) How a register is selected for read-out.
2) How an arbitrary pattern of ONE's and ZERO's can be written in a register.

Both of these functions must be accomplished without changing the information held by the cores in other registers in the array.

If the current necessary to switch a core is defined as I_m and the core material is such that $I_m/2$ will not switch the core, then by applying $+I_m/2$ to an X line and $+I_m/2$ to a Y line, it is possible to select the register at the intersection of these two co-ordinates without destroying the information held in other registers. In order to write in this register $-I_m/2$ is applied on the X and $-I_m/2$ on the Y line. The appropriate Z-windings are used to inhibit with $+I_m/2$ on those planes in which ZEROs are to be written. This selection process depends entirely on the ability of the core to switch at an excitation of I_m and to remain unchanged when subjected to an excitation of $I_m/2$. At this point it is convenient to define the noise in this system as that part of the sense-winding output during read-out which is not contributed by the selected core. The noise would be made up of capacitive and inductive coupling from driving lines as well as the outputs due to half-selected cores which are subjected to $I_m/2$ excitation.

4-4.3 MAGNETIC-CORE SWITCH. In order to match the characteristics of the driver tubes to the X and Y selection lines, current step-down devices called magnetic-core switches are used. These switch cores also perform the final stage of the decoding of the address.

4-4.3.1 MECHANICAL FEATURES. A schematic of a magnetic-core switch is illustrated in Fig. 4-8. Each of the two magnetic-core switches is made up of 256 tape-wound cores. Four windings are placed on each core: two 12-turn input windings, a 16-turn output winding and a 2-turn bias winding. These cores are connected into a square array to form a 2-coordinate switch. The inputs to the X switch are called $XU_0, \ldots, XU_{15}$, $XV_0, \ldots, XV_{15}$, and the inputs to the Y switch are called $YU_0, \ldots, YU_{15}$, $YV_0, \ldots, YV_{15}$.

The decoders select one core in each of the two magnetic-core switches. The switching of these two cores generates currents in the memory X and Y windings. The currents switch the memory cores which contain ONES in the selected memory register. The flux reversal generates signals in the sense windings of the switched cores. When the two cores in the magnetic-core switch are deselected, they generate opposite currents in the memory cores of the selected register. This writes ONES in the memory cores that have no current in the inhibit (Z) winding.

4-4.3.2 OPERATING PRINCIPLE. The operation of the switch is illustrated in Fig. 4-9. A bias current is used to bias all the cores in the switch to point A of the hysteresis loop. When the core is switched, the flux reverses to point C of the hysteresis loop. The magnetomotive force required to switch the core is generated by two current pulses called U and V. The application of either the U or V current pulses alone is not sufficient to switch the core. However, the occurrence of both the U and V current pulses in the windings of a core is sufficient to cause the core to switch. The switching of the core generates a read current pulse at its secondary.

When the U and V current pulses end, the bias current switches the core back to point A. This generates a write pulse.

4-4.4 SWITCH DRIVING. Since the magnitude of the current levels from the memory address level decoders is insufficient to actuate the memory array, switch drivers, switch driver input amplifiers, and switch driver current regulators are used to amplify and regulate the memory current.

4-4.4.1 BLOCK DIAGRAM. Fig. 4-10 shows a block diagram of the switch core driving system associated with the XU inputs to the X magnetic-core switches. Similar systems exist for the XV, YU and YV magnetic-core switch inputs. Thus, there are 4 switch core driving systems, each set containing 16 switch drivers, 8 switch driver input amplifiers and 2 switch driver current regulators.

4-4.4.2 OPERATING PRINCIPLE. Fig. 4-10 shows the 16 switch drivers as two groups of eight, with one current regulator feeding each group. The switch driver is made up of several power triodes connected in parallel. The current regulator signal is applied to the cathode and the input amplifier signal is applied to the grid of the associated switch drivers. The grid input lines to the first group of eight drivers are connected consecutively from 0 to 7. The grid input lines to the second group of eight drivers are connected in parallel with the first group.

A particular line in the switch is selected by first grounding one of the grid input lines and then pulsing one of the current regulators. For example, to select line 0, grid input 0 is grounded and the current regulator input is pulsed, thus actuating driver 0. Current regulator input 0 is pulsed for switch drivers 0 through 7 and current input 1 is pulsed for switch drivers 8 through 15.

4-4.5 DIGIT DRIVERS. The digit drivers are used during the WRITE phase of a memory cycle. They provide the inhibit currents that prevent the cores, in which it is desired to write ZEROS, from reversing to the ONE state when the write current is applied.

4-4.5.1 BLOCK DIAGRAM. The digit-plane driver block diagram is shown in Fig. 4-11. It is similar to the current regulator in the switch drive circuit. Four such circuits are associated with each 256 by 256 plane, one for each quarter of the digit plane winding. Each quarter is made up of the digit winding of four subassemblies connected in series.

4-4.5.2 OPERATING PRINCIPLE. The input to the digit driver is a transistor "AND" gate. The quarter selection level is applied to one input and the timing level to the second input. The output of the gate circuit controls an amplifier which supplies a regulated current pulse to the inhibit winding.

4-4.6 SENSE AMPLIFIERS

4-4.6.1 BLOCK DIAGRAM. The block diagram of a typical sense amplifier is shown in Fig. 4-12. Each sense amplifier has four bi-polar inputs. Each bi-polar input comes from a sense winding section of a Z plane in the memory array and goes to one of four differential amplifiers in the sense amplifier. The four amplifier outputs are then mixed in an emitter follower and finally amplified by a pulse amplifier. One four-input sense amplifier is associated with each 256 by 256 plane. Thus, there is a total of 38 sense amplifiers for the entire memory.

The sense windings in a 256 X 256 plane are broken up into four sections. Each section consists of the sense windings from four diagonally adjacent (in the sense of Fig. 4-12) 64 X 64 subassemblies. By connecting the sense windings on the diagonal, there is a minimum coupling between the active X and Y windings and the sense winding. (Ideally the coupling should occur in only one case in the plane.) Note that a given X or Y drive line intersects only one of the subassemblies in the diagonal section which the sense windings are interconnected. It should also be noted that with this method of connection, the voltage induced in the sense windings by the half selected cores (i.e., the cores in which either the X or Y winding is pulsed but not both) is equal to that in one 64 X 64 subassembly memory.

Each sense winding is also a delay line. To reduce the delay and resultant signal distortion, the four assemblies on a given sense winding section are connected in series-parallel as shown in Fig. 4-12 rather than just in series.

4-4.6.2 OPERATING PRINCIPLE. The bi-polar signal is fed into the differential amplifier in the sense amplifier. The outputs of the differential amplifiers are mixed (ORed) and rectified in the emitter-follower circuit, and then further amplified in the pulse amplifier to a certain voltage. If a ONE is read out, the voltage will be 3 volts, and if a ZERO is read out, the voltage will be zero volts. The signal is transmitted to the memory buffer register where it is sampled by a 0.1 microsecond strobe pulse.

4-5 T MEMORY

4-5.1 GENERAL DESCRIPTION. The T Memory is a high-speed, random-access, coincident-current magnetic-core unit with a storage capacity of 4096 38-bit words (registers). It contains only transistors and diodes in the read, write, and selection circuitry in contrast to the S Memory which uses principally vacuum tubes and magnetic cores. The bits in a word are read out in parallel. The cycle time is 4.4 microseconds and the access time is 2.0 microseconds. As in the S Memory, a 2:1 current selection ratio is used, with two coordinates used to select a register during read-out and three coordinates used during write.

4-5.2 BLOCK DIAGRAM. Fig. 4-13 is a block diagram of the T Memory. It is very similar to that of the S Memory. The basic difference is that transistor drivers, called selection line drivers, are used to select and drive the X and Y memory lines instead of the magnetic-core switches used in the S Memory.

Like the S Memory, the T Memory uses two coordinate selection of memory registers. The X and Y lines are selected in a two level decoder. The decoded coordinate selection levels are amplified by two sets of 64 selection line drivers to the magnitude required to operate the cores. Other components are 30 digit plane drivers, 38 sensing amplifiers, a read-write selector and 4 read-write current generators.

The T Memory is made up of thirty-eight 64 by 64 planes and contains 155,648 ferrite cores. The cores are composed of the same material as those of the S Memory, but are dimensionally smaller. The outside diameter of each core is 50 mils, the inside diameter 30 mils and the thickness 12 mils.

4-5.3 OPERATING PRINCIPLE. The operation of the T Memory is very similar to that of the S Memory. The first and second level decoders operate as standard decoders. The drivers operate as switches in a similar manner to those of the S Memory except that they drive the cores directly instead of through core switches. Thirty-two selection line drivers are used in conjunction with each read-write current generator in order to reduce the capacitive load on the read-write generator. The READ-WRITE operation is the same as the S Memory.

The operation of the digit inhibit circuits and sense amplifiers is also the same as those of the S Memory. However, since there is only one 64 X 64 array per digit, one digit plane driver and one sense amplifier per digit suffice.

4-6 U MEMORY

The U Memory is currently undefined. However, it is expected to be similar in construction and operation to the T Memory. It will have a storage capacity of 4096 38-bit words (registers).

4-7 V MEMORY

4-7.1 GENERAL DESCRIPTION. Fig. 4-14 is a simplified block diagram of the V Memory. The selection of V_{FF} or $V_{\overline{FF}}$ is determined by the V Memory selector control which decodes bits 1.7 - 1.5. Components of $V_{\overline{FF}}$ include 32 plugboard registers, 16 toggle switch registers, a real-time clock register, and a shaft encoder register. Components of V_{FF} include the 5 flip-flop registers A, B, C, D, and E.

4-7.2 V_{FF} MEMORY (A, B, C, D AND E REGISTERS)

4-7.2.1 GENERAL DESCRIPTION. The A, B, C, D and E registers located in the Arithmetic Element and the Exchange Element are also used as memory storage registers. They have a non-destructive READ cycle. Thus, a write cycle is not required, unless a store type instruction is being performed.

4-7.2.2 MECHANICAL FEATURES. The five V_{FF} registers contain 36 bits numbered from left to right: 4.9 - 4.1, 3.9 - 3.1, 2.9 - 2.1 and 1.9 - 1.1. Since there is no 2.10 bit, parity checking is excluded. Note that there is also no 4.10 (meta) bit associated with these registers.

4-7.2.3 OPERATING PRINCIPLE. During the READ phase of the READ-WRITE cycle, the information is transferred out of the A, B, C or D register into the E register and from there into the N or M register. First, the content of the E register is read into the M register. Next, the content of the selected A, B, C or D register is read into the E register. Then the content of the E register is transferred into the N or M register and, at the same time, the content of the M register is placed back in the E register.

When writing into the A, B, C or D registers during the WRITE phase, the content of the M register is transferred into the E register. The selected register in the Arithmetic Element is then cleared and the content of the E register is transferred into that register. Since read-out of the V_{FF} registers is non-destructive, the content of the N register never needs to be rewritten.

4-7.3 $V_{\overline{FF}}$ MEMORY

4-7.3.1 TOGGLE SWITCH REGISTERS. The toggle switch register is a manually set static memory which, like the V_{FF} Memory, has a non-destructive read cycle.

The 16 toggle switch registers are arranged in horizontal rows in groups of four on the control console. (See Fig. 4-15.) The addresses are labelled in octal code from 377720 through 377737. Each register consists of 37 bits (vertical columns) numbered from left to right as follows: 4.10 - 4.1, 3.9 - 3.1, 2.9 - 2.1, and 1.9 - 1.1. Bit 2.10 is excluded because there is no parity checking for the V Memory. The circuitry is built so that another 8 registers can be easily added.

Fig. 4-16 illustrates the basic operating principles of the toggle switch storage. The basic mechanism of the storage is a resistor-switch matrix which contains 592 resistors and 592 switches (16 X 37 = 592). Each resistor has one end tied to a digit line (16 resistors per digit line) and the other end tied to the common terminal of a switch. One side of each switch, the ZERO or normally closed position, is tied to ground. The other side, the ONE or normally open position, is tied to a resistor driver (37 resistors per driver). One resistor from each digit line goes to each resistor driver.

The resistor driver of a register is selected by a -3 volt level on one of three lines and by a ground level on one of eight lines. The output of the selected resistor driver swings from ground to -25 volts. All unselected resistor drivers are held at ground.

Those resistors that are tied to switches set in the ONE position are connected to resistor drivers. They drive their digit lines to -1 volt. Hence, a -1 volt signal on a digit line means that the appropriate digit of the selected register is a ONE.

The digit line signal is detected, amplified to a standard level, and inverted in a digit detector unit. The digit detector, in turn, drives a cable driver which transfers the information to the sense amplifier and strobe selector, where it is strobed and used.

4-7.3.2 PLUGBOARD REGISTERS. The plugboard registers, like the toggle switch register, form a manually set static memory which has a non-destructive read cycle. Provisions are made so that two plugboards can be connected to the computer at a given time. These plugboards are designated A and B. They are used as storage for utility and maintenance routines. The sequence switch priority plugboard is not considered part of $V_{\overline{FF}}$.

Fig. 4-17 shows a typical plugboard. The octal addresses are numbered from 0 to 17, and the register bits are numbered from left to right: 4.10 - 4.1, 3.9 - 3.1, 2.9 - 2.1, and 1.9 - 1.1. Again, bit 2.10 is excluded because there is no parity checking required.

The operation of the plugboard is similar to the toggle switch storage previously described. In this case, insertion of a jumper (dual-prong patchcord) has the same effect as closing a switch. (See Fig. 4-17.)

4-7.3.3 SHAFT ENCODER. The shaft encoder is an analog-to-digital converter which translates shaft position information into digital code by means of a self-contained, non-ambiguous, dual-brush selection logic. The unit itself and one of the two internal converter discs are shown in Fig. 4-18.

The voltage appearing on each lead of the two sets of output terminals represents a coefficient in the natural binary system. A terminal voltage of -20 volts develops the coefficient ONE and zero voltage develops the coefficient ZERO. The complement of the binary output is provided on the second set of terminals.

4-7.3.4 REAL-TIME CLOCK. The real-time clock uses a 100-kc pulse source and a 36-bit counter. Pulses are applied continuously to the counter so that a time reference is always available.

4-8 PROGRAM ELEMENT MEMORIES

4-8.1 GENERAL DESCRIPTION. The Program Element contains two memories designated X and F. X is the 64-register 19-bit index memory and F is the 32-register 10-bit configuration memory. Both memories have a parity bit, but no meta bit. The operation of both memories is described in somewhat more detail in Chapter 12.

4-8.2 X MEMORY

4-8.2.1 GENERAL DESCRIPTION. There are three modes of operation for the magnetic-core X Memory: READ-WRITE, READ, AND CLEAR-WRITE. Unlike the S, T, and U memories, two cores per bit in a single memory plane are used. The bits of a word are read out in parallel with a cycle time of 3.6 microseconds and an access time of 0.6 microsecond. In the X Memory, cycle time is the time between successive strobe pulses during a repetitive READ-WRITE cycle; access time is the minimum delay between setting the address register and strobing.

The X Memory system is shown in Fig. 4-19. The word selection method used is determined by the manner in which the N_J bits are decoded. The first five bits ($N_{3.5 - 3.1}$) of the six N_J bits are decoded by the J Decoder into 32 J Decoder levels. These levels represent the addresses of 32 pairs of registers in the X Memory. The selection of which of the two registers in a pair is determined by the sixth J bit ($N_{3.6}$). This bit is used in a second level selector, with the selection determined by the value of the bit.

4-8.2.2 MECHANICAL FEATURES. The cores used for the X Memory are 47 mils OD, 27 mils ID, and 12 mils thick. Both the digit and word selection windings make 4 turns on each core through which they pass. Fig. 4-20(a) shows the complete memory plane. Fig. 4-20(b) shows a portion of it enlarged. The cores are mounted on a lucite plane. The wires pass through openings made by the intersection of slots milled on one side of the plate with similar slots milled at right angles on the other side. The digit current is 8 ma, the write driver output current is 18 ma, and the read driver currents is 117 ma.

4-8.2.3 OPERATING PRINCIPLE. Each bit in the X Memory has an A and B core associated with it. Each of these cores may be in either a CLEARED or SET state, but both cores cannot be in the SET state at the same time. If an A core is SET, a ONE will be read-out during the READ process when this core is switched. Similarly, if the B core is SET, a ZERO will be read-out during the read process if this core is switched. Whichever core is switched, both will be left in the CLEARED state at the end of the READ process.

The winding configuration is shown in Fig. 4-21. A word is selected by connecting the upper end of a word line, e.g., Y, to a specified voltage (-3 V). The READ driver then puts out a current pulse 4-1/3 times that required to switch a core (on a 2:1 basis). (See Fig. 4-22.) Only one of the two cores (per bit) A or B is switched to the CLEARED state, since any previous WRITE operation will have left one core SET, and one core CLEARED. The switched core generates a pulse in its digit line. This line passes through one of the cores in the same direction as the word line and through the other core in a direction opposite to the word line. Thus, the polarity of the induced pulse on the digit line during READ indicates whether a ONE or a ZERO is being read out.

The ends of the digit line are connected to two differential amplifiers, each of which responds to pulses of only one polarity. The output of one amplifier is fed to the SET TO ONE input side of the buffer flip-flop and the output of the other amplifier is fed to the SET TO ZERO side of the buffer flip-flop. Thus the output of the memory is jammed into the buffer register by the strobe pulse.

At all times, a digit bias current flows in the digit winding. The direction of this current is determined by the state of the buffer register flip-flop associated with the digit. The amplitude of the current is 1/3 that required to switch the core in a 2:1 system. This digit bias current is very small compared with the read current and thus has no effect on the READ process. However, the digit bias current does enter into the logic of the WRITE process. During WRITE, a current of 2/3 that required to switch the core is placed on the selected word line. If the buffer register bit contains a ONE, the polarity of the digit bias current will be such as to add to the select line current in the A core. The current in this core is now sufficient to switch it, and accordingly it will be SET. If the buffer register bit contained a ZERO the B core would have been SET in the same manner. Notice that while the digit bias current is adding to the select word current in one core, because of the winding configuration it will be subtracting from the select word current in the other core, i.e., the net current in that core will be 1/3 that required to switch the core. That core will be undisturbed and left in the CLEARED state.

Thus, the current ratio used during WRITE is 3:1 (i.e., 1:1/3) with a disturb current of no more than 1/3. Fig. 4-22 shows the timing and current relationships in cores A and B of Fig. 4-21. Note that during both the READ and WRITE processes, as well as during quiescent periods, there is a current of ± 1/3 in all the cores.

The complete READ-WRITE cycle has been described above. Sometimes the READ operation is used alone, when there is no need or time to perform a complete READ-WRITE cycle. When the computer returns to WRITE in registers have had a READ cycle only, the CLEAR-WRITE cycle is used. CLEAR-WRITE is the same as READ-WRITE except that the strobe into the buffer register is eliminated. The CLEAR-WRITE cycle is necessary to insure correct writing, since if a WRITE operation should follow a previous WRITE operation in the same register, some bits might have both A and B cores SET.

4-8.3 F MEMORY

4-8.3.1 GENERAL DESCRIPTION. The configuration memory is used to store 32 configuration words. Each word has a word length of 9 bits plus a parity bit.

This memory differs from the other memories in the computer in that it is a thin magnetic film memory. A film memory has several potential advantages over the familiar ferrite toroidal core memory: faster cycle time, lower power dissipation, greater compactness, and simpler fabrication of the wiring arrays.

4-8.3.2 MECHANICAL FEATURES. Two 16 X 16 arrays, one of which is shown in Fig. 4-23, are placed side by side to form the memory. Each of the 256 individual film elements of an array are round spots 0.060" in diameter and 5×10^{-6} inches thick and are composed of 81% nickel and 19% iron. They are deposited on a small plate of glass 1.6 inches square with a thickness of 0.007 inch. The center to center spacing of the spots is 0.10 inches. The films are relatively stable in ordinary environments and no special treatment or surface coatings are required. However, a coating of "Krylon" is applied to resist abrasion during handling.

There are three lines associated with each spot: (1) word, (2) digit and (3) sense lines. Fig. 4-24 shows a complete set of word, digit and sense lines for one spot. Fig. 4-25 is a schematic view of the drive and sense line winding configuration.

Each word line is driven by a magnetic core. These cores serve a purpose similar to that of the switch cores in the S Memory. Each core has two selection windings, one of which is based on a decoding of three of the address bits and the other on the remaining two address bits. The diode

in the word lines permit current to flow only when the core is selected. When a core is deselected, a bias current in the selection windings switches the core back to its normal state.

The digit and sense lines are parallel. The digit lines usually have current flowing in one direction. The current is reversed only during a write cycle when a ONE is being written. The sense line picks up the signals generated by a film switching during read-out. Note that the sense windings are crossed between the two memory arrays to minimize noise pickup.

4-8.3.3 OPERATING PRINCIPLE. The thin films are fabricated in such a way as to have an easy axis of magnetization. Application of an external field along this axis simultaneously with the application of a small transverse field can cause the film to switch from one magnetized state to the other. The transverse field is generated by the word line current pulse and the longitudinal field by the digit line current level.

One of the directions of the film's magnetization along the easy axis is arbitrarily chosen to represent a ONE, and the other direction to represent a ZERO. During a READ the field generated by the digit line is in the ZERO direction. If the film is in the ONE state, it will switch to a ZERO and the resulting rotating field induces a voltage in the sense line. If the film is a ZERO, there is no switching but merely a slight disturbance of the film's field and no voltage is induced in the sense line. Thus it can be seen that the READ is a destructive type in that it clears the memory word register. During a write operation, if a ONE is to be written, the digit current is reversed so that the field switches the film into the original ONE state. If a ZERO is to be written, the digit current is not changed and the film remains in the ZERO state it was placed in during the destructive read out.

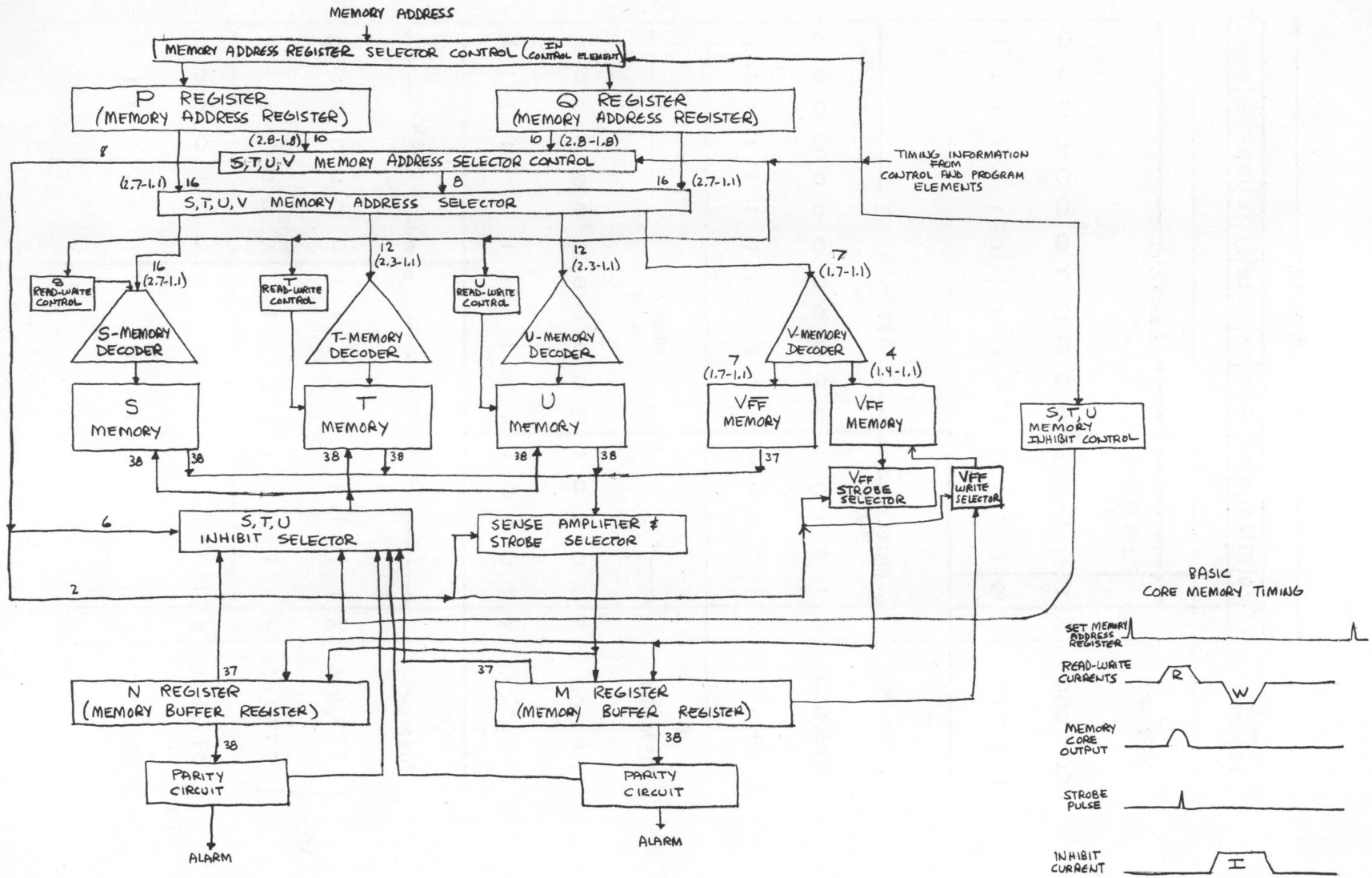


FIGURE 4-1 MEMORY ELEMENT, SIMPLIFIED BLOCK DIAGRAM

P OR Q REGISTER	2.9	2.8	2.7	2.6	2.5	2.4	2.3	2.2	2.1	1.9	1.8	1.7	1.6	1.5	1.4	1.3	1.2	1.1
	18 BITS																	
S MEMORY		SELECTION (1)	ADDRESS (16)															
000,000	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
⋮																		
177,777	X	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
T MEMORY		SELECTION (5)					ADDRESS (12)											
200,000	X	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
⋮																		
207,777	X	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
U MEMORY		SELECTION (5)					ADDRESS (12)											
210,000	X	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
⋮																		
217,777	X	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
V MEMORY		SELECTION (10)										ADDRESS (7)						
VFF 377,600	X	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
⋮																		
VFF 377,617	X	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1
$\overline{\text{VFF}}$ 377,620	X	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0
⋮																		
$\overline{\text{VFF}}$ 377,777	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FIGURE 4-2. MEMORY ADDRESS SELECTOR CODING

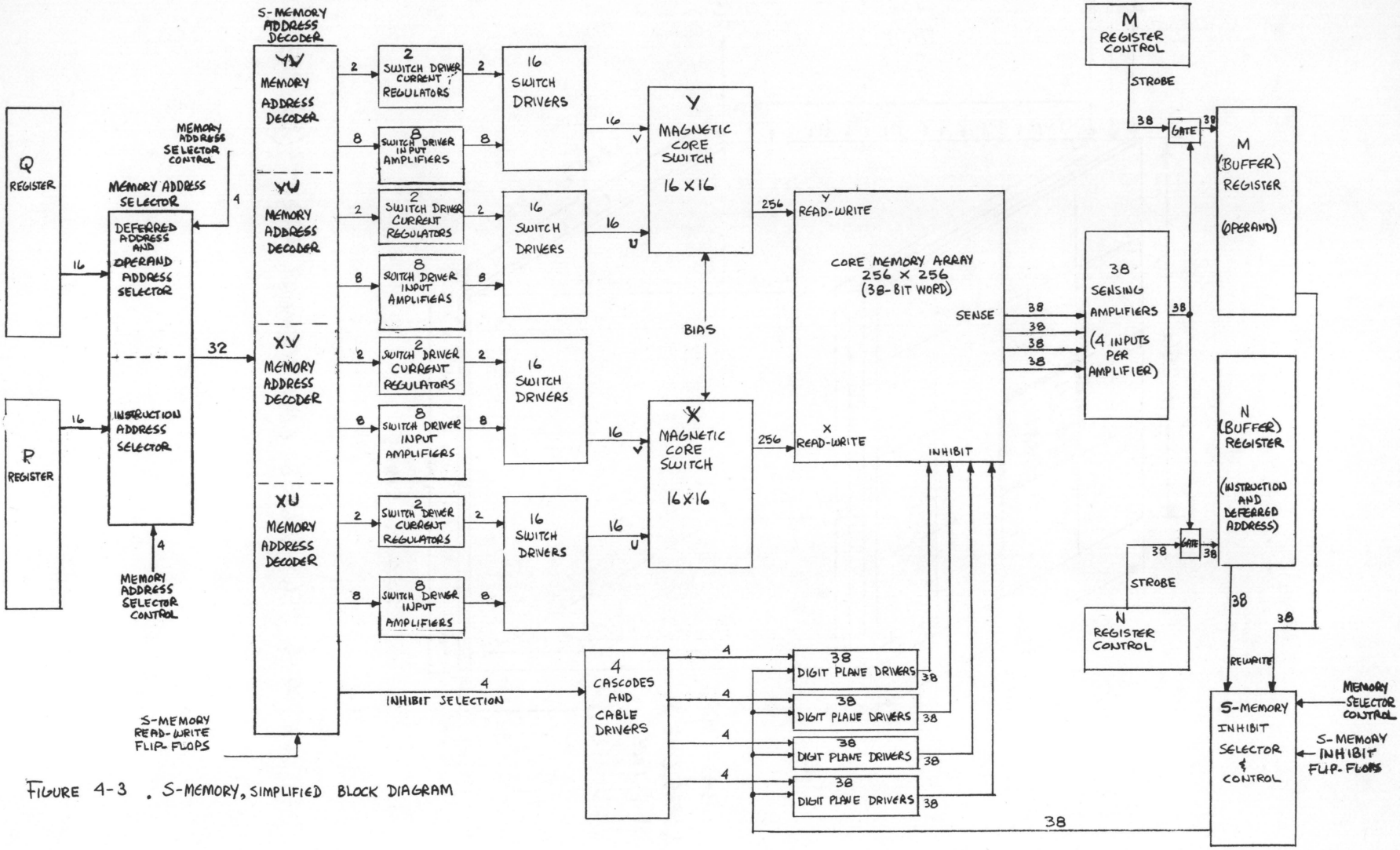


FIGURE 4-3 . S-MEMORY, SIMPLIFIED BLOCK DIAGRAM

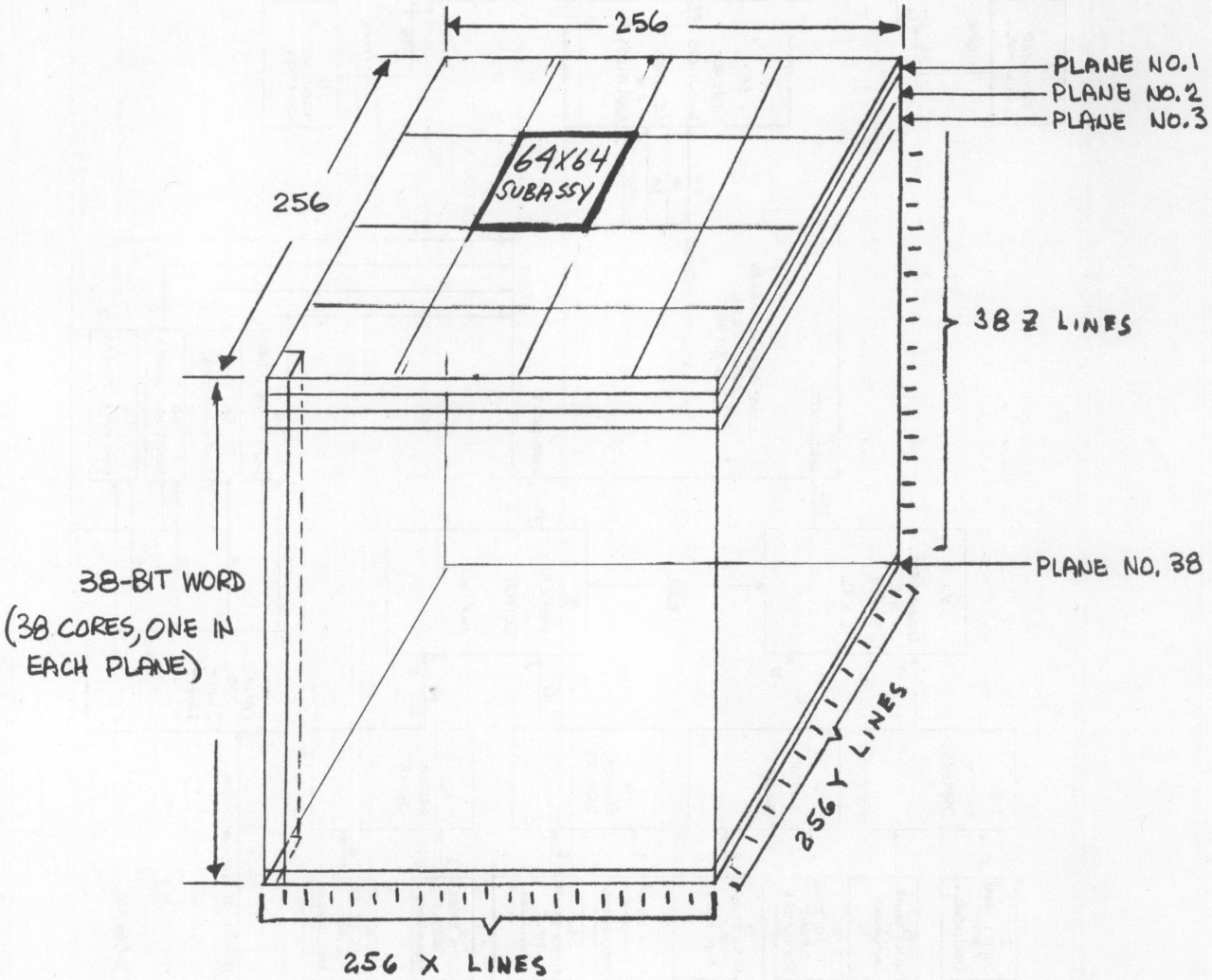
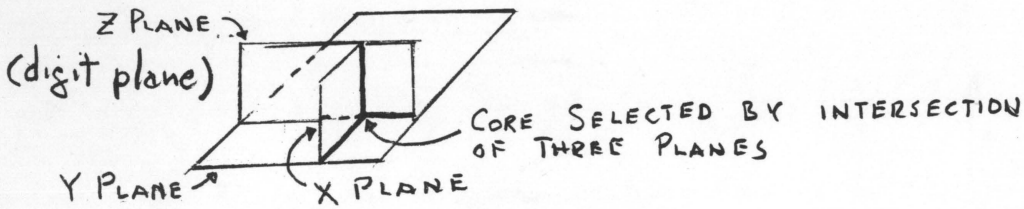
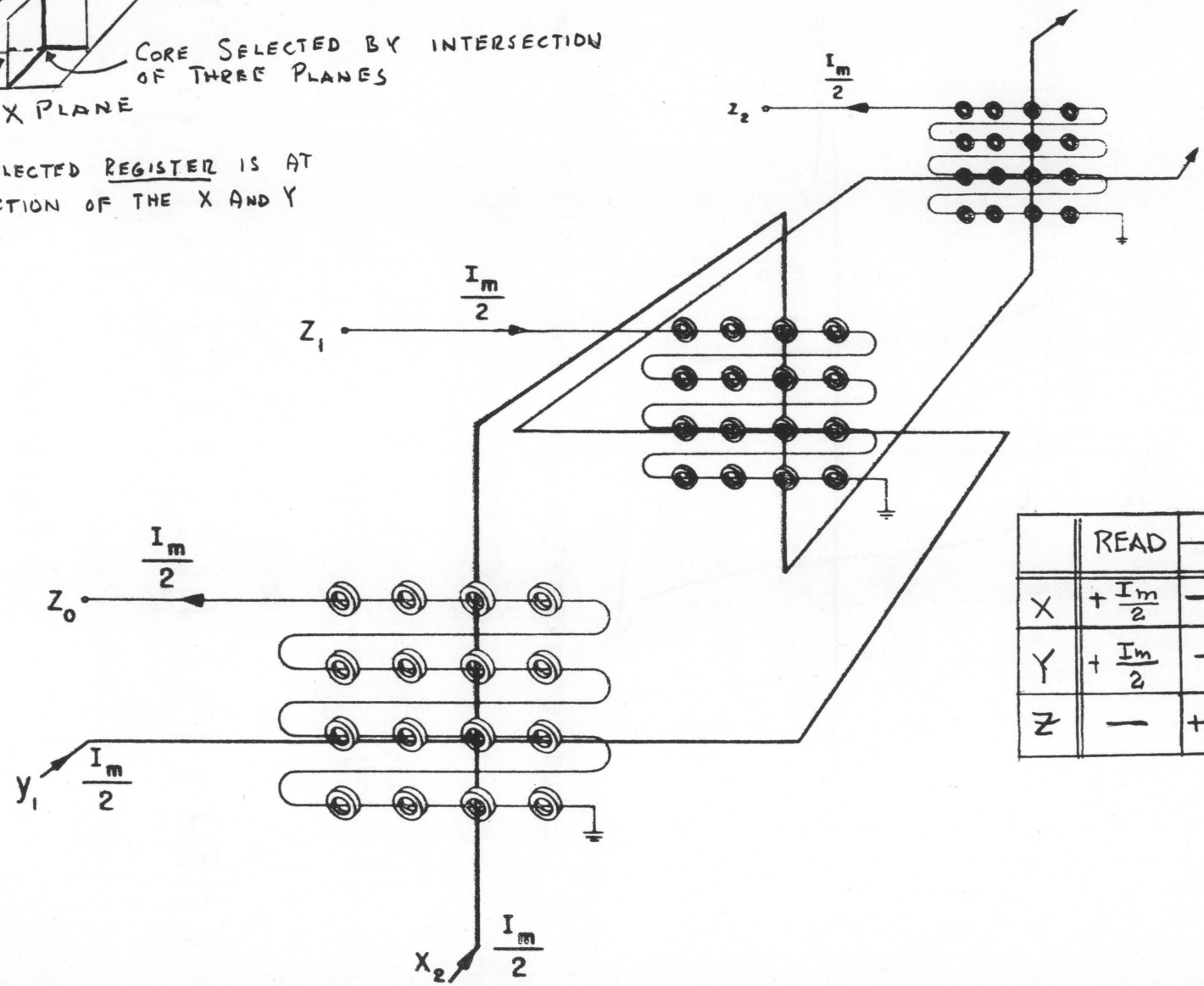


FIGURE 4-4. 256 X 256 X 38 CORE MEMORY ARRAY (S MEMORY)

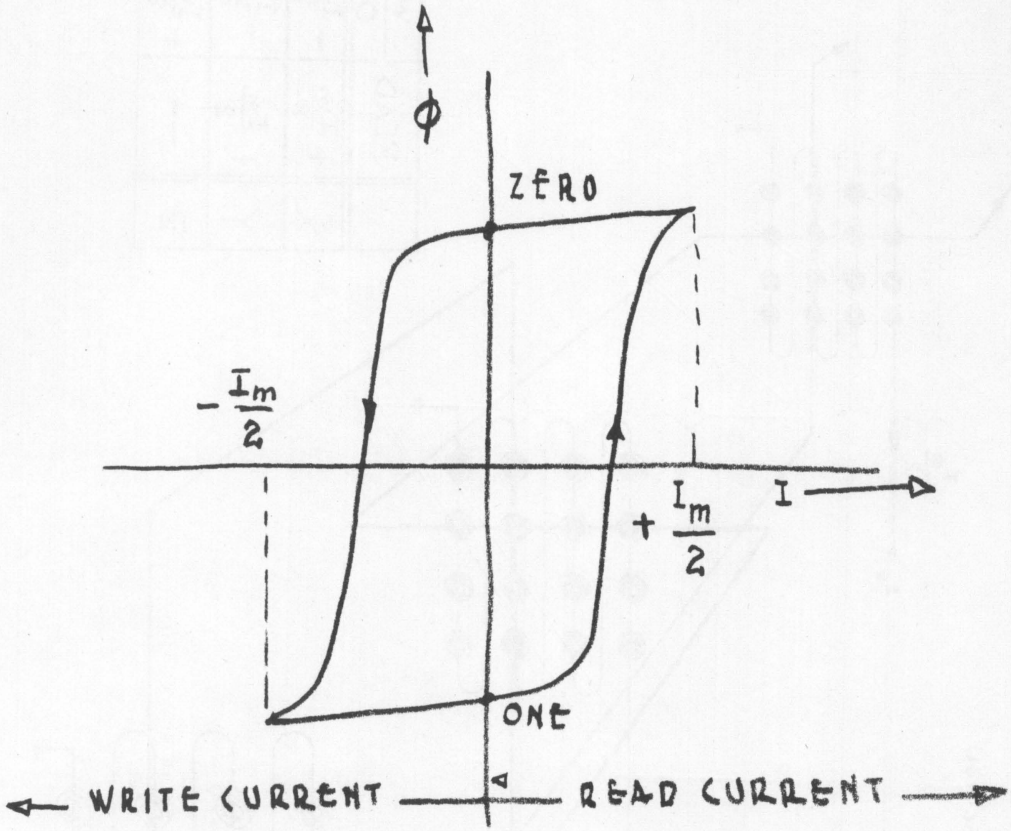


NOTE: THE SELECTED REGISTER IS AT THE INTERSECTION OF THE X AND Y PLANES.



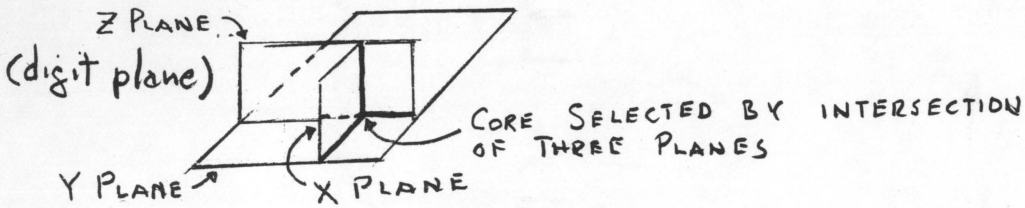
	READ	WRITE 0	WRITE 1
X	$+\frac{I_m}{2}$	$-\frac{I_m}{2}$	$-\frac{I_m}{2}$
Y	$+\frac{I_m}{2}$	$-\frac{I_m}{2}$	$-\frac{I_m}{2}$
Z	—	$+\frac{I_m}{2}$	—

Fig. 4-5 S MEMORY WINDING CONFIGURATION

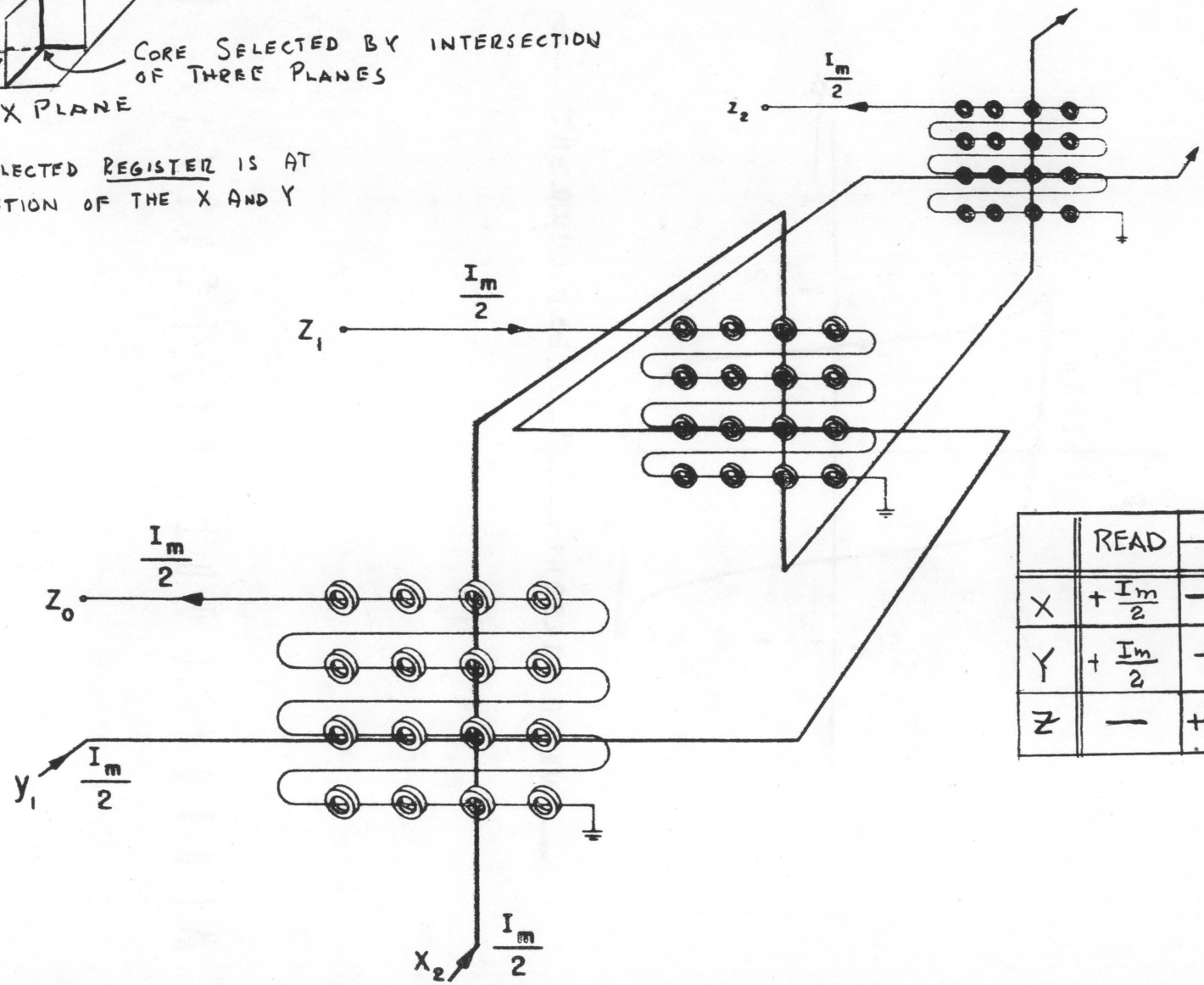


MEMORY CORE HYSTERESIS LOOP

FIG 4-6

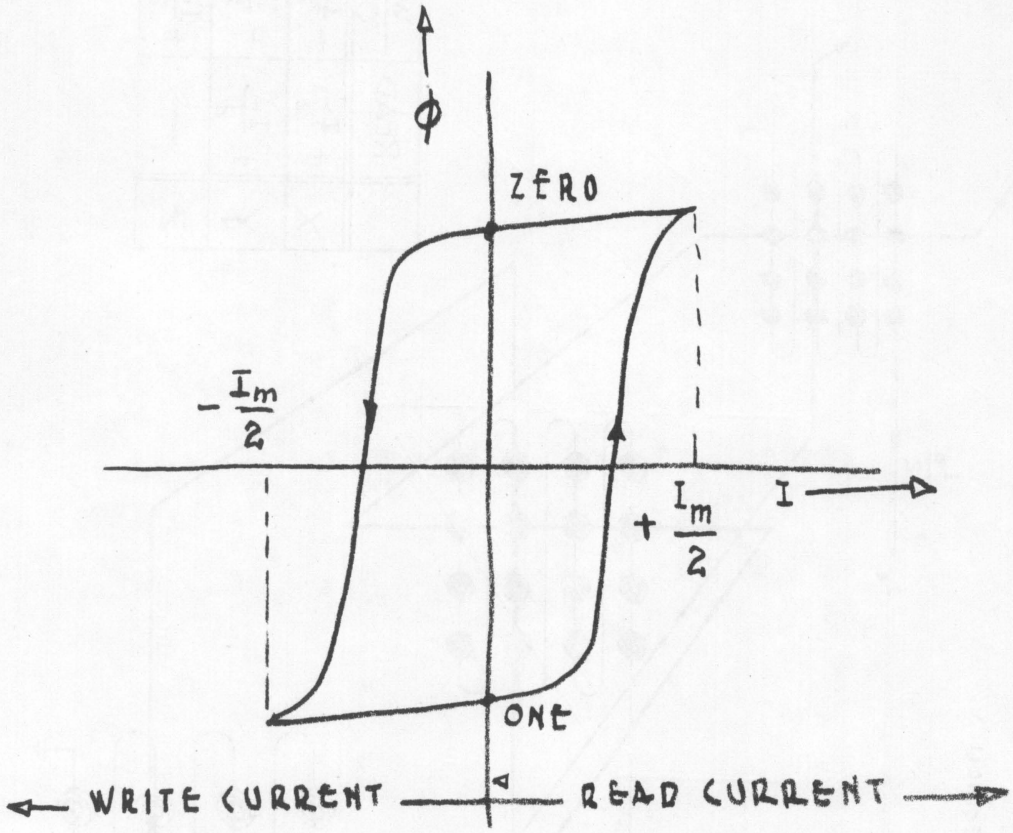


NOTE: THE SELECTED REGISTER IS AT THE INTERSECTION OF THE X AND Y PLANES.



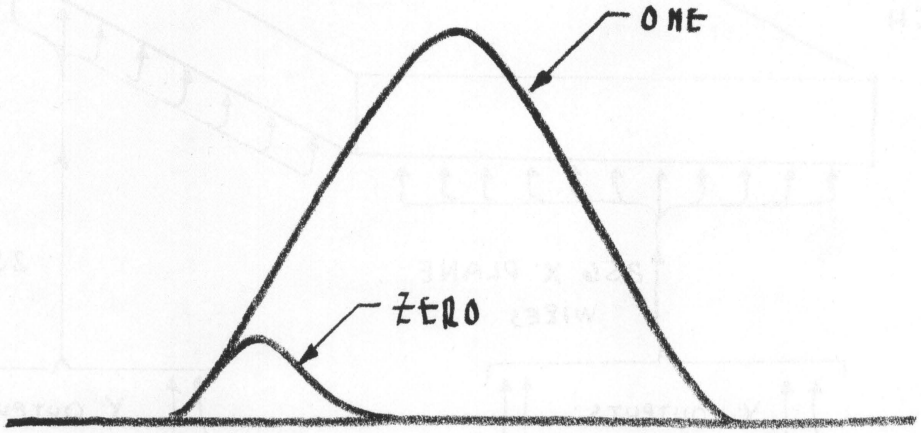
	READ	WRITE 0	WRITE 1
X	$+\frac{I_m}{2}$	$-\frac{I_m}{2}$	$-\frac{I_m}{2}$
Y	$+\frac{I_m}{2}$	$-\frac{I_m}{2}$	$-\frac{I_m}{2}$
Z	—	$+\frac{I_m}{2}$	—

Fig. 4-5 SMEMORY WINDING CONFIGURATION



MEMORY CORE HYSTERESIS LOOP

FIG 4-6



ZERO AND ONE OUTPUTS FROM A MEMORY CORE

FIG 4-7

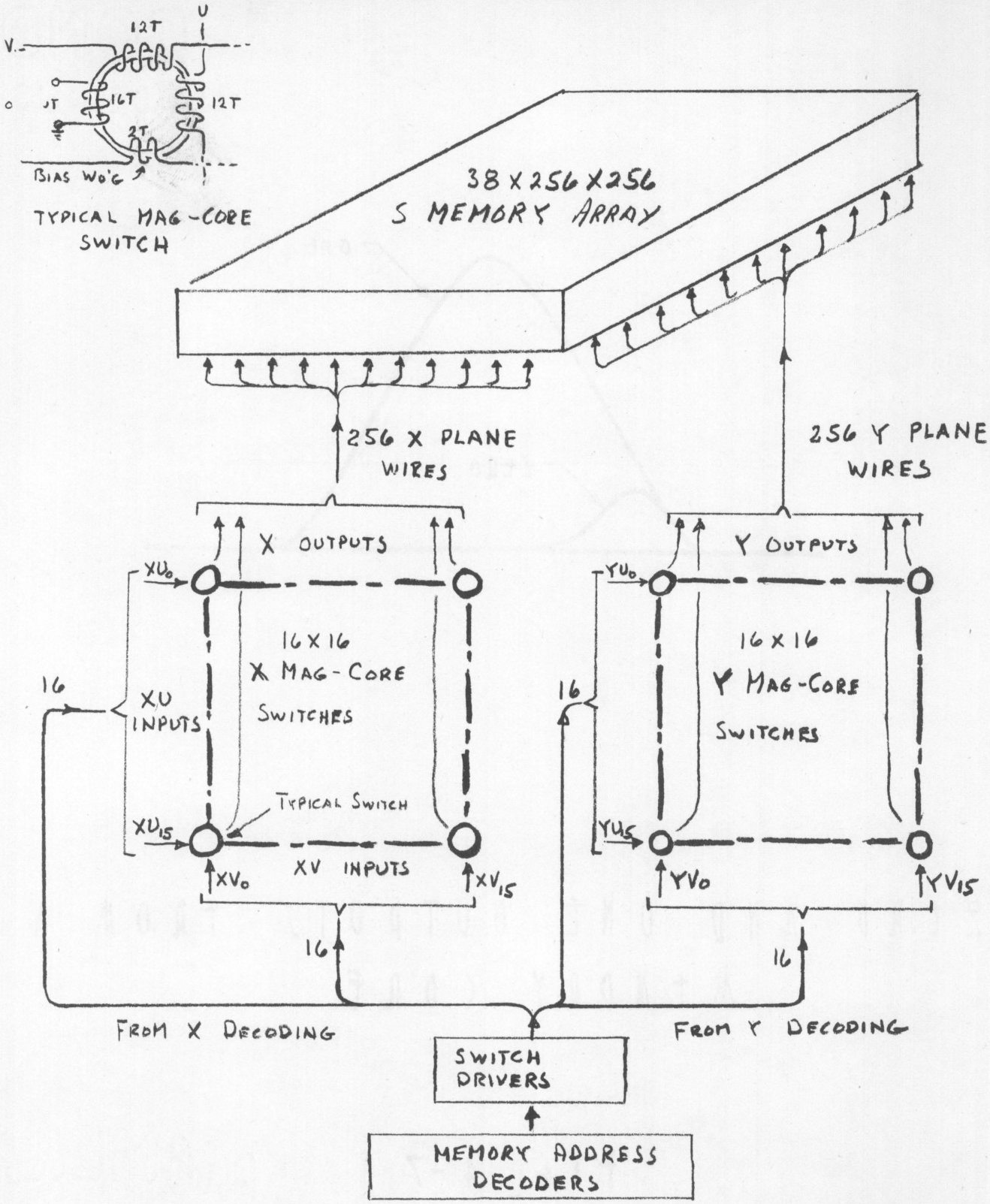


FIG. 4-8 MAGNETIC-CORE SWITCHING

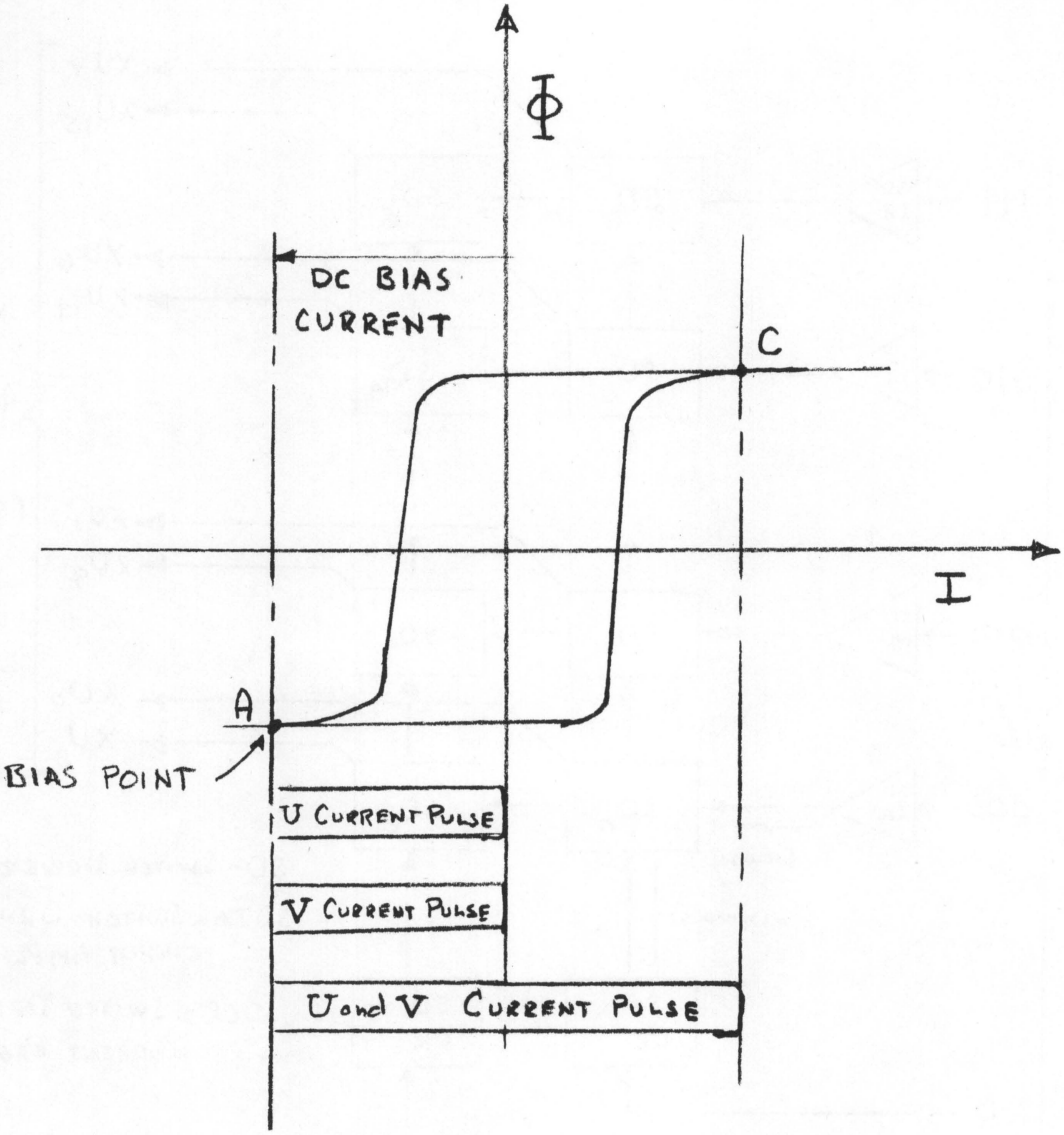


Fig. 4-9 MAGNETIC-CORE SWITCH CHARACTERISTIC

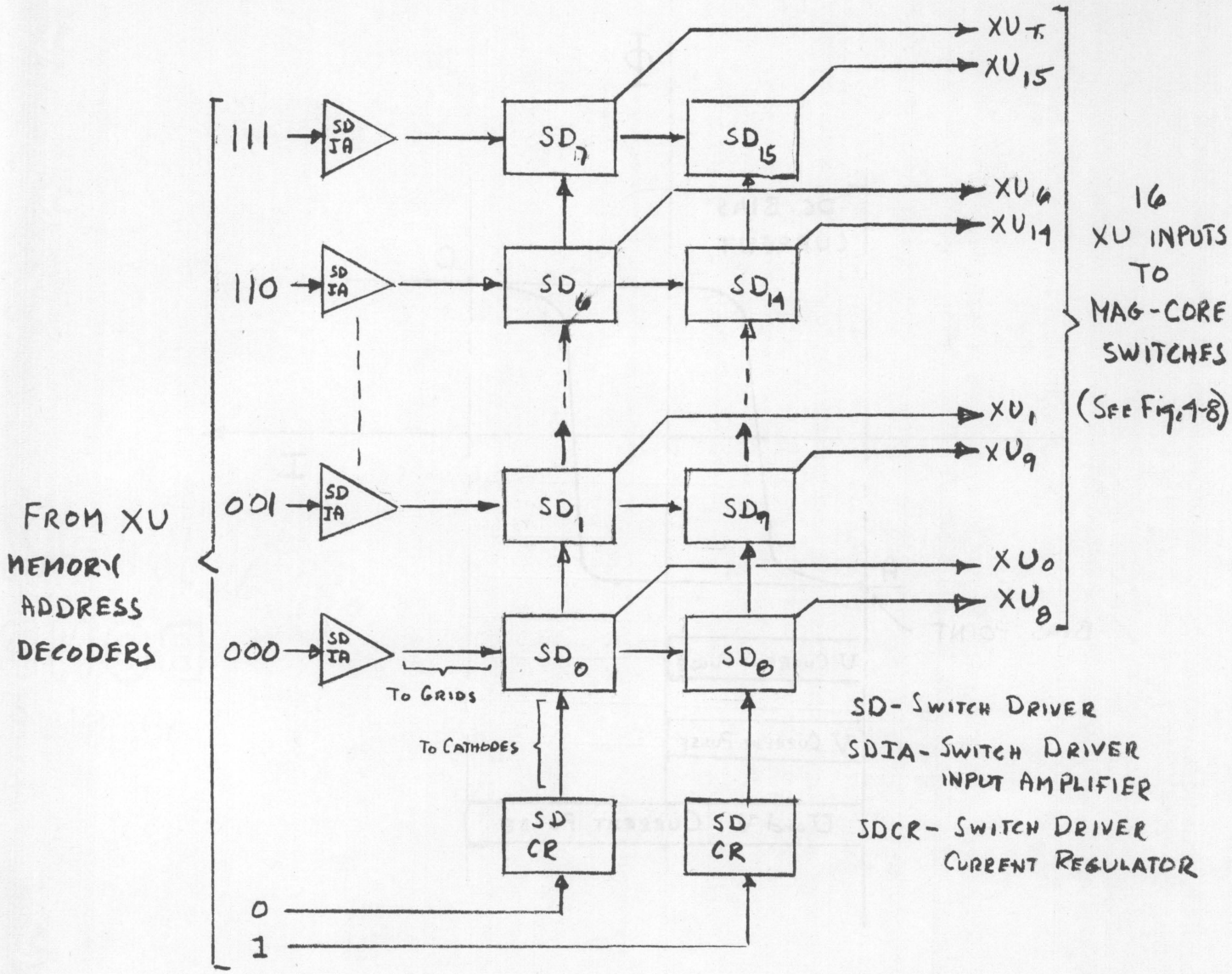


FIG. 4-10 BLOCK DIAGRAM OF XU SWITCH CORE DRIVING SYSTEM (SIMILAR DRIVING SYSTEMS EXIST FOR THE XV, YU, and YV MAG-CORE SWITCH INPUTS)

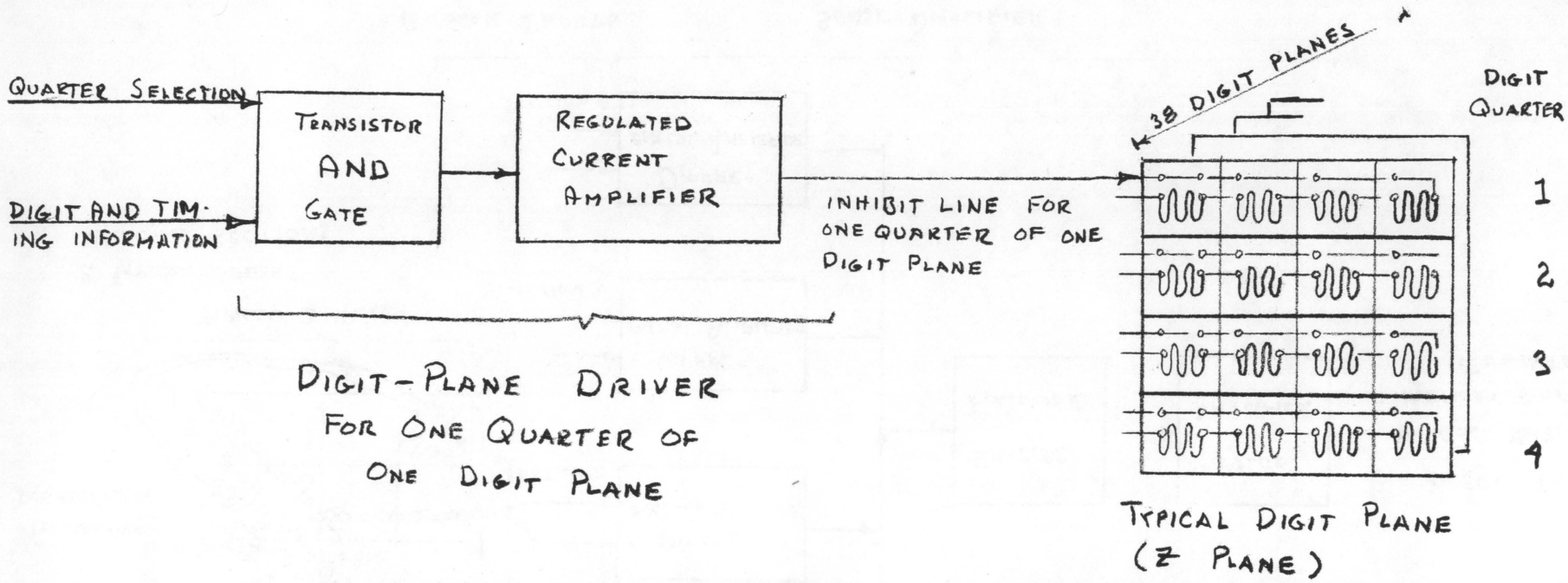


FIG 4-11 BLOCK DIAGRAM OF DIGIT-PLANE DRIVER.

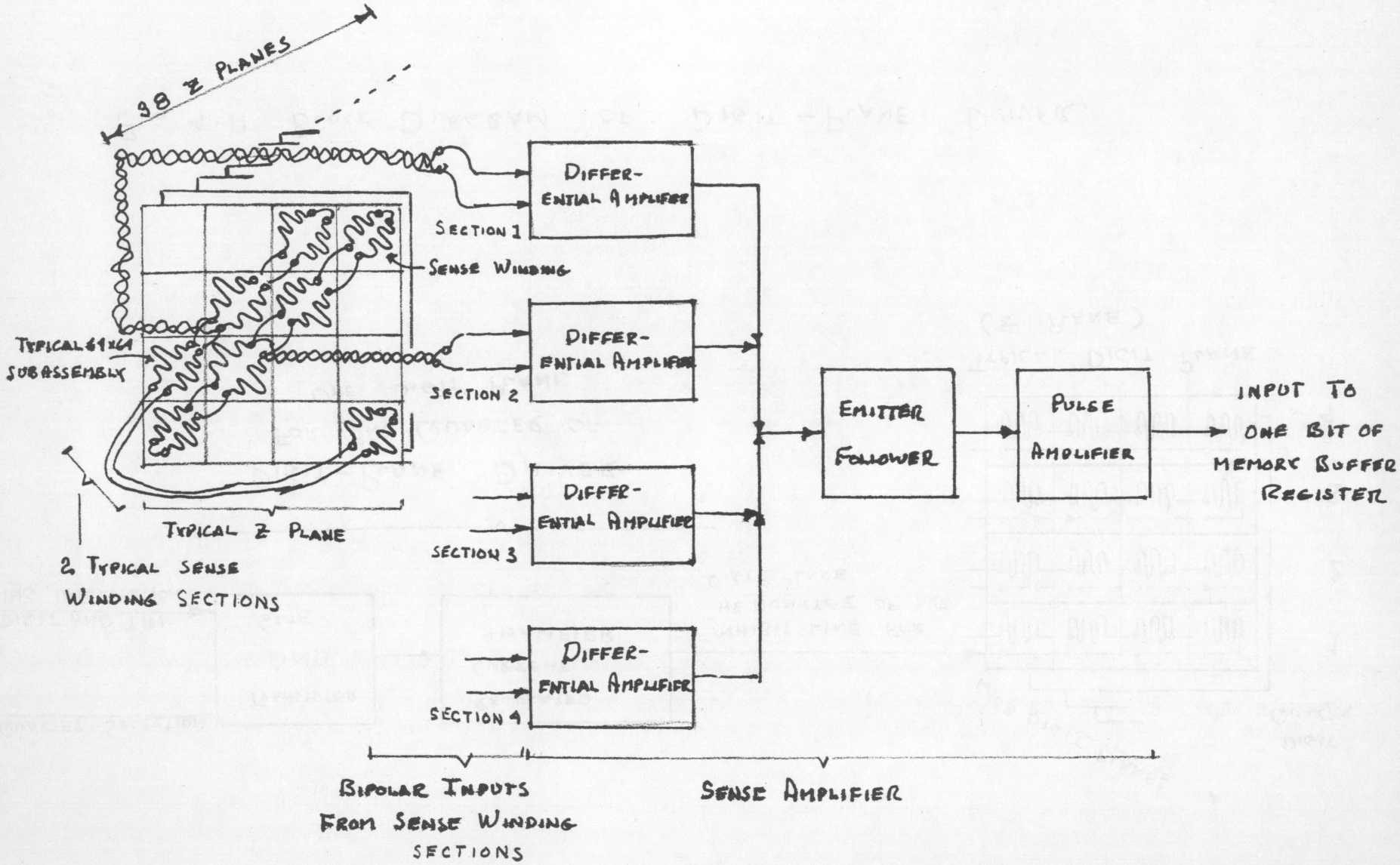
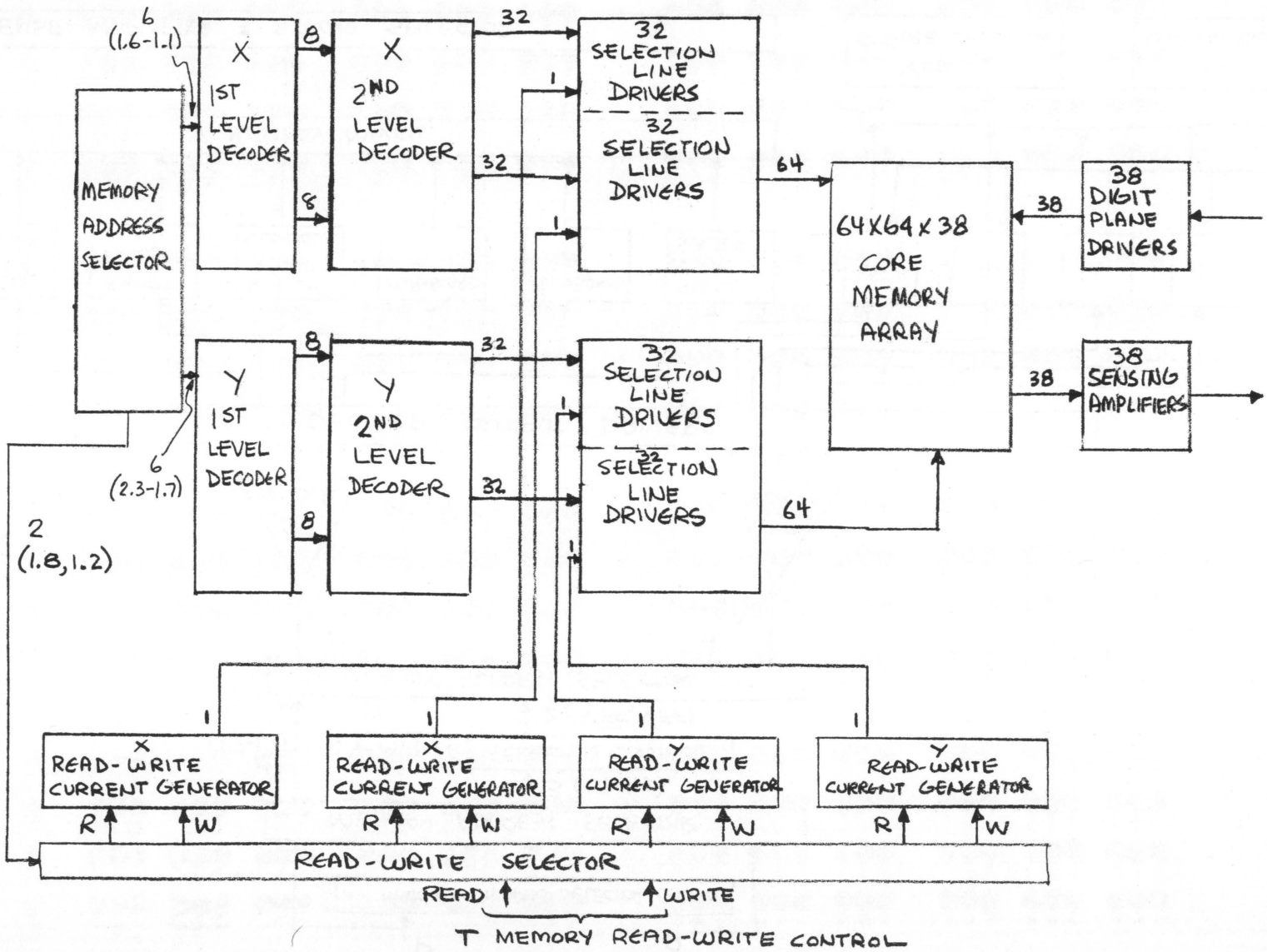
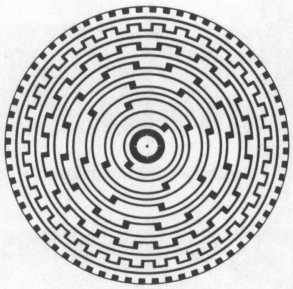


FIG 1-12. BLOCK DIAGRAM OF SENSE AMPLIFIER

FIGURE 4-13. T MEMORY, SIMPLIFIED BLOCK DIAGRAM



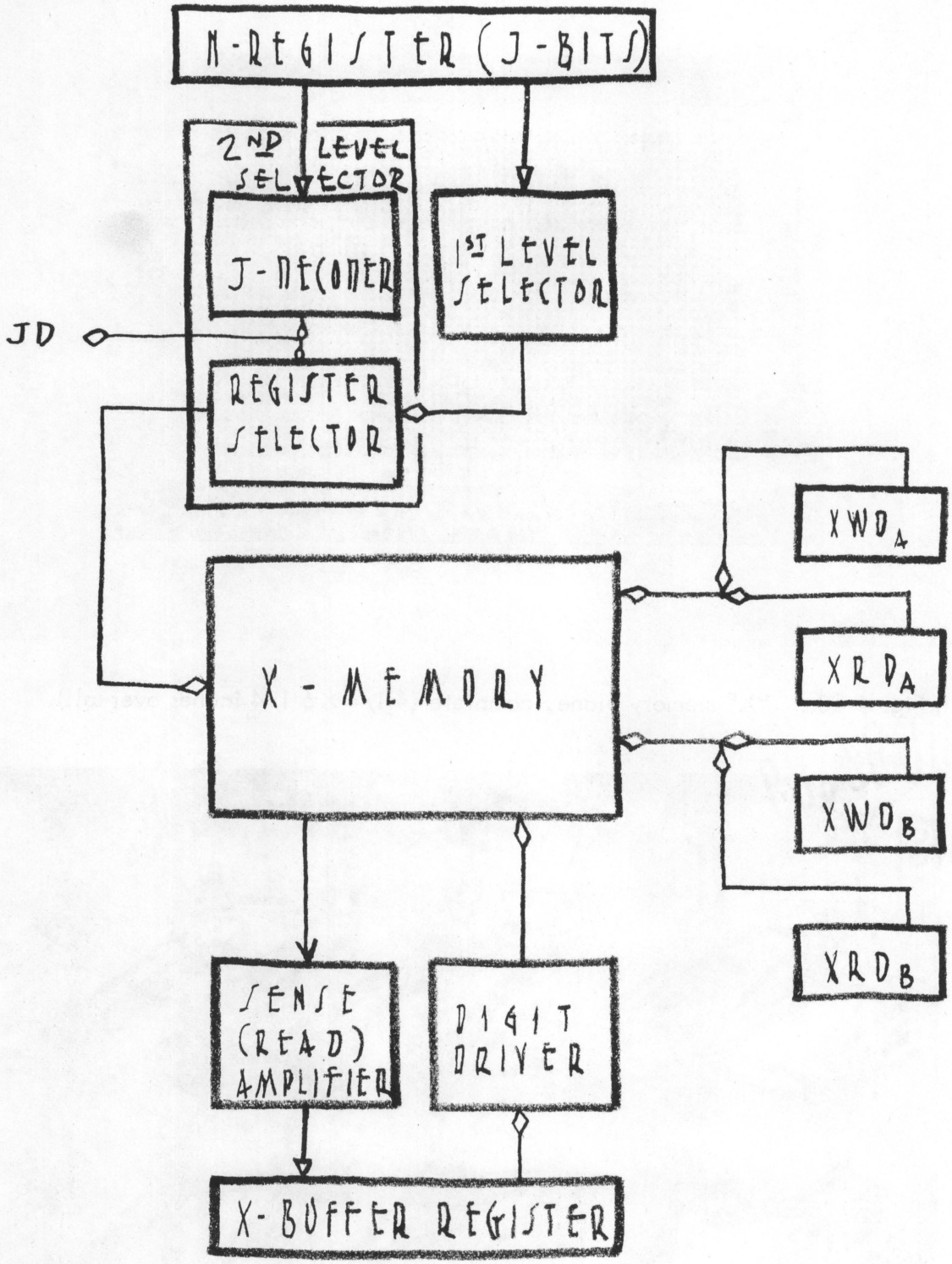
A. Photograph of shaft encoder.



B. Converter disc.*

Fig. 4-18. Shaft encoder.

* Reproduced by permission.



X-MEMORY SYSTEM

FIG 4-19

HM 8-25-60

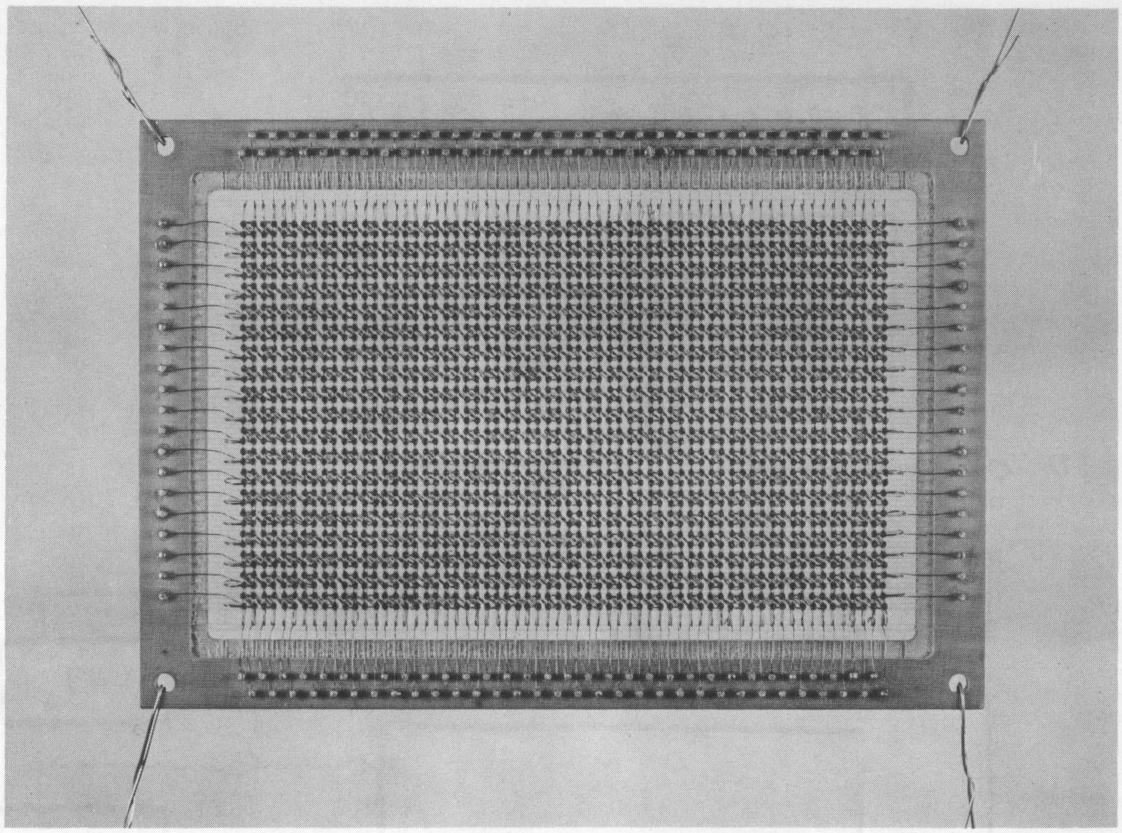


Fig. 4-20a. "X" memory plane, complete (4 1/4 × 6 1/4 inches over-all).

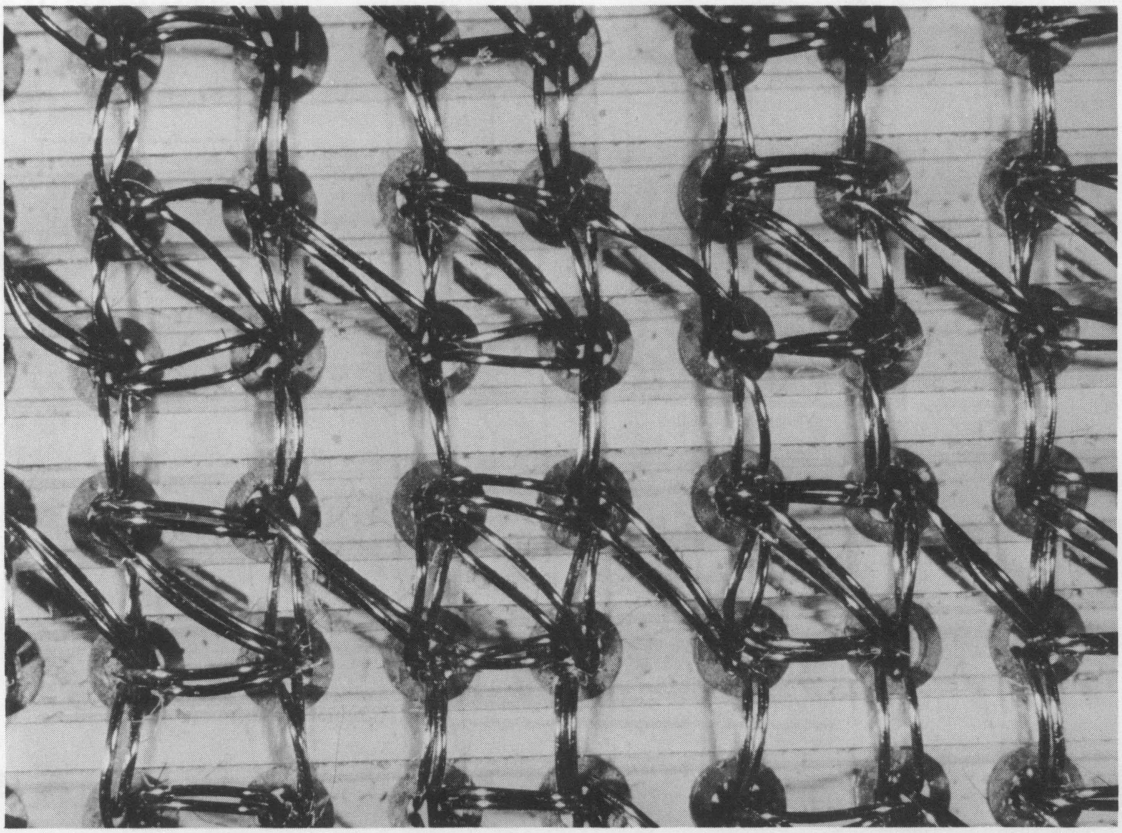


Fig. 4-20b. "X" memory plane enlarged.

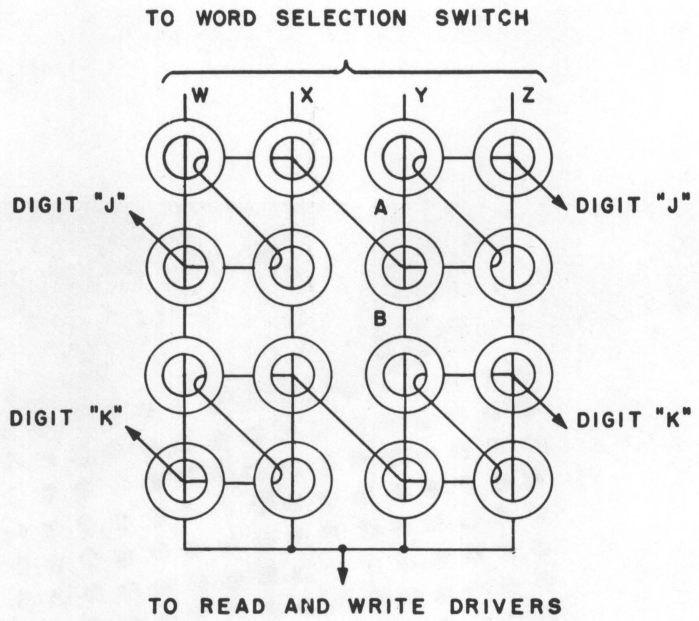


Fig. 4-21. "X" memory winding configuration.

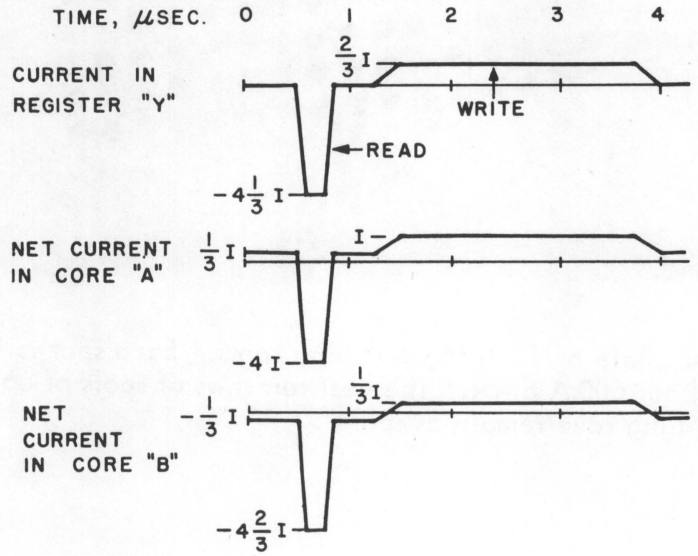


Fig. 4-22. "X" memory timing diagram.

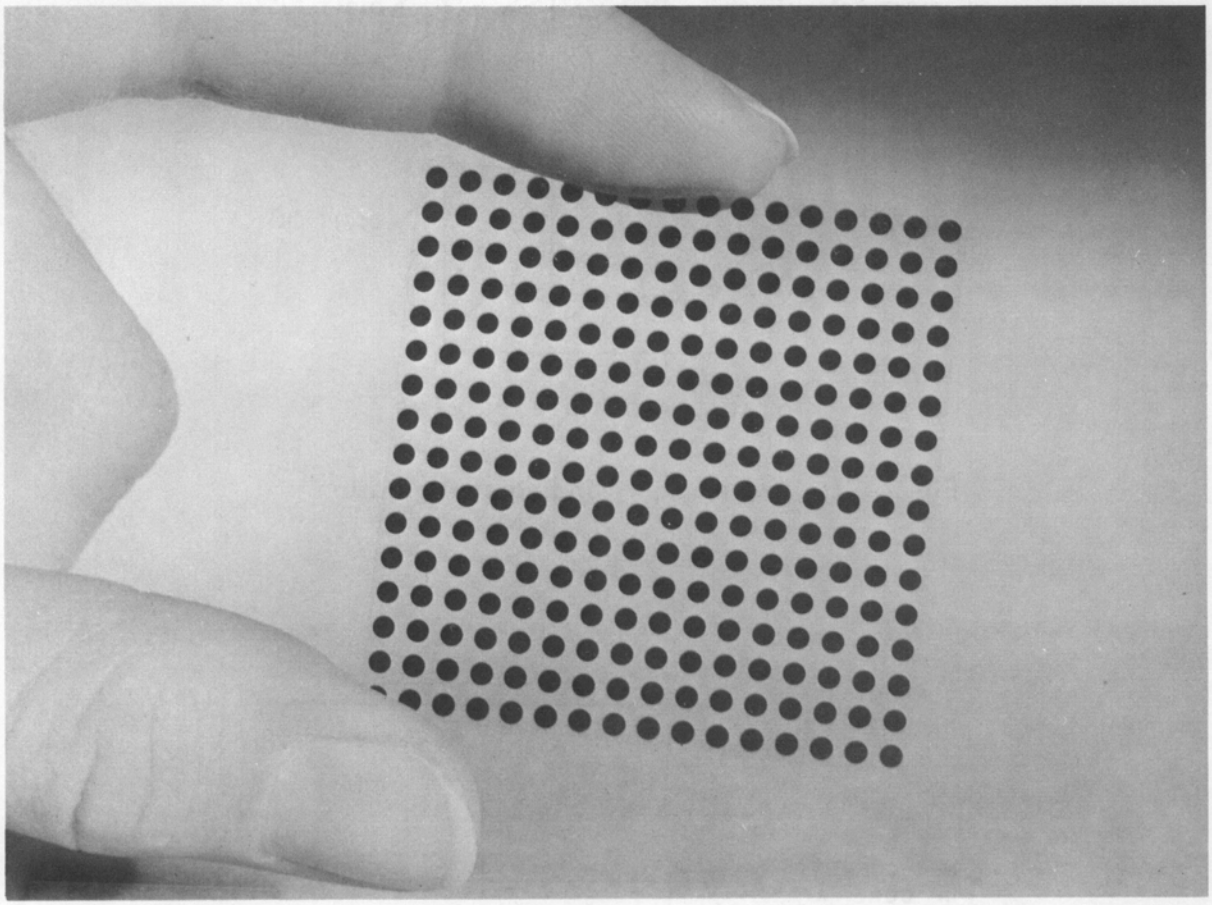


Fig. 4-23. One plate of 16 × 16 permalloy spots. Each spot is 1/16 inch in diameter and about 600 Å thick. The best ten rows of spots of each plate are used; the remaining rows remain as spares.

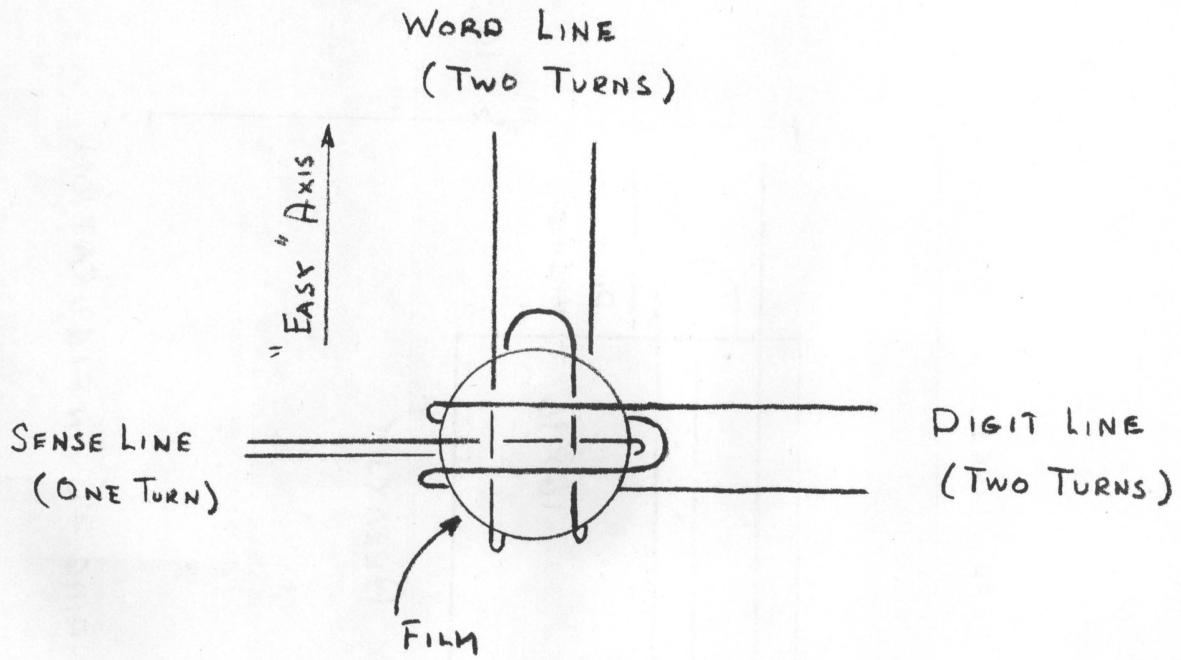


FIG. 4-24 WORD, DIGIT AND SENSE WINDINGS ON THIN FILM MEMORY SPOT

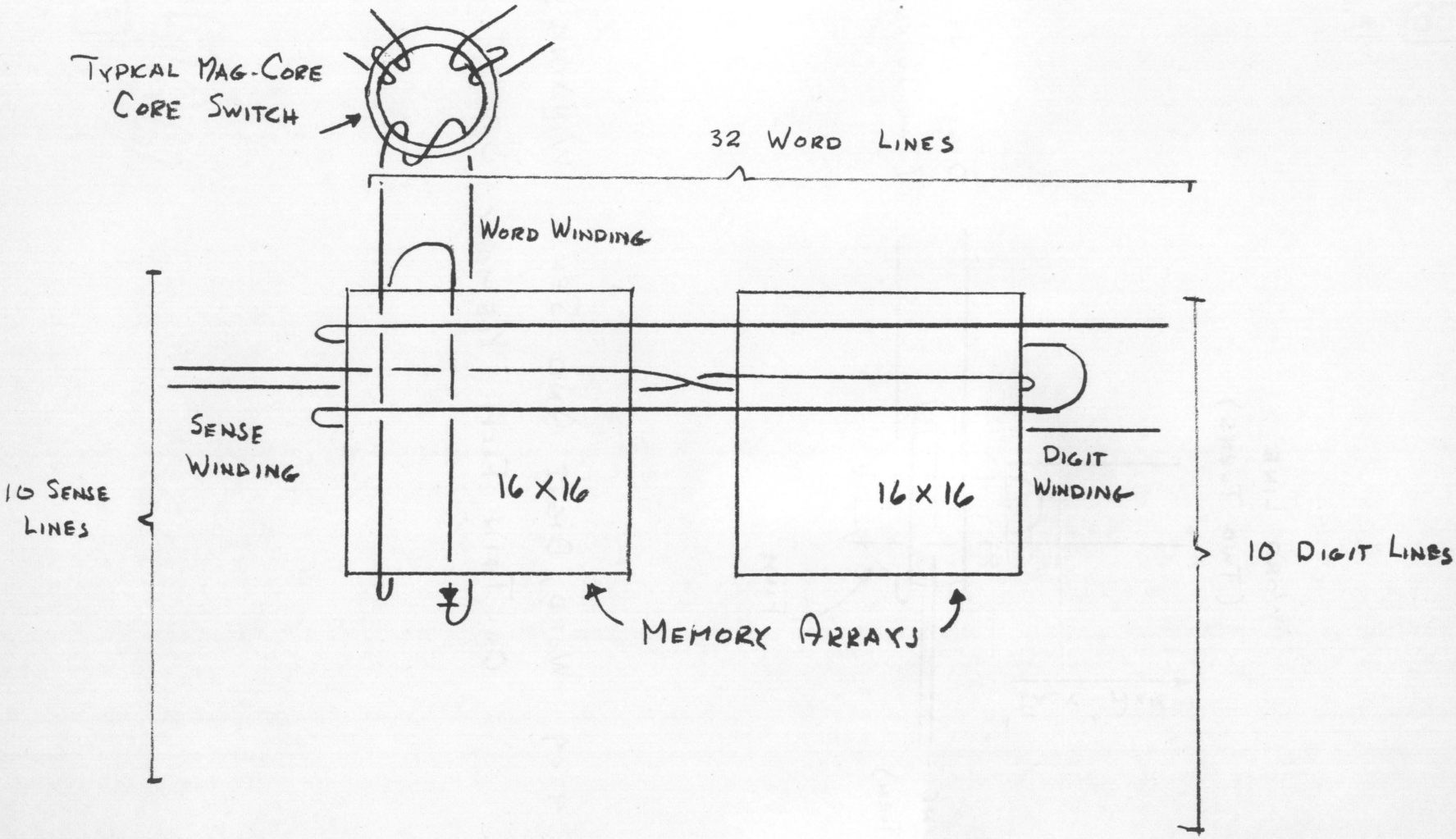


FIG 4-25 F MEMORY WINDING CONFIGURATION

CHAPTER 5

TIMING AND CONTROL

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 5

TIMING AND CONTROL

5-1 INTRODUCTION

The pattern of activity in time that takes place in the computer is determined by the Control Element. This chapter will first describe the different sequences of events that make up this pattern. It will then explain the basic features of control and timing that keep the computer running in a disciplined fashion. Chapter 6 will give an integrated functional description of the Control Element itself.

Previous chapters established that memories, logic nets, flip-flops, and other components require finite but determinate lengths of time to operate. The fact that the operating times are determinate means that maximum times to satisfy the "worst condition" situations can be established. After initiating one event, it is only necessary to meter out the required time before initiating the next event. The hardware for performing this metering is usually a counter, although sometimes a delay line is used.

The simplest dynamic picture of the computer illustrates how data is transferred in and out of storage devices as a function of time. It is only necessary to understand the time patterns during this shuffling of data in order to understand the basic control and timing features of the computer.

5-2 PATTERN OF ACTIVITY

Two basic cycles are dominant in the general pattern of recurrent execution of instructions: the instruction cycle and the operand cycle. Other subordinate and auxiliary cycles occur but they play a secondary and dependent role. Fig. 5-1(a) shows what is occurring in time from this elementary viewpoint. Some of the subordinate activity is shown in Fig. 5-1(b). During a typical instruction cycle an instruction is first strobed out of the main memory into the N register. The contents of the N register are then decoded. At some point, the base address in the N register may be modified by indexing it with the contents of an X Memory register. During a typical operand cycle, an operand is first strobed out of memory into the M register of the Exchange Element. The operations on the operand called for by the instructions are then performed. The basic mechanisms involved in these operations were described in Chapter 2.

Within the design limits of the computer, there is an obvious advantage in compressing the amount of time required by the various patterns of activity. One method of reducing the over-all time is to overlap the basic cycles and the subordinate activities. Usually one cycle or phase need not be completed before the next begins. Fig. 5-2 shows how the pattern illustrated in Fig. 5-1(b) can be compressed in time.

5-3 OVERLAPPING

Some explanation of the points at which cverlap can generally occur is in order. Once an instruction word has been read out of a memory element register and strobed into the N register, it is possible to immediately start decoding the N register even though the instruction word memory cycle is not completed until the instruction word is rewritten back into the memory. Similarly, it is not necessary that all the bits of the N register be decoded before the address modification process begins since only the Y base address and J index bits are required for this process. The address modification process is completed when the address of the operand appears at the output of the X Adder, ready for insertion in the Q register. At this point the process of reading the operand out of memory can begin.

The execution of the operation called for by the instruction can be initiated as soon as the instruction in the N register is decoded. Usually, the instruction calls for an operand. In Fig. 5-2, the operation is shown being initiated after the operand is strobed out of memory into the M register. The operation process can overlap the operand rewrite process whenever the operation does not involve modifying the operand before it is rewritten in memory. Thus, the processes performed during the execution of an instruction can be overlapped to a considerable extent.

In Fig. 5-3, the next instruction word memory cycle is shown overlapping the operand word memory cycle of the current instruction. This is allowable whenever the instruction word and operand word are obtained from different memories in the Memory Element. This kind of overlapping yields nearly all the speed advantages which can possibly accrue from overlapping memory cycles. In fact, the total effective time required to execute an instruction can be reduced to little more than one memory cycle.

It should be noted that the preceding description of overlapping has been highly simplified. Both the basic cycles and the subordinate forms of activity vary widely depending on the specific instruction and the prior state of the computer. More details on overlapping are covered in Chapter 9.

5-4 CONTROL SCHEME

5-4.1 GENERAL. The basic control requirements involve some scheme for metering out time for all the various activities that occur. This metering is done by counters. Each of the basic cycles has its own time meter, or counter. In addition, several other types of computer activity, such as those involving the X and F Memories, Change of Sequence, the Arithmetic Element, etc., have a counter associated with them. These counters will be described in Chapter 6 and Chapter 10.

It is also important to have a control scheme for determining precisely when a new activity can begin. This is accomplished by an interlock control. An interlock is

a storage device (a flip-flop) which remembers when various conditions have occurred. A number of interlocks are used to determine when counters can begin their cycles and thus when the associated processes can be performed.

5-4.2 COUNTERS. The general characteristics of counters as they relate to the control of the computer will now be described. The basic function of the counter is to convert a stream of indistinguishable clock pulses into distinguishable time levels. These levels can then be used to select, or "gate" clock pulses at specific times. Consider for the moment a free running counter (Fig. 5-4), that is, one in which an uninterrupted stream of clock pulses is emanating from the related counter register driver. Levels of 0.4 microsecond duration appear on the output wires of the time level decoders in the time order shown in Fig. 5-4. After the counter reaches the 6α state, it reverts to the 0α state and begins another cycle.

One point should be emphasized. The clock pulse that causes the counter to generate the 5α time level occurs 0.4 microsecond prior to the clock pulse that can be gated by the 5α time level. The clock pulse that causes the 5α time level, occurs while the counter is in state 4α, and can in fact also be gated by the 4α time level. This distinction should be borne in mind when attempting to determine when events, caused by interlocks and time levels, occur with respect to each other.

As mentioned earlier, the various counters in the computer are associated with specific kinds of computer activity. When the counter has completed its cycle (which is tantamount to completing the associated activity), it will be inhibited from recycling by interlocks. When this activity is again required, the interlock will permit the counter to start again.

5-4.3 INTERLOCKS. Interlock level logic controls the counter register driver logic nets. The inputs to these nets come from interlock flip-flops and from other sources of interlock information in the computer. Basically, the interlock logic indicates that all the things which must be done before starting a counter have in fact been done. As indicated above, nothing prevents two counters from operating simultaneously. The only obvious limitation is that two counters may not control the operation of the same device simultaneously. Nothing prevents them from alternately doing this, however. The other somewhat arbitrary limitation is that usually no two counters start at the same time. All of the information necessary to impose these limitations finds its way into the interlock logic nets.

5-5 TIMING

There is considerable value in understanding how the clock pulses and the time levels from the time level decoders are specifically used in controlling the timing of events. All events occurring in the computer are initiated by gated clock pulses. This gating occurs in register drivers and is determined by the output of logic nets. The event itself may

arbitrarily be defined as the appearance of the gated pulse at the input to a flip-flop or memory device. Fig. 5-5 shows the form of the logic at the register driver. Many control levels of various kinds will enter the register driver logic nets. These levels will always be ANDed with time levels from one or more counters. This logic will in turn be ANDed with a clock pulse in the register driver itself to produce a gated clock pulse which can initiate an event.

Certain generalities may now be stated: (1) every clock pulse which is used in the computer is gated by some time level; and (2) every time level is coincident with some clock pulse. This coincidence is shown in Fig. 5-6. These two generalities are keystone ideas in time control in the computer.

Fig. 5-7 shows how the counters are integrated with the rest of the computer by means of their output time levels. This figure also indicates the feedback paths that can occur. Note that the counter register driver logic is similar to that of any of the other register drivers. It may in fact look at time levels from the same counter it is driving. After the register driver pulses have transferred data between registers, cleared and set flip-flops, read information out of memory, etc., the new states of the affected storage devices feedback through logic nets into the register drivers and then another cycle of events is repeated.

5-6 SYNCHRONISM

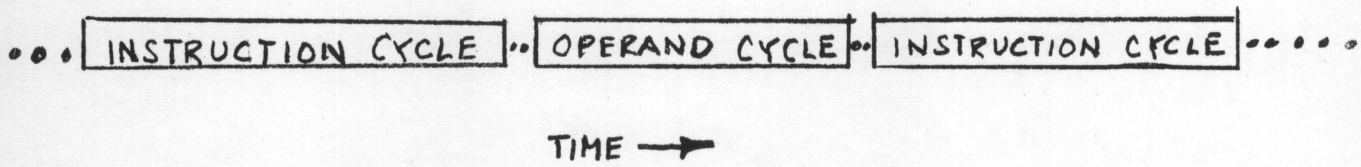
The operation of most computers is described as either synchronous or asynchronous. The term asynchronous implies the completion of an event is indicated by the occurrence of the event itself; that is, it is not sufficient that the event be initiated, there must be positive indication by the event itself that the event is completed. Then and only then can a succeeding event take place. The term synchronous implies that each event takes a known (maximum) length of time and that it is only necessary to meter out a time interval* before initiating a succeeding event. In this case, no positive indication is required that the event has in fact taken place before the next event is initiated. These terms have somewhat loose meanings and are in practice difficult to use in precisely describing the behavior of a computer. It may be stated, however, that TX-2 is dominantly a synchronous machine. It is asynchronous to the extent that interlocks, and not fixed intervals of time, determine when the basic memory cycles and change of sequence cycles can be initiated.

Because the computer is synchronous in nature, events initiated in the outside world (that is, in the In-Out Element or at the pushbutton console) must be synchronized with the computer. Chapter 3 described a synchronizer type circuit for doing this. The important feature of this circuit is that the level outputs initiated by asynchronous pulses are coincident in the sense of Fig. 5-6 with the computer clock pulses.

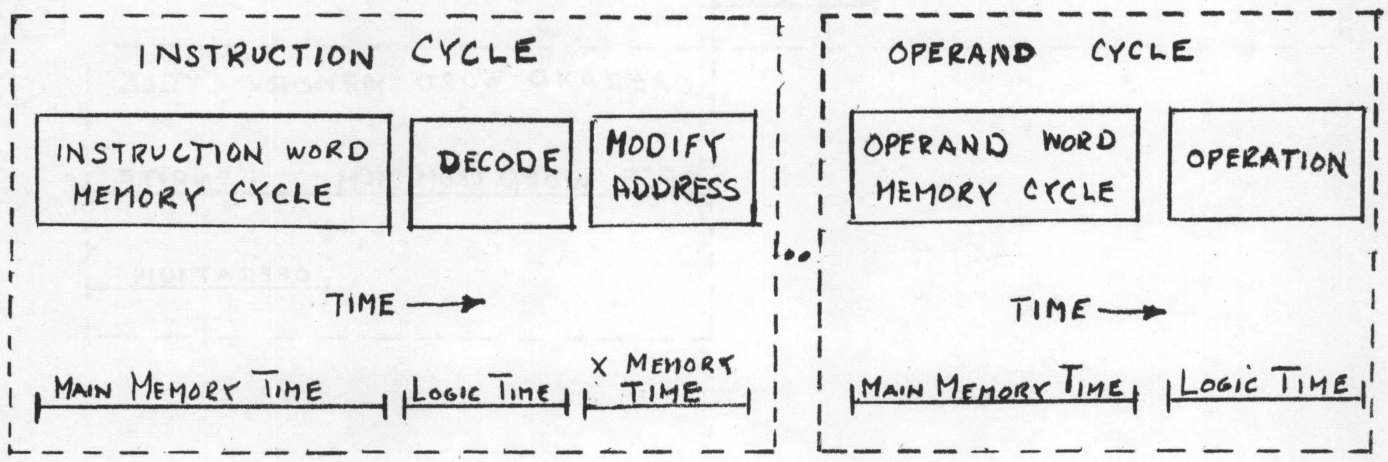
*This time interval is usually some integer number of basic time increments. These increments are 0.2 or 0.4 microseconds, in the case of TX-2.

5-7 SUMMARY

The dynamic operation of the TX-2 computer is determined by the counters and the interlock control. The time levels from the counters find their way into logic nets which are used to gate clock pulses. These gated pulses initiate events such as register transfers. The sum of the events associated with a given counter constitute some basic process in the computer such as a memory cycle. The interlock control determines when the specific counters should start, that is, when it is required and permissible for one of the basic processes to be initiated.



a) BASIC CYCLIC ACTIVITY



b) SUBORDINATE ACTIVITY

FIG. 5-1 TIME ACTIVITY PATTERNS

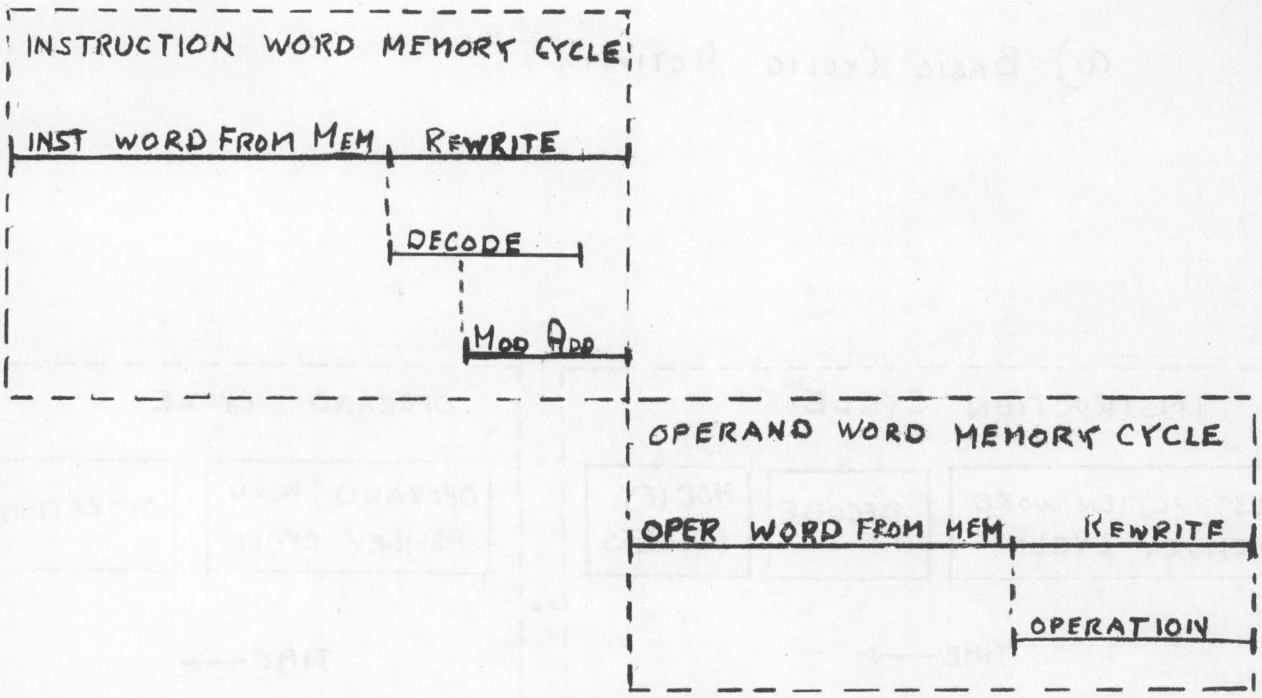


Fig 5-2 COMPRESSED ACTIVITY PATTERNS

(Compare with Fig. 5-1)

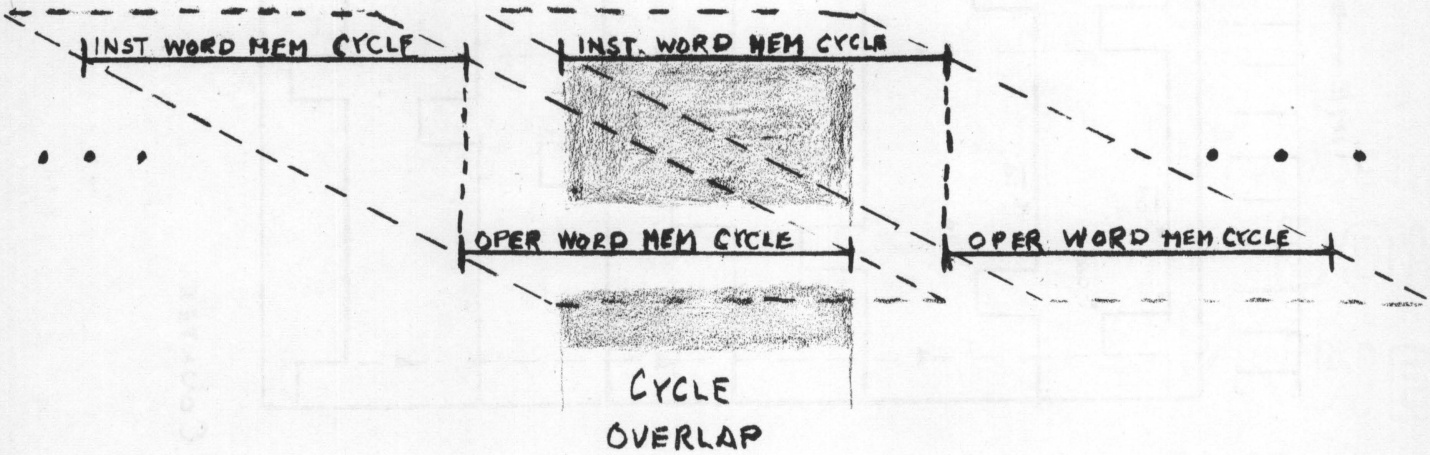


FIG. 5-3 BASIC CYCLE OVERLAP WHEN INSTRUCTION WORD AND OPERAND WORD ARE LOCATED IN <u>DIFFERENT</u> MEMORIES.

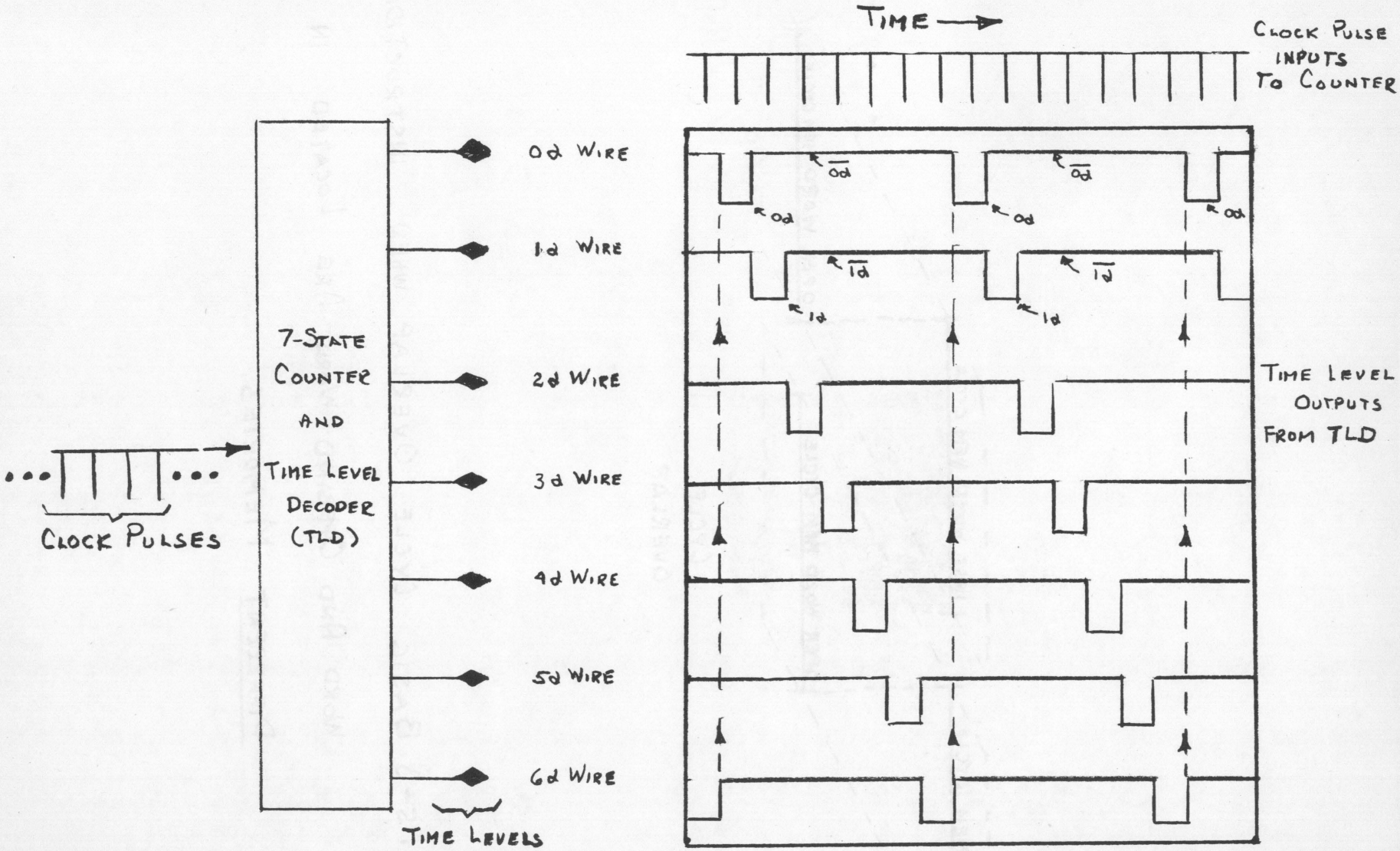
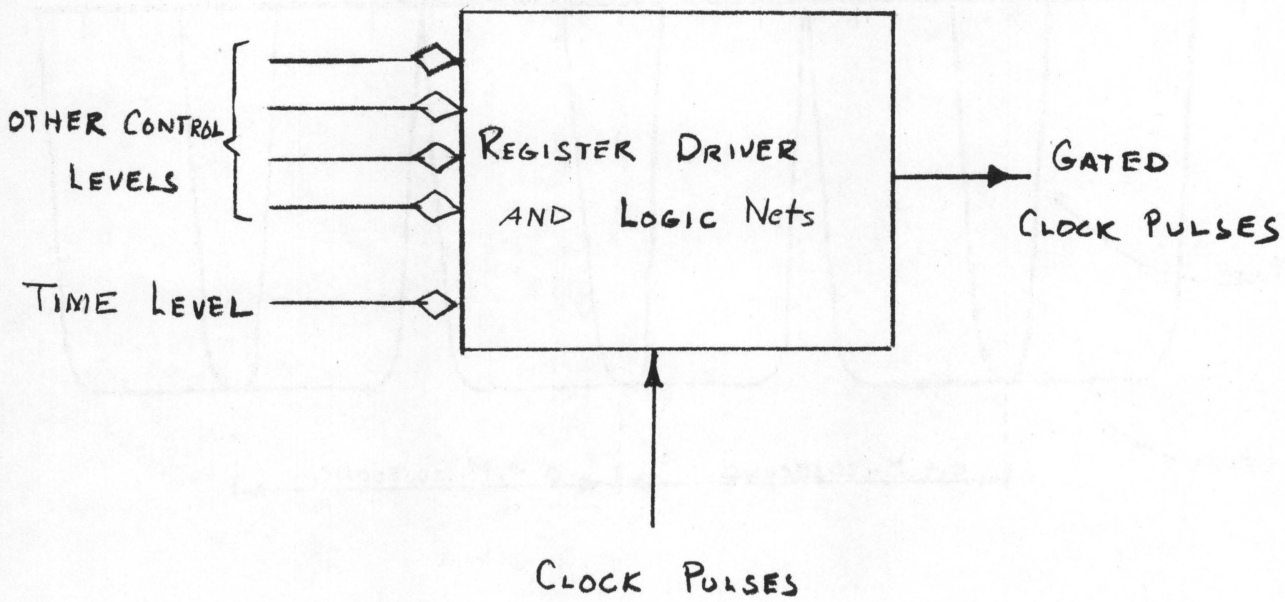


FIG 5-4 FREE-RUNNING COUNTER



GATING OCCURS WHEN TIME LEVELS AND CONTROL LEVELS ARE COINCIDENT WITH CLOCK PULSES (SEE FIG 5-6)

FIG 5-5 REGISTER DRIVER TIME LOGIC

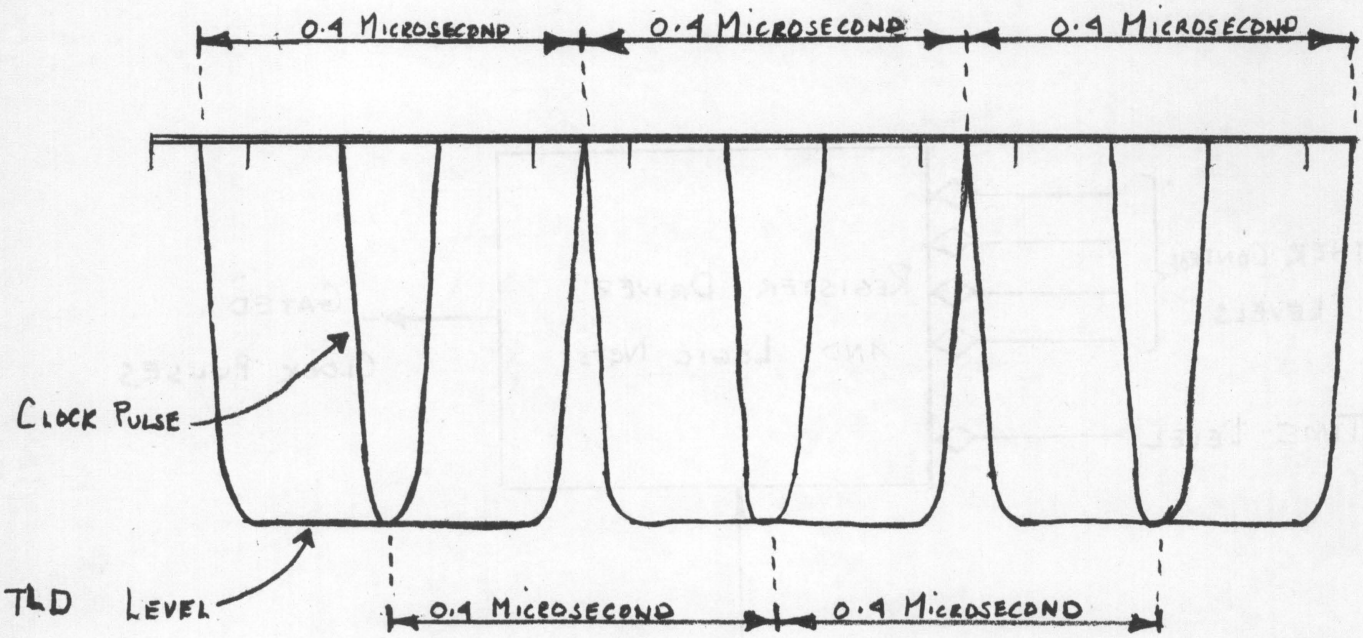


FIG 5-6 COINCIDENCE OF TIME LEVEL DECODER LEVELS AND CLOCK PULSES

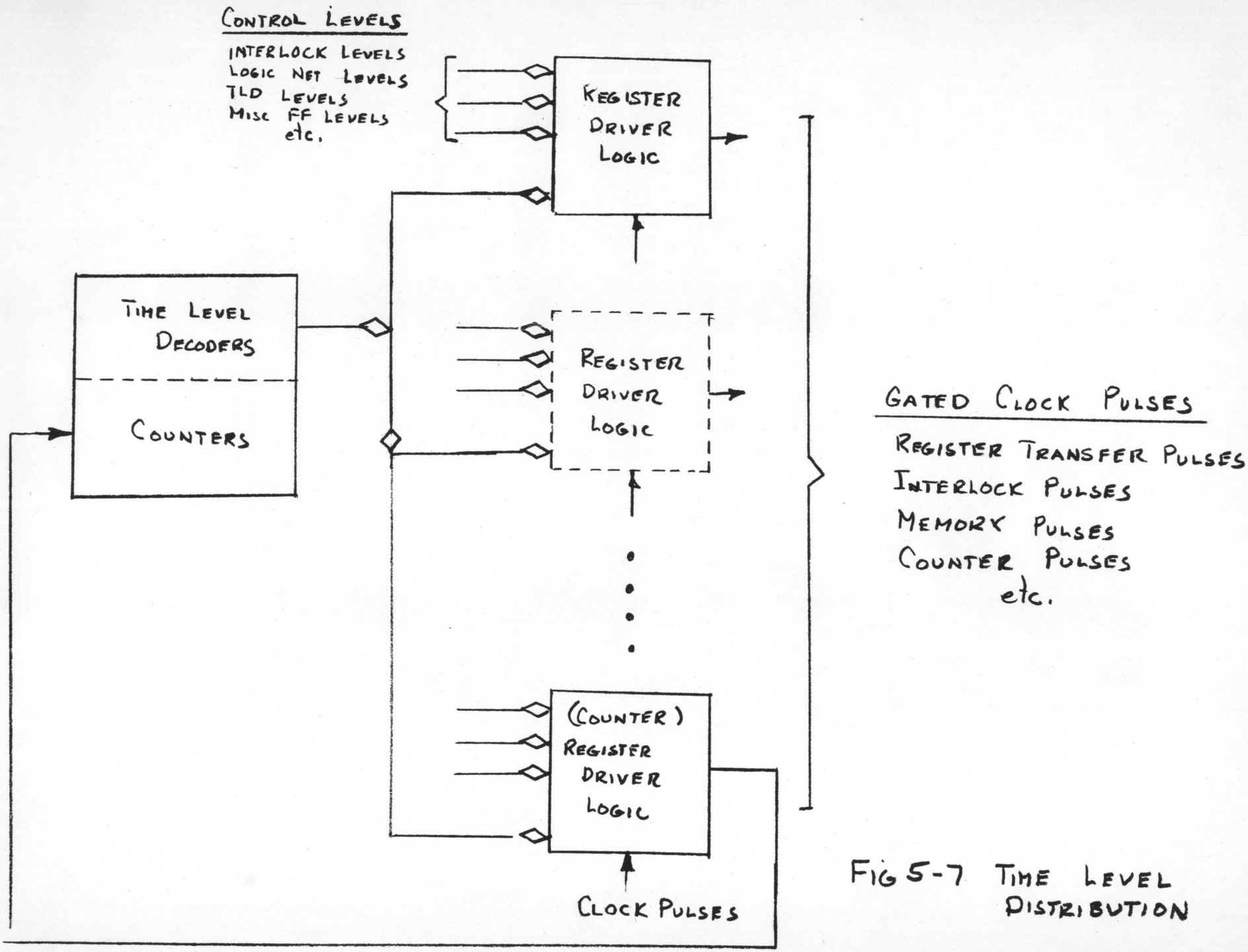


FIG 5-7 TIME LEVEL DISTRIBUTION

CHAPTER 6

FUNCTIONAL ORGANIZATION

OF THE

CONTROL ELEMENT

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 6
FUNCTIONAL ORGANIZATION OF THE CONTROL ELEMENT

6-1 INTRODUCTION

This chapter will describe the principle parts of the Control Element.

One of the basic problems in discussing the Control Element is that of properly establishing the boundary lines between the other Elements, i.e., the Memory Element, Exchange Element, Arithmetic Element, Program Element and In-Out Element, and the Control Element itself. (The problem is analogous to defining the boundary between the organs of a body and the nervous system controlling the organs.) In the computer the demarcation line is somewhat arbitrarily established by locating the information registers (exclusive of the counter registers) in the Elements and the associated register drivers in the Control Element. One can visualize the Control Element as reaching into all the Elements via the register drivers and controlling the dynamic activity taking place in the registers that comprise each Element. (See Fig. 6-1)

Behind the register drivers are a variety of Control Element devices whose chief function is to coordinate the various activities going on in the computer, remember critical events in the computer's previous history, and provide indication and a means of manual control of the computer.

The Control Element consists of console controls and indicators, interlocks, counters and register drivers, as well as the logic interconnecting them. Fig. 6-2 illustrates the general paths of communication within the Control Element.

The control console contains all the manual controls, i.e., the pushbuttons and toggle switches. The indicators on the console provide useful information to the operator on the current state of the computer.

When an operand, instruction or other cycle is initiated an associated counter is used to time control the activity. Time levels from this counter find their way into the register driver logic involved in the activity. Interlocks are used to determine when the various counters can begin their cycles.

6-2 CONSOLE CONTROL

6-2.1 GENERAL DESCRIPTION. The console control consists of a number of pushbuttons and toggle switches as shown in Fig. 6-3. All the pushbuttons are of the momentary type. They and the toggle switches trigger level producing mercury relays. The levels generated are asynchronous with respect to the computer clock pulses and are usually synchronized before being used by the Control Element.

The indicators shown in Fig. 6-4 consist of banks of lamps arranged to display both in octal and binary form the contents of the registers and counters. Individual indicator lamps are also used to display the state of interlocks, alarms and in-out units.

6-2.2 PUSHBUTTON CONTROL

6-2.2.1 START-STOP CONTROL. The two pushbuttons STOP and CALACO* (Clear Alarms and Continue) provide a manual means for stopping and restarting the computer.

The two push-push switches LOW SPEED and LOW SPEED REPEAT control the "speed" of operation of the computer. If both switches are off, the computer will operate in the high speed mode. When the computer is in the high speed mode, the interlock control will allow the basic cycles of the computer to occur as soon as these cycles are needed and can be performed. However, when the computer is not in the high speed mode it is in either the low speed mode or the low speed repeat mode, and four of the basic cycles instruction word, operand word, deferred address word and change of sequence cycles are controlled by the push-push switches PK $STOP_1$, PK $STOP_2$, QK STOP and CSK STOP, respectively. When the computer is in either of these low speed modes, i.e., not in the high speed mode, then the computer will stop whenever it attempts to perform one of the four basic cycles for which the corresponding console stop switch is on. If the LOW SPEED switch is on, the computer can be restarted by pressing the CALACO pushbutton, but, if the LOW SPEED REPEAT switch is on, the computer will be automatically restarted by a pulse generated from an internal oscillator called the LOW SPEED OSCILLATOR. The frequency of this oscillator can be continuously adjusted from 0 to 500 KC by knobs on the console.

Thus, if the computer is in the low speed repeat mode, and if only the PK $STOP_1$ stop switch is on and the LSO is running at 1 cycle per second, then every time the computer attempts to read an instruction word out of the Memory Element it will stop and wait for (about) 1 second before continuing.

Note that the CALACO pushbutton clears all the existing alarms before it starts the computer.

6-2.2.2 STARTOVER CONTROL. Pressing the STARTOVER pushbutton raises the flag of Sequence 0. Since Sequence 0 is the highest priority sequence, pressing the STARTOVER pushbutton will eventually cause a change of sequence to Sequence 0 to occur. Sequence 0 will usually perform some special program at that time.

*CALACO generates three pulses called CLEAR SUPPRESSED ALARMS, CLEAR UNSUPPRESSED ALARMS and START.

6-2.2.3 PRESET CONTROL. Pressing the PRESET pushbutton places certain critical flip-flops in a pre-determined state. The states of these flip-flops guarantee that pressing other pushbuttons will be meaningful.

6-2.2.4 CLEAR SUPPRESSED ALARMS CONTROL. Pressing the CLEAR SUPPRESSED ALARM pushbutton will clear the computer of all the existing suppressed alarms. The dozen or so alarms in the computer have various individually associated controls. One of these controls can suppress the effect of the alarm, although the alarm is still indicated when it occurs. This pushbutton clears the indication of such alarms.

6-2.2.5 CLEAR UNSUPPRESSED ALARMS CONTROL. Pressing the CLEAR UNSUPPRESSED ALARMS pushbutton will clear the computer of all the existing alarms that have not been suppressed. The unsuppressed alarms cause the computer to stop. They must be cleared before the computer can be started again.

6-2.2.6 CLEAR REAL-TIME CLOCK CONTROL. Pressing the CLEAR REAL-TIME CLOCK pushbutton will reset the clock to zero.

6-2.2.7 COMPUTER MASTER START CONTROL. Pressing the CODABO (Count Down and Blast Off) pushbutton initiates a succession of actions which start the computer from any condition. The CODABO pushbutton actuates the following other pushbuttons in the indicated order:

1) STOP
2) CLEAR SUPPRESSED ALARM
3) CLEAR UNSUPPRESSED ALARM
4) PRESET
5) STARTOVER
6) START

6-2.3 ALARM INDICATIONS AND CONTROL

6-2.3.1 GENERAL DESCRIPTION. Alarms are generated by computer or programming errors. These alarms may be classified into five categories:

1) Memory selection alarms
2) Memory parity alarms
3) In-out alarms
4) Operation code alarms
5) Miscellaneous alarms

Associated with each of most of the alarms is an indicator light, and a toggle switch used to suppress the effect of the alarm condition. (Only the miscellaneous alarms are not suppressible.) A two-tone chime emits one tone whenever a suppressed alarm occurs and the other tone whenever an unsuppressed

alarm occurs. The occurrence of an unsuppressed alarm will cause the computer to stop, while a suppressed alarm will not interrupt the computer operation.

Alarms are cleared by depressing the CLEAR SUPPRESSED ALARMS or CLEAR UNSUPPRESSED ALARMS pushbuttons.

6-2.3.2 MEMORY SELECTION ALARMS (PSAL AND QSAL). If either the P or Q register contains an illegal address and is used to select a Memory Element register, the associated memory selection alarm will be generated. The specified address is illegal if either the address is in the range 220 000 to 377 577 (octal) or if the S, T, or U memories are addressed and these memories are turned off. These memories may be turned off by the SMOFF, TMOFF, or UMOFF console switches respectively.

6-2.3.3 OPERATION CODE SELECTION ALARM (OCSAL). This alarm occurs whenever the computer selects one of the 14 undefined operation codes. The occurrence may be caused either by a computer error or a programming error.

6-2.3.4 MEMORY PARITY ALARMS (MPAL, NPAL, XPAL AND FPAL). Whenever a bad memory readout into a buffer is made, the parity alarm associated with this buffer will occur. For example, if a Memory Element register is read into the M register with a bad (even) parity, then an MPAL will occur.

6-2.3.5 IN-OUT ALARMS (IOSAL, MISAL). The IOSAL (In-Out Selection Alarm) alarm will be generated whenever an IOS 3X XXX or IOS 6X XXX type instruction* selects an in-out unit which is in the maintenance mode. The MISAL (Misindication Alarm) alarm occurs whenever an in-out unit loses a line of data because the central computer fails to perform the proper in-out instruction. Note that MISAL alarms are associated only with "free running" in-out units.

6-2.3.6 MISCELLANEOUS ALARMS (TSAL, SYAL AND MOUSETRAP). A TSAL (T Memory Selection Alarm) alarm is generated whenever the T Memory selection circuits fail to perform properly.

A SYAL (Synch System Alarm) alarm is generated whenever the synch system stops the computer.

The "Mousetrap" alarm is used to stop the computer during special, maintenance operations.

*See Chapter 7.

6-2.4 AUTOMATIC START FROM ALARMS

6-2.4.1 AUTO START AFTER UNSUPPRESSED ALARM TOGGLE SWITCH. This switch automatically starts the computer after it has stopped on an unsuppressed alarm. The action of the switch is equivalent to pressing the CLEAR UNSUPPRESSED ALARM pushbutton and then pressing the START pushbutton*.

6-2.4.2 PRESET AND STARTOVER AFTER SUPPRESSED ALARM TOGGLE SWITCH. This switch has effect only when the computer has stopped on a suppressed alarm. The action of the switch is equivalent to pressing the PRESET pushbutton and then pressing the START pushbutton*.

6-2.5 TOGGLE SWITCH PROGRAM SWITCHES (TSP). The Toggle Switch Program counter consists of 18 toggle switches. The binary number set up in these switches is used to specify the value of the program counter read out, whenever a change to Sequence 0 occurs. This number is then loaded into the P register.

6-2.6 REMOTE PUSHBUTTON AND TOGGLE SWITCH PROGRAM. For convenience, a portable control console is provided with several of the control pushbuttons and switches located on it.

6-2.7 NO OVERLAP TOGGLE SWITCH (NO). This switch can be used to inhibit the simultaneous execution of two Memory Element word read out cycles.

6-2.8 SYNCH TRAP TOGGLE SWITCH. This switch determines whether the trapping sequence control or the external Synchronization System will provide the signals which raise the flag of the trapping sequence.

6-2.9 STOP AE ON SELECTED SYNCHRONIZATION TOGGLE SWITCH. When this switch is turned on, the computer will stop whenever the particular conditions specified by the Synchronization System exist in the Arithmetic Element.

6-3 INTERLOCK CONTROL

6-3.1 GENERAL DESCRIPTION. The interlocks themselves are individual flip-flops which are set to ONE and cleared to ZERO when certain specific interlock logic associated with them is satisfied. The important characteristic of these interlocks is that they remember certain conditions have occurred after the conditions themselves have disappeared. By means of the interlock control the past history of the computer is used to control the future activity of the computer.

6-3.2 ARITHMETIC ELEMENT PREDICT INTERLOCK (AEP). This interlock is used to predict when the Arithmetic Element will be finished with its current activity and be again available for another use. E.g., if a division is being performed, the AEP interlock

*Note that the action of the START button has been incorporated into the action of the CALACO button.

predicts when this division will be completed and the Arithmetic Element free to perform another operation, such as an addition.

6-3.3 E REGISTER BUSY INTERLOCK (EB). EB^1 indicates that the E register is currently in use and is not yet available for some new use.

6-3.4 INSTRUCTION $INTERLOCK_1$ (PI_1). PI_1^0 is one of the conditions that is required before an instruction word memory cycle or a change of sequence cycle can begin. PI_1^1 is one of the conditions that is required before an operand memory word cycle can begin.

6-3.5 INSTRUCTION $INTERLOCK_2$ (PI_2). PI_2^1 indicates that a deferred address cycle is required.

6-3.6 INSTRUCTION $INTERLOCK_3$ (PI_3). PI_3^1 indicates that a change of sequence cycle is to occur.

6-3.7 INSTRUCTION $INTERLOCK_4$ (PI_4). This interlock remembers the value of the hold bit of the last instruction executed.

6-3.8 INSTRUCTION $INTERLOCK_5$ (PI_5). PI_5^1 indicates that an intermediate deferred address cycle is required.

6-3.9 Q REGISTER BUSY INTERLOCK (QB). QB^1 indicates that the Q register is currently being used in an operand cycle and is not yet ready for some new purpose.

6-3.10 X REGISTER BUSY INTERLOCK (XB). XB^1 indicates that the X (index) register is being used and is not yet ready for some new purpose.

6-3.11 X MEMORY WRITE INTERLOCK (XW). XW^1 indicates that both the X (index) register and the X (index) memory are being used and are not yet ready for some new purpose.

6-3.12 F MEMORY INTERLOCK (FI). FI^1 indicates, in certain circumstances, whether the F (configuration) memory is to be used.

6-4 INTERLOCK LEVEL CONTROL

6-4.1 GENERAL DESCRIPTION. Interlock control levels are used to start up counters, or place them in waiting states. These interlock control levels are usually not effective until the counters which use them are in some specific state.

The interlock control levels are generated by certain conditions in the computer including, most importantly, the state of the interlock flip-flops.

6-4.2 INSTRUCTION CYCLE INTERLOCK START$_1$ LEVEL (PI^{START_1}). This level is used to start the PK counter in an instruction cycle. PK must be in its 00 resting state before this level can be used.

6-4.3 INSTRUCTION CYCLE INTERLOCK START$_2$ LEVEL (PI^{START_2}). This level is used to start the PK counter in a deferred address cycle. PK must be in its 00 resting state before this level can be used.

6-4.4 OPERAND CYCLE INTERLOCK START LEVEL (QI^{START}). This level is used to start the QK counter in an operand cycle. QK must be in its 00 resting state before this level can be used.

6-4.5 CHANGE OF SEQUENCE CYCLE INTERLOCK START LEVEL (CSI^{START}). This level is used to start the CSK counter in a change of sequence cycle. CSK must be in its 00 resting state before this level can be used.

6-4.6 INSTRUCTION CYCLE CHANGE OF SEQUENCE INTERLOCK LEVEL ($PI^{CH\ SEQ}$). This level indicates that a CSK counter cycle will occur after the current PK counter cycle. This level is generated while the PK counter is running and interpreted by the logic controlling PI_3. PI_3 then remembers whether a change of sequence has been requested.

6-4.7 INSTRUCTION CYCLE INTERLOCK WAIT LEVEL (PI^{WAIT}). This level occurs if PK is required to stop and wait in an intermediate state of the counter (state 23) until some other interlock condition is satisfied. Various interlock conditions can cause PI^{WAIT} to be generated.

6-4.8 INSTRUCTION CYCLE INTERLOCK LEAVE SEQUENCE LEVEL ($PI^{LV\ SEQ}$). This level occurs if PK is required to leave the current instruction cycle uncompleted in order that a change of sequence cycle can occur. PK is in the 22nd or 23rd state when the $PI^{LV\ SEQ}$ level becomes effective. Various interlock conditions can cause $PI^{LV\ SEQ}$ to be generated.

6-4.9 F MEMORY COUNTER START LEVEL ($\xrightarrow{START}$ FK). This level is used to start the FK counter in an F (configuration) memory cycle. FK must be in its 00 resting state before this level can be used.

6-4.10 X MEMORY COUNTER START LEVEL ($\xrightarrow{START}$ XWK). This level is used to start the XWK counter in an X (index) memory cycle. XWK must be in its 00 resting state before this level can be used.

6-4.11 ARITHMETIC COUNTER START LEVEL ($\xrightarrow{START}$ AK). This level is used to start the AK counter. AK must be in its 00 resting state before this level can be used.

6-4.12 ARITHMETIC STEP COUNTER START LEVEL ($\xrightarrow{START}$ ASK). This level is used to start the ASK counter. ASK must be in its 00 resting state before this level can be used.

6-4.13 IN-OUT DELAY SYNCHRONIZATION COUNTER START LEVEL ($\xrightarrow{\text{START}}$ IODK). This level is used to start the IODK counter part of the CSK counter. (Part of this counter is used during an in-out delay synchronization cycle, while the other part is used during a change of sequence cycle.) The counter must be in its 8th state before this level can be interpreted.

6-4.14 ALARM DELAY COUNTER START LEVEL ($\xrightarrow{\text{START}}$ ADK). This level is used to start the ADK counter in an alarm delay cycle. ADK must be in its 00 resting state before this level can be used.

6-5 COUNTER REGISTER DRIVER CONTROL

6-5.1 GENERAL DESCRIPTION. The counter register drivers gate clock pulses. These gated clock pulses are then used to change the state of the associated counter.

Usually a counter will proceed from one state to the next in step with a stream of gated clock pulses. However, the register driver logic may alter this pattern and give use to one of three possibilities. The possibilities are that the counter may "rest", "wait", and "skip" when it is in particular states. These possibilities are determined by the counter logic and the general condition of the computer at the moment.

Usually a counter will rest in its 00 state until it has a reason to start counting or because it is inhibited from starting by other conditions. The interlock levels in the counter register driver logic determine when the counter can start. As mentioned above, frequently a counter will not count through all its states successively, but, instead will skip a number of states. These skips are controlled in the register driver logic by levels which define the kind of counter cycle being performed. In other circumstances a counter will stop and wait in some intermediate state. This usually occurs because some interlock level prevents clock pulses from getting through the register driver to change the state of the counter.

6-6 COUNTERS

6-6.1 GENERAL DESCRIPTION. The physical structure of time level counters was described in Chapter III. The time levels generated by a counter are used as factors in register driver logic. There they serve to identify the particular clock pulses which, when ANDed with the remainder of the register driver logic, affect the contents of the associated register.

There are eight counters in the Control Element. These counters time control the execution of memory cycles, change of sequence, arithmetic operations, and other miscellaneous activities.

The logic controlling the counters reflects conditions existing throughout the computer including the state of the counters themselves.

6-6.2 CHANGE OF SEQUENCE COUNTER (CSK). This is a three stage counter with one additional flip-flop used as an interlock. Although 16 time level states can be decoded, the counter actually generates either one or the other of two sets of 8 time levels.

The first set of 8 states from 00 to 07 comprise a part of the counter which time controls the change of sequence cycle. This part of the counter is called CSK. The second set of 8 states from 08 to 15 comprise a part of the counter which time controls the in-out delay synchronization cycle. This part of the counter is called IODK*. As will be seen later, 1.6 microseconds is sufficient to perform an in-out delay synchronization cycle. For this reason only states 08 to 11 of IODK are used.

6-6.3 INSTRUCTION COUNTER (PK). This is a five stage counter consisting of two sets of five flip-flops. One set generates alpha time levels and the other set generates beta time levels. The basic function of this counter is to time control the execution of an instruction or deferred address cycle.

6-6.4 OPERAND COUNTER (QK). This is a five stage, two phase counter similar in structure to PK. The basic function of this counter is to time control the execution of an operand cycle.

6-6.5 ARITHMETIC OPERATION COUNTER (AK). This counter differs from the other counters in that it is actually a shift register. Like the other counters it has both an alpha and beta phase. The basic function of this counter is to time control the execution of arithmetic operations. Thus AK time controls the basic addition, subtraction, scale and cycle operations. Certain other arithmetic operations require the ASK counter to operate concurrently with the AK counter in the time control process.

6-6.6 ARITHMETIC ELEMENT STEP COUNTER (ASK). This is a seven stage counter. The basic function of this counter is to count the iterations of certain sub-operations that occur during the AK cycle, when the divide, multiply, tally, and, in a special sense, normalize operations are being executed.

6-6.7 F MEMORY COUNTER (FK). This is a four stage, two phase counter. Only 9 of the 16 possible alpha states are decoded, the beta states are only partially decoded. The basic function of this counter is to time control the execution of the F (configuration) memory cycle.

6-6.8 X MEMORY COUNTER (XWK). This is a three stage counter. The basic function of this counter is to time control the execution of the write part of an X (index) memory read-write cycle.

*As will be seen, this part of the counter is just as frequently called DSK, the delay synchronization counter.

6-6.9 ALARM DELAY COUNTER (ADK). This is a two stage counter where the counting mode is determined by variable delay units. The basic function of this counter is to control time delays, after an alarm occurs, so as to stop the computer and, in some cases, start it again in a controlled manner.

6-7 REGISTER DRIVERS FOR REGISTERS

6-7.1 GENERAL DESCRIPTION. Information transfers in the central computer are effected by pulses from register driver units. These register drivers are bit by either alpha or beta clock pulses every 0.4 microsecond. These pulses are gated in the register drivers in the usual way by logic levels, including time levels.

6-8 REGISTERS

6-8.1 GENERAL DESCRIPTION. The registers in the central computer hold information that is either in process of being transferred to or from the various memories and in-out units, or is the intermediate result of a computation or other operation. The information contained in the register frequently makes a significant contribution to the decisions made in the Control Element. The registers themselves, however, are classified as belonging to the various other elements, i.e., the Arithmetic Element, Program Element, Exchange Element, In-Out Element and Memory Element.

Information is placed in registers by register driver pulses. The information remains there until new information is placed there by new register driver pulses.

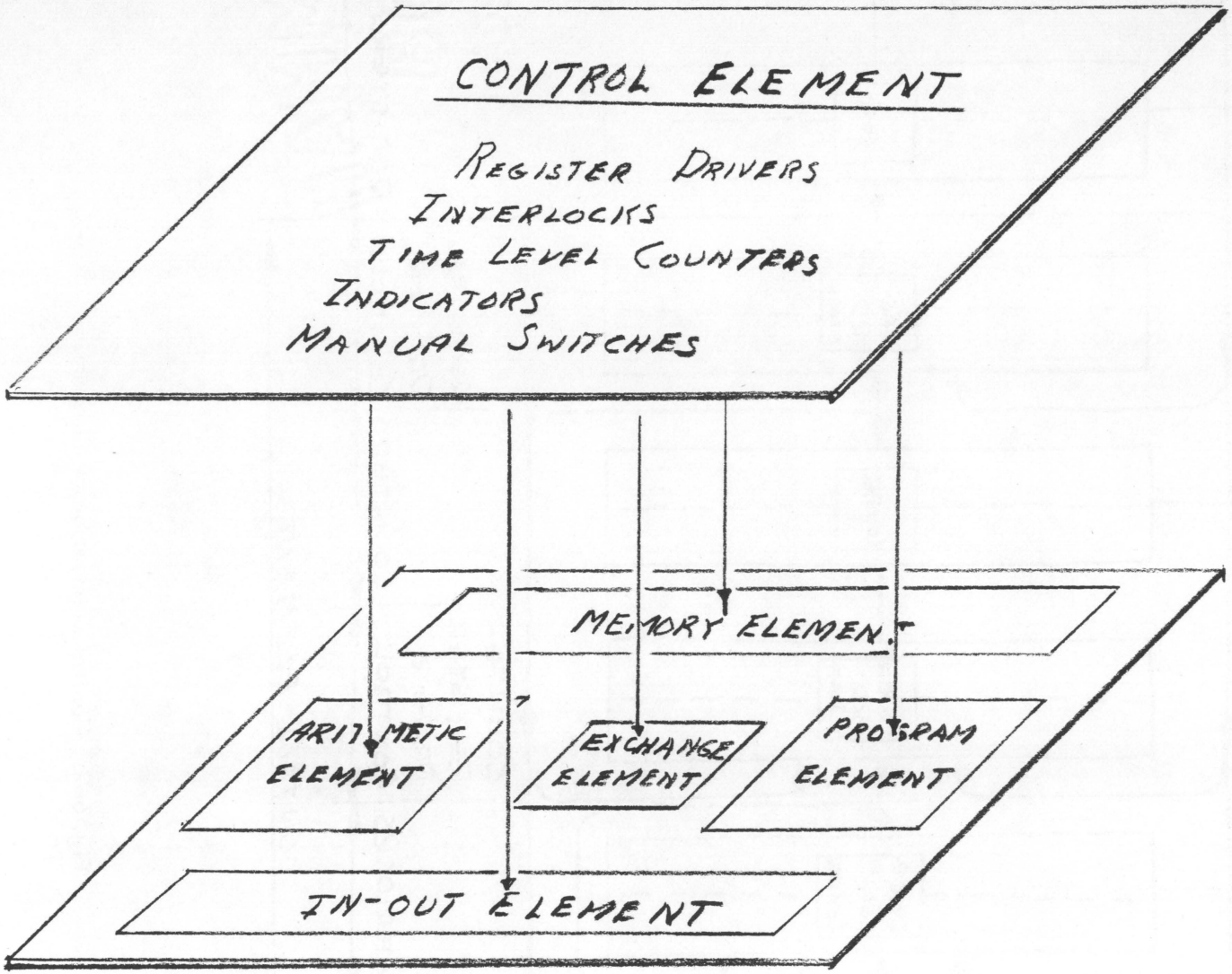


Fig. 6-1. Relationship of control element to other elements in computer.

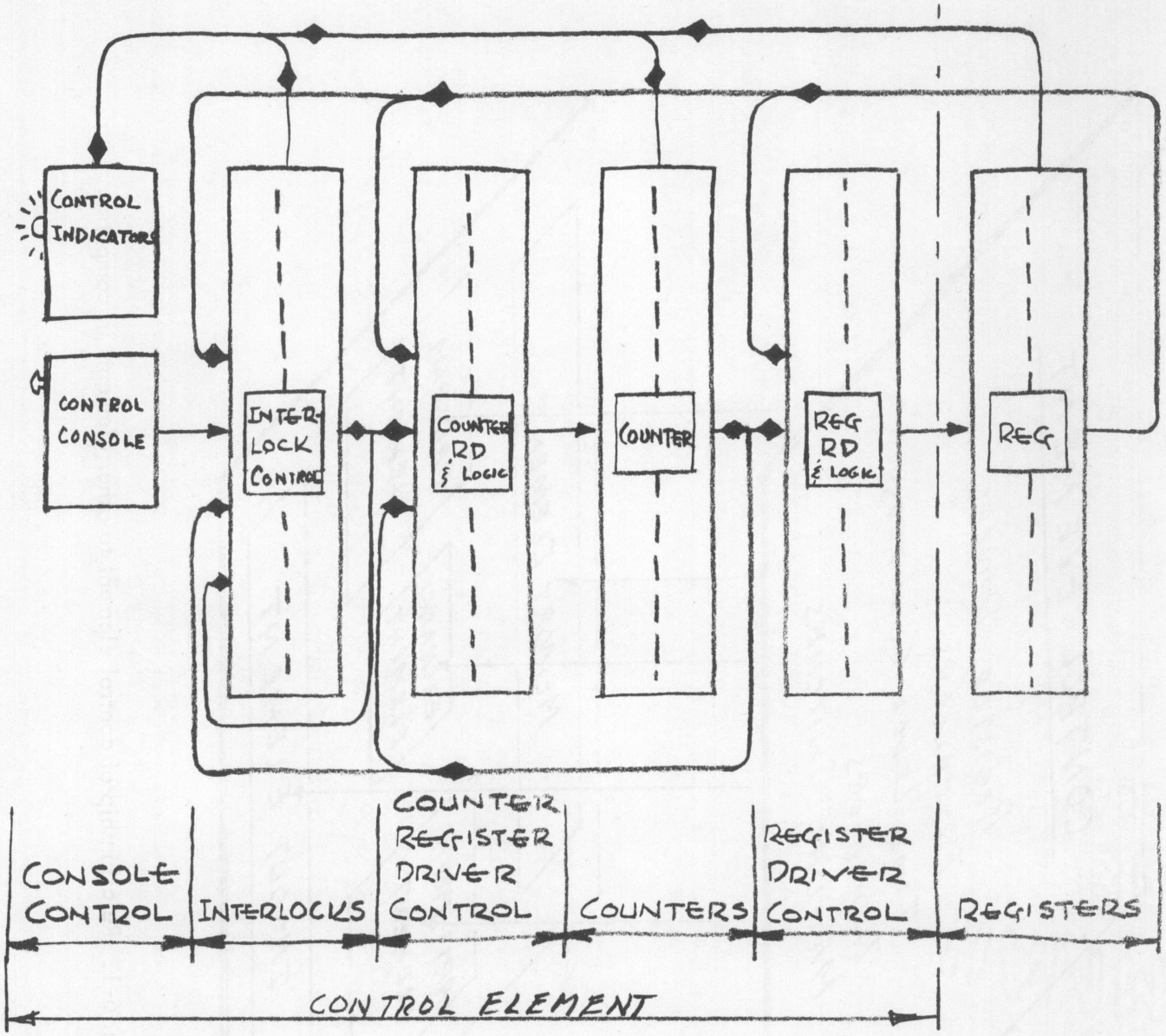


Fig. 6-2. Control element block diagram.

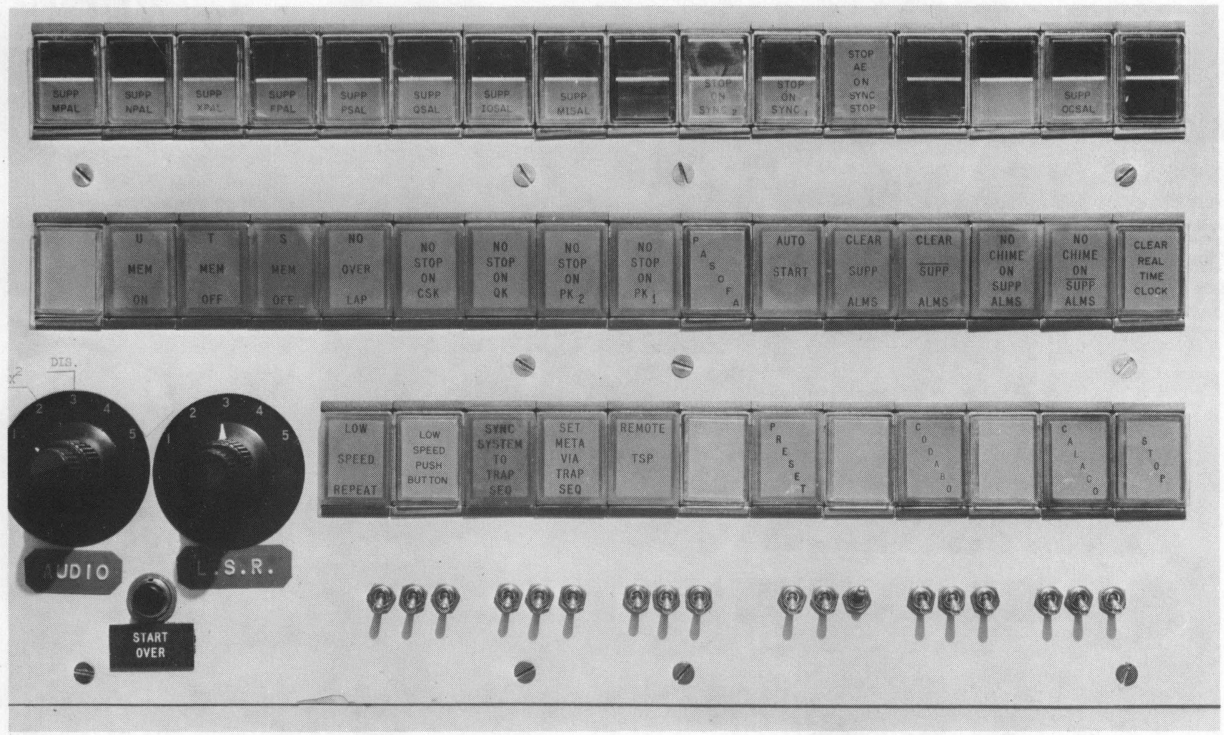


Fig. 6-3. Console control pushbuttons and toggle switches.

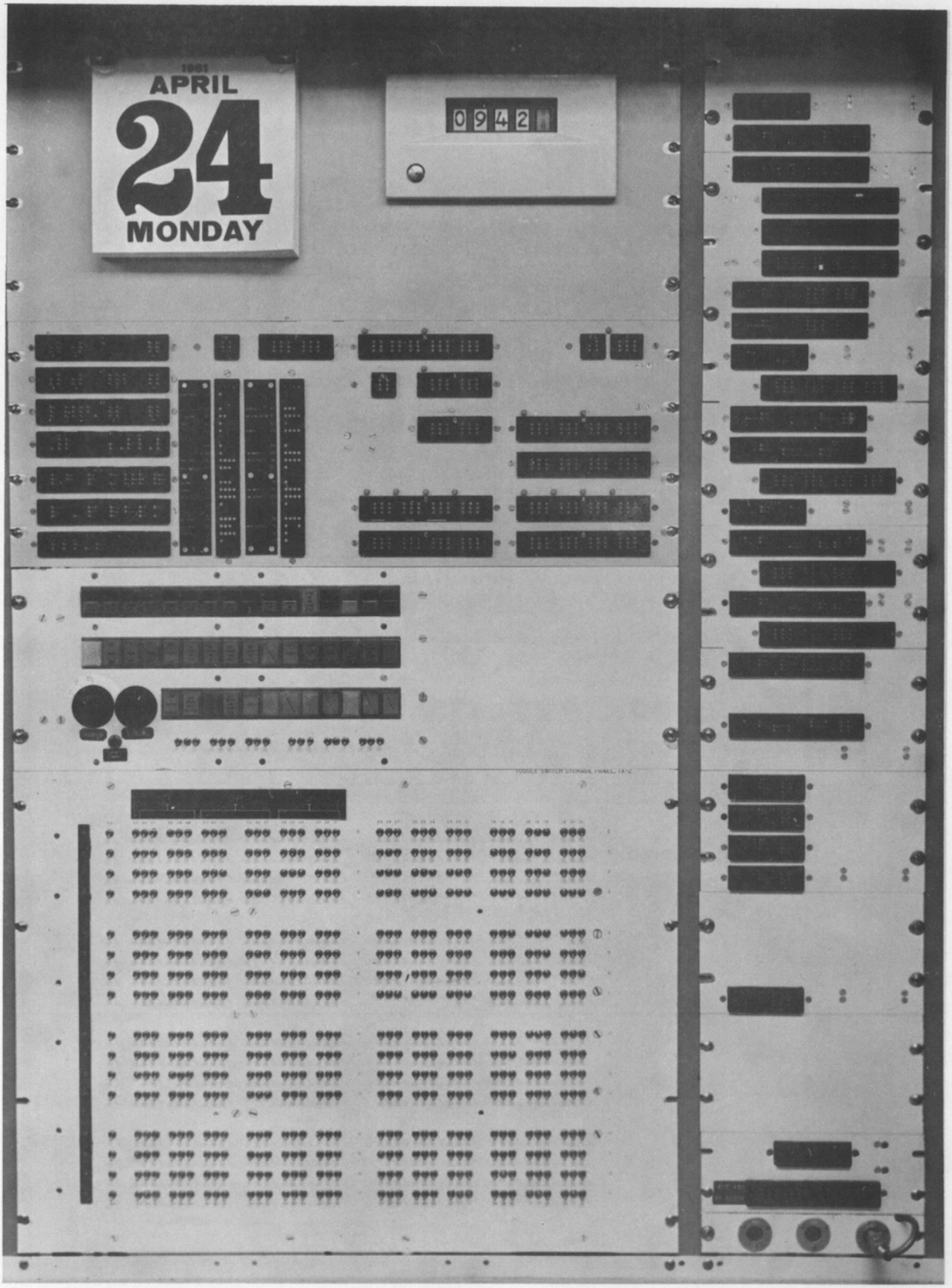


Fig. 6-4.

CHAPTER 7

OPERATION CODES

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 7
OPERATION CODES

7-1 INTRODUCTION

As mentioned earlier, the P register specifies the address of instruction words in the Memory Element. An instruction word is strobed into the N register where it is interpreted. Bits $N_{4.3 - 3.7}$ determine the basic operation to be performed during the execution of the instruction, e.g., multiplication, addition, etc.

In Chapter 2, the over-all pattern of activity occurring in the computer during the execution of a typical instruction was described. This chapter emphasizes the variety of operations that can be specified by the operation codes and discusses the basic features of these operations. The details of the logic of each operation is covered in Chapter 16.

7-2 DEFINED OPERATION CODES

Of the 64 possible operation codes, only 50 are currently defined. These defined codes are listed in Fig. 7-1. (The attempted execution of an instruction containing an undefined operation code causes an OCSAL alarm.) New codes can be defined in the future as new programming needs arise.

It should be noted that AOP and IOS are variations of the single operation code OPR. The selection of OPR^{AE} or OPR^{IOS} is determined by the value of bits 2.8 and 2.7 in the address section of the instruction.

7-3 INSTRUCTION CHARACTERISTICS

During the execution of indexable and configurable type instructions, the CF, OP and J bits are interpreted in the normal manner. However, during the execution of nonindexable or nonconfigurable type instructions, these bits are either interpreted in a different way or not at all. Whether the instruction is indexable or nonindexable, or configurable or nonconfigurable usually depends only on the operation code specified in the instruction.

Note that, as described in Chapter 2, the base address of indexable instructions is modified by adding the content of the specified X Memory register X_J. The result of this addition is called the <u>indexed base address</u>. However, any instruction can call for deferred addressing. If deferred addressing is called for, then the final base address, obtained as a result of the deferred addressing, is modified by the content of the index register specified by the original instruction word. In this case it is more precise to speak of the <u>indexed final base address</u>.

Many instructions can perform the same operation upon several active subwords simultaneously. In the following discussions usually the effect of the operation upon only one active subword is described.

During the execution of most instructions the operand word written in the selected Memory Element register at the end of the instruction is also copied into the E register. An instruction usually involves both an operand register in the Memory Element and an operand register in the central computer, i.e., the Arithmetic Element, Program Element, Exchange Element, or In-Out Element. When either operand register is the E register in the Exchange Element, then the rule about placing the final content of the selected Memory Element register in the E register does not apply.

7-4 ARITHMETIC ELEMENT OPERATION CODES

With the exception of AOP, all Arithmetic Element instructions are both indexable and configurable.

7-4.1 LOAD OPERATIONS. During the execution of a LDA instruction, the content of a selected Memory Element register is configured and "loaded" into the A register of the Arithmetic Element. The LDB, LDC and LDD instructions are similar except that registers B, C, and D, respectively, are loaded with the configured operand.

The LD- type instructions rewrite the original (unconfigured) operand back into the selected Memory Element register and also leave this original operand copied in the E register.

7-4.2 STORE OPERATIONS. During the execution of a STA instruction, the content of the A register is inversely configured and "stored" in the selected Memory Element register. The STB, STC and STD instructions are similar except that the inversely configured contents of registers B, C and D, respectively, are stored in the selected memory register.

The final content of the selected memory register is also placed in the E register.

7-4.3 EXCHANGE A OPERATION. The EXA instruction is a combination load and store type instruction. It exchanges the content of the selected Memory Element register with the content of the A register.

The content of the A register is inversely configured and stored in the selected memory register (as in a STA instruction) and, at the same time, the content of the selected memory register is configured and loaded into the A register (as in a LDA instruction).

7-4.4 ARITHMETIC OPERATIONS

7-4.4.1 ADD OPERATION. During an ADD instruction, the content of the selected Memory Element register is configured and added to the content of the A register. The signed sum is left in the A register. If an overflow occurs

in any of the active subwords of A as a result of this addition, the associated overflow flip-flop Z_i is left set to ONE. (Z_i is always cleared at the beginning of the ADD instruction in all the active subwords.)

7-4.4.2 SUBTRACT OPERATION. During a SUB instruction, the configured content of the selected Memory Element register is subtracted from the content of the A register. The signed difference is left in the A register. Except for the fact that the configured operand loaded into D from memory is initially complemented, the addition and subtraction process are identical.

7-4.4.3 MULTIPLY OPERATION. During a MUL instruction, the configured content of the selected Memory Element register is multiplied by the content of the A register. The signed product is left in the AB register. Note that the right most bit of the active subword in the B register is not part of the product, and is in fact a copy of the sign bit in the A register.

7-4.4.4 DIVIDE OPERATION. During a DIV instruction, the content of the AB register is divided by the configured content of the selected Memory Element register. The dividend in the AB register before the DIV is performed has the same form as a product after an MUL. The signed quotient is left in the A register and the signed remainder is left in the B register. If an overflow occurs (as it will unless the divisor is smaller than the part of the dividend in A), the associated overflow flip-flop Z_i is left set to ONE. (Z_i is always cleared at the beginning of the division process.)

7-4.4.5 SCALE OPERATIONS. In the SCA instruction, the content of the A register is shifted the number of places to the left or right specified by the configured content of the selected Memory Element register.

The configured operand is first loaded into the D register from memory. Each active subword in the A register is then arithmetically shifted "n" places (where "n" is the number now located in the sign quarter of each corresponding subword in the D register). The subword in A is shifted to the left if "n" is positive and to the right if "n" is negative.

The contents of the overflow flip-flops may be shifted by the scaling process, but the true sign of the data in the A register is not altered by the scaling process.

The SCB instruction is the same as the SCA instruction, except that the shifting process occurs in the B register instead of in the A register. Note that there are no overflow bits associated with the B register.

The SAB instruction is the same as the other scale instructions, except that the shifting takes place in the AB register.

7-4.4.6 CYCLE OPERATIONS. The CYA, CYB and CAB instructions are similar to the SCA, SCB and SAB instructions respectively, except that the CY- type instructions ignore the state of the overflow flip-flops and give no special significance to the sign bits in the data registers. The CY- type instructions simply rotate the data in the registers as if it had no arithmetic significance.

7-4.4.7 NORMALIZE OPERATION. In the NOA instruction, the content of the A register is shifted to the left or right until the value of the data in the A register (and the overflow flip-flops) has a value between 1/2 and 1.

The configured content of the selected Memory Element register is first loaded into the D register from memory. Each active subword in the A register is then arithmetically shifted until its content is "normalized". The number of shifts required to do this is added to (if shifting to the right occurred) or subtracted from (if shifting to the left occurred) the current number in the sign quarter of each corresponding subword in the D register.

The data will be shifted to the right only if a previous instruction left the overflow flip-flop associated with the data set to ONE. Note that a shift of only one place to the right is required to normalize the data in this case. In all other cases, the normalizing process shifts the data to the left.

7-4.4.8 TALLY OPERATION. During a TLY instruction, the number of ONES in the configured content of the selected Memory Element register are counted. The total is then added to the content of the D register.

The configured operand is first loaded into the A register from memory. The number of ONES in each active subword of the A register is added to the current number in the sign quarter of each corresponding subword in the D register.

7-4.5 LOGICAL OPERATIONS

7-4.5.1 INTERSECT OPERATION. During an ITA instruction, the configured content of the selected Memory Element register is logically ANDed ("intersected") with the content of the A register. The "intersection" is left in the A register.

7-4.5.2 UNITE OPERATION. During a UNA instruction, the configured content of the selected Memory Element register is inclusively ORed ("united") with the content of the A register. The "union" is left in the A register.

7-4.5.3 DISTINGUISH OPERATION. During a DSA instruction, the configured content of the selected Memory Element register is exclusively ORed ("distinguished") with the content of the A register. The result of the "distinguishing" process is left in the A register.

7-4.5.4 INSERT OPERATION. The INS instruction is similar to a STA instruction in which certain bits in the A register are "masked". The INS instruction copies the configured contents of those bits in the A register that have corresponding bits in the B register in the ONE state into the selected Memory Element register. The remaining bits in the memory register are left undisturbed.

7-4.6 ARITHMETIC ELEMENT COMMAND. See Miscellaneous Operation Codes (OPR^{AE}).

7-5 X (INDEX) MEMORY OPERATION CODES.

The X Memory type instructions are similar to the Arithmetic Element load, store and exchange type instructions. Some of the X Memory type instructions involve arithmetic modification of the content of an X Memory register by the content of a Memory Element register, or the arithmetic modification of a Memory Element register by the content of an X Memory register. None of these instructions are indexable, but all are configurable.

7-5.1 RESET X OPERATION. During an RSX instruction, the content of the X Memory register X_J, specified by the J bits of the instruction word, is "reset" by the right 18 bits of the configured content of the selected Memory Element register. By means of the RSX instruction data in the Memory Element can be loaded into the X Memory.

7-5.2 DEPOSIT X OPERATION. During a DPX instruction, the inversely configured content of the X Memory register X_J, specified by the J bits of the instruction word, is "deposited" in the selected Memory Element register. By means of the DPX instruction, data in the X Memory can be stored in the Memory Element.

In this instruction the content of X_J is considered as a 36 bit number, where the left 18 bits are the same as the sign bit of the right 18 bits.

7-5.3 EXCHANGE X OPERATION. The EXX instruction is a combination reset and deposit instruction. It "exchanges" the configured content of the selected Memory Element register with the content of the X Memory register X_J specified by the J bits of the instruction word. The content of the X Memory register is inversely configured and stored in the selected Memory register (as in a DPX instruction), and at the same time the content of the selected Memory Element register is configured and loaded into the specified X Memory register (as in a RSX instruction).

7-5.4 AUGMENT X OPERATION. During an AUX instruction, the content of the X Memory register X_J specified by the J bits of the instruction word is "augmented" by the configured content of the selected Memory Element register.

Specifically, the ONE's complement sum of the right 18 bits of the configured content of the selected Memory Element register (with zeroes in the inactive subwords) and the content of the X Memory register X_J is placed in X_J.

7-5.5 ADD X OPERATION. During an ADX instruction, the content of the X Memory register X_J specified by the J bits of the instruction word is added to the content of the selected Memory Element register.

Specifically, the ONE's complement sum of the right 18 bits of the configured content of the selected Memory Element register (with zeroes in the inactive subwords) and the content of the X Memory register X_J is placed in the selected Memory Element register.

7-6 F (CONFIGURATION) MEMORY OPERATION CODES

The F Memory operations perform only load and store operations. These operations are not configurable, but are indexable.

7-6.1 SPECIFY FORM OPERATION. During a SPF instruction, the content of the rightmost (first) quarter of the selected Memory Element register is loaded into the F Memory register specified by the CF bits of the instruction word.

7-6.2 SPECIFY GROUP OF FORMS OPERATION. During a SPG instruction, the 36 bit content of the selected Memory Element register is loaded into four successive F Memory registers. In the execution of the instruction the content of the rightmost (first) quarter of the memory register is loaded into the F Memory register specified by the CF bits of the instruction word. The contents of the second, third, and fourth quarters of the same Memory Element register are then loaded into the following F Memory registers, successively.

Should the CF bits specify the last F Memory register, then the content of the next quarter of the Memory Element register is loaded into the first F Memory register (F_0), and so forth. (Note that the contents of F_0 always remain zero.)

7-6.3 FILE FORM OPERATION. The FLF instruction is the reverse of the SPG instruction in that the content of the F Memory register specified by the CF bits of the instruction word is stored in the first quarter of the selected Memory Element register.

7-6.4 FILE GROUP OF FORMS OPERATION. The FLG instruction is the reverse of the SPG instruction in that the contents of four successive F Memory registers are stored in the first, second, third and fourth quarters of one selected Memory Element register.

7-7 EXCHANGE ELEMENT OPERATION CODES

All of the Exchange Element instructions are indexable and configurable.

7-7.1 LOAD E OPERATION. The LDE instruction is similar to the Arithmetic Element load type instructions. In this case the configured content of the selected Memory Element register is loaded into the E register.

7-7.2 STORE E OPERATION. The STE instruction is similar to the Arithmetic Element store type instructions. In this case, the inversely configured content of the E register is stored in the selected Memory Element register.

7-7.3 INTERSECT E OPERATION. The ITE instruction is similar to the ITA instruction. In this case, the configured content of the selected Memory Element register is "intersected" (logically ANDed) with the content of the E register. The result is left in the E register.

7-7.4 PERMUTE AND COMPLEMENT OPERATIONS. During a PCM instruction, the content of the selected Memory Element register is first configured and placed in the E register. All the active subwords in the E register are then complemented. The final content of the E register (whether active or not) is then placed in the selected Memory Element register without inverse configuration. This instruction alters the original contents of the selected memory register.

If the configuration bits (CF) specify permutation only, then the operation is usually called PERMUTE (PMT); if the configuration bits specify activity only, the active subwords are complemented and the operation is usually called COMPLEMENT (COM).

7-8 SKIP OPERATION CODES

Of the three skip instructions, SED is indexable and configurable. The other two, SKM and SKX, are neither indexable nor configurable.

7-8.1 SKIP IF E DIFFERS OPERATION. During the SED instruction, the content of the E register is compared with the content of the selected Memory Element register. If the contents differ, then the next instruction is not executed (that is, it is skipped). Specifically, the configured content of the selected Memory Element register is compared with the content of the E register to determine whether any subwords differ. If there is a difference, then the next instruction is skipped.

The content of the E register remains unchanged during an SED operation.

7-8.2 SKIP ON M OPERATION. The SKM instruction is a bit-setting and decision-making instruction. In this instruction, the J bits of the instruction word are used to select a particular bit in the selected Memory Element register. The CF bits are then used to specify such functions as changing the binary value of the selected bit and making a decision based on the value of the selected bit. The exact interpretation of the J and CF bits is illustrated in Fig. 7-2.

The four J bits (3.4 - 3.1) are used to specify the bit number, and the remaining two J bits (3.6 - 3.5) are used to specify the quarter number. For example, 01 0001 specifies bit 1.1 of the selected memory register. Similarly, 00 1010 specifies bit 4.10.

The following points should be noted:

1) Decision making is always done first.
2) Bit changing comes next.
3) Cycling, or rotation, comes last.
4) The final content of the E register is the same as the final content of the selected Memory Element register.
5) If all the cf bits are zero, SKM does nothing other than changing the content of the E register.
6) SKM is the only operation that can affect the operand word meta bit. (It can not change the parity bit.)

7-8.3 SKIP ON INDEX OPERATION. The SKX instruction permits a programmer to dismiss, raise a flag, skip the next instruction and either reset or augment an index (X Memory) register. Neither an operand from the Memory Element nor a configuration from the F Memory is used. The X Memory register specified by the J bits of the instruction word is the only operand register affected by this operation.

The seventeen base address bits (2.8 - 1.1) are used as an operand rather than as the address of an operand. Since X Memory registers contain 18 bits, this operand is normally treated as a positive 18-bit number. If any deferred address cycles are executed, then it is possible for the resulting base address (which is to be used as the operand) to be negative.

The J bits address both an X Memory register, X_J, and the flag of a sequence, $FLAG_J$ (if one exists).

The CF bits are decoded to specify the desired variation of the SKX operation:

1) If the CF_5 bit is a ONE and if the hold bit is a ZERO, then a dismiss is performed, lowering the flag of the current sequence.

2) If the CF_4 is a ONE, then $FLAG_J$ will be raised (if such a flag exists). Since flag raising occurs after a flag lowering caused by a dismiss, an SKX operation which dismisses and raises the flag of its own sequence will have no apparent effect on the flag.

3) The remaining three CF bits are decoded as shown in Fig. 7-3.

7-9 JUMP OPERATION CODES

JPA, JNA and JOV are indexable, configurable instructions. JPX and JNX are nonindexable, nonconfigurable instructions. JMP is an indexable, nonconfigurable instruction.

7-9.1 JUMP OPERATION. The JMP instruction has several variants. These variants are specified by the CF bits:

1) If the CF_1 bit of the instruction word is a ZERO, then a jump is performed to the Memory Element register specified by the final base address. If the CF_1 bit is a ONE, then a jump is performed to the Memory Element register specified by the indexed final base address.

2) When the CF_2 bit of the instruction word is a ONE, then the address plus ONE of the JMP instruction is placed in register X_J of the X Memory.

3) When the CF_3 bit of the instruction word is a ONE, then the address plus ONE of the JMP instruction is placed in the first two quarters of the E register.

4) When the CF_4 bit of the instruction word is a ONE, then the content of the Q register is placed in the third and fourth quarters of the E register.

5) When the CF_5 bit is a ONE and the hold bit (H) is a ZERO, then the present sequence is dismissed.

Fig. 7-4 summarizes the interpretation of the CF bits during a JMP operation.

7-9.2 JUMP ON POSITIVE A OPERATION. During a JPA instruction, a jump is performed to the selected Memory Element register if any one of the active subwords in the A register is positive and non-zero.

If a jump occurs, the right half of the E register is set to the return address (the address plus ONE of the JPA instruction); otherwise, no change is made in the E register.

7-9.3 JUMP ON NEGATIVE A OPERATION. The JNA instruction is the same as the JPA instruction, except that a jump is performed only if one of the active subwords in the A register is negative and non-zero.

7-9.4 JUMP ON OVERFLOW OPERATION. The JOV instruction is the same as the JPA instruction, except that a jump is performed only if the overflow flip-flop (Z_i) of one of the active subwords in the A register is set to a ONE.

The JOV instruction does not clear overflow.

7-9.5 JUMP ON POSITIVE INDEX OPERATION. During a JPX instruction, a jump is performed to the selected Memory Element register if the initial content of X Memory register X_J is positive and non-zero.

If a jump is made, the return address is saved in the right half of the E register. Also, if a jump is made and the hold bit (H) is a ZERO, then the present sequence is dismissed.

Whether a jump occurs or not, the JPX instruction adds an increment to the content of X_J. This increment is represented by the CF bits. These 5 bits are interpreted as a 4 bit one's complement number with a sign digit.

7-9.6 JUMP ON NEGATIVE INDEX OPERATION. The JNX instruction is the same as the JPX instruction except that a jump is performed only if the initial content of the X Memory register X_J is negative and non-zero.

7-10 IN-OUT OPERATION CODES

TSD is indexable and, sometimes, configurable; IOS is neither indexable nor configurable.

7-10.1 TRANSFER DATA OPERATION. The TSD instruction transfers data between the selected Memory Element register and the K-th In-Out buffer register. If the K-th In-Out unit is an input device, data is transferred from the K-th In-Out buffer register to the selected Memory Element register. The transfer path is reversed when the K-th In-Out unit is an output device.

For certain In-Out units, the data can be transferred in either a NORMAL or an ASSEMBLY mode. When a TSD is executed in the NORMAL mode, data is transferred as a contiguous block of bits. In this case, the data is configured. If a TSD is executed in the ASSEMBLY mode, the data is not configured; instead, the Memory Element word is rotated one position to the left or right after the In-Out data transfer is completed.

The CF bits specify configuration when TSD is in the NORMAL mode. However, when TSD is in the ASSEMBLY mode, the CF bits are not used.

TSD uses the hold bit (H) in a unique manner. Normally, if the hold bit of the previous instruction is a ONE, no change of sequence can occur before the current instruction is executed. However, if the current instruction is a TSD and the K-th In-Out unit is not ready for an In-Out data transfer, then a dismiss is performed

(before the TSD is executed) independent of the value of the hold bit of the previous instruction. This is called "dismiss and wait" to distinguish it from ordinary dismissing, since the TSD instruction which causes the dismiss has not yet been executed. When the K-th In-Out unit is ready to perform the In-Out data transfer, then the flag of the K-th sequence is raised and the TSD is finally executed.

7-10.2 IN-OUT SELECT OPERATION. See Miscellaneous Operation Codes (OPR^{IO}).

7-11 MISCELLANEOUS OPERATION CODES

At present, there is one Miscellaneous Operation Code, namely OPR. Depending on the value of bits 2.8 and 2.7 in the final address section of the instruction word, OPR may be interpreted as an AOP or IOS instruction. Specifically, if bits 2.8 and 2.7 are both ZERO, an IOS is executed; if bit 2.8 is a ZERO and bit 2.7 is a ONE, then an AOP is executed. That is,

$N_{2.8}$	$N_{2.7}$	OPR
0	0	IOS
0	1	AOP
1	0	undefined
1	1	undefined

Both AOP and IOS are nonindexable, nonconfigurable instructions.

Fig. 7-5 shows the interpretation of the Y bits in the instruction word for both the IOS and AOP variation of this instruction.

7-11.1 ARITHMETIC ELEMENT COMMAND OPERATION. The AOP (OPR^{AE}) instruction is used when it is desired to manipulate directly existing data in the Arithmetic Element.

In this operation, the J and CF bits of the instruction word are not used, and the H bit serves its normal "hold" function. The Y bits, as shown in Fig. 7-5, determine the Arithmetic Element operation (addition, subtraction, etc.) and also the configuration (activity and subword form only).

7-11.2 IN-OUT SELECT OPERATION. The IOS (OPR^{IO}) instruction is used to control and/or report on the status of the In-Out system. For example, it can be used to raise and lower flags, and to connect and disconnect the various In-Out units.

The J bits of the instruction word are used to specify the sequence controlled by the IOS operation. The interpretation of the Y bits is shown in Fig. 7-5.

Only two of the CF bits are used. If CF_1 is a ONE, then a report on the current state of the In-Out unit is placed in the E register. The sequence specified by the J bits is dismissed when CF_5 is a ONE and H is a ZERO.

ARITHMETIC ELEMENT OPERATION CODES

TYPE	MNEMONIC CODE	INTERPRETATION	OCTAL CODE
LOAD	LDA	LOAD A	24
	LDB	LOAD B	25
	LDC	LOAD C	26
	LDD	LOAD D	27
STORE	STA	STORE A	34
	STB	STORE B	35
	STC	STORE C	36
	STD	STORE D	37
EXCHANGE	EXA	EXCHANGE A	54
ARITHMETIC OPERATIONS	ADD	ADD	67
	SUB	SUBTRACT	77
	MUL	MULTIPLY	76
	DIV	DIVIDE	75
	SCA	SCALE A	70
	SCB	SCALE B	71
	SAB	SCALE A & B	72
	CYA	CYCLE A	60
	CYB	CYCLE B	61
	CAB	CYCLE A & B	62
	NOA	NORMALIZE A	64
	NAB	NORMALIZE A & B	66
	TLY	TALLY	74
LOGIC OPERATIONS	ITA	INTERSECT A	41
	UNA	UNITE A	42
	DSA	DISTINGUISH A	65
	INS	INSERT	55
AE COMMAND	AOP	ARITHMETIC ELEMENT OPERATE (OPR^{AE})	04

FIG. 7-1. BASIC OPERATION CODES

TYPE	MNEMONIC CODE	INTERPRETATION	OCTAL CODE
INDEX MEMORY OPERATION CODES			
	RSX	RESET X	11
	DPX	DEPOSIT X	16
	EXX	EXCHANGE X	14
	AUX	AUGMENT X	10
	ADX	ADD X	15
CONFIGURATION MEMORY OPERATION CODES			
	SPF	SPECIFY FORM	21
	SPG	SPECIFY GROUP OF FORMS	22
	FLF	FILE FORM	31
	FLG	FILE GROUP OF FORMS	32
EXCHANGE ELEMENT OPERATION CODES			
	LDE	LOAD E	20
	STE	STORE E	30
	ITE	INTERSECT E	40
	PCM	PERMUTE & COMPLEMENT	56
SKIP OPERATION CODES			
	SED	SKIP IF E DIFFERS	43
	SKM	SKIP-MAKE	17
	SKX	SKIP ON INDEX	12
JUMP OPERATION CODES			
	JMP	JUMP	05
	JPA	JUMP ON POSITIVE A	46
	JNA	JUMP ON NEGATIVE A	47
	JOV	JUMP ON OVERFLOW	44
	JPX	JUMP ON POSITIVE INDEX	06
	JNX	JUMP ON NEGATIVE INDEX	07
IN-OUT OPERATION CODES			
	TSD	TRANSFER DATA	57
	IOS	IN-OUT SELECT (OPR IO)	04
MISCELLANEOUS OPERATION CODES			
	OPR	OPERATE	04

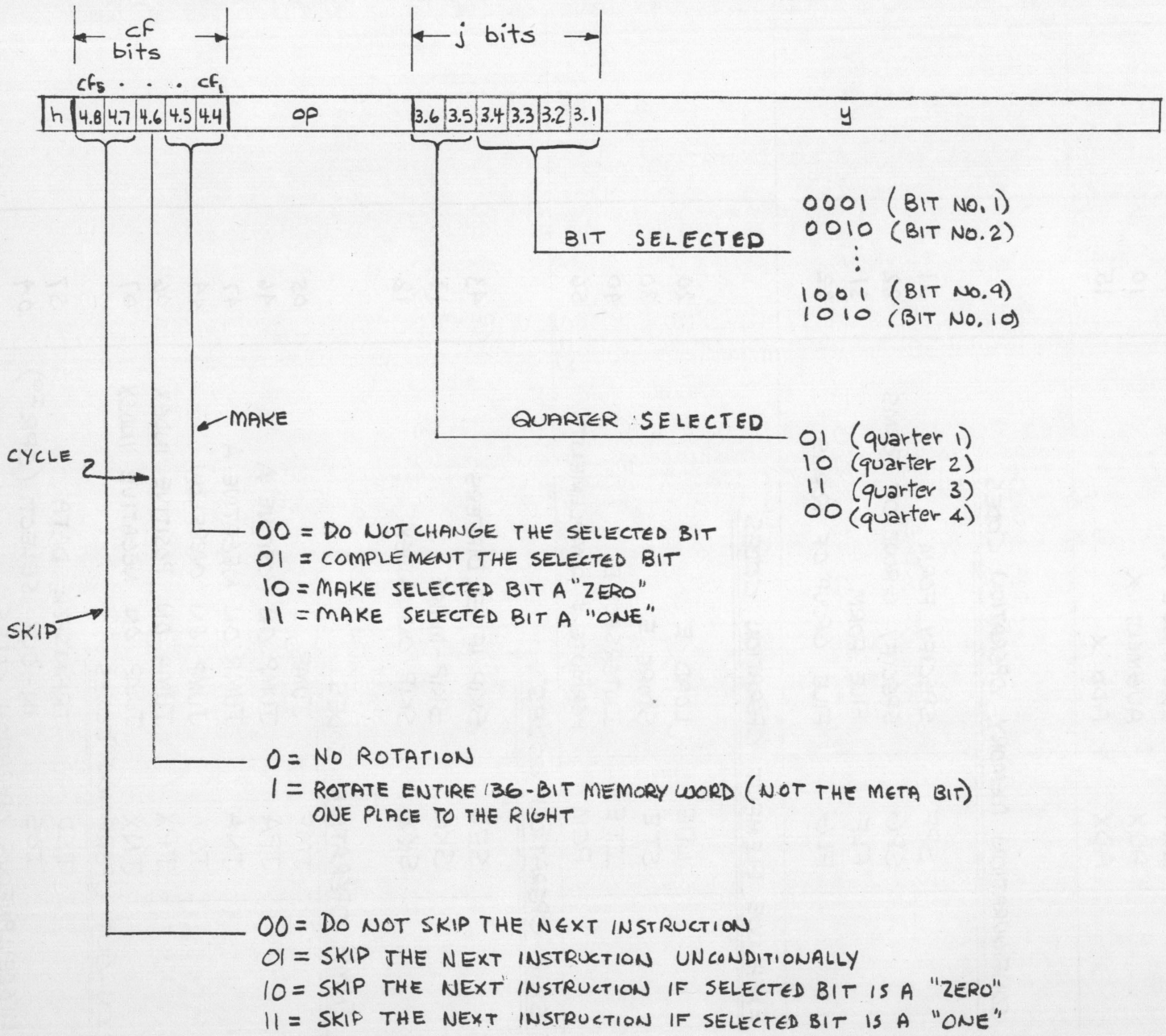


FIG. 7-2 INTERPRETATION OF BITS IN SKM INSTRUCTION

CF BITS			SYMBOLIC DESCRIPTION	FUNCTIONAL DESCRIPTION
CF_3	CF_2	CF_1		
0	0	0	$\Omega \longrightarrow X_j$	PLACE FINAL BASE ADDRESS Ω IN INDEX REGISTER X_j.
0	0	1	$\overline{\Omega} \longrightarrow X_j$	PLACE 18-BIT ONE'S COMPLEMENT OF FINAL BASE ADDRESS Ω IN INDEX REGISTER X_j.
0	1	0	$\Omega + \overline{X_j} \longrightarrow X_j$	AUGMENT CONTENTS OF INDEX REGISTER X_j WITH FINAL BASE ADDRESS Ω.
0	1	1	$\overline{\Omega} + \overline{X_j} \longrightarrow X_j$	AUGMENT CONTENTS OF INDEX REGISTER X_j WITH 18-BIT ONE'S COMPLEMENT OF FINAL BASE ADDRESS Ω.
1	0	0	$\overline{X_j} \neq \Omega \supset$ SKIP	SKIP NEXT INSTRUCTION IF CONTENTS OF INDEX REGISTER X_j ARE ALGEBRAICALLY NOT EQUAL TO FINAL BASE ADDRESS Ω.
1	0	1	$\overline{X_j} \neq \overline{\Omega} \supset$ SKIP	SKIP NEXT INSTRUCTION IF CONTENTS OF INDEX REGISTER X_j ARE ALGEBRAICALLY NOT EQUAL TO ONE'S COMPLEMENT OF FBA Ω.
1	1	0	$\overline{X_j} < \Omega \supset$ SKIP	SKIP NEXT INSTRUCTION IF CONTENTS OF INDEX REGISTER X_j ARE ALGEBRAICALLY LESS THAN FINAL BASE ADDRESS Ω.
1	1	1	$\overline{X_j} > \overline{\Omega} \supset$ SKIP	SKIP NEXT INSTRUCTION IF CONTENTS OF INDEX REGISTER X_j ARE ALGEBRAICALLY GREATER THAN ONE'S COMPLEMENT OF FBA Ω.

SKX (SKIP ON INDEX)

FIG 7-3 INTERPRETATION OF CF_{3-1} IN SKX INSTRUCTION

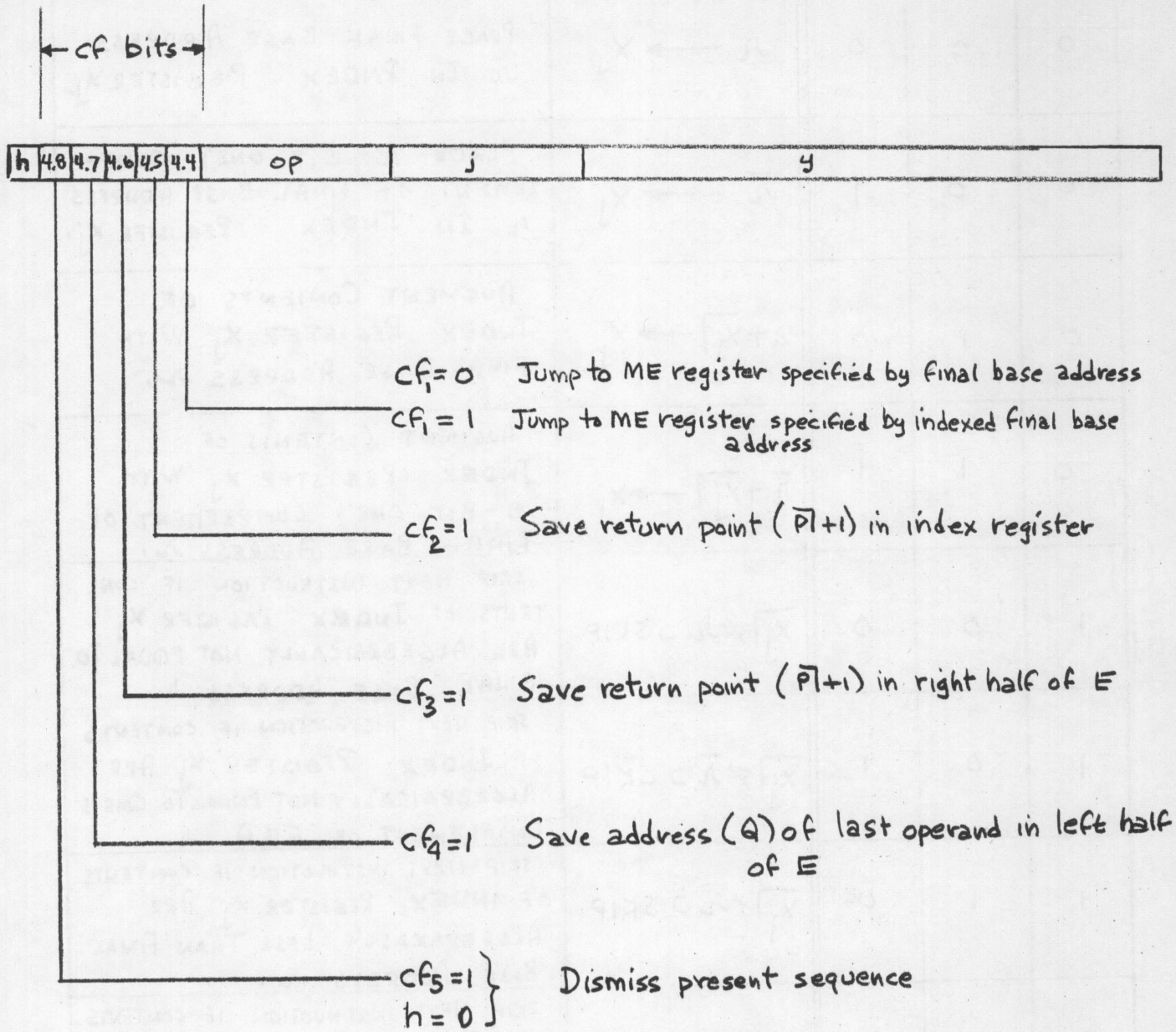


FIG 7-4 INTERPRETATION OF CF BITS DURING JMP OPERATION

OP CODE	Y BITS								FUNCTIONAL DESCRIPTION
	2.8	2.7	2.6	2.5	2.4	2.3→2.1	1.9→1.4	1.3→1.1	
IOS	0	0				—	—	—	SPECIFIES AN IOS OPERATION
			0	1	0	—	—	—	DISCONNECT IOU_j
			0	1	1	XXX	XXX XXX	XXX	CONNECT IOU_j (PLACE IN MODE SPECIFIED BY $N_{2.3-1.1}$)
			1	1	0	XXX	XXX XXX	XXX	SELECT THE DEVICE IN IOU_j SPECIFIED BY $N_{2.3-1.1}$
			1	0	1	—	—	—	RAISE $FLAG_j$
			1	0	0	—	—	—	LOWER $FLAG_j$
AOP	0	1							SPECIFIES AN AOP OPERATION
			← OP CODE NO. →						BITS $N_{2.6-2.1}$ USED TO SPECIFY OP CODE DIRECTLY TO AE.
							← CONFIGURATION →		BITS $N_{1.9-1.4}$ USED TO SPECIFY CONFIGURATION DIRECTLY TO AE.

FIG. 7-5 INTERPRETATION OF Y BITS IN OPR INSTRUCTION