3–32–0

THEORY OF OPERATION

OF

CENTRAL COMPUTER SYSTEM

FOR

AN/FSQ-7
COMBAT DIRECTION CENTRAL

AND

AN/FSQ-8
COMBAT CONTROL CENTRAL

VOLUME I

1 February 1959

MILITARY PRODUCTS DIVISION
INTERNATIONAL BUSINESS MACHINES CORPORATION
KINGSTON, NEW YORK

LIST OF REVISED PAGES

INSERT LATEST REVISED PAGES. DESTROY SUPERSEDED PAGES.

NOTE: The portion of the text affected by the current revision is indicated by a vertical rule in the left margin of a left-hand page and in the right margin of a right-hand page.

*The asterisk indicates pages revised, added or deleted by the current revision.

CONTENTS

Heading *Page*

VOLUME I

CONTENTS (cont'd)

CONTENTS (cont'd)

CONTENTS (cont'd)

CONTENTS (cont'd)

CONTENTS (cont'd)

CONTENTS (cont'd)

CONTENTS (cont'd)

CONTENTS (cont'd)

CONTENTS (cont'd)

VOLUME II

CONTENTS (cont'd)

CONTENTS (cont'd)

CONTENTS (cont'd)

CONTENTS (cont'd)

CONTENTS (cont'd)

CONTENTS (cont'd)

CONTENTS (cont'd)

CONTENTS (cont'd)

CONTENTS (cont'd)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (cont'd)

LIST OF ILLUSTRATIONS (cont'd)

LIST OF ILLUSTRATIONS (cont'd)

LIST OF ILLUSTRATIONS (cont'd)

LIST OF ILLUSTRATIONS (cont'd)

LIST OF TABLES

LIST OF TABLES (cont'd)

LIST OF TABLES (cont'd)

PART 1
INTRODUCTION

CHAPTER 1
GENERAL

1.1 PURPOSE, SCOPE, AND ORGANIZATION OF MANUAL

1.1.1 Purpose

This manual is intended primarily as an aid in training field engineers in the detailed theory of operation of the Central Computer, Magnetic Tape, and Warning Light Systems of the AN/FSQ-7 Combat Direction Central and AN/FSQ-8 Combat Control Central equipment. Its secondary purpose is to furnish a reference source for field engineers in the maintenance of these systems.

1.1.2 Scope

In general, the scope of this manual is restricted to the basic Central Computer, Magnetic Tape, and Warning Light Systems. However, portions of the other units and systems of the equipment which are involved in the transfer of information into and out of these systems are discussed in varying degrees, as required for adequate coverage. Whenever applicable, direct reference is made to the theory of operation manuals of the other systems of the equipment. The numbers and titles of these manuals follow:

3–12–0 *Introduction to AN/FSQ-7 Combat Direction Central and AN/FSQ-8 Combat Control Central*

3–42–0 *Theory of Operation of Drum System for AN/FSQ-7 Combat Direction Central and AN/FSQ-8 Combat Control Central*

3–52–0 *Theory of Operation of Input System for AN/FSQ-7 Combat Direction Central and AN/FSQ-8 Combat Control Central*

3–62–0 *Theory of Operation of Display System for AN/FSQ-7 Combat Direction Central and AN/FSQ-8 Combat Control Central*

3–72–0 *Theory of Operation of Output System for AN/FSQ-7 Combat Direction Central and AN/FSQ-8 Combat Control Central*

3–82–0 *Theory of Operation of Power Supply System for AN/FSQ-7 Combat Direction Central and AN/FSQ-8 Combat Control Central*

3–92–0 *Theory of Operation of Marginal Checking System for AN/FSQ-7 Combat Direction Central and AN/FSQ-8 Combat Control Central*

3–713–0 *Card Reader Type 713*

3–718–0 *Line Printer Type 718*

3–723–0 *Card Recorder Type 723*

3–728–0 *Tape Drive Type 728*

The *Introduction to AN/FSQ-7 Combat Direction Central and AN/FSQ-8 Combat Control Central* manual is provided to define the function of the equipments in relation to the SAGE System, which is the overall air defense system, and to define the function and relationships of the major units and systems of the equipment. Each of the other manuals listed above provides the detailed theory of operation of one of the major divisions of the equipment.

The theory of operation contained in this manual is based on the engineering logic drawings that represent the 12th AN/FSQ-7 (DC-12) and the 3rd AN/FSQ-8 (CC-3) equipments. In providing the detailed theory of operations, direct reference is made to the logic drawings whenever possible; however, simplified block diagrams are included in the manual if they serve to simplify a particular logic analysis.

Since the theory of operation contained in this manual is presented at the block diagram level, the schematic diagram analysis of the various circuits contained in these systems are not included in the logic discussions except as required to define the function of specific blocks. The detailed theory of operation of all the circuits of these systems is contained in the following manuals:

3–22–0 *Basic Circuits for AN/FSQ-7 Combat Direction Central and AN/FSQ-8 Combat Control Central*

3–3–0 *Special Circuits for AN/FSQ-7 Combat Direction Central and AN/FSQ-8 Combat Control Central*

In general, it is assumed that the reader is familiar with the logic function of each of the basic circuits; therefore, direct reference is not made to the basic circuits manual. However, since the special circuits are defined as being the unique circuits of a system, direct reference is made to the special circuits manual whenever a special circuit is first encountered.

1.1.3 Organization

Parts 1 through 10 of this manual contain an introduction to, and a detailed analysis of, the theory of operation of the Central Computer, Magnetic Tape, and Warning Light Systems of the AN/FSQ-7 equipment. Part 1 contains the introduction to and a general description of these systems. Paragraphs 1.2 and 1.3 of this chapter define the function of these systems and their relationship to the other portions of the equipment. Chapter 2 of this Part is the only portion of this manual that deals with the physical aspects of these systems. Chapters 3, 4, and 5 define the primary characteristics and functional elements of the Central Computer and provide a broad overall explanation of computer operation. Chapter 5 provides a statement of function of each of the 59 instructions that the Central Computer is capable of executing.

Parts 2 through 8 each contain a detailed analysis of one of the functional elements of the Central Computer System.

Parts 9 and 10 contain a detailed analysis of the Magnetic Tape and Warning Light Systems, respectively.

Part 11 contains a detailed analysis of the differences in the internal operation of the Central Computer System of the AN/FSQ-8 Combat Control Central as compared with the Central Computer System of the AN/FSQ-7 Combat Direction Central.

1.2 DESCRIPTION OF AN/FSQ-7 EQUIPMENT

The Central Computer Systems of the AN/FSQ-7 Combat Direction Central and the AN/FSQ-8 Combat Control Central are functionally identical in performing their overall tasks, but some differences exist in the internal operation of the two systems. Since the two systems perform essentially the same overall function, the following brief discussion of the relationship of the Central Computer System to the other systems of the two equipments will be based on the AN/FSQ-7 equipment only. A more detailed description of the AN/FSQ-7 and AN/FSQ-8 equipments is contained in manual 3-12-0, *Introduction to AN/FSQ-7 Combat Direction Central and AN/FSQ-8 Combat Control Central.*

The AN/FSQ-7 Combat Direction Central is a data-processing machine designed to handle large amounts of military tactical data associated with a number of aircraft detection and weapon devices. Basically, the equipment receives radar data, converts it into meaningful target information (location, speed, and direction of aircraft), and displays this information for human interpretation and decision-making. Commands and requests issued by human operators that are pertinent to specific aircraft are then processed by the equipment and then sent back to the scope operators or to weapons centers, friendly aircraft, missile bases, etc., so that defense action can be carried out.

Because of operational considerations, the AN/FSQ-7 Combat Direction Central is divided into 10 major groups of circuits, each of which has its own special function to perform. These major groups, shown in figure 1–1, are as follows:

a. Input System
b. Drum System
c. Central Computer System
d. Output System
e. Display System
f. Warning Light System
g. Card Machines
h. Tape System
j. Maintenance Consoles
k. Power and Marginal Checking System

As noted in figure 1–1, the Input, Output, Display, Drum, and Central Computer Systems are closely related in the performance of the air defense task. One of the functions of the Input System is to control the flow of target data that is transferred to the central from numerous radar sites, to convert this data from serial form into parallel form, and to store it on the appropriate input fields of the Drum System. The Drum System is a medium-speed, medium-capacity storage device used as an intermediate time buffer between the Central Computer System and the Input, Output, and Display Systems. Periodically, the accumulated input data is transferred from the input drum fields to the Central Computer System under computer control. Thus, although the input data is received at random, it is transferred to the computer as a block of data. Further, since the Drum System is a medium-speed storage device, the input data is transferred to the computer at a speed compatible with the high information-handling speed of the Central Computer System. It is the function of the Central Computer System to correlate and process this input data and to assemble the results into a block of data for subsequent transfer to the Display System or to the Output System. Periodically, and under computer controls, the block of processed target data is transferred to the display fields or output fields of the Drum System for subsequent transfer to the Display System or to the Output System.

Since the Display System is a relatively slow-operating device, the Drum System acts as a time buffer in that it controls the transfer of discrete pieces of display

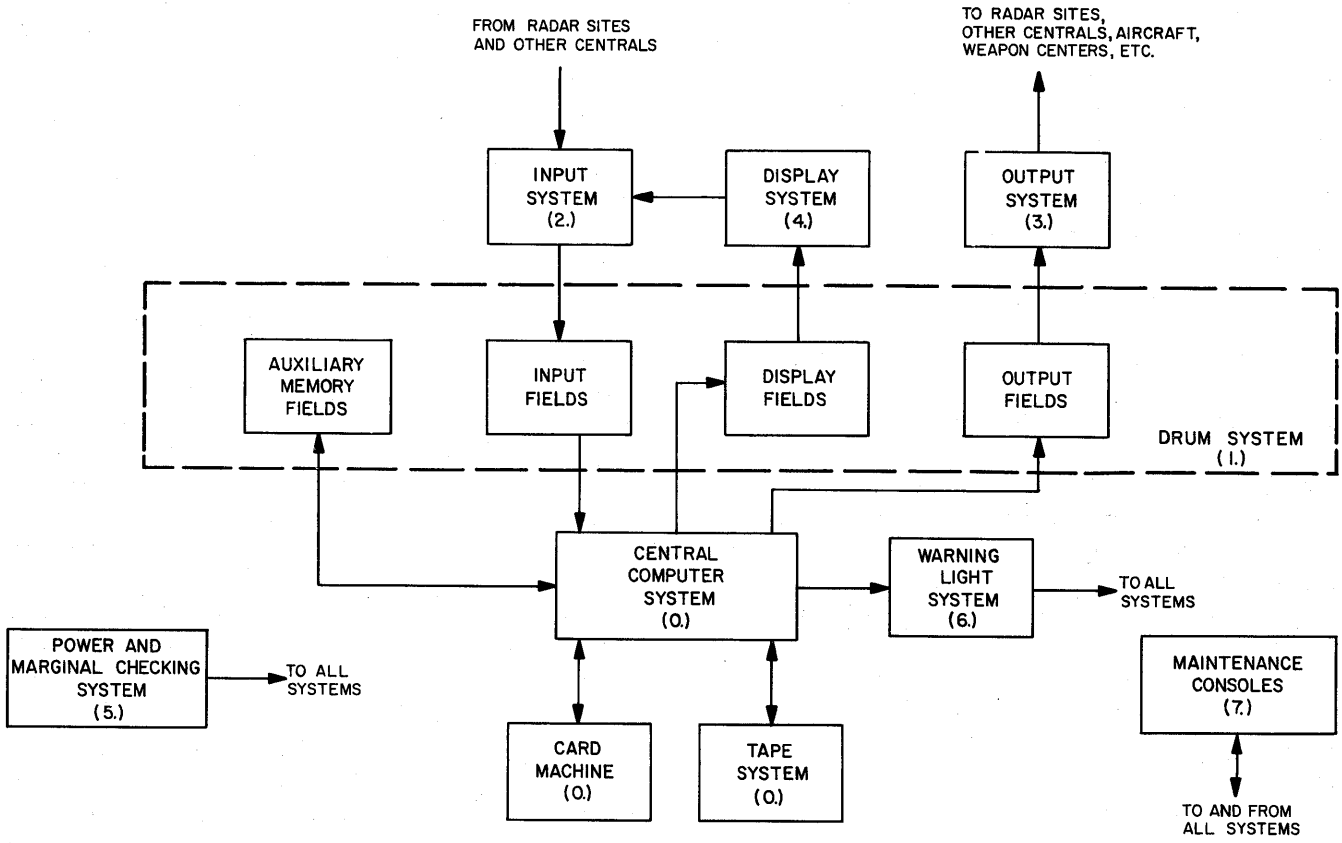


Figure 1–1. AN/FSQ-7 Combat Direction Central, System Diagram

data at the optimum operating speed of the Display System. Each piece of data thus transferred is decoded for visual presentation to human operators for interpretation and control. If one of the operators desires more information about a specific target or that some action be taken, he will communicate his requests to the Central Computer System through the Input System.

This input information is transferred under computer control in essentially the same manner as input target data. After the transfer is accomplished, the Central Computer System processes these requests and/or commands and assembles the resulting information into a block of data for subsequent transfer back to the Display System or to the Output System. The Output System is the interconnecting link between the Central Computer System and devices external to the AN/FSQ-7 equipment (i.e., air bases, missile bases, other centrals, etc.). Periodically, under computer control, this block of output data is transferred to the output fields of the Drum System. Discrete portions of the output data are then transferred to the Output System at its own operating speed. The Output System forms this data into the specified messages, converts each message from parallel form into serial form, and transmits the data to its destination through the selected telephone links or radio transmitters of the output channel equipment.

The other systems and units (fig. 1–1) of the AN/FSQ-7 equipment do not control the flow of operating data within the central; therefore, they are not as interrelated as the systems previously discussed. The Warning Light System of the AN/FSQ-7 equipment is a passive alarm system actuated by the Central Computer System to warn operators at the various consoles of unusual air defense situations, impending operations, action requests, and/or malfunctions which require human intervention. The card machines and tape system are large-capacity, slow-speed devices which are primarily used as direct input and output devices for initial program-loading and maintenance-testing of the equipment. The maintenance consoles of the equipment contain all the maintenance and switching controls used for setting up initial operations, and for maintenance control of the equipment. These consoles also contain indicator lights and neons which indicate the status of the various registers and controls of the equipment. The Power Supply and Marginal Checking System contains the required d-c power supplies and all the a-c and d-c distribution equipment to supply electrical power to all the other systems. In addition, this system contains the controls and equipment required to provide for the marginal checking of selected circuits during reliability testing of the equipment.

Because the nature of the air defense task requires that the equipment be operational 100 percent of the time, spare equipment is provided so that each portion of the operating equipment can be replaced for maintenance checking purposes. This need for keeping the equipment continually operational resulted in the philosophy of duplex operation, wherein most of the equipment is duplicated as one operating unit. This duplicated (duplex) equipment, referred to as computer A and computer B, consists of the Central Computer System, the Drum System, the Warning Light System, the Magnetic Tape System, the Output System, the common portion of the Input System, the decoding portion and some of the display consoles of the Display System, the associated (duplex) power supply and marginal checking equipment, and the associated maintenance console. The remainder of the equipment, referred to as simplex units, consists primarily of the conversion (channel) equipment of the Input System, the majority of the display consoles of the Display System, the associated (simplex) power and marginal checking equipment, and the various indicators of the Warning Light System.

Since the input channel equipment contains three groups of identical units, only a few spare units are provided for each group. Spare simplex display consoles are not required in this equipment because the distribution of display data is such that the function and coverage of each simplex display console is overlapped by other simplex display consoles in the system. The simplex power supply units are duplicated so that the simplex load units can receive power from one of two sources.

As noted in figure 1–2, the duplex equipment is controlled so that only one portion of this equipment is always in active status (that is, performing the air defense task) while the other portion is in standby status. Individual portions of the duplex equipment cannot be controlled separately. The status of each of the simplex units, including the spare input units, is individually controlled so that each unit may be connected to the active duplex equipment or to the standby duplex equipment, or disconnected from both portions of the duplex equipment. During normal operation (fig. 1–2), the input channel units, the output channel equipment, the warning light indicators, and all the simplex display consoles are connected to the active duplex equipment to perform the air defense task; the spare input units are connected to the standby duplex equipment. When maintenance is to be performed on a given active simplex input unit, one of the spare input units is substituted to take over its function, and the deactivated input unit is connected to the standby duplex equipment for maintenance purposes. After the maintenance operation has been completed, the statuses of the tested input unit and the spare unit are reversed so that the spare unit may be used as a substitute for another simplex input unit. This principle of substitution can be used to check each of the simplex input units to ensure their reliable operation. Prior to performing maintenance on a given simplex display console, the maintenance operator must ensure that the function and coverage of the specific console has been taken over by the other active consoles in the system. The specific console is then connected to the standby duplex equipment for maintenance purposes. After reliable operation of the display console is assured, it is again connected to the active duplex equipment to resume its normal function and coverage. When maintenance is to be performed on the active duplex equipment, the statuses of the two duplex equipments are reversed; that is, the standby duplex equipment is made active and the active duplex equipment becomes standby. The active simplex equipment becomes automatically connected to the newly active duplex equipment. The previously active duplex equipment is then available for maintenance purposes. Reliable operation of all standby equipment is continually maintained so that, if some portion of the active equipment becomes faulty or erratic, it can be replaced with a minimum loss of operational time.

1.3 FUNCTION OF CENTRAL COMPUTER SYSTEM

As previously stated, the primary function of the Central Computer System is to algebraically and logically process the information supplied to it from the Input System via the Drum System, and to arrive at results pertinent to the air defense situation. This processing takes the form of numerical computation with binary-coded data concerning tactical conditions in the geographical sector of AN/FSQ-7 Combat Direction Central. All tactical information supplied to the Central Computer System is in binary numerical form, and the Central Computer System processes the data as if it were numerical in both content and nature.

As tactical information is introduced into the Central Computer System, it is processed and correlated in a fixed pattern. The actual manner of processing is predetermined by additional information (a program) introduced into the Central Computer System. This programmed information consists of a series of orders which, when executed in the prescribed sequence, cause the Central Computer System to process the available tactical data and to arrive at the desired results; that is, to update the previously processed data. After the computations are completed, the processed data is transferred to the Drum System for temporary storage and subsequent transfer to the Output and Display Systems. This sequence of events – the transfer in of new raw data, the processing of this data to update the old data, and the transfer out of the latest data for display and/or output purposes – constitutes the basic operational cycle of the equipment. This cycle of events is controlled by the computer program and is continuously repeated to

constantly inform the Air Defense Command personnel and associated weapons systems of the status of the air defense situation.

In addition to its function of processing raw information, the Central Computer System acts as the control center for the AN/FSQ-7 Combat Direction Central. As information is processed, signals are generated by the Central Computer System and sent to the Drum System for use by the Input, Output, and Display Systems. These signals control the transfer of data between systems, initiate operational cycles, set up control circuits for impending operations, and generally synchro-

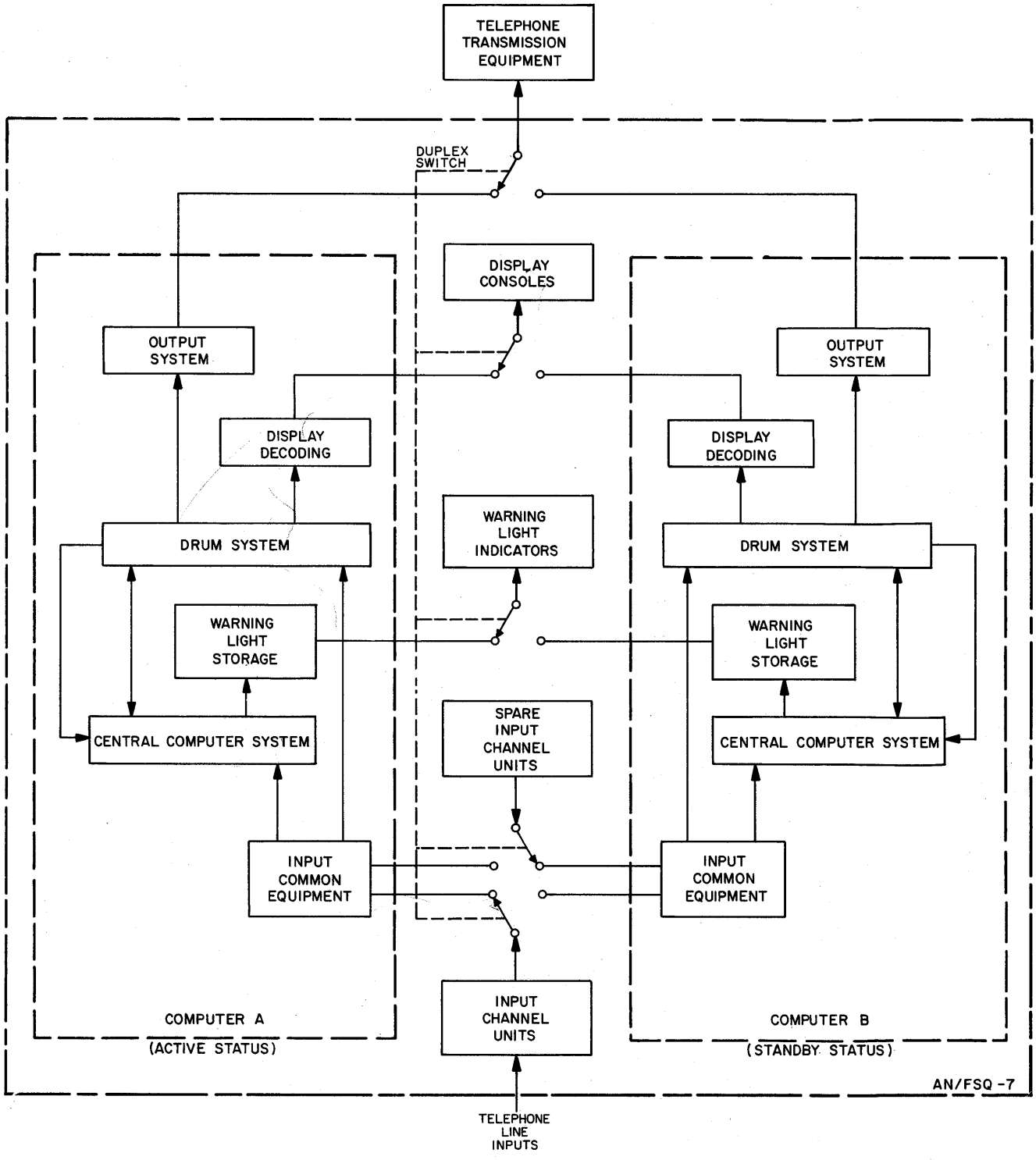


Figure 1–2. Duplex Switching Facility

nize the action of each of these systems with that of the Central Computer System.

In summary, the purpose of the Central Computer System may be separated into two parts: (1) information-processing and (2) synchronization and control. Information-processing takes place within the Central Computer System and involves the computation of results from supplied information. The synchronization and control function of the Central Computer System supervises and co-ordinates the acquisition, transfer, transmittal, and storage of information through the AN/FSQ-7 Combat Direction Central, so that all operations take place at the proper time and in the proper sequence.

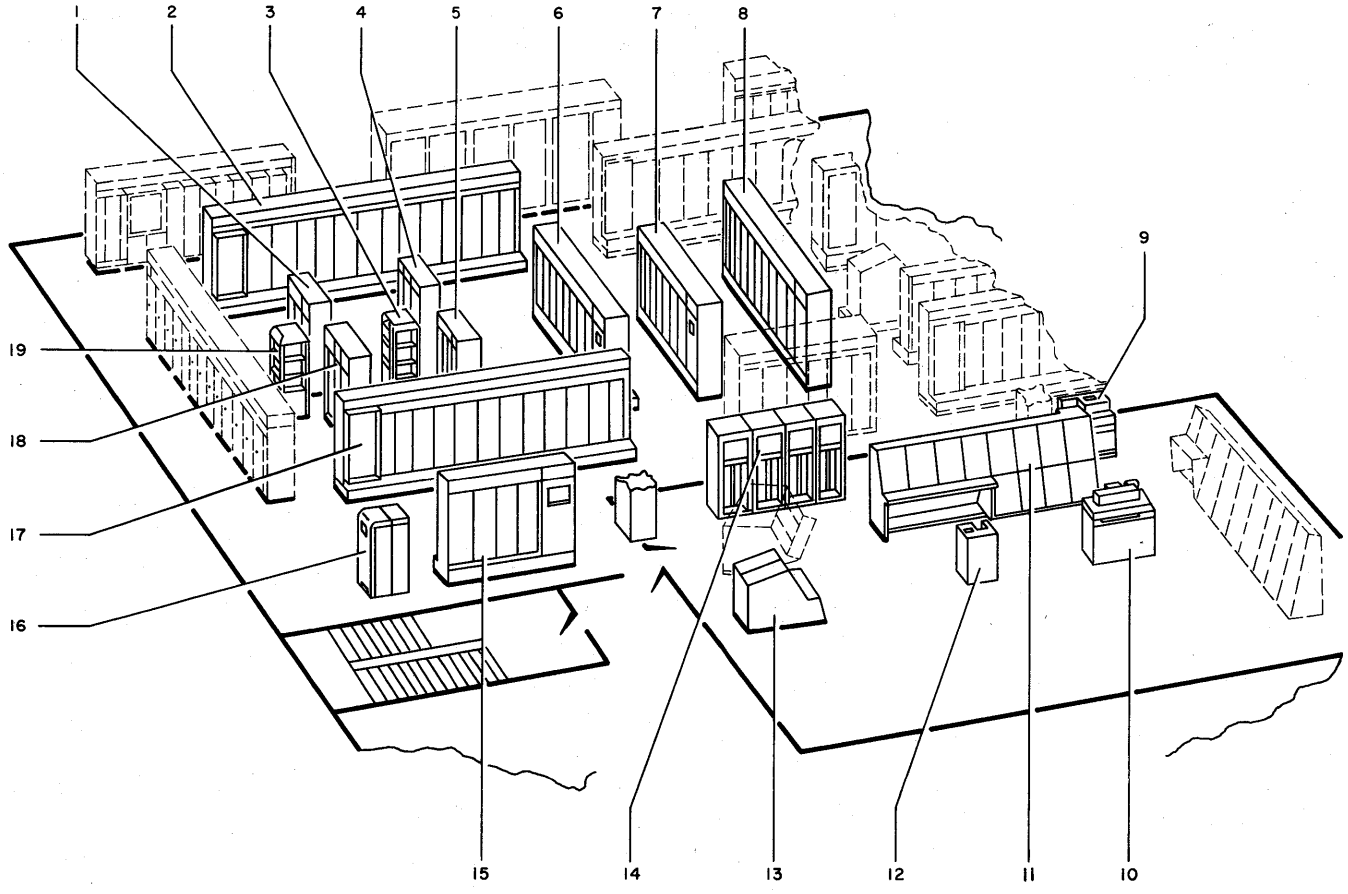
CHAPTER 2
PHYSICAL DESCRIPTION

2.1 INTRODUCTION

Although the theory of operation presented in this manual is based on the functional rather than on the physical division of the systems, a description of the physical units which these systems comprise is included in the manual for reference purposes. Figure 1–3, which is a partial floor plan of the first floor of the SAGE Site building which houses the AN/FSQ-7 equipment, identifies and shows the relative position of the physical units of the Central Computer and Magnetic Tape Systems. In addition, this figure also identifies and shows the relative location of the IBM card machines and the duplex switching console, which are functionally associated with the Central Computer System.

The physical units of the Warning Light System consist of the warning light storage unit, the warning light interconnection unit, and the warning light indicators. The warning light storage and warning light interconnection units, although not included in figure 1–3, are also located on the second floor of the SAGE Site building. The individual warning light indicators, usually discussed as a group, are actually distributed among approximately 100 consoles.

The physical units of the Central Computer, Mag-



1. LEFT MEMORY 1 UNIT
2. LEFT ARITHMETIC UNIT
3. CORE MEMORY 2 ARRAY UNIT
4. LEFT MEMORY 2 UNIT
5. RIGHT MEMORY 2 UNIT
6. INSTRUCTION CONTROL UNIT
7. SELECTION CONTROL UNIT
8. PROGRAM UNIT
9. IBM 723 CARD PUNCH
10. IBM 718 LINE PRINTER
11. DUPLEX MAINTENANCE CONSOLE
12. IBM 713 CARD READER
13. DUPLEX SWITCHING CONSOLE
14. IBM 728 MAGNETIC TAPE DRIVE UNITS
15. MAGNETIC TAPE ADAPTER UNIT
16. MAGNETIC TAPE POWER SUPPLY UNIT
17. RIGHT ARITHMETIC UNIT
18. RIGHT MEMORY 1 UNIT
19. CORE MEMORY 1 ARRAY UNIT

Figure 1–3. Second Floor of SAGE Building, Partial View

netic Tape, and Warning Light Systems represent three types of construction: standard pluggable unit type modular construction, panel type modular construction, and nonmodular construction.

2.1.1 Standard Modular Construction

Most of the physical units of the Central Computer System and the magnetic tape adapter unit of the Magnetic Tape System and the warning light storage unit of the Warning Light System belong in this group. Standard modular units are defined as consisting of several standard pluggable unit racks (modules) bolted together to form a unit. Each standard module has the capacity to hold 20 pluggable units. These pluggable unit locations are designated, from top to bottom, as pluggable unit row C, D, E, F, G, H, J, K, L, M, N, P, R, S, T, U, V, W, X, and Y.

The following Central Computer units use standard modular construction:

a. Instruction control unit
b. Left arithmetic unit
c. Right arithmetic unit
d. Program unit
e. Selection control unit
f. Left memory 2 unit
g. Right memory 2 unit

In addition to the above, the left and right memory 1 units utilize a modified version of standard modular construction in that these two units are composed of modules that contain 23 pluggable units each. The three additional pluggable unit locations are designated as pluggable unit row AA, BB, and CC.

2.1.2 Panel Type Modular Construction

The duplex maintenance and duplex switching consoles of the AN/FSQ-7 equipment which contain the manual controls and indicators associated with the du-

TABLE 1–1. LIST OF PHYSICAL UNITS

SYSTEM	UNIT	FIGURE
Central Computer	Left arithmetic (unit 2)	1–4
	Right arithmetic (unit 3)	1–5
	Instruction control (unit 4)	1–6
	Selection control (unit 5)	1–7
	Program (unit 6)	1–8
	Left memory 1 (unit 65)	1–9
	Core memory 1 array (unit 66)	1–9
	Right memory 1 (unit 67)	1–9
	Left memory 2 (unit 10)	1–10
	Core memory 2 array (unit 11)	1–10
	Right memory 2 (unit 12)	1–10
	Duplex maintenance console (unit 1)	1–11
Tape	Magnetic tape adapter (unit 13)	1–12
	IBM 728 magnetic tape drive (unit 14)	1–13
	IBM 728 magnetic tape drive (unit 15)	1–13
	IBM 728 magnetic tape drive (unit 16)	1–13
	IBM 728 magnetic tape drive (unit 17)	1–13
	Magnetic tape power supply (unit 18)	1–14
Warning Light	Warning light storage (unit 30)	1–15
	Warning light interconnection (unit 91)	1–16
Card Machines	IBM 713 card reader (unit 51)	1–17
	IBM 718 line printer (unit 52)	1–18
	IBM 723 card punch (unit 53)	1–19
Auxiliary Console	Duplex switching console (unit 45)	1–20

plex portion of equipment are made up of panel type modules. Panel type modular units are defined as consisting of several 4-sided verticle chambers (modules) which are bolted together to form a unit. The switches and indicators associated with a specific module are mounted on the front panel of that module, and the relays and connectors associated with these components are mounted on the inner side of the left and right panels of the module.

2.1.3 Nonmodular Construction

The two memory array units of the Central Computer System, the magnetic tape drive units and magnetic tape power supply unit of the Magnetic Tape System, and the warning light interconnection unit of the Warning Light System belong in this group. The magnetic tape drive units are modified commercial IBM equipment adapted for use with the AN/FSQ-7 equipment. The other units listed above are specially constructed pieces of equipment which could not be manufactured in standard modular form. All the units in this group consist of free-standing assemblies.

2.2 DESCRIPTION OF INDIVIDUAL UNITS

Since the physical description contained in this manual is included for reference purposes only, the scope of the description is purposely limited and consists almost entirely of photographs. In addition to providing coverage for the physical units of the Central Computer, Magnetic Tape, and Warning Light Systems, photographs are also included of each of the IBM card machines and the duplex switching console. These later units, although not part of the scope of this manual, are included here because of their functional relationship with the Central Computer System.

Table 1–1 lists the individual units by group, name, and number and provides a reference to the photographs that are included in this chapter.

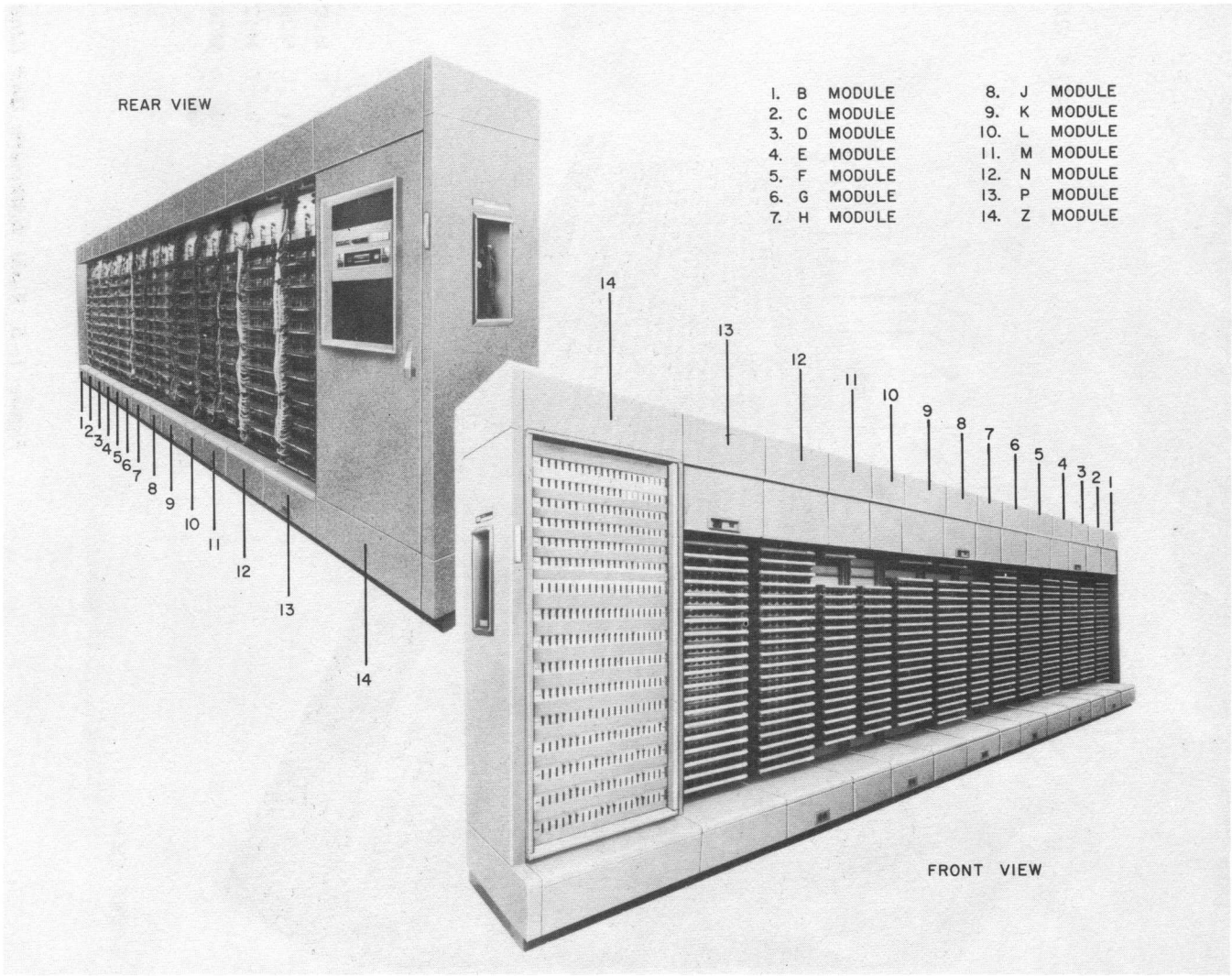


Figure 1–4. Left Arithmetic Unit (Unit 2)

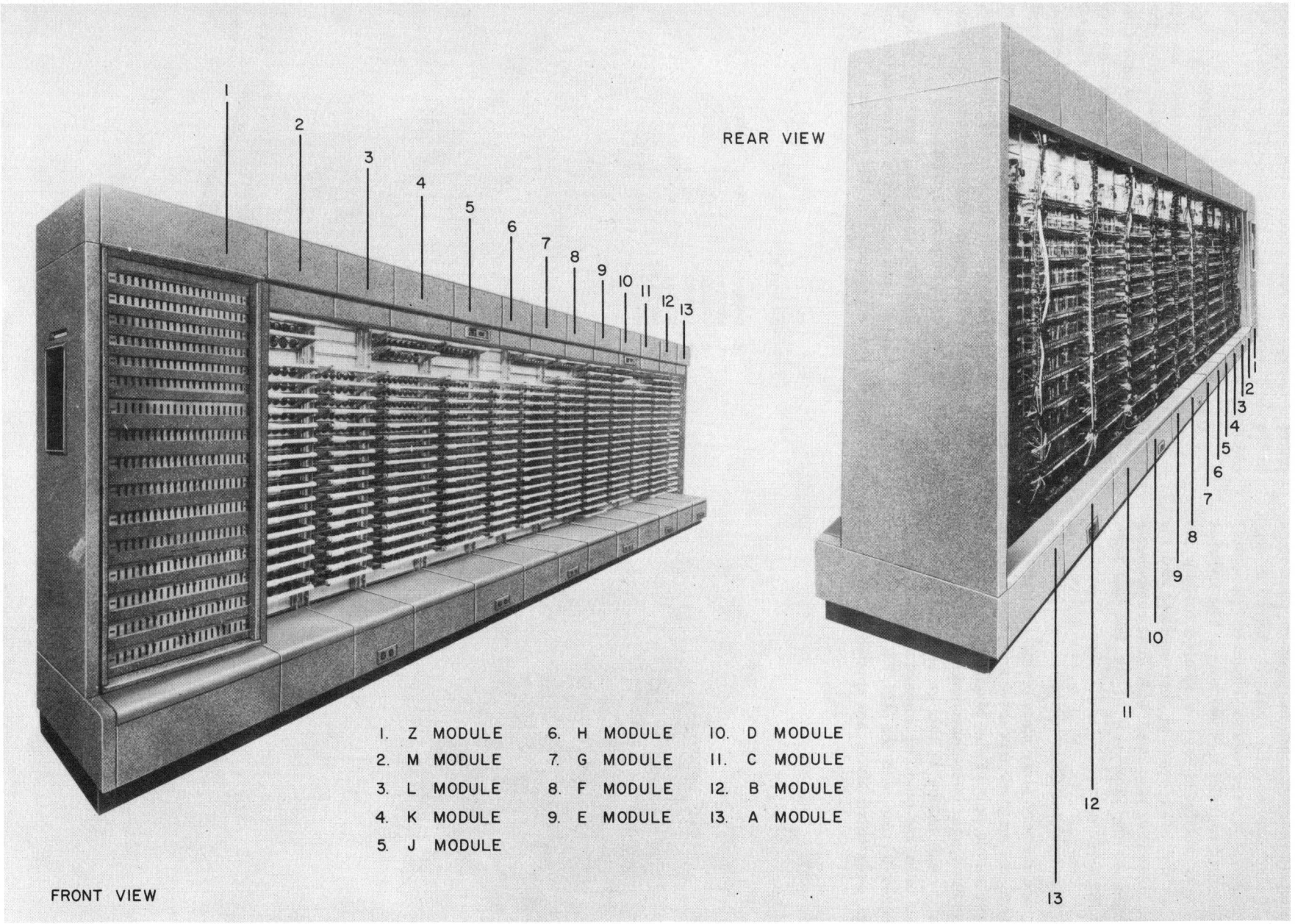


Figure 1–5. Right Arithmetic Unit (Unit 3)

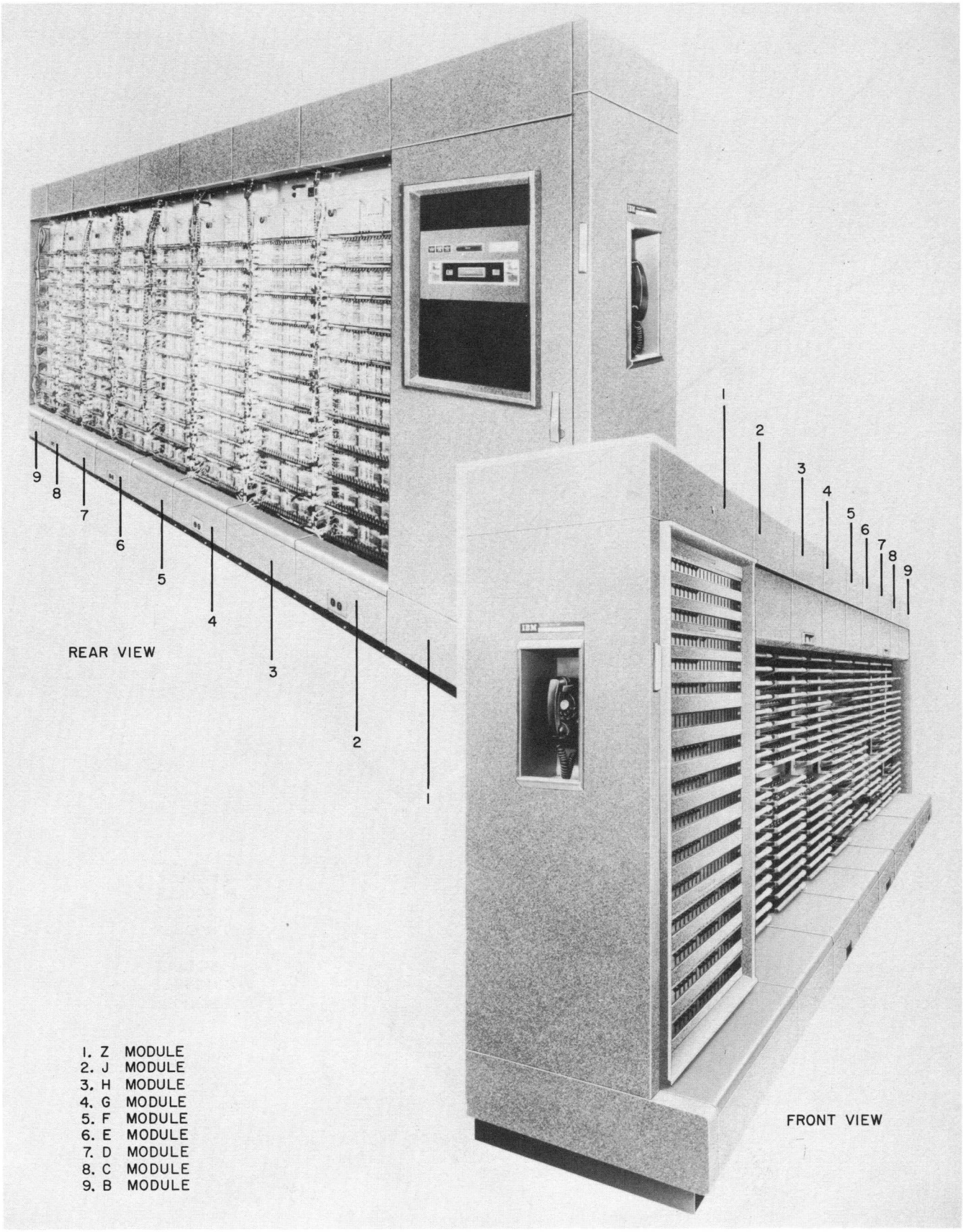


Figure 1-6. Instruction Control Unit (Unit 4)

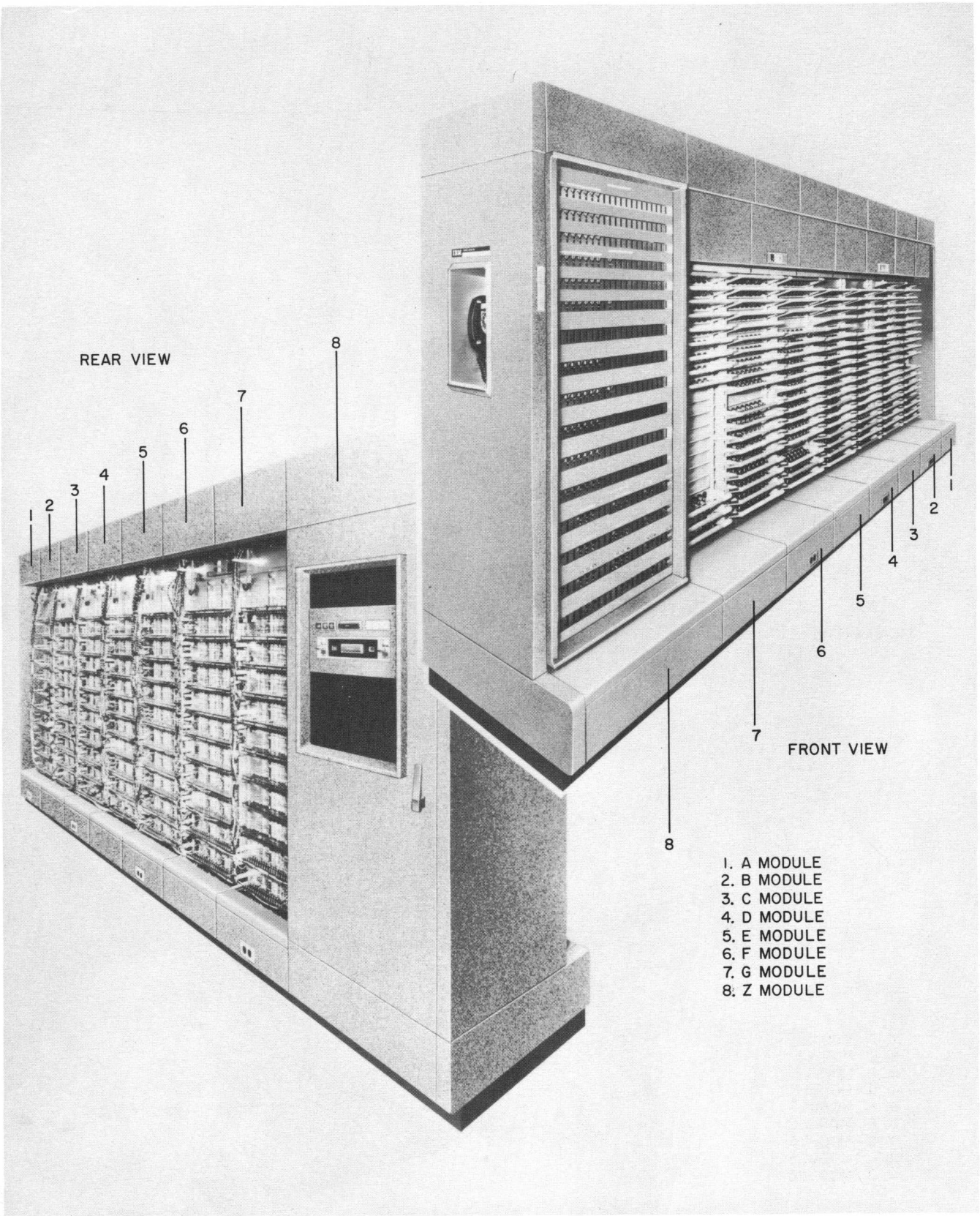


Figure 1–7. Selection Control Unit (Unit 5)

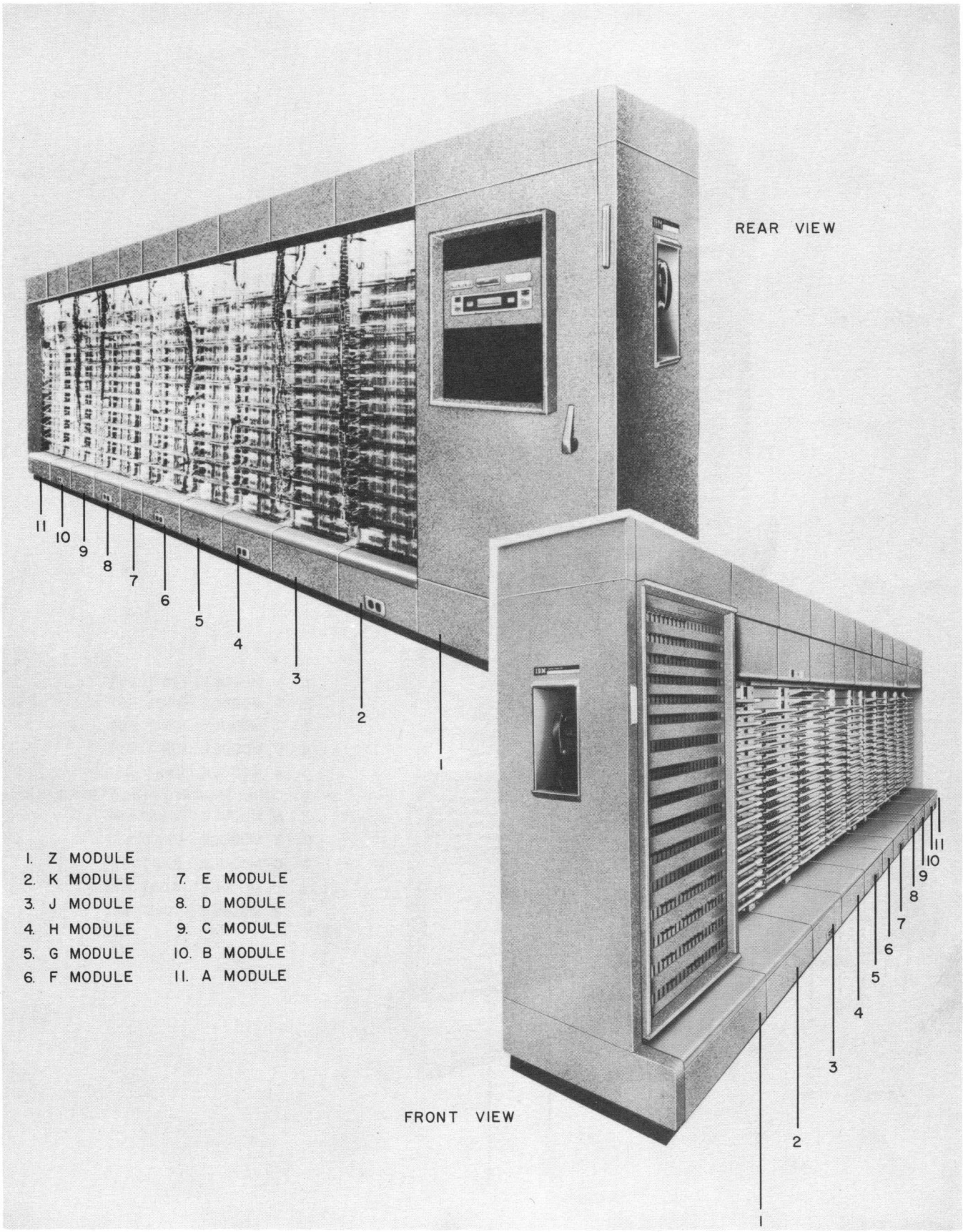


Figure 1–8. Program Unit (Unit 6)

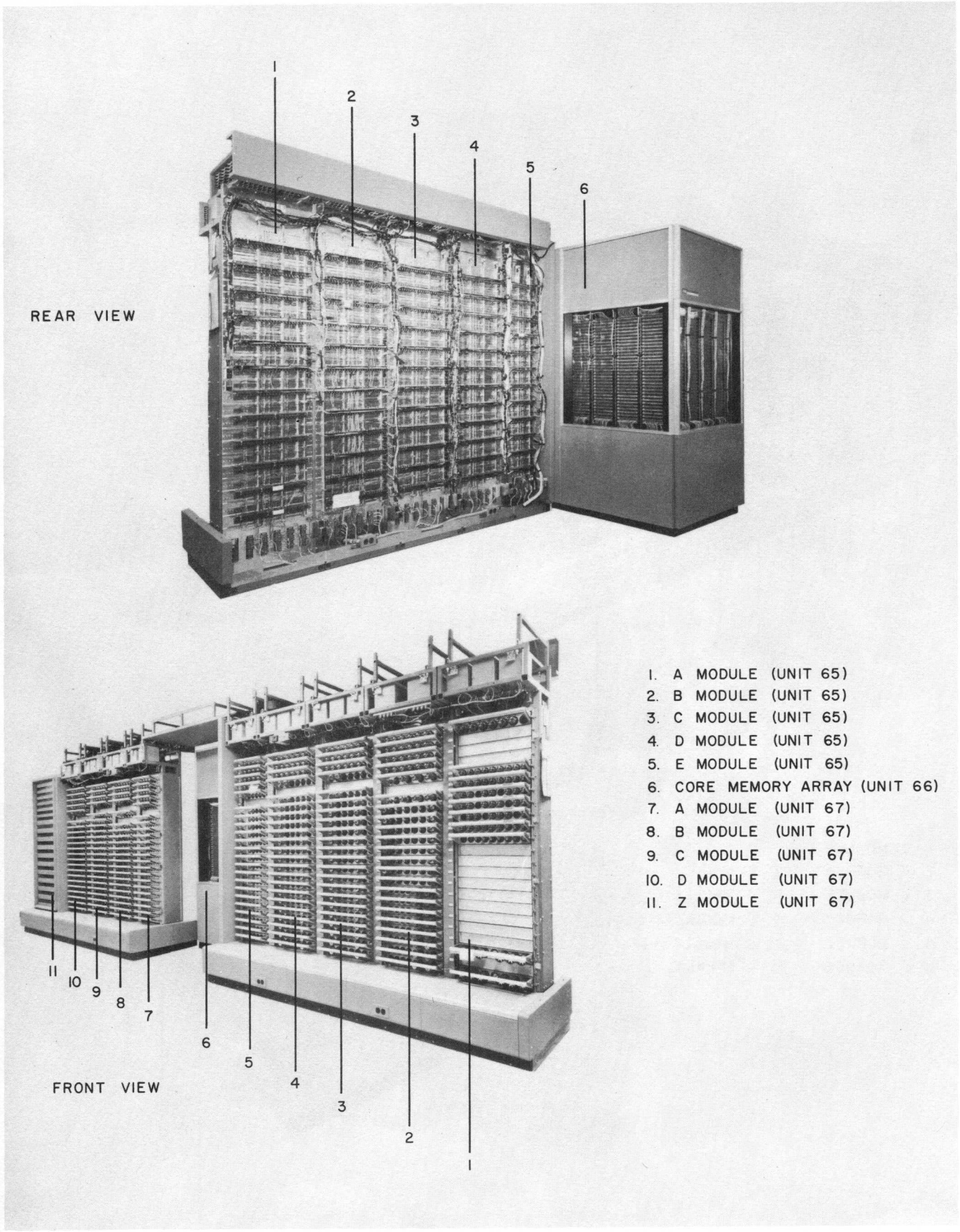


Figure 1–9. Core Memory 1 Units (Units 65, 66, and 67)

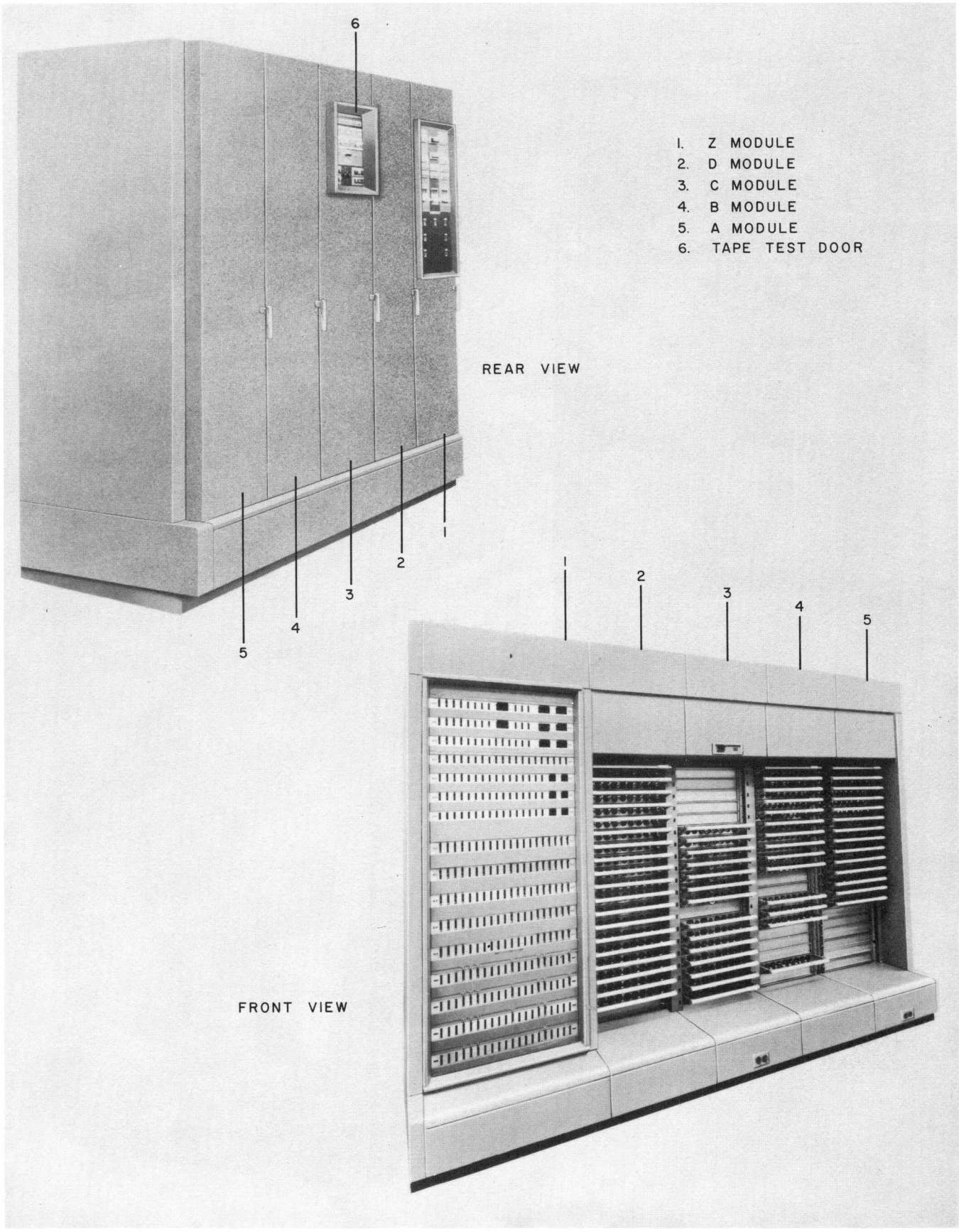


Figure 1–12. Magnetic Tape Adapter Unit (Unit 13)

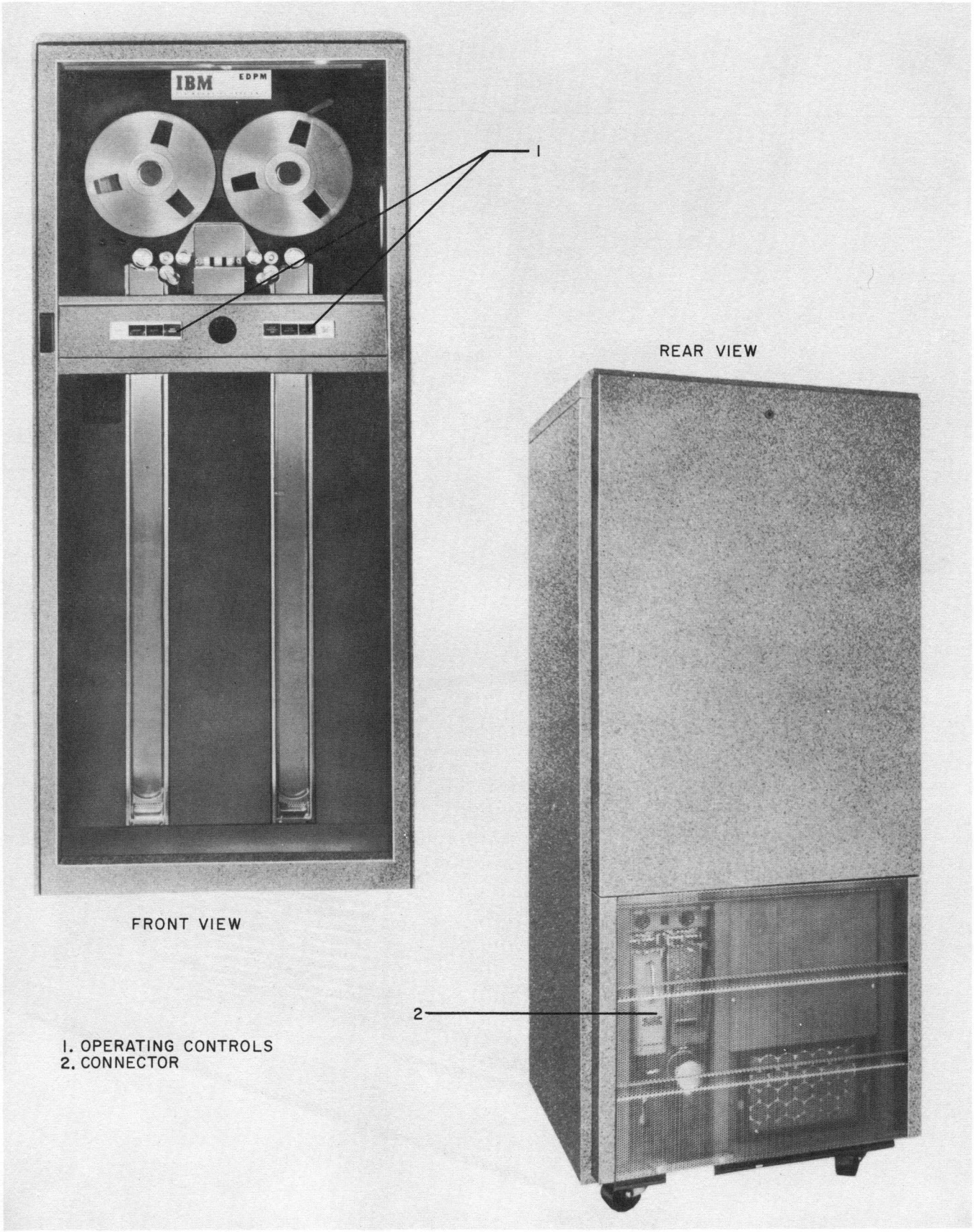
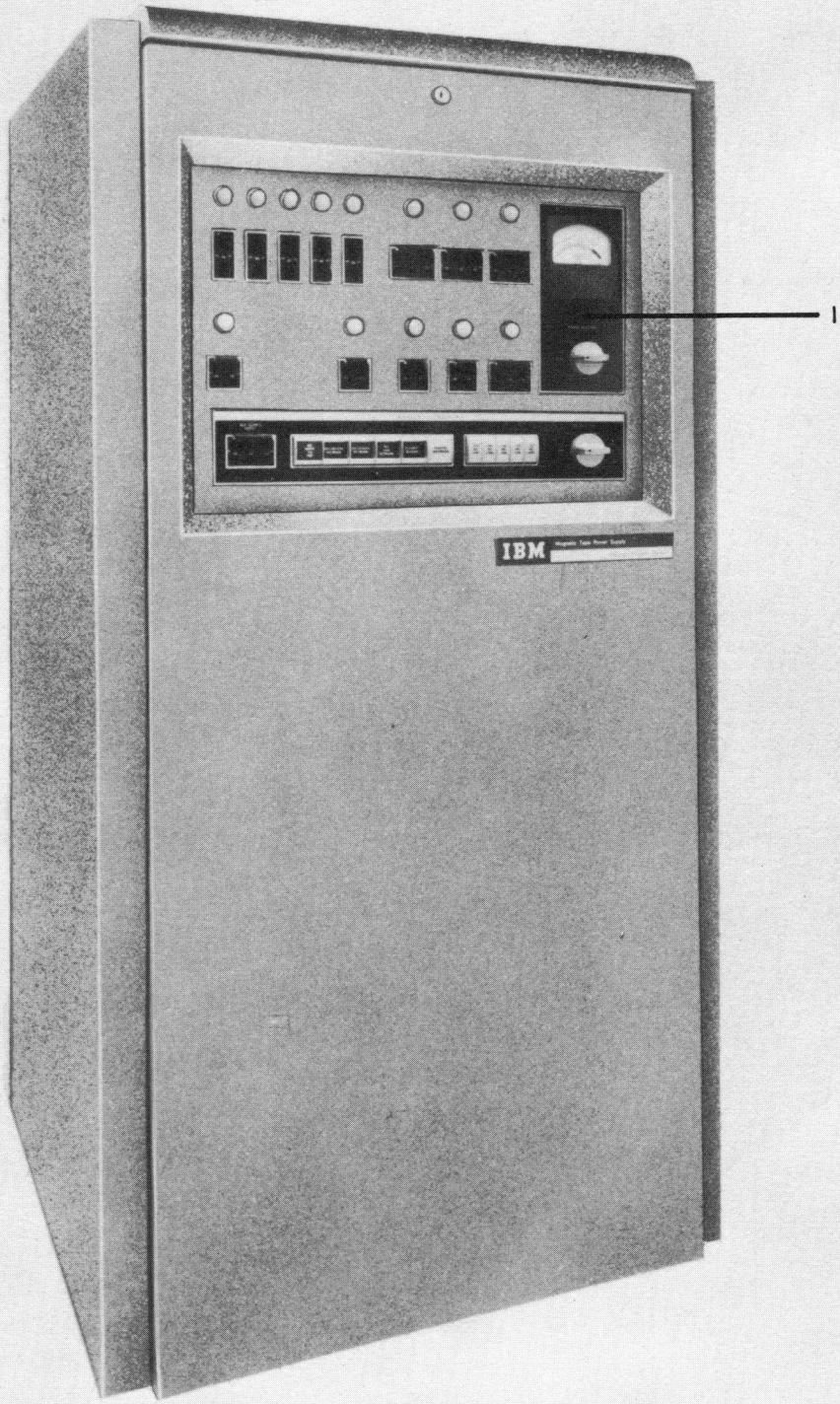


Figure 1–13. IBM 728 Magnetic Tape Drive Unit (Units 14, 15, 16, and 17)



1. OPERATING PANEL

Figure 1–14. Magnetic Tape Power Supply Unit (Unit 18)

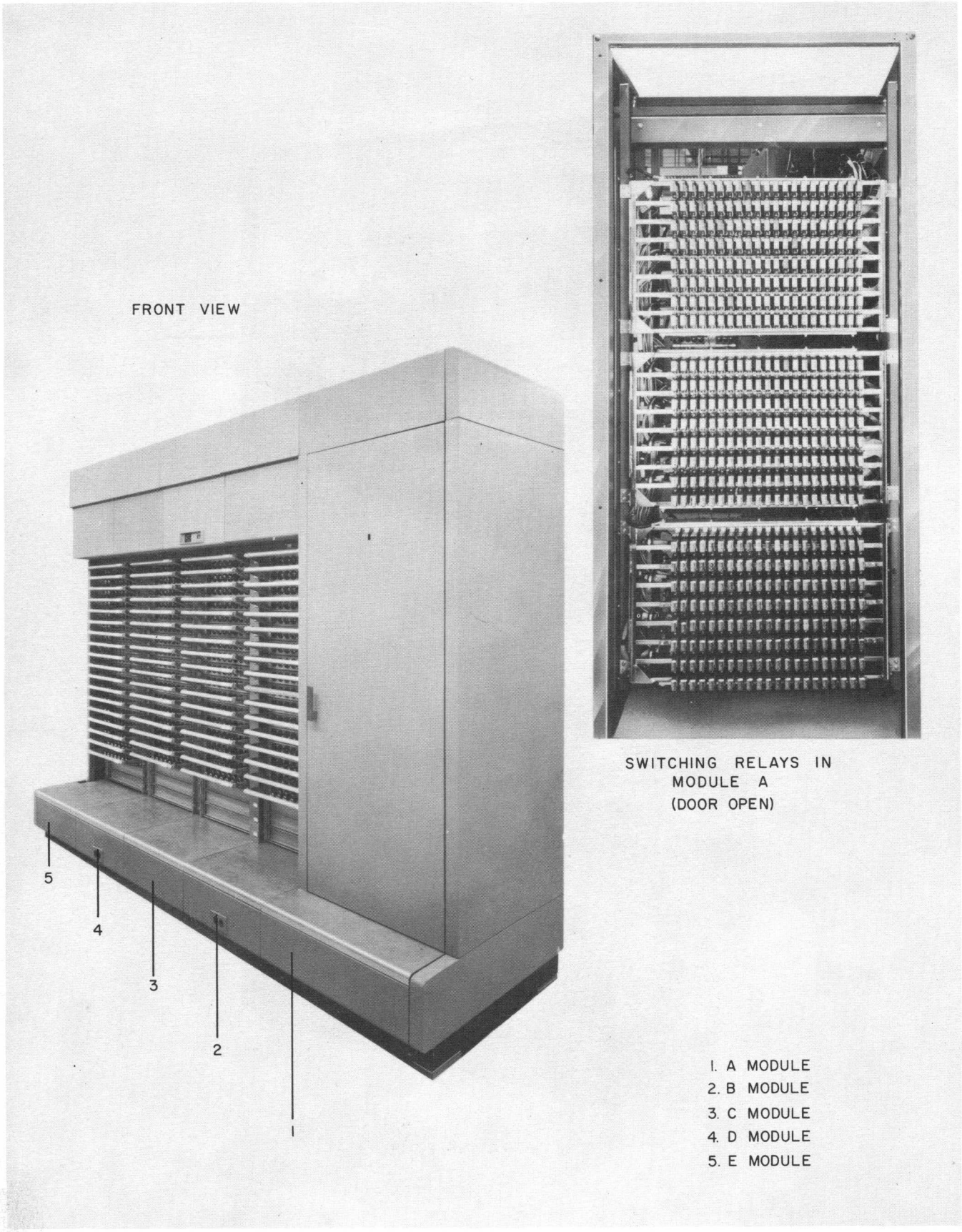


Figure 1–15. Warning Light Storage Unit (Unit 30)

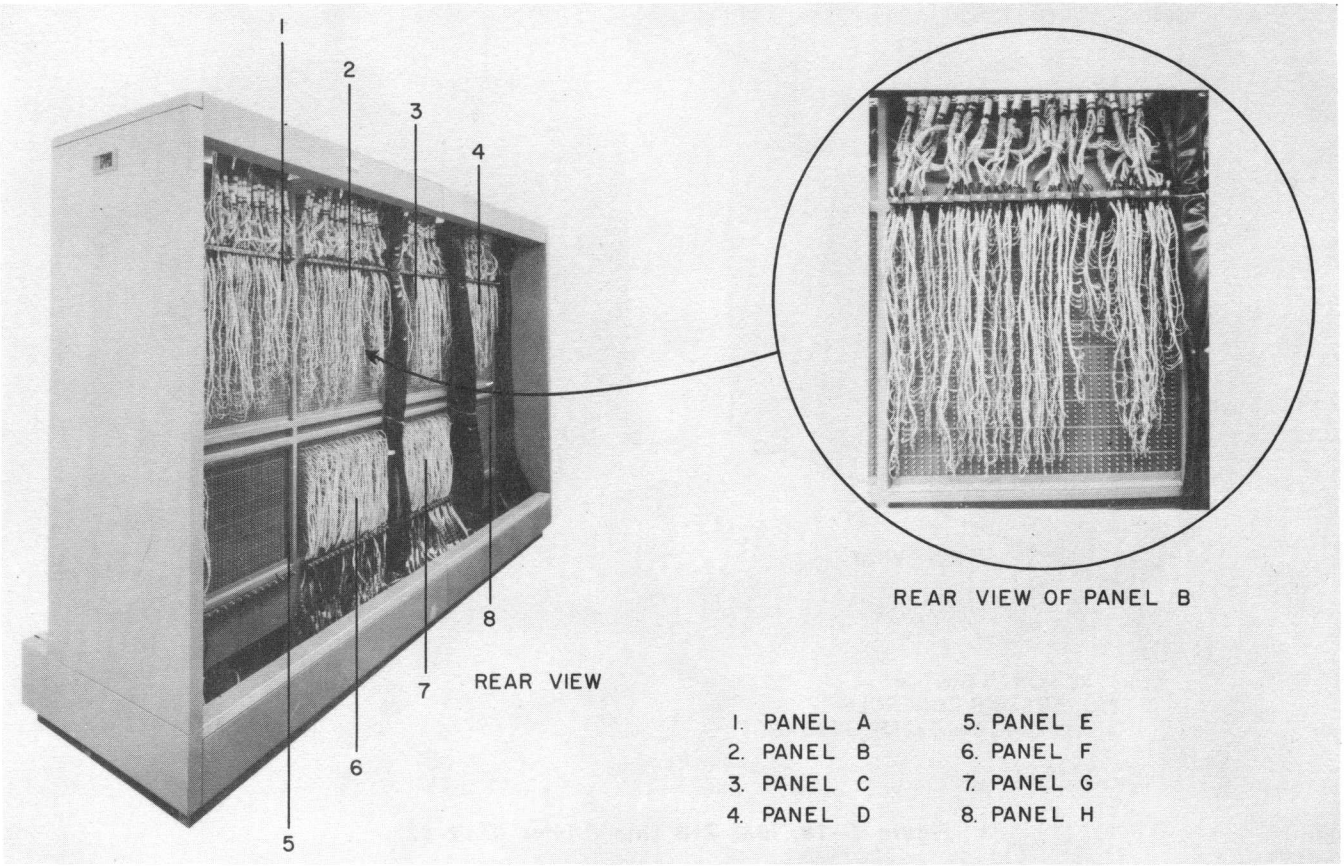


Figure 1–16. Warning Light Interconnection Unit (Unit 91)

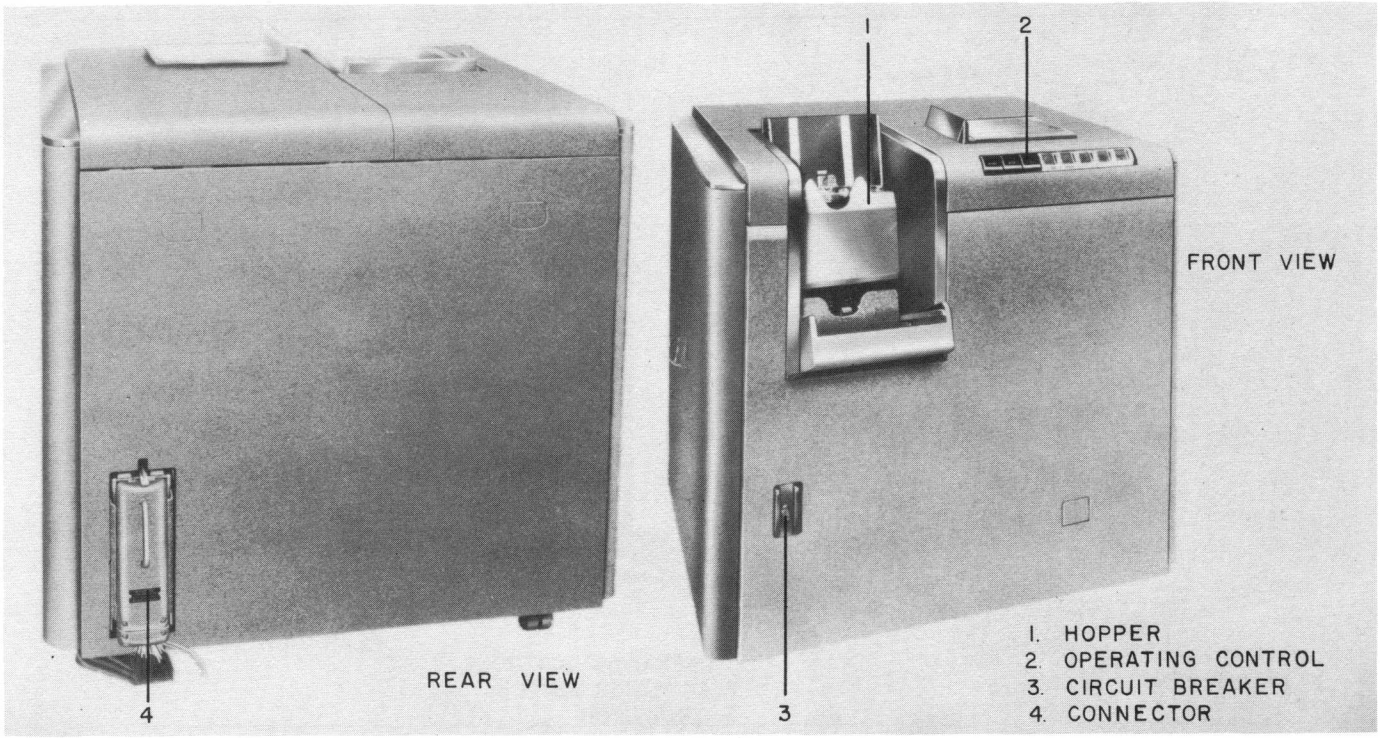


Figure 1–17. IBM 713 Card Reader (Unit 51)

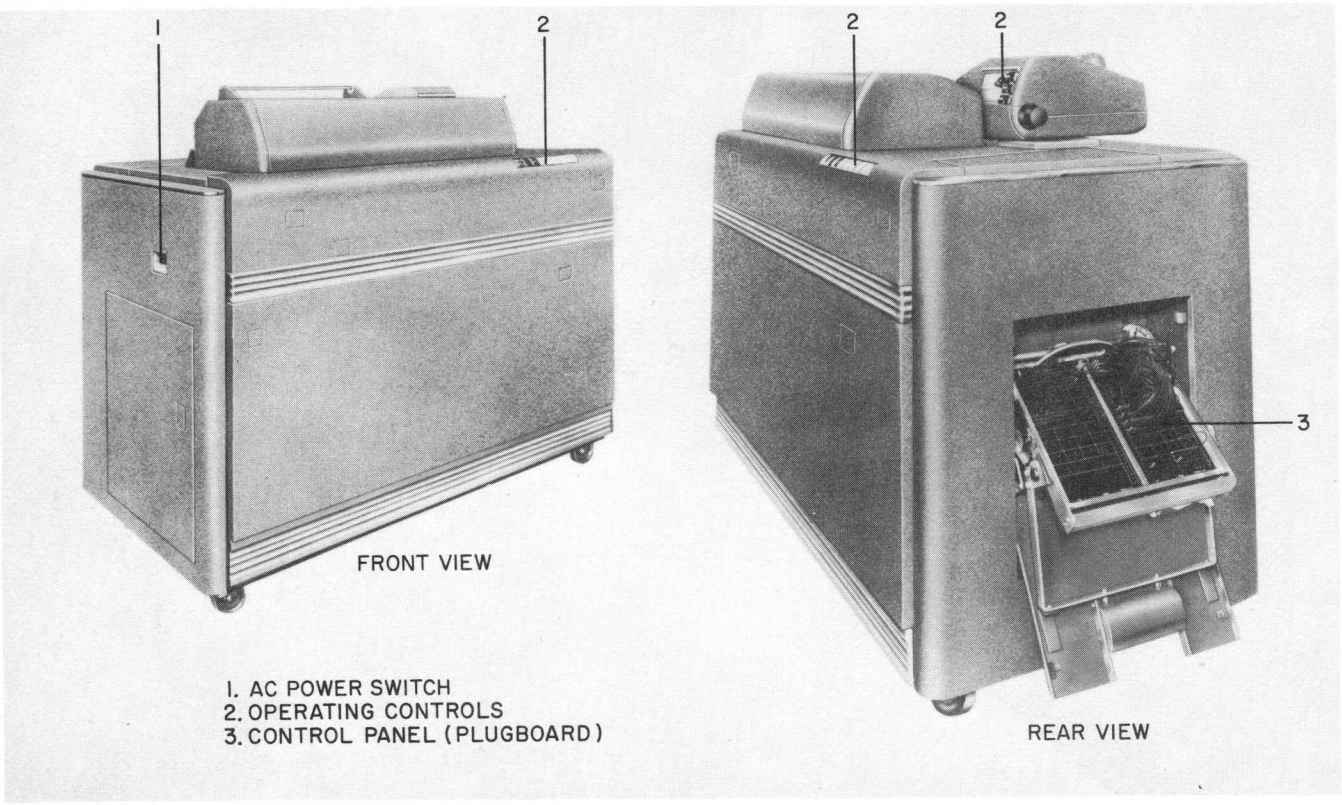


Figure 1–18. IBM 718 Line Printer (Unit 52)

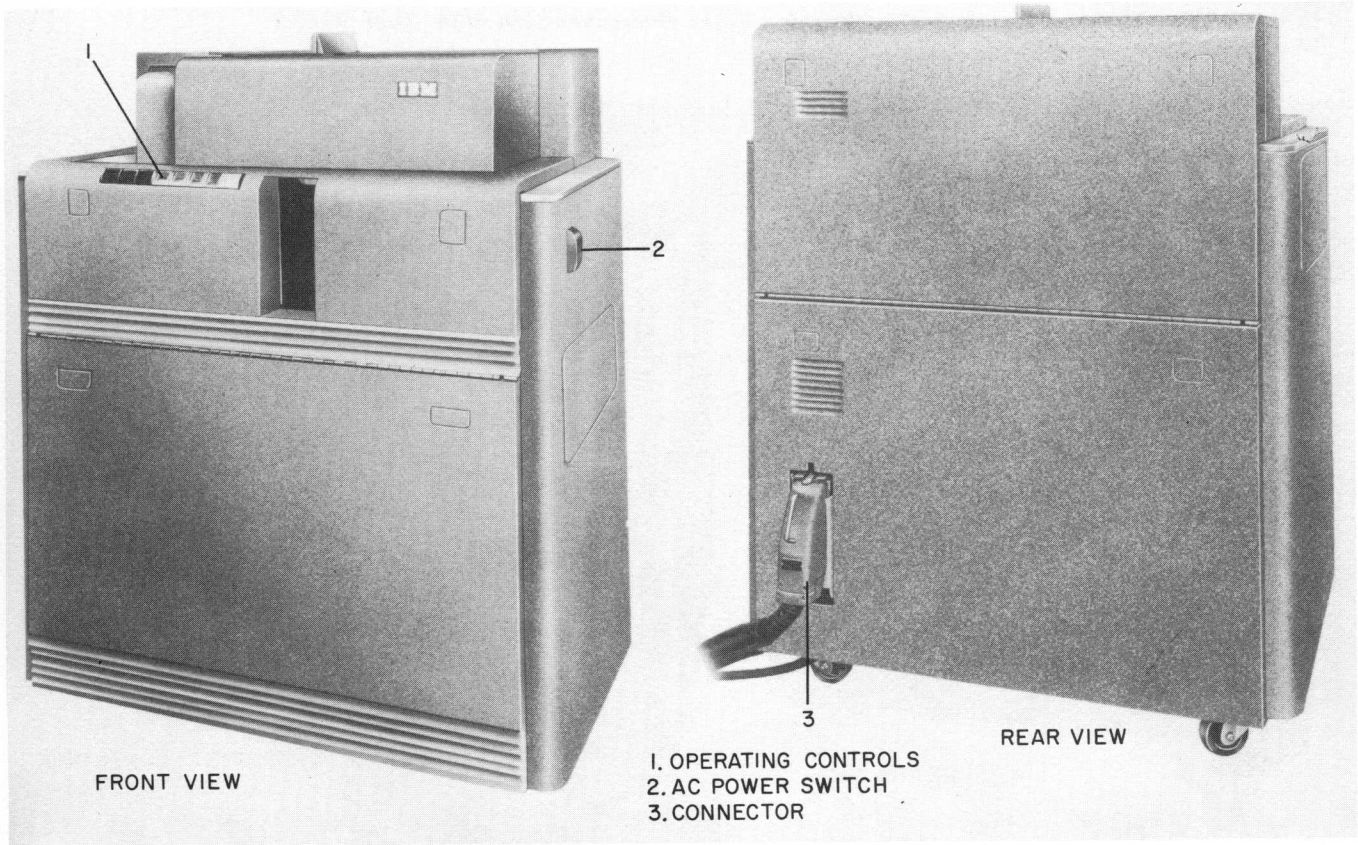


Figure 1–19. IBM 723 Card Punch (Unit 53)

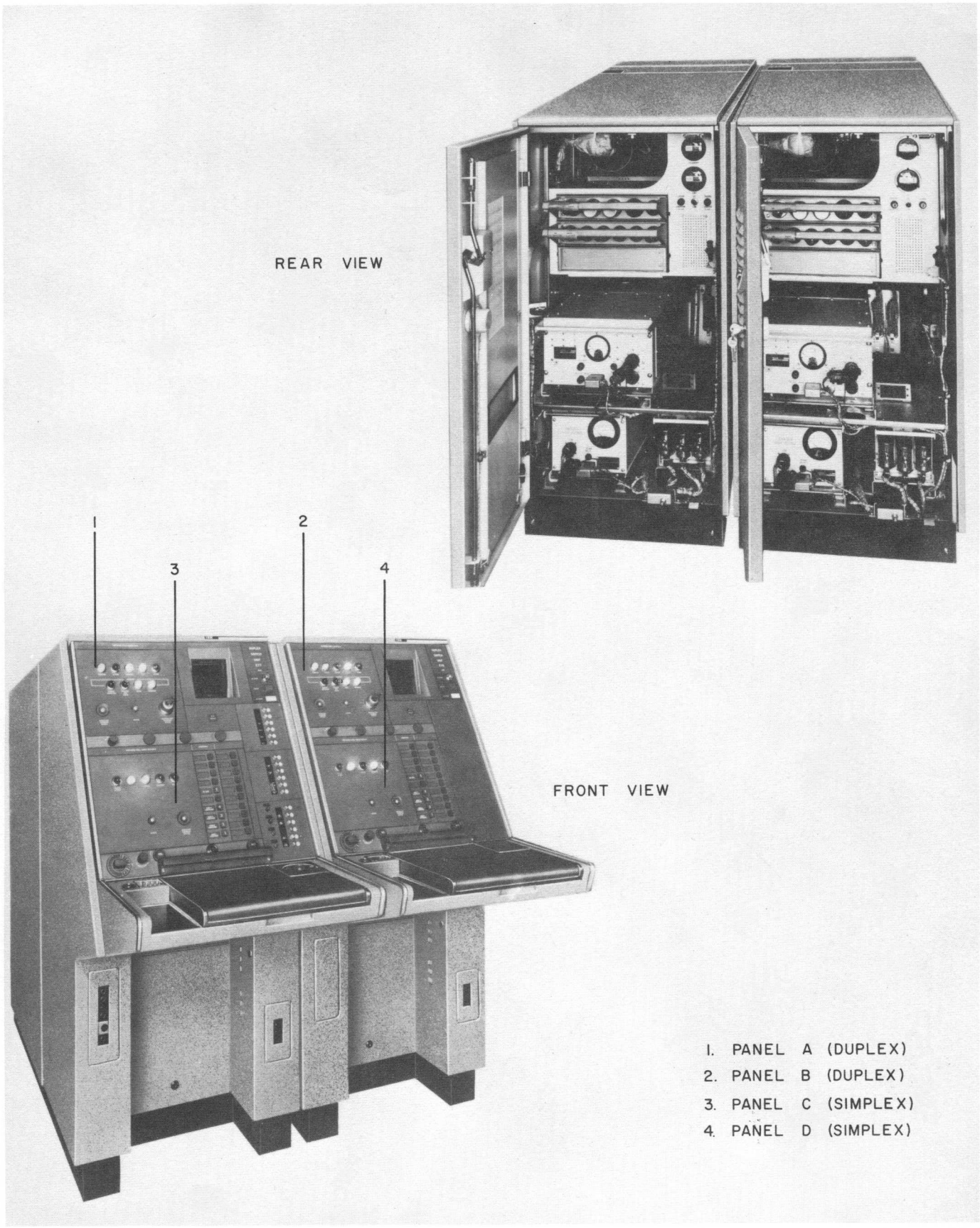


Figure 1–20. Duplex Switching Console (Unit 45)

CHAPTER 3

PRINCIPAL CHARACTERISTICS OF CENTRAL COMPUTER SYSTEM

3.1 INTRODUCTION

This chapter presents a brief discussion of the principal characteristics of the Central Computer System and, therefore, serves as an introduction to the theory of operation of the Central Computer and its associated IO devices. The principal characteristics of the Central Computer may be summarized as follows:

a. General purpose
b. Stored program
c. Single address
d. Binary internal operation
e. Parallel operation
f. Internal memory capacity
g. Memory word description
h. Internal timing
i. Real-time control

Although the material contained in this chapter is not used to define the function of the various circuits of the Central Computer, figure 1–21, which shows the functional elements of the Central Computer, is included here for reference purposes.

3.2 GENERAL-PURPOSE COMPUTER

The primary function of the Central Computer System is to process military tactical data, but the nature of this processing is such that standard digital-arithmetic computing techniques may be used. For this reason, the logic of the Central Computer (fig. 1–21) is similar to that of existing commercial computers and it may be described as a general-purpose digital computer. As such it could be used for standard commercial and scientific calculations.

3.3 STORED PROGRAM

To provide the high-speed operation necessary for processing large amounts of tactical data, the Central Computer must be provided with a list of the operations it is to perform. This list of operations, which consists of sequential instructions, is called a program and is stored within the Central Computer in an internal memory device (memory element). As used in the AN/FSQ-7 equipment, the stored program functions to direct the transfer of new tactical data from the various input devices to the memory element, to process this data and store the results in the memory element, and to direct the transfer of processed data from the memory

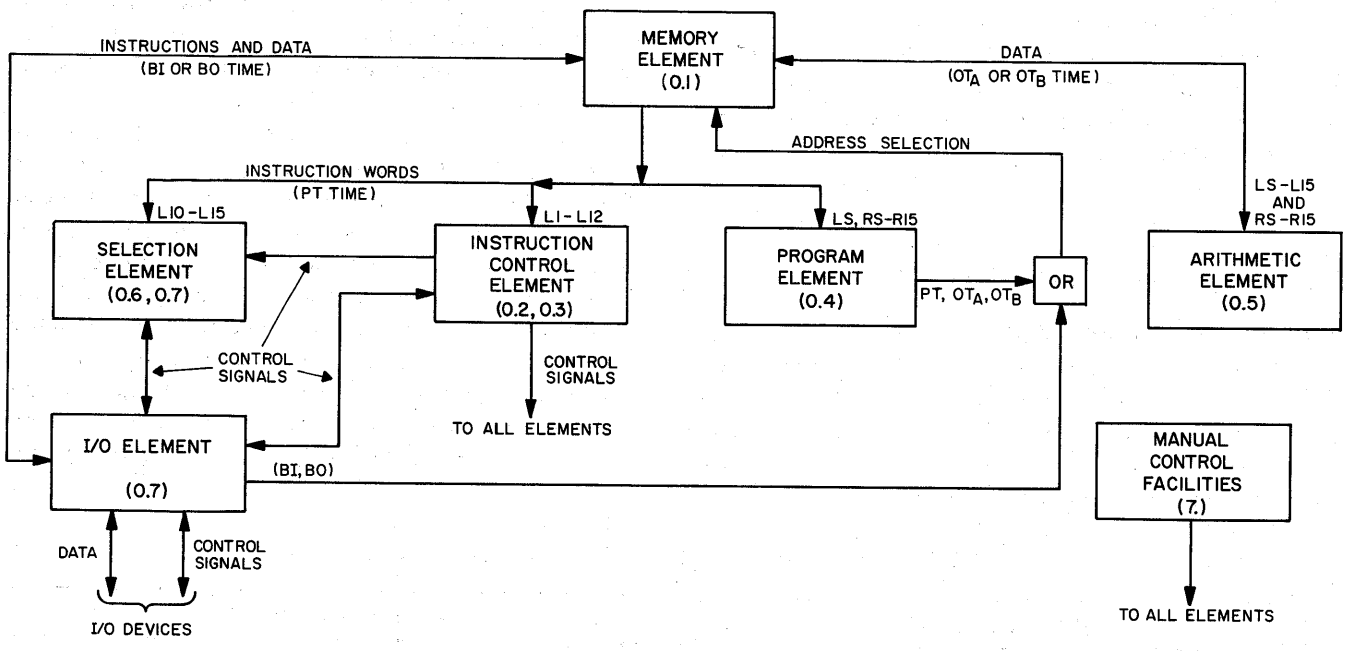


Figure 1–21. Functional Elements of Central Computer System

element to the appropriate output devices. Since the AN/FSQ-7 deals with live rather than static data, the stored program is continuously repeated to act on the constantly changing input data.

3.4 SINGLE ADDRESS

In considering the internal operation of the computer, the instructions that the program comprises and the numerical data to be processed by the program are both contained in the memory element of the computer. During the execution of the program, each instruction is in turn transferred to the computer and decoded to determine what specific operation is to be performed. Many of the instructions that the computer is capable of executing require that numerical data be obtained from a specific location in the memory element for processing. For instance, an instruction ordering the computer to add requires that the number to be added (addend) be obtained from the memory element and supplied to the arithmetic equipment (arithmetic element). In this case, the address or location in the memory element where the addend is stored is specified by the instruction itself. Since many of the program instructions specify a single location in the memory element where additional information pertinent to the instruction may be obtained, the Central Computer System is designated as a single-address computer, as differentiated from a multiple-address computer. In the latter type, each instruction specifies two or more locations in the internal memory where additional information pertinent to the instruction may be obtained.

3.5 BINARY INTERNAL OPERATION

All information utilized within the computer is in binary-coded form. This applies to the stored program instructions and to all types of numerical data. For instance, that part of the instruction word that orders the computer to add appears in the computer in binary form as 0010001. Similarly, the number $+5/8$ appears in the computer in binary form as 0.101 000 000 000 000 (the 0 before the binary point indicates that the number is positive). In binary coded form, a digit is either a 1 or a 0, and either numeral is called a bit.

3.6 PARALLEL OPERATION

All the internal operations performed by the Central Computer are performed in parallel rather than serial form. In parallel operation, multiple-bit information (memory words, data words, etc.) is transferred from one register to another by using separate lines to transfer each bit simultaneously. In serial operation, each bit of the information is transferred in sequence over one line. Similarly, parallel addition of two 16-bit words is accomplished by adding pairs of corresponding-order bits simultaneously in 16 different adders. In serial addition, each pair of corresponding-order bits is added sequentially in the same adder. Although parallel operation requires more equipment than serial operation, the increase in speed is sufficient justification for the added equipment.

3.7 STORAGE CAPACITY

An important factor in the ability of the computer to process information at high speed is the relatively large quantity of data which may be stored internally. Data storage is accomplished by the memory element (fig. 1–21), which is composed of two core memories and a test memory device. The largest of these, core memory 1, has a storage capacity of 65,536 memory words. Core memory 2 has a storage capacity of ~~4,096~~ memory words, and the test memory has a storage capacity of 16 memory words. Thus, the total storage capacity of the memory element is ~~69,648~~ memory words.

In addition to containing a large amount of internal storage, the memory element of the computer is very flexible in that random access (random memory address selection) is possible. The memory devices of the memory element are classed as high-speed devices since memory words can be stored in, or obtained from, the element at a rate of one word per 6.0 μsec.

3.8 MEMORY WORD DESCRIPTION

The unit of information stored in the internal memory of the Central Computer is a 32-bit word composed of two 16-bit half-words. The two half-words are distinguished from each other by the terms right half-word and left half-word. The symbolic layout of the 32-bit memory word is shown in table 1–2.

As noted in the table, a 16-bit half-word consists of a sign (S) bit and 15 bits numbered from 1 through 15. The sign bit has significance only if the half-word contains numerical information. By definition, if the sign bit is 0, the number contained in bits 1 through 15 is positive and in true binary form; if the sign bit is 1, the number is negative and is in 1's complement form. When referring to a single bit in a 32-bit memory word, an L or an R is prefixed to the bit number to indicate the half-word in which it is located. For example, L8 refers to the 8th bit of the left half-word; R14 refers to the 14th bit of the right half-word.

TABLE 1–2. COMPOSITION OF A MEMORY WORD

LEFT HALF-WORD																RIGHT HALF-WORD															
S	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	S	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Since the internal memory of the Central Computer is used for storage of both instruction words and data words, the character of each type of word is discussed separately under the following subheadings.

3.8.1 Instruction Words

An instruction word is a 32-bit memory word containing a binary-coded order to the computer. When the order is decoded and acted upon, it causes the computer to perform a designated operation. Because of their functions, bits L1 through L15 of the instruction words are termed the operation portion and bits LS and RS through R15 are termed the address portion. The operation portion of the instruction contains a binary-coded instruction, or order, specifying the operation the computer is to perform. The 15 bits which compose the code may be divided into four groups, each having its own operational significance (fig. 1–21). The manner in which the operation portion bits are grouped and the names assigned to these groups are shown in table 1–3.

Bits L1 through L3 of the instruction word are termed the index register selection bits since they are used to specify whether one of the five index registers of the Central Computer will be used with the instruction. The instruction itself determines whether the index register is to be controlled or whether it will exercise control. If the instruction calls for exercising control on the selected index register, pertinent information will be contained in the other portions of the instruction word. If the instruction utilizes the index register to exercise control, the process is termed indexing and results in modifying the address portion of the instruction word.

The Central Computer System of the AN/FSQ-7 equipment is capable of executing 59 instructions, which are grouped into 8 distinct classes. The instructions within a class are designated as variations of that class. (A complete listing of these instructions together with a statement of their function is given in Ch 5 of Part 1.) Bits L4 through L6 of the instruction word are used to indicate the class within which a specified instruction falls. For all instructions except the *Branch and Index, Branch and Sense, Operate, Test One Bit, Test Two Bits, Select,* and *Select Drums* instructions, bits L7 through L12 of the instruction word are used to specify which particular instruction within a class is selected. For the *Branch and Index, Branch and Sense, Operate, Test One Bit, Test Two Bits, Select,* and *Select Drums* instructions, bits L7 through L9 of the instruction word fully specify these instructions as variations of their particular class. Bits L10 through L15 further specify which of the numerous pieces of equipment under control of the particular instruction is to be operated upon, or what type of action, if any, is to be effected on particular pieces of equipment.

It should be noted that bits L10 through L12 function as either a portion of the variation bits of the operation code or as a portion of the index interval bits. However, this causes no ambiguity or conflict in the computer logic, since, by design, those instructions which utilize the index interval bits do not require bits for their complete identification. Thus, when the index interval is utilized, the appropriate instruction is completely specified by only six bits of the operation code instead of the usual nine bits.

Bits L13, L14, and L15 of the index interval also have a double function: besides being used to form an integral part of the *Branch and Index, Branch and Sense, Operate, Test One Bit, Test Two Bits, Select,* and *Select Drums* instructions, they are used to exercise alarm control on all instructions which can cause an arithmetic overflow. When these bits are properly coded, an alarm will sound if an overflow occurs in either of the two arithmetic units. Further action may be taken if an overflow occurs, but this action is determined by control switch settings which are effective when an overflow occurs.

As previously stated, many of the instructions that the computer is capable of executing require data (an operand) for their execution and this operand must be located in the memory elements of the Central Computer. For this type of instruction, the address portion of this instruction word is used to specify in which of three memory devices and at what address location within the selected devices the operand is stored. For example, when the computer is to execute an *Add* instruction, the augend (the number to be supplemented) is already contained in the arithmetic registers, whereas the addend (the number to be added) is stored in one of the internal memory devices. It is the purpose of the

TABLE 1–3. BIT DESIGNATION OF OPERATION PORTION OF INSTRUCTION WORD

L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15
											Index Interval			
Index Register Selection			Class			Variation								
Operation Code														

address portion of the *Add* instruction to specify the location of the addend. The operation portion of the *Add* instruction, when decoded, causes the addend to be obtained from the internal memory device and to be added to the augend.

The location of a memory word in one of the three internal storage devices of the memory element is designated by the 17 bits of the address portion of the instruction word shown in table 1–4. Since the capacities of each of the devices are different, specific groups of bits (table 1–4) are used to designate an address in each of the devices; that is, bits R12 through R15 specify an address in test memory, bits R4 through R15 specify an address in core memory 2, and bits RS through R15 specify an address in core memory 1. In operations requiring an operand, all three of the memory address registers (located in the memory element) are loaded with the address information; however, only one of these devices is activated to read out the information contained in the selected address. Information to determine which of the three memory devices is to be activated is contained in bits LS and RS through R11 of the instruction word. If bit LS contains a 0, core memory 1 is activated; if bit LS contains a 1, bits RS through R11 are examined to determine whether core memory 2 or test memory should be activated. In the latter case, if bits RS through R11 all contain 1's, test memory is specified; if any of these bits contain a 0, core memory 2 is specified.

Although the above discussion was based on the premise that the address portion of an instruction specifies a location in an internal storage device from which an operand may be obtained for use in the computer, it applies equally to the process of storing the results of calculations in the selected internal storage device. The address portion of an instruction is used only to designate a specific location in an internal storage device; the direction of information transfer between the memory and the computer depends upon the operation portion of the instruction with which the address portion is associated.

Many of the instructions which do not require an operand make use of the address portion of the instruction word to store pertinent information required for their execution. Generally, the address portion of such instructions contains numerical data that is to be loaded into one of the computer registers so that the latter may control the performance of the instruction. For example, if the computer is to execute any of the shift-class instructions, the address portion of the instruction word contains a number specifying the number of times the shift operation is to be repeated. As a further example, some of the IO-class instructions utilize the address portion of the instruction word to store a number which, when loaded into the proper register, is used to coordinate the transfer of data between the computer and one of the various IO devices. Other instructions do not require the use of the address portion of the instruction word; in such cases, the information contained in that portion of the word is meaningless.

3.8.2 Data Words

As used in the AN/FSQ-7 equipment, the basic unit of numerical data consists of 16 bits. Thus, a 32-bit memory word contains two binary-coded numbers, one in the left half-word and one in the right half-word, each having a word length of 16 bits. Each 16-bit word consists of a sign bit and 15 data bits.

Since many operations used in the air defense program involve identical calculations with two related quantities, the two half-words composing a full 32-bit memory word are used to represent these two related quantities. The arithmetic element, which performs arithmetic operations on these numbers, is designed for dual arithmetic operation to simultaneously process both the right and left half-words in an identical manner. Except in a few instructions, the dual sections of the arithmetic element are controlled together and must perform the same operation on both 16-bit half-words.

All numerical data in the computer has an absolute magnitude less than unity; i.e., a number, X, may be anywhere in the range where X is less than $+1$ and greater than -1 ($-1 < X < +1$). Therefore, the binary point (corresponding to the decimal point in decimal or denary arithmetic) is located between the sign bit and the first data bit. If the sign bit of a data word is 0, the number is positive and is in true binary notation; on the other hand, if the sign bit of a data word is 1, the number is negative and is in the 1's complement form. Since the mathematical capacity of the computer must never be exceeded, it is necessary that all numerical data fed to it be scaled down to the range where the absolute magnitude of the data is less than unity. Thus, if the

TABLE 1–4. BIT DESIGNATION OF ADDRESS PORTION OF INSTRUCTION WORD

LS	RS	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15
Memory Unit Selector Bits													Test Memory Address Bits			
Core Memory 2 Address Bits																
Core Memory 1 Address Bits																

multiplication 256 x 33 is to be performed, the numbers are loaded into the computer as 0.256 and 0.033 (that is, scaled down by a factor of 1/1000). The result of the multiplication is 0.008448, which the equipment using the results interprets as differing from the true product by a factor of 1/1,000,000. By this means, and under normal circumstances, numbers greater in magnitude than unity are prevented from occurring in the computer.

A special condition exists when an instruction is used to arithmetically modify the 17-bit address portion of another instruction. Since the 17-bit address is a positive whole number, the sign bits and binary points, usually associated with data, have no significance. Further, the dual arithmetic units which normally function independently to process two 16-bit data words are connected together to arithmetically process a single 17-bit number. The control option to provide for 17-bit arithmetic operation is provided by bit L12 of the instruction being executed.

3.9 COMPUTER TIMING

The interval of time required to transfer a word into or out of the memory element is 6.0 μsec and is called a memory cycle. When referenced to the computer, memory cycles are termed machine cycles to distinguish them from internal operations associated with the memory element. In addition to memory and machine cycles, operation of the computer may be discussed in terms of instruction cycles. An instruction cycle is defined as the time required for the computer to execute one instruction and is usually composed of from one to three memory cycles, each of which performs a particular function. The reason for this becomes apparent when the characteristics of the various instructions associated with the computer are considered. Of the 59 instructions used in AN/FSQ-7 Combat Direction Central, 11 involve simple operations such as setting up control circuits or transferring words between two registers. These operations may be completed in 6.0 μsec or less. Because of their simplicity of operation, these instructions can be obtained from the memory element, decoded, and executed in one memory cycle, and are called 1-memory-cycle instructions. However, other instructions require that data (an operand) be obtained from the memory element before they can be completed. There are 29 instructions of this type, called 2-memory-cycle instructions. These instructions require a second memory cycle during which the memory element is activated, the operand obtained from memory, and the instruction completed. There are six instructions which require three memory cycles for completion. In these, the instruction is obtained and decoded during the first memory cycle, the information contained in the specified memory location is obtained during the second memory cycle, and a new word is stored in the memory element during the third memory cycle. The memory cycles which compose an instruction cycle have been assigned distinctive names for easy reference. The names and characteristics of these are listed in table 1–5.

Within a machine cycle, computer operations are synchronized and identified by timing pulses (TP's). These pulses are not used directly to execute instructions, but serve as basic timing to synchronize, control, and co-ordinate all computer operations. The timing pulses control the generation of instruction pulses (IP's), breakout (BO) pulses, or break-in (BI) pulses, as required by the computer operations. The instruction pulses are used to generate the command required to execute the operations specified by the 59 instructions. The BO pulses are used to effect the transfer of information from the memory element to a selected IO unit; the BI pulses are used to effect the transfer of information from a selected IO unit to the memory element. Instruction, BO, and BI pulses cannot be generated simultaneously because only one type of machine cycle can be performed at one time. During instruction execution, instruction pulses are generated to provide for internal machine cycles (PT, OTA, or OTB). However, during transfer of data between the memory element and IO devices, BI or BO pulses are generated to provide for IO machine cycles (BI or BO). In either case, the machine cycle begins with TP 0 time and ends with TP 11.

Although an internal machine cycle begins at TP 0 time, an instruction cycle starts with TP 7 of a program time cycle, which is denoted PT 7. The instruction cycle does not start until PT 7 because, during the time interval of PT 0 to PT 7, the coded instruction word is being transferred from the memory element to the other elements of the computer. This is indicated by the chart in figure 1–22, which shows the basic internal machine and instruction cycles. In this chart, the internal machine cycles required by an instruction cycle are shown

TABLE 1–5. FUNCTION OF MEMORY (MACHINE) CYCLES OF AN INSTRUCTION

MEMORY CYCLE	TITLE	FUNCTION
First	Program Time (PT)	Decodes instruction and initiates execution.
Second	Operate Time A (OTA)	Obtains operand and performs operation.
Third	Operate Time B (OTB)	Stores result of operation in core memory.

as cross-hatched areas. Thus, in a 1-memory-cycle instruction, the decoding process starts at PT 7, and the instruction is completed by PT 6 of the subsequent PT cycle, at which time decoding of the next instruction is initiated. Similarly, a 2-memory-cycle instruction starts at PT 7, continues through the subsequent OTA cycle, and is completed by PT 6 of the next PT cycle, at which time the next instruction will begin to be executed.

An exception to the usual sequence of internal machine cycles occurs in the case of 13 of the 59 instructions associated with AN/FSQ-7 Combat Direction Central operations. In these instructions, additional time is required to perform a series of repetitive operations: multiplication, which requires repetitive addition; division, which requires repetitive subtraction; or shifting, which may require any number of repetitive shifts. This additional time is supplied by temporarily stopping the generation of TP and IP pulses (instruction control element of fig. 1–21), which also stops the operation of instruction and machine-cycle sequencing. This additional time is called an arithmetic pause, since the computer pauses in its usual sequential operation long enough to complete the repetitive operations. An example of an instruction cycle utilizing an arithmetic pause is shown in figure 1–22 (2-memory-cycle plus pause). This cycle configuration is typical for the multiplication and divide instructions. As illustrated, decoding of the instruction begins at PT 7; the operand, which, in this case, is either the multiplicand or the divisor, is obtained during the subsequent OTA cycle. At the end of this cycle, the generation of time and instruction pulses is stopped, and the computer goes into an arithmetic pause during which the repetitive operations (addition or subtraction) which constitute a multiplication or division operation are executed. When the pause condition is terminated (automatically), the next PT cycle begins and the multiply or divide instruction being executed is completed by PT 6 of that PT cycle.

The operations which take place in the computer during the memory and internal machine cycles are illustrated in figure 1–23. As noted in the figure, the time interval from TP 0 to TP 6 of a PT cycle is used to complete an instruction previously begun and to bring the next instruction out of the memory element. The new instruction is placed in the computer control registers (operation register and address register) at PT 7.

In addition to the internal machine cycles just discussed, two other operational cycles are used in the computer. These are associated with IO operations and the transfer of information into and out of the memory element (BI and BO cycles). A BI cycle is initiated in the computer whenever a word is to be transferred from one of the IO devices to the memory element for storage. A BI cycle is initiated as a result of a request from the IO unit (break request) and occurs at the end of any one of the machine cycles. For example, consider the cycle sequencing if a break request is received during the first PT cycle of a 2-memory-cycle instruction. (Refer to figs. 1–22 and 1–23.) During the first PT cycle, a break request is received by the computer. At the end of this PT cycle, and before the initiation of the succeeding OTA cycle, a BI cycle is executed. This BI cycle is 6.0 μsec in duration. During the BI cycle, the word received from the selected IO unit is transferred into the memory element. After the BI cycle has been completed, the OTA cycle is initiated and the program continues. Thus, a BI cycle has no effect upon the execution of an instruction or the execution of the program other than to delay it.

A BO cycle is similar to the BI cycle except that information is obtained from the memory element and transferred to a selected IO unit. Thus, if a break re-

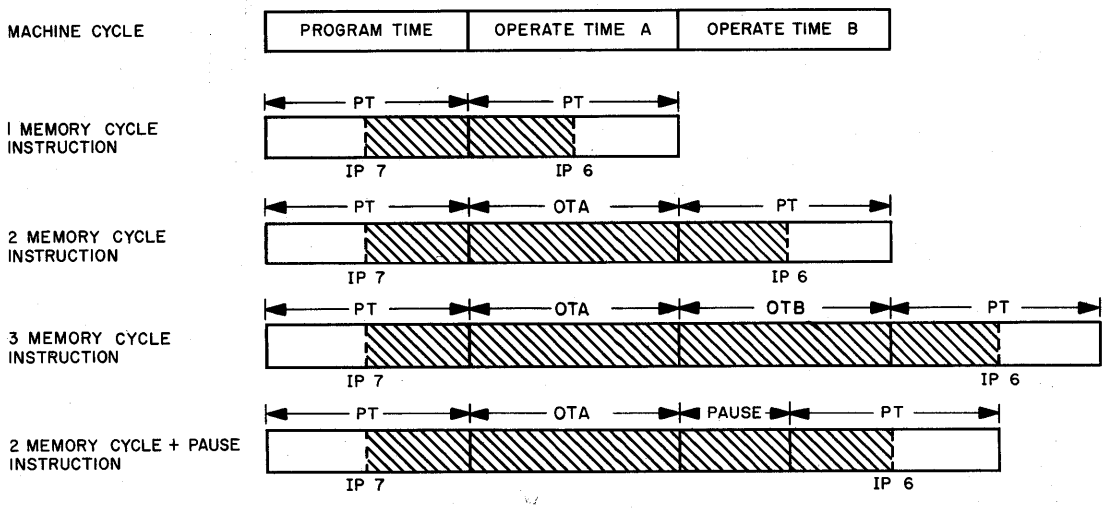


Figure 1–22. Machine and Instruction Cycles

CHAPTER 4

GENERAL THEORY OF OPERATION OF CENTRAL COMPUTER SYSTEM

4.1 INTRODUCTION

This chapter defines the function of each of the Central Computer elements and provides a brief analysis of how the major circuit groups within each element operate to produce the desired results. The amount of detail provided in this chapter is purposely limited, since this material is an introduction to the detailed theory of the computer elements presented in Parts 2 through 8.

As illustrated in figure 1–21, the Central Computer System is divided into seven functional elements:

a. Memory element
b. Instruction control element
c. Program element
d. Arithmetic element
e. Selection element
f. IO element
g. Manual control facilities

The memory element of the Central Computer is a passive, large-capacity, random-access, high-speed storage facility which is operated at a rate of 1 memory cycle per 6.0 μsec. During the execution of a memory cycle, a memory word of 32 bits (plus parity) can be either stored into or read out of the memory location specified by the computer. In normal operation, the memory element is used to store the operating program (sequential list of instructions) and the input data that is to be processed by the program. This information is initially transferred to the memory element from one of the various input units. During the execution of the program, which directs all internal operations of the computer, the input data is transferred to the arithmetic element for processing, and the results of the arithmetic operation are subsequently stored back into memory. During further execution of the program, the processed data is transferred from the memory element to one of the various output units and new input data is obtained for processing. The stored program is then repeated to process the new input data.

The instruction control element is the internal control center of the computer since it contains the master-timing and control circuits which are necessary to sequence, co-ordinate, and control all internal operations required to execute the stored program. In performing its function, the instruction control element decodes the individual program instructions when they are read out of the memory element so that the required control signals will be generated at the proper time to perform the specified operations. The control signals generated by this element are distributed to all operating elements of the computer. If the instruction being executed requires that data be transferred between the memory element and the arithmetic element, control signals will be generated to set up the memory element circuits and to initiate the required memory cycle (OTA or OTB). When an instruction has been completed, additional control signals are generated to initiate the transfer of the next program instruction from the memory element (PT cycle) to the control circuits of the instruction control element.

The program element of the Central Computer contains the addressing registers which are used to specify the memory address associated with the subsequent internal machine cycles (PT, OTA, and OTB). Since the instruction control element determines the sequence of internal machine cycles required for the execution of the stored program, control signals generated by the latter element direct the transfer of the memory address information to the memory element at the beginning of each memory cycle.

The arithmetic element of the Central Computer contains the circuits required for the arithmetic manipulation of data. During the execution of an arithmetic instruction, the arithmetic circuits are actuated by control signals (generated in the instruction control element) to provide for the addition, subtraction, multiplication, division, or comparison of data words. The final results of these arithmetic operations are contained in the arithmetic element. Execution of subsequent instructions of the stored program will determine whether these results will be further processed or stored in the memory element.

The selection element of the Central Computer operates in conjunction with the instruction control element to perform a unique function during the execution of 6 of the 59 instructions that the computer is capable of executing. These 6 instructions have one thing in common: each can perform an operation on any of an associated group of similar circuits. During the execution of these instructions, the selection element specifies which one of a group of similar circuits will be operated on by the control signals generated by the instruction control element.

The IO element of the Central Computer synchronizes and controls the transfer of information (instruction or data words) between the memory element and the various synchronous, relatively slow-speed, IO devices associated with the Central Computer. Since the transfer of such information must be initiated by the stored program instructions, the instruction control element, in executing the required instructions, will generate control signals to select a specific IO device, specify the number of words to be transferred, specify the locations in the memory element from which or into which the information is to be transferred, and start the selected device. Once started, the selected IO device informs the computer whenever any IO word transfer is to take place so that the computer can initiate the appropriate BI or BO cycle to transfer data into or out of the memory element. This process is repeated until the proper number of IO words have been transferred, at which time the IO operation is terminated. Since the Central Computer is associated with relatively slow IO devices, the computer controls have been designed so that the computer continues the execution of the stored program after setting up the IO control circuits and initiating an IO operation. As a result, the BI or BO machine cycles, which accomplish the IO word transfers to or from the memory element, are actually interleaved with the internal machine cycles. The BI and BO machine cycles can, therefore, be considered as providing 6.0-μsec breaks in the computer execution of the stored program.

The manual control facilities of the Central Computer contain the circuits and controls required by operating personnel to initially load a program into the memory element, to start and stop computer operation, and to operate the computer for maintenance purposes.

The subsequent analyses of the major circuit groups of each Central Computer element are based on the assumption that both the operating program and the required data are stored in the memory element and that the computer is operating to execute the program instructions. To emphasize the primary interrelation of the various circuit groups, the following discussion is referenced to figure 1–25, foldout. This figure, which shows the major circuit groups of the Central Computer, has been purposely simplified by excluding many internal transfer and control lines.

4.2 MEMORY ELEMENT

As shown in figure 1–25, foldout, the memory element of the Central Computer comprises three distinct storage devices: core memory 1, core memory 2, and test memory. Although the memory address registers are shown as individual blocks, each register is actually an integral part of the associated memory device. The memory buffer register is a transfer buffer between the three memory devices and various registers of other elements.

Core memory 1 has a storage capacity of 65,536 words, core memory 2 has a storage capacity of 4,096 words, and the test memory has an operating storage capacity of 16 words. Since the basic unit of information (instructions and data) used in the computer is a 32-bit binary word, the memory word length is also 32 bits. However, a special condition exists when either of the core memories is utilized, since a 33rd bit is added to each word when it is initially stored in either of these devices. This 33rd bit, called a parity bit, does not affect the information contained in the other 32 bits and is used only to detect errors in the transfer of words between the core memory devices and the memory buffer registers.

4.2.1 Core Memory Devices

Although the capacity and internal circuitry of the two core memory devices are different, the two memories operate on the same basic principles since the primary component in each device is a 3-dimensional, magnetic core array. In each memory, the magnetic core array is composed of 33 horizontal planes of ferrite cores. All the cores in a single plane are used to store one specific bit of each memory word that can be stored in that core array. Thus, each plane of the memory 1 array contains 65,536 cores, and each plane of the memory 2 array contains 4,096 cores. In each memory plane, the ferrite cores (having an outside diameter of 0.080 inch, an inside diameter of 0.050 inch, and a thickness of 0.025 inch) are wired into a square configuration; that is, each plane has an equal number of rows and columns of cores. The individual row (X) and column (Y) wires, which represent the selection windings of the planes of the array, are connected in series so that the coincident selection of one row and one column will specify a vertical column of 33 cores, one core in each plane. In this manner, the selected X and Y windings represent the location or address of the 33 cores of a memory word. In addition to being linked by X and Y windings, the cores in each individual plane are linked together by two additional wires, the sense and inhibit windings.

The magnetic cores of the core memory arrays are capable of being magnetized in either a clockwise or a counterclockwise direction by passing current pulses through the various core windings. The energy thus stored in the magnetic flux of any core corresponds to either a 1 or a 0, depending upon the direction of magnetization. A binary word is written in the magnetic core array by passing coincident write current pulses of the proper amplitude and polarity through these selected X and Y windings of the array to actuate the 33 cores that correspond to the 33 binary digits of the memory word. In cores that are to contain a 1, the X and Y current pulses magnetize each core in such a direction that it contains a 1. In each individual core that is to contain a 0, the writing of a 1 is prevented

by passing an inhibit current pulse through the inhibit winding of that plane in coincidence with the write current pulses. A binary word is read from the magnetic core array by passing coincident read current pulses of the proper amplitude and direction through the selected X and Y windings of the array so that each selected core is magnetized in such a direction that it will contain a 0. Since the read current pulses cause a reversal in the magnetic flux of each core that undergoes a change of magnetic state (from the 1 condition to the 0 condition), the read operation is referred to as destructive readout. When the magnetic flux in a core is reversed, a voltage is induced in the sense winding of the associated plane. This signal, when properly amplified, can be used to set a flip-flop, thus indicating the past status of the core. If a core was originally in the 0 state, no change in magnetic flux will be produced and, therefore, no voltage will be induced in the sense winding of the associated plane.

Since the core memory devices process all data in exactly the same manner without distinguishing its source or destination, memory cycles that transfer data out of memory (equivalent to PT, OTA, and BO machine cycles) are referred to as readout cycles. The memory cycles that store data into memory (equivalent to OTB and BI machine cycles) are referred to as store cycles. These two cycles are very similar since both cycles involve setting up the X and Y selection circuits to specify the desired memory word and then performing a read and write operation in sequence. At the start of a memory readout cycle, the core memory control circuits are cleared, and the desired memory address is transferred to the associated memory address register to set up the control circuits which select the specified X and Y windings. Core memory operation is then initiated by a start-memory-control signal to control the read and write operations. As a result, coincident read current pulses are applied to the selected X and Y windings to reverse the flux in those selected cores that contain a 1. The resultant voltage induced in each associated sense winding is used to set the associated bit in the previously cleared memory buffer register. Thus, at the end of the read operation (approximately 3.0 μsec after the start of the readout cycle), the memory buffer register will contain the selected memory word, and the selected cores will be in the 0 state. Because the read operation effectively erases the selected word from the memory array, it is necessary to rewrite this word into the same location so that it will be available for future reference. This latter action is accomplished by the write operation, which consists of applying coincident write current pulses to the selected X and Y windings, and a coincident inhibit current pulse to the inhibit winding of each specified plane. The inhibit current pulse is applied to a plane only if the associated bit of the memory buffer register contains a 0. Thus, the write operation will result in writing 1's in all cores of the selected memory address except those that should remain at 0.

The primary difference between the readout cycle and the store cycle is that the read operation of the latter cycle is used to effectively erase the contents of the selected core memory address (the memory buffer register is not affected by the original contents of the selected cores), and the write operation is used to control the writing of the new memory word which was loaded into the memory buffer register from an external source prior to the write operation.

4.2.2 Test Memory

Basically, the test memory portion of the memory element is a manual storage device into which 32-bit binary words (either instruction words or data words) may be manually inserted, and which the computer is able to read at the standard operating speed of one memory word in 6.0 μsec. This device is used primarily for the storage of short maintenance programs containing up to 16 instructions, or for the storage of constants or control words used in special programs. Although provision is made for the storage of as many as 19 binary words in the test memory (in 19 separate registers), no more than 16 words can be made available to the computer for use in any one program. The 19 registers which comprise the test memory storage facilities consist of the following: toggle switch register A, toggle switch register B, the 16 registers on the test memory control panel, and a test register. The two toggle switch registers and the test memory control panel, which are manually loaded, are physically mounted on the duplex maintenance console. The test register, which is a 32-bit flip-flop register, is the only register in test memory that can be loaded by the computer under program control.

Each of the two toggle switch registers, which provide a very flexible means of manually loading a memory word, consist of a bank of 32 toggle switches. A memory word is inserted by manually setting the individual switches to either the 1 or 0 position.

The 16-register control panel, which is the primary component of the test memory, contains 16 double rows of hubs, which when bridged by connectors are said to contain 1's and when not bridged are said to contain 0's. Each of the 16 rows of double hubs, which are identified as the 16 control panel addresses, contains 32 information hubs which are used to store a memory word, and 4 control hubs which provide the means of specifying the source of information that will be transferred to the memory buffer register when that control panel address is selected. These control hubs can be bridged singly or in any combination, so that the information transferred from the test memory will be obtained from any of the following: the word stored at

that address in the control panel; the word stored in toggle switch register A; the word stored in toggle switch register B; the word stored in the test register; or the logical sum of any two, three, or four of these words.

During the execution of a memory cycle that involves the test memory, the location of a specific memory word is designated by the test memory address register and the setting of the test memory UNASSIGN/ASSIGN switch, which is located on the duplex maintenance console. The purpose of this switch is to provide additional flexibility in selecting the sequence of test memory registers that are selected by the 16 consecutive test memory addresses. When this switch is set to the UNASSIGN position, the 16 consecutive test memory addresses will specify the control hubs associated with the 16 consecutive control panel addresses. The control hubs then specify the source of information that will be transferred for each test memory address selected. When the switch is set to the ASSIGN position, test memory addresses 3.77760_8 and 3.77761_8 always specify toggle switch registers A and B, respectively; test memory addresses 3.77762_8 through 3.77776_8 specify the control hubs associated with control panel addresses 2 through 16_8, respectively; and test memory address 3.77777_8 will always specify the test register. In the latter case, only 13 of the 16 control panel words can be transferred to the memory buffer register; the memory words contained in control panel addresses 0, 1, and 15_8 cannot be addressed.

4.3 INSTRUCTION CONTROL ELEMENT

As previously stated, the function of this element is to decode each program instruction as it is read out of the memory element and to generate all the properly timed control signals required to perform the specified task. To perform the various operations required by the 59 instructions that the computer is capable of executing, a total of 163 identified control signals (commands) may be generated. A command is defined as a control pulse generated at a specific time which performs a single operation, such as transferring the contents of one register to another register, clearing a specific register, complementing a specific register, etc. Since the instruction control element is the control center of the computer, the commands generated by this element are distributed among the other elements of the computer.

To simplify the block diagram analysis of this element, the circuits of the element are grouped into three sections, each of which has a specific function:

a. Instruction decoder
b. Pulse generation and control
c. Command generators

During the execution of an instruction, the instruction decoder combines the coded information of the instruction word (which specifies the instruction to be executed) with the internal cycle control circuits (which determine whether a PT, an OTA, or an OTB machine cycle is in process) to cause a conditioning d-c level to be applied to specific groups of command generators. Thus, during the execution of an instruction, a different group of command generators will be conditioned for each of the machine cycles associated with that instruction. During every internal machine cycle, all the command generators (163) are sensed by timing pulses generated by the pulse generation and control circuits, with the result that the proper commands will be generated during each of the machine cycles required by the instruction.

4.3.1 Instruction Decoder

As shown in figure 1–25, the instruction decoder consists of the operation register, the cycle control circuits, and the class-cycle, variation, index selection, and instruction matrices. The operation register initially receives the coded instruction word from the memory element during a PT cycle (program time machine cycle). This coded information remains in this register for the duration of the instruction execution time and is used to condition the control matrices; that is, the class-cycle, variation, and index selection matrices.

The class-cycle matrix, which receives inputs from the class selection portion of the operation register and the cycle control circuits, determines to which of the eight classes a particular instruction belongs and the types of machine cycles required for its execution. Some of the individual output lines of the class-cycle matrix are used to condition command generators directly, and commands thus generated are termed class-common commands. The majority of the output lines of the class-cycle matrix are fed to the instruction matrices, where they are combined with the outputs of the variation and index selector matrices. The outputs from the cycle control are also used to control command generators directly, and the commands thus generated are termed cycle commands. Since three types of internal machine cycles (PT, OTA, and OTB) are possible, there are three such groups of commands.

The variation matrix, which receives inputs from the variation selection portion of the operation register, determines which variation of the selected class of instructions is being executed. The outputs of the variation matrix are fed to the instruction matrices, where they are combined with the outputs of the class-cycle and index selector matrices.

The index selection matrix, which receives inputs from the index selection portion of the operation register, determines which, if any, of the index registers (in the program element) is to be used during the execution of the instruction. The outputs of the index selector matrix are fed to the instruction matrices, where they

are combined with the outputs of the class-cycle and variation matrices.

The instruction matrices, which combine the outputs of the class-cycle, variation, and index selector matrices, are used to condition command generators which are not common to any particular class of instruction or to a particular machine cycle of operation. In general, only one command generator is provided for the generation of each unique command. As a result, if a common operation is required at the same cycle time during the execution of several instructions, the outputs of the instruction matrices (which specify these instructions and machine cycles) are combined so that the command will be generated during the execution of any of these instructions.

4.3.2 Pulse Generation and Control

Three types of pulses are used within the computer during internal operation: 2-mc pulses, time pulses, and instruction pulses. The purpose of the pulse generation and control section is to develop and distribute these pulses, which are subsequently gated by the command generators. The pulse generation and control section (fig. 1–25) consists of an oscillator, a time pulse distributor control (TPDC), a step counter, a divide time pulse distributor, and a time pulse distributor, all of which are interconnected as shown in figure 1–26.

The primary source of all control pulses within the computer is the 2-mc oscillator. Time pulses and instruction pulses used during internal computer operation are supplied by the time pulse distributor, which converts two single-line 2-mc inputs into 12 time pulse outputs and 12 instruction pulse outputs. The pulse repetition rate for individual time pulses and instruction pulses is 6.0 μsec. The TPDC functions to control the generation of the various timing pulses used during all computer operations; that is, the TPDC controls the time pulse distributor to generate time pulses and instruction pulses simultaneously, to generate time pulses alone, or to generate neither. The primary function of the time pulses is to supply overall computer timing and to sample command generators in order to develop the machine cycle common commands. The primary function of the instruction pulses is to sample the majority of the command generators and, thus, to generate the unique commands required to perform the majority of the operations specified by the instruction being executed. The 2-mc pulses, besides being converted into time and instruction pulses, provide overall computer timing synchronization by sampling a great number of control pulse generators. These pulses are also used to supply 2-mc operate pulses for certain types of computer operation.

During the normal execution of the sequentially stored instructions which constitute the computer program, the time pulses and the instruction pulses develop commands which perform the various operations required by the instructions. During the execution of some instructions, conditions arise wherein many repetitive operations are performed and where the normally allotted machine-cycle time is insufficient for the instruction to be completed (primarily shift- and multiply-class instructions). Under these conditions, time and instruction pulses are temporarily discontinued at the end

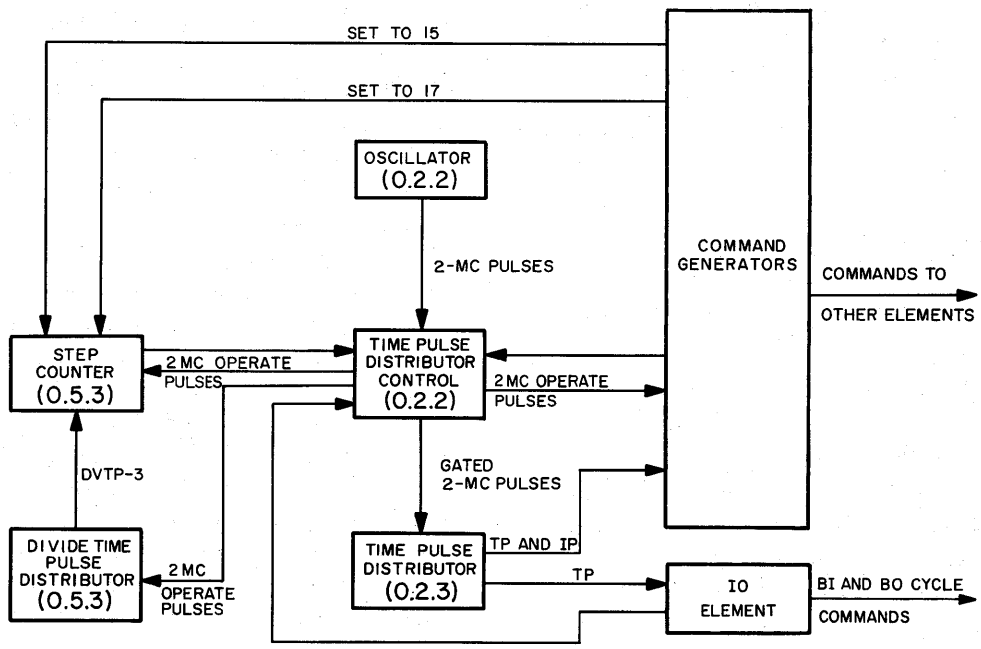


Figure 1–26. Pulse Generation and Control

of the last machine cycle required by the instruction, and 2-mc operate pulses (which are not the same as 2-mc control pulses) are used to perform the repetitive operations. The time during which the 2-mc operate pulses are being generated is termed an arithmetic pause because, during this time, normal computer timing, as controlled by the time pulses, is not being maintained. The duration of the pause varies with the instruction and is controlled by the step counter. An auxiliary control (the divide-time pulse distributor) is used with the step counter during the execution of a *Divide* instruction. When the step counter terminates the pause condition, time and instruction pulses are restored and the computer continues to execute the instruction in process.

When an IO unit is selected and actuated by specific instructions of the stored program, computer time must be provided to effect the subsequent transfer of IO words. The IO units are relatively slow compared with the computer, and, since it is desirable to have the computer operating as much of the time as possible, information is transferred between the selected IO unit and the memory element on a word-by-word basis whenever the IO unit is ready for a word transfer, and normal internal computer operation is maintained during the intervals. Thus, when a selected IO unit is ready to effect a word transfer, the continuity of internal computer timing is temporarily broken, and a single IO word is transferred during the resulting break cycle.

Since information can be transferred to as well as from the memory element, the break cycles are further distinguished as BI or BO machine cycles. During the execution of either of these cycles, normal internal computer cycles are not executed, with the result that the instruction pulses which produce the internal sequenced operations are halted. Time pulses, however, are not halted since these pulses provide the required basic machine timing. During a break cycle, time pulses are converted into BI or BO pulses in the IO element to provide the break commands required to effect the IO word transfer.

4.4 PROGRAM ELEMENT

As shown in figure 1–25, the program element of the Central Computer contains the following circuits:

a. Address register
b. Index registers
c. Index adder
d. Program counter
e. Memory unit selection circuits

These circuits, which are controlled by commands generated by the instruction control element, specify the memory element location to be used during the execution of subsequent internal machine cycles (PT, OT_A, and OT_B). Since each of the registers contained in this element is used to perform a unique control function, the analysis of operation of these circuits in selecting a memory address is preceded by individual discussions on the function of each register.

4.4.1 Address Register

During the execution of the stored program instructions, the address portion of each instruction word is transferred to the address register (and the step counter of the instruction control element) when the instruction word is first obtained from the memory element. For the majority of the instructions that the Central Computer is capable of executing (32 of the 59), the contents of the address register specify the memory address of the data to be processed. During the execution of these instructions, the contents of the address register are transferred to the memory element (at the proper time), and the data obtained from memory is transferred to the arithmetic element, where it is processed to fulfill the requirement of the instruction.

During the execution of a control instruction (one of six branch-class instructions), the contents of the address register specify the memory address of the next instruction that will be executed if the condition specified by the control instruction exists. If the condition does not exist, no meaningful computer action will occur and the next sequential program instruction will be executed in step.

Five of the instructions that the computer is capable of executing function to perform the task of loading a constant into one of the control registers of the computer. These instructions utilize the address register as a temporary intermediate storage register.

The other 16 instructions that the computer is capable of executing (which include the 9 arithmetic shift instructions) do not utilize the address register for control. Thus, for these instructions, the address register contents are meaningless. During the execution of any one of the 9 arithmetic shift instructions, the contents of the step counter (which was loaded when the instruction word was obtained from memory) are used to specify the number of shift operations to be performed.

4.4.2 Index Register

During the normal operation of the AN/FSQ-7 equipment, the Central Computer System is provided with numerous sets of similar data which are to be processed in a fixed manner to produce similar results. Since these sets of data are normally stored in sequential memory addresses, a process called indexing can be utilized, in which a given fixed sequence of instructions can be used to sequentially process each set of data. Indexing is defined as a process in which a fixed set of instructions is controlled so that the instructions will be repeatedly executed to process numerous sets of data.

As used in the Central Computer, the indexing operation results in modifying the address portion of an

instruction word by the contents of a selected index register prior to the time that the address information is used to select the memory address of the data to be processed. This operation is accomplished by the address register, index registers, and the index adder (which is an integral part of the address register) of the program element.

If indexing is to be used during the execution of a stored program, two special instructions must be included in the program routine. The first of these provides for the loading of the desired constant into a specified index register; the other (a branch-class instruction) provides for the subtraction of a fixed quantity from the contents of the selected index register. The constant used represents the numerical difference between the highest address of the data to be processed and the address specified by the address portion of the instruction to be executed. During the execution of an indexed instruction, the contents of the selected index register are added to the contents of the address register, so that the desired data will be transferred from the memory element during the subsequent OT cycle. Before this instruction can be repeated to process the next set of data, the contents of the selected index register are reduced by a fixed quantity which represents the numerical difference between sequential data addresses. This sequence of events is continuously repeated until the selected index register contents are reduced to zero.

The program element of the Central Computer contains four physical index registers; however, the right accumulator register in the arithmetic element may also be used as an index register. It is the sole function of an index register to index the address portion of an instruction word when this type of operation is specified by the operation portion of the instruction word.

4.4.3 Program Counter

In initially loading a program into the memory element, the individual instructions which constitute the program are stored in consecutive memory addresses. As its name implies, the program counter provides for the sequential execution of these instructions. Thus, each time an instruction word is transferred from the memory element to the computer control circuits, a 1 is added to the program counter so that the next consecutive program instruction will be read out of memory during the subsequent program time (PT) machine cycle.

In some instances, the execution of consecutive program instructions is not desired. In these cases, the program counter contents are modified directly by the instruction being executed. The program counter may be modified in one of two ways, depending upon the type of instruction being executed. During the execution of one of the 10 logical (compare) instructions, the program counter may be directly stepped up to three times, depending upon the instruction being executed and the conditions that exist. Thus, the computer may skip 1, 2, or 3 program instructions. The second method of control occurs during the execution of one of the six control instructions (branch-class instructions). During the execution of any of these instructions, a specific condition is sensed for and, if it exists, a branch of program control is executed. Since the address part of these instructions specifies the next program address that is desired, the program counter is cleared and the new program address is transferred from the address register to the program counter. When a branch of control is executed, the program continues sequentially from this new program address.

4.4.4 Memory Address Selection Circuits

The program counter and the address register (fig. 1–25) of the program element are the internal-machine-cycle addressing registers of the Central Computer. Although the IO address counter of the IO element is controlled by break cycle commands, it performs the same function as the other addressing registers and is therefore included in the following discussion.

During the execution of every machine cycle, the contents of the selected addressing register must be transferred to the memory element to specify which memory address is to be involved in the operation. Since the memory element contains three separate storage devices, the contents of the addressing register must specify which of these three devices is to be used as well as the memory address within the selected memory device. To accomplish this task, each of the addressing registers contains 17 bits, which are designated as bits LS and RS through R15. Since the capacities of each of the memory storage devices are different, specific groups of these bits are used to designate an address in each of the devices. Bits R12 through R15 specify one of the 16 addresses of test memory, bits R4 through R15 specify one of the 4,096 addresses in core memory 2, and bits RS through R15 specify one of the 65,536 addresses in core memory 1. Information to determine which of the three memory devices contains the desired memory word is specified by the contents of LS and RS through R11.

When a memory cycle is initiated, the contents of bits RS through R15 of the specified register are transferred to each of the memory address registers, and the contents of bits LS and RS through R11 are decoded by the memory unit selection circuits to determine which memory device is to be activated by a start-memory pulse. If bit LS contains a 0, core memory 1 is activated; if bit LS contains a 1, bits RS through R11 are examined to determine whether core memory 2 or test memory should be activated. In the latter case, if bits RS through R11 all contain 1's, test memory is specified; if any of these bits contain a 0, core memory 2 is specified.

4.5 ARITHMETIC ELEMENT

The arithmetic element of the Central Computer contains the arithmetic registers and circuits required to perform all arithmetic computations and logic operations on the numerical data supplied to it during the execution of the stored program. Since all the circuits contained in this element are passive, the manner in which they function and operate during the execution of an arithmetic instruction (an instruction that affects the arithmetic element) is completely controlled by commands generated by the instruction control element. As shown in figure 1–25, the arithmetic element is composed of two functional groups of registers and circuits; each individual register and circuit is prefixed by LEFT or RIGHT to show its relationship to one of the groups. Since each register within the arithmetic element contains 16 bits of information (except for the 17-bit left A register), each of the associated adder and compare circuit blocks contains 16 identical circuits. The register bits and individual circuits within a functional group are designated bits LS through L15 or RS through R15 to identify each individual circuit with the element. The auxiliary circuit blocks contain the associated flip-flops and gating circuits required to provide control for dealing with end-carry, overflow, divide control, and sign storage.

The circuits of the arithmetic element are divided into two functional groups to increase the effective operating speed of the AN/FSQ-7 equipment in processing tactical data. Since each discrete quantity of tactical input information (such as the X, Y, R, or Θ target co-ordinate, relative time, site identity, console identity, address code, etc., quantities which make up the various types of data words) is expressed in 16 binary bits or less, each 32-bit data word stored in the memory element will contain at least two related quantities of information. As a result, during the execution of an arithmetic instruction that deals with the manipulation and/or processing of tactical data, the left and right arithmetic circuits function independently of each other as complete entities. For the majority of these instructions, the left and right arithmetic circuit groups are controlled identically to perform the same type of operation on two related quantities of the data word. For the remainder of these instructions, the registers of the left and right arithmetic groups are controlled individually to fulfill special program requirements.

During the execution of the stored program, special conditions occur in which an instruction is used to arithmetically modify the address portion of another instruction. Since the address portion of an instruction word consists of 17 bits of information, the 16-bit arithmetic circuit groups which normally function independently must be reconnected to provide for 17-bit operation. Because the address portion of an instruction is specified by bits LS and RS through R15 of the instruction word, the bit LS circuits are connected to the right arithmetic group during the execution of such an instruction. In this case, the right auxiliary circuits are not used, and the bit L1 through L15 circuits are only used to provide for an inherent shift right so that the sum of the bit LS adder can be temporarily stored in bit L1 of the accumulator. The manner in which the arithmetic circuit groups are to function during the processing of either tactical data or instruction addresses is specified by the contents of bit L12 of the instruction word. If this bit contains a 0, the two 16-bit arithmetic circuit groups will function independently; if this bit contains a 1, the two arithmetic circuit groups will be connected together to provide for 17-bit operation.

The circuits of the arithmetic element are used during the execution of four general categories of arithmetic instructions: those that implement the four basic arithmetic operations (add, subtract, multiply, and divide), those that cause the contents of the arithmetic registers to be shifted in one direction or the other, those that compare the contents of a data word with the contents of the accumulator registers, and those that store the contents of designated arithmetic registers into a specified memory location. Since the two arithmetic circuit groups function independently for the majority of the arithmetic instructions, the following brief discussion of these instructions will not include the 17-bit mode of operation. During the execution of an arithmetic instruction that implements one of the four basic arithmetic operations (add and multiply instructions), the adders and the associated A, B, and accumulator registers are connected together to form the two required operating circuit groups. During the operate cycle (OTA) of such an instruction, each half of the data word obtained from the memory element is transferred to its associated A register so that it can be either added to, subtracted from, multiplied by, or divided into the contents of the associated accumulator register. Since the adder circuits can only function to add two numbers, subtraction is accomplished by means of a complement system wherein the number to be subtracted is first complemented and then added. Multiplication of two binary numbers is accomplished by repetitive addition of the multiplicand to each previous partial product as directed by each more-significant bit of the multiplier. The final product is contained in the associated accumulator B register. Division of two binary numbers is accomplished by repetitive trial subtraction of the divisor, first from the dividend and subsequently from the current remainder, to develop the individual bits of the quotient. The quotient and final remainder are contained in the associated accumulator B register.

The various shift-type instructions that the computer is capable of executing are used to position the contents

of the individual accumulator registers or the combined accumulator B registers for future operations. For each of these instructions, the instruction portion of the instruction word specifies which arithmetic registers are to be affected and in what direction the register contents are to be shifted. The address portion of the instruction word, which is transferred to the step counter of the instruction control element, is used to specify the number of shifts to be performed. During the execution of a shift instruction, the instruction control element generates the required shifting commands on a repetitive basis to perform the required number of shifts.

During the execution of an arithmetic instruction that specifies that a logical comparison is to be made between specific bits of two registers (compare instruction), the compare circuits and the associated A and accumulator registers are connected together to form the two required operating circuit groups. During the operate time (OTA) of such an instruction, each half of the data word obtained from the memory element is transferred to its associated A register so that its contents can be compared bit by bit with the contents of the associated accumulator. If the specified bits of the A and accumulator registers do not compare, a no-compare signal will be generated which will step the program counter so that the next sequential instruction will be skipped in the execution of the stored program. These instructions are generally used to determine the identity code of data words and set up the desired program control to take appropriate action.

The majority of the store type instructions that the computer is capable of executing specify that the contents of a specific arithmetic register be stored in either the left or right half portion of the designated memory location. However, these instructions are complicated by the fact that the memory element functions either to read out or to store a full 32-bit memory word during a memory cycle. Thus, these instructions require two memory cycles to perform the specified operation. During the first of these (OTA cycle), the memory word stored in the specified memory location is transferred to the A registers so that the unchanging portion will be retained. During the second memory cycle (OTB cycle), the old contents of the specified memory location are erased, and the new word to be stored (which consists of an unchanged and a changed portion) is transferred to the memory buffer register from the specified arithmetic registers.

4.6 SELECTION ELEMENT

As previously stated, the selection element of the Central Computer performs a selection function during the execution of six specific instructions:

a. *Test One Bit (TOB)*
b. *Test Two Bits (TTB)*
c. *Operate (PER)*
d. *Branch and Sense (BSN)*
e. *Select (SEL)*
f. *Select Drum (SDR)*

Each of these instructions uses the selection element circuits to determine which specific circuit of an associated group is to be operated on by a command generated by the instruction control element.

As shown in figure 1–25, the selection element contains the index interval register, the *PERSELBSN* matrix, the test, *PER,* and *BSN* circuits, and the select gates. During the execution of any of the above instructions, bits L10 through L15 of the instruction word (which contain the coded information to specify the desired circuit to be operated on) are transferred to the index interval register. The output of the index interval register is fed to the *PERSELBSN* matrix, which determines which of the 64 possible codes is specified. The selected line of the *PERSELBSN* matrix is supplied to the test, *PER, BSN,* and selected circuit groups, conditioning only one circuit in each group. The instruction control element specifies which circuit group is involved in the operation by generating the proper command pulse.

The test-bit circuits of the selection element are used during the execution of the *Test One Bit (TOB)* and the *Test Two Bits (TTB)* instructions to specify which bit (or bits) of the selected memory word is to be tested. If the specified bit (or bits) of the memory word is a 1, the program counter is stepped one or more times so that the next one or more sequential program instructions will be skipped.

During the execution of the *PER* instruction, the selected *PER* gate generates an operate pulse to perform the specified operation.

The *BSN* circuits of the selection element contain the *BSN* selection gates and the associated flip-flops which are used to indicate the status of a large number of operating conditions in the AN/FSQ-7 equipment. During the execution of a *BSN* instruction, a specific condition is sensed to determine whether a branch of program control is to be performed.

The selection gates of the selection element are used during the execution of the *Select (SEL)* and *Select Drum (SDR)* instructions to set the specified IO selection circuits and IO transfer control circuits of the IO element. The selected circuits in the IO element specify the IO devices to be used during the subsequent IO operation and the mode of data transfer to be used.

4.7 IO ELEMENT

The IO element of the Central Computer (fig. 1–25) contains the control circuits and transfer registers required to accomplish the transfer of information

between the memory element and the various asynchronous, slow-speed IO devices associated with the Central Computer. Since the transfer of IO information is initiated by the stored program (except for initial program loading), the Central Computer must execute specific instructions to prepare the IO element circuits before any information transfer can occur. These instructions must specify the following:

a. The IO device or drum field to be involved in the subsequent IO operation.
b. The drum starting address or identity code to be used if the subsequent IO operation involves the Drum System.
c. The starting memory address from which or into which information is to be transferred.
d. The direction of information transfer; that is, whether information is to be transferred into or out of the memory element.
e. The number of words to be transferred during the subsequent IO operation.

This basic information, which must be supplied to the IO element each time a block transfer of IO data is to take place, is supplied by a group of three instructions which constitute an IO program. The basic IO program is composed of the following instructions:

a. *Select* (*SEL*) or *Select Drum* (*SDR*)
b. *Load IO Address Counter* (*LDC*)
c. *Read* (*RDS*) or *Write* (*WRT*)

Generally, the three required instructions are executed in the sequence noted above; the only restriction is that the *Read (RDS)* or *Write (WRT)* instruction must be executed last.

As noted above, one of two separate selection instructions is used to specify which IO device is to be involved in the subsequent IO operation. Two separate instructions are required because the selective capacity of a single instruction is not great enough to specify all the IO devices associated with the Central Computer. If one of the individual IO devices (other than a drum field) is to be selected, a *Select* instruction is used. During the execution of a *Select* instruction, the IO selection and transfer control circuits of the IO element are cleared, and the desired controls (flip-flops) are set as directed by the control signal generated by the conditioned select gate of the selection element. In this case, the selected controls indicate which IO device is selected and the type of transfer control that will be exercised during the subsequent transfer of data. If the Central Computer is to be associated with one of the 39 fields of the main drum, or with one of the 36 fields of the auxiliary drum, a *Select Drum* instruction is used. The *Select Drum* instruction performs three distinct functions: it sets up the specified field selection circuits in the Drum System, sets up the IO selection and transfer control circuits in the IO element, and designates the drum starting address or identity code if this information is required by the selected drum field. During the initial execution of this instruction, the IO selection and transfer control circuits of the IO element and the main and auxiliary drum field selection registers of the Drum System are cleared in preparation for the new selection. During the latter part of the instruction cycle, the contents of the index interval register (which specifies the selected field in either drum) are transferred to the selected drum field selection register (specified by the contents of R1 of the instruction word). At the same time, the conditioned select gate of the selection element will generate a control signal to set the associated IO transfer control circuits. In addition, during the latter part of the instruction cycle, the contents of the address register (which specifies the drum address or identity code to be used) are transferred to the drum control register of the IO transfer control circuits. Thus, the *Select Drum* instruction functions to set up the specified conditions in both the Drum System and the IO element of the Central Computer.

The *Load IO Address Counter (LDC)* instruction of the basic IO program specifies the starting memory address to be used during the subsequent IO operation. During the execution of this instruction, the contents of the address register (which specifies the memory starting address) are transferred to the IO address counter of the IO element. Subsequently, as each data word is transferred, the contents of the IO address counter are increased by 1, with the result that consecutive memory locations are utilized during any IO operation.

The final instruction executed during the processing of a particular IO program depends upon the direction that the IO data is to be transferred. If IO data is to be transferred from the selected IO device to the memory element, the *Read (RDS)* instruction is executed; if IO data is to be transferred from the memory element to the selected IO device, the *Write (WRT)* instruction is executed. The address portion of each of these instructions is used to specify the number of words to be transferred during the subsequent IO operation. During the execution of either of these instructions, the following operations are performed:

a. The IO interlock is set.
b. The complement of the address register content is transferred to the IO word counter.
c. The specified read or write control circuit is set.
d. A *start read* or a *start write* command is generated.

The IO interlock, which forms a part of the IO termination controls of the IO element, is always set at the beginning of an IO operation and remains set for the

duration of the operation. The IO interlock functions to prevent subsequent IO program instructions from disturbing the IO operation in process. If such a condition occurs, all internal operations will be halted until the IO operation in process is terminated.

The IO word counter of the IO element functions to keep a running record of the number of data words that remain to be transferred during the remainder of the IO operation. As each data word is transferred, the contents of this counter are increased by 1. When the contents of this counter are stepped from negative to positive zero, an end-carry pulse is generated which is used to terminate the IO operation in process.

The read and write control circuits, which form a part of the break cycle generation and control circuits of the IO element, are used to specify which type of break cycle will be generated during the subsequent data word transfers. If the read flip-flop is set, the subsequent break cycles will be defined as BI cycles; if the write flip-flop is set, the subsequent break cycles will be defined as BO cycles.

The *start read* or *start write* command generated during the execution of the *Read (RDS)* or *Write (WRT)* instruction functions to start the IO operation. These commands, which are generated in the instruction control element, are used to sense the IO operation circuits of the IO element, so that a start pulse may be applied to the selected IO device. This start pulse functions either to set the selected IO device in motion or to condition its control circuits. From this time until the IO operation is terminated, the Central Computer is under the control of the selected IO device. During the IO operation, the selected IO device informs the computer, by means of the break request circuits, each time it is ready to transfer data between its associated IO transfer registers and the memory element. The manner in which the IO devices are connected to the IO transfer registers (IO buffer register and IO register) is shown in detail in figure 1–25. As noted in the figure, the transfer of IO data into or out of the memory element is accomplished between the memory buffer register and the IO register. The IO buffer register, which is used only during the transfer of input data, provides temporary storage. As indicated in figure 1–27, the transfer of data from the IO buffer register to the IO register is effected by the IO transfer and control circuits.

After the IO program instructions are executed, the Central Computer continues to execute the internal operations as directed by the stored program. At the end of each machine cycle, the break request circuits of the IO element are sensed to determine whether IO data is to be transferred into or out of the memory element. If the break request circuits are cleared, the computer continues with its internal operations. If the break circuits are set, the internal computer operations are stopped for one machine cycle, and the next memory cycle is designated as a BI or BO cycle. During the break cycle, the time pulses generated by the instruction control element are converted into BI or BO pulses to accomplish the required operations. These break pulses effect the data word transfer between the IO register and the memory element and step the IO word counter and IO address counter in preparation for the next break cycle. After the break cycle is completed, the computer resumes its internal operations. This process of interleaving break cycles with internal machine cycles is repeated until the proper number of IO data words have been transferred. When the IO operation is completed, the IO terminating circuits generate a disconnect signal, which either stops the selected IO device or deconditions its control circuits.

4.8 MANUAL CONTROL FACILITIES

The manual controls and indicators associated with the Central Computer System are located on the duplex maintenance and duplex switching consoles, which contain the controls and indicators associated with the duplex equipment. Since the scope of this manual is limited to the Central Computer System and associated IO devices, only the associated controls and indicators are discussed in this manual. The remaining controls are discussed in the applicable systems theory manuals.

The duplex maintenance console (fig. 1–11) plays an important role in the maintenance operation of the Central Computer and associated IO devices. Basically, the computer manual controls can be grouped into two types: those that are used to initiate some specific action, and those that are used to supplement program control. The first type consists of pushbutton switches which, when depressed, will generate a control pulse to execute the desired action. These pushbutton controls enable the operator to perform any of the following:

a. Reset the computer controls to a starting state.
b. Clear all the core memory locations.
c. Initially load a program into core memory from either the card reader or an auxiliary memory drum field.
d. Start and stop the execution of the stored program.
e. Select and start to execute the program contained in test memory.
f. Execute the stored program in discrete steps; that is, by single instruction, memory cycle, or timing pulse.
g. Clear alarm circuits.

The second type of manual control consists of toggle switches which are preset before the execution of the stored program is initiated. These toggle switches are set to determine what action is to be taken if an alarm

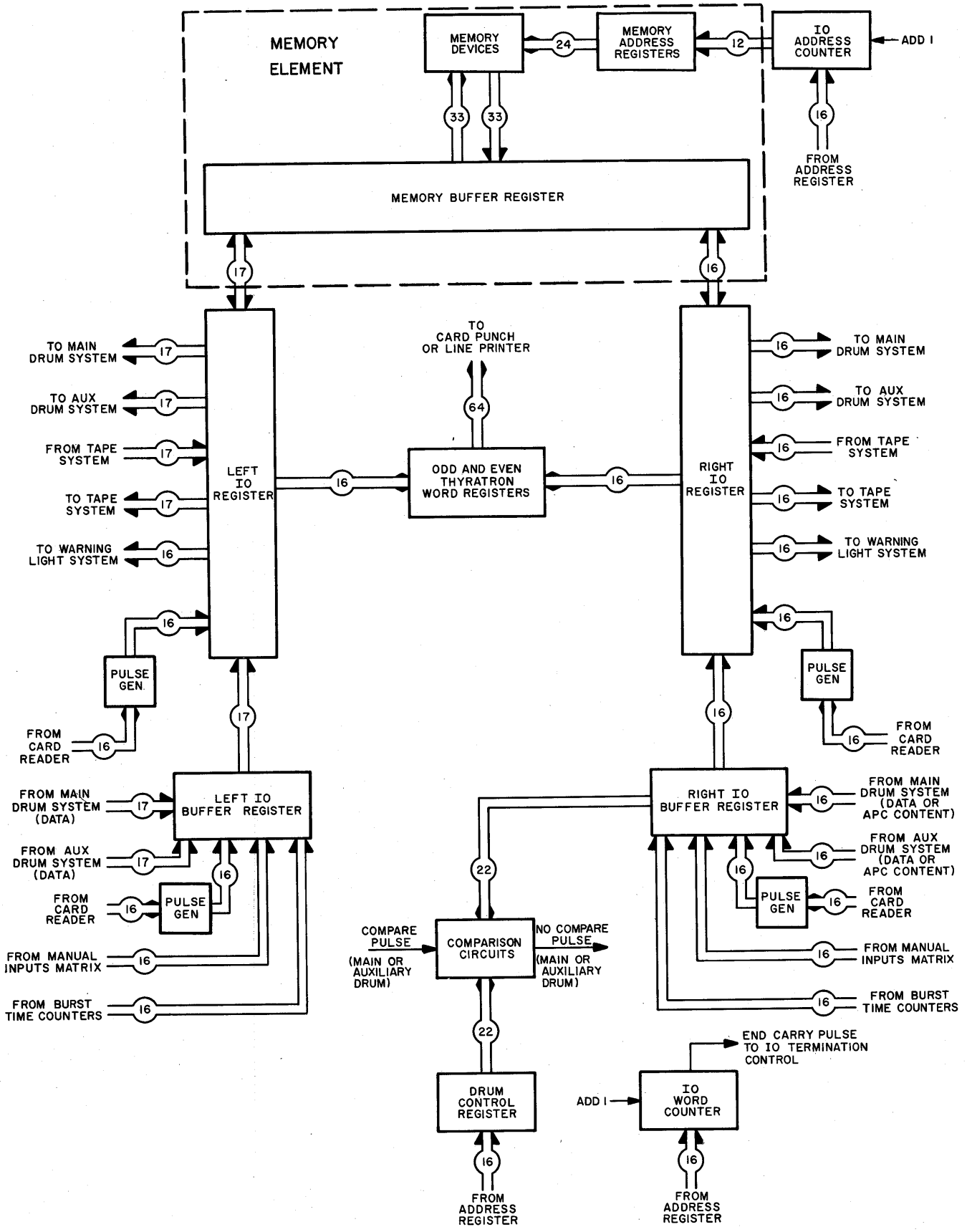


Figure 1–27. IO Element Registers, Simplified Block Diagram

condition occurs during the execution of the program. These alarm conditions are:

a. Memory parity
b. Addressable drum parity
c. Status drum parity
d. Tape parity
e. Arithmetic overflow
f. Inactivity

If a specific alarm condition occurs during the execution of the program, the setting of the associated switch is sensed to determine whether the computer should continue with the sequential execution of the program, execute an alarm branch to test memory, or execute an alarm halt.

The neon indicators mounted on the duplex maintenance console are used to inform the operator of the status of every Central Computer register and control flip-flop. Thus, if a programmed error halt or an alarm halt occurs during the execution of a maintenance program, the operator can determine the cause of the halt by analyzing the contents of various computer registers and control flip-flops. If the computer stopped because of a programmed error halt, reference to the program listing and/or program writeup will usually enable the operator to determine the cause of the trouble.

The duplex switching console (fig. 1–20) contains neon indicators to specify the active-standby status of the individual sets of duplex equipment, the active-standby status of the two simplex power supplies, and the pushbutton switches required to produce a reversal of this status.

CHAPTER 5
INSTRUCTION ANALYSIS

5.1 INTRODUCTION

This chapter contains a brief analysis of each of the 59 instructions the Central Computer System is capable of executing. The discussions are not detailed because they are only intended to provide a basic understanding of each instruction. A detailed analysis of these instructions is contained in Parts 3 through 6.

5.2 ADD CLASS

The 10 instructions grouped within this class involve adding the contents of a specified memory location or register to the contents of the accumulator registers. Because subtraction in the Central Computer System is accomplished by the addition of complements, the instructions which involve algebraic subtractions are also contained within the add class. The control circuits to provide for 17-bit adder operation are conditioned during the execution of any add class instruction. However, 17-bit adder operation is meaningful only for the *ADD*, *SUB*, and *ADB* instructions of this class. Similarly, the indexing control circuits are conditioned during the execution of any add class instruction. However, indexing is meaningless during the execution of either the *ADB* or *CAC* instruction. All the instructions in this class require a PT cycle and an OT cycle for execution. The add-class instructions and their functions are listed in table 1–6.

TABLE 1–6. ADD CLASS INSTRUCTIONS

NAME	SYMBOL	OCTAL CODE	FUNCTION
Clear and Add	*CAD*	100	The left and right accumulator registers are cleared, and the left and right half-words contained in the memory location (specified by the address part of the instruction and the selected index register) are added to the corresponding accumulator registers.
Add	*ADD*	104	This instruction option is used for data manipulation. During the execution of this instruction, the left and right half-words contained in the memory location (specified by the address part of the instruction and the selected index register) are added to the contents of the corresponding accumulator registers. Execution of this instruction can cause an overflow.
	ADD 10	105	Because this instruction option is used for instruction address modification, the carry lines from the bit RS adder are connected to the bit LS adder to provide for 17-bit operation. During the execution of this instruction, bits LS and RS through R15 of the memory word (specified by the address part of the instruction and the selected index register) are added to bits LS and RS through R15 of the accumulator registers. Execution of this instruction can cause an overflow.
Twin and Add	*TAD*	110	The left half-word contained in the memory location (specified by the address part of the instruction and the selected index register) is added to the contents of both accumulator registers. Execution of this instruction can cause an overflow.
Add B Registers	*ADB*	114	This instruction option is used for data manipulation. During the execution of this instruction, the contents of the left and right B registers are added to the contents of the left and right accumulator registers, respectively. The contents of the B

TABLE 1–6. ADD CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
			registers remain unchanged. The address portion of this instruction is meaningless. Execution of this instruction can cause an overflow.
	ADB 10	115	Because this instruction option is used for instruction address modification, the carry lines from the bit RS adder are connected to the bit LS adder to provide for 17-bit operation. During the execution of this instruction, the contents of B register bits LS and RS through R15 are added to bits LS and RS through R15 of the accumulator registers. The contents of the B registers remain unchanged. The address portion of this instruction is meaningless. Execution of this instruction can cause overflow.
Clear and Subtract	*CSU*	130	The left and right accumulator registers are cleared, and the complement of the left and right half-words (contained in the memory location specified by the address part of the instruction and the selected index register) is added to the corresponding accumulator registers.
Subtract	*SUB*	134	This instruction option is used for data manipulation. During the execution of this instruction, the complement of the left and right half-words (contained in the memory location specified by the address part of the instruction and contents of the selected index register) is added to the corresponding accumulator registers. Execution of this instruction can cause an overflow.
	SUB 10	135	Because this instruction option is used for instruction address modification, the carry lines from the bit RS adder are connected to the bit LS adder to provide for 17-bit operation. During the execution of this instruction, the complement of bits LS and RS through R15 of the memory word (specified by the address part of the instruction and the selected index register) is added to bits LS and RS through R15 of the accumulator registers. Execution of this instruction can cause an overflow.
Twin and Subtract	*TSU*	140	The complement of the left half-word (contained in the memory location specified by the address part of the instruction and the selected index register) is added to the contents of both accumulator registers. Execution of this instruction can cause an overflow.
Clear and Add Magnitude	*CAM*	160	The left and right accumulator registers are cleared, and the positive absolute values of the left and right half-words (contained in the memory location specified by the address part of the instruction and the selected index register) are added to the corresponding accumulator register.
Difference Magnitude	*DIM*	164	The negative absolute values of the left and right half-words (contained in the memory location specified by the address part of the instructions and the selected index register) are added to the positive absolute values of the contents of the corresponding accumulator registers. The original contents of the accumulator registers are transferred to the B registers.

TABLE 1–6. ADD CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
Clear and Add Clock	*CAC*	170	The left and right accumulator registers are cleared, and the contents of the clock register are added to the right accumulator. This instruction is not indexable, and the address part of this instruction must specify a test memory address (preferably address 3.77777) to avoid starting either of the core memory units.

5.3 MULTIPLY CLASS

The four instructions grouped within the multiply class utilize the contents of the A registers in a repetitive manner to execute the instructions. Multiplication is accomplished by developing the individual partial products. If a particular multiplier bit (bit 15 of the B register) is a 1, then the contents of the A registers (the multiplicand) are added to the most significant bits of the partial product (contained in the accumulator registers) and the contents of the B registers and accumulators are shifted one bit-position to the right. If the multiplier bit is a 0, the contents of the combined accumulator registers and the B registers are simply shifted one bit-position to the right. The instructions which involve division are also contained in the multiply class. Division is accomplished by repetitive subtraction of the contents of the A registers (divisor) from the accumulator registers (most significant bits of the dividend or the current remainder). As each one in a series of subtractions is performed, a 1 is recorded in bit 15 of the B registers if the result of the subtraction is positive, and a 0 is recorded if the result is negative. All instructions in this class are indexable, and all require a PT cycle, an OT cycle, and an arithmetic pause for their execution. The arithmetic pause for multiply instructions is 4.5 μsec, and the arithmetic pause for divide instructions is 39.0 μsec. The multiply-class instructions and their functions are listed in table 1–7.

TABLE 1–7. MULTIPLY CLASS INSTRUCTIONS

NAME	SYMBOL	OCTAL CODE	FUNCTION
Multiply	*MUL*	250	The left and right half-words contained in the memory location specified by the address part of the instruction and the selected index registers are multiplied by the contents of the left and right accumulator registers, respectively, to form two 30-bit products plus the associated signs. The left and right products are contained in the two 32-bit registers formed by the left accumulator and left B register, and the right accumulator and right B register, respectively. The sign of each product appears in bit S of the associated accumulator register, and the 30-bit products appear in bits 1 through 15 and S through 14 of the associated accumulator and B registers, respectively. Bit 15 of the B register actually forms the 31st bit of the product. However, since this bit is meaningless in the product of two 15-bit numbers, the contents of this bit are made to duplicate the sign of the associated product so that this bit does not add significance to the product. The sign of each product is determined by the usual rule of algebraic sign manipulation.
Twin and Multiply	*TMU*	254	The left half-word (contained in the memory location specified by the address part of the instruction and the selected index register) is multiplied by the contents of the left and right accumulator registers to form two 30-bit products plus the associated signs. The left and right products are contained in the two 32-bit registers formed by the left accumulator and left B registers and the right accumulator and right B registers, respectively. The sign of each product appears in bit

TABLE 1–7. MULTIPLY CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
			5 of the associated accumulator register, and the 30-bit product appears in bits 1 through 15 and S through 14 of the associated accumulator and B registers, respectively. Bit 15 of the B register actually forms the 31st bit of the product. This bit is meaningless in the product of two 15-bit numbers. The contents of this bit are made to duplicate the sign of the associated product so that this bit does not add significance to the product. The sign of each product is determined by the usual rule of algebraic sign manipulation.
Divide	*DVD*	260	The left and right half-words (contained in the memory location specified by the address part of the instruction and the selected index register) are divided into the contents of the 32-bit left accumulator B register and the 32-bit right accumulator B register, respectively, to form two 16-bit quotients plus the associated signs. The sign of each quotient appears in bit S of the associated accumulator register, the 16-bit quotients appear in the associated B registers, and the remainders appear in bits 1 through 15 of the associated accumulator register. The sign of each quotient is determined by the usual rule of algebraic manipulation.
Twin and Divide	*TDV*	264	The left half-word (contained in the memory location specified by the address part of the instruction and selectd index register) is divided into the contents of the 32-bit left accumulator B register and the 32-bit right accumulator B register to form two 16-bit quotients plus the associated signs and remainders. The sign of each quotient appears in bit S of the associated accumulator register, the 16-bit quotients appear in the associated B registers, and the remainders appear in bits 1 through 15 of the associated accumulator register. The sign of each quotient is determined by the usual rule of algebraic manipulation.

5.4 STORE CLASS

Each of the seven instructions grouped within this class involves the transfer and storage of a word from the arithmetic element into a specified location in the memory element. Some of these instructions only store a word in the memory element; others obtain a word from the memory element, alter it by specific operations, and store the altered word in the original location in the memory element. Each instruction within this class is indexable. The *RST, STA,* and *AOR* instructions can be controlled to provide either 16- or 17-bit arithmetic operation. All instructions in this class except the *FST* instruction require a PT cycle, an OTA cycle, and an OTB cycle for its execution. The *FST* instruction requires only a PT cycle and an OTB cycle. The store-class instructions and their functions are listed in table 1–8.

TABLE 1–8. STORE CLASS INSTRUCTIONS

NAME	SYMBOL	OCTAL CODE	FUNCTION
Full Store	*FST*	324	The contents of the left and right accumulator registers are stored in the memory location specified by the address part of the instruction and the selected index register. The contents of the accumulator registers are not changed, but the original contents of the specified memory location are destroyed.

TABLE 1–8. STORE CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
Left Store	*LST*	330	The contents of the left accumulator register replace the left half-word in the memory location specified by the address part of the instruction and the selected index register. The contents of the accumulator registers and the right half-word of the specified memory location are not changed, but the original contents of the left half-word of the specified memory location are destroyed.
Right Store	*RST*	334	This instruction option is used for data manipulation. The contents of the right accumulator register replace the right half-word in the memory location specified by the address part of the instruction and the selected index register. The contents of the accumulator registers and the left half-word of the specified memory location are not changed, but the original contents of the right half-word of the specified memory location are destroyed.
	RST 10	335	This instruction option is used for instruction address modification. The contents of accumulator bit LS and RS through R15 replace bits LS and RS through R15 of the memory word specified by the address part of the instruction and the selected index register. The contents of the accumulator registers and bits L1 through L15 of the specified memory word are not changed, but the original contents of bits LS and RS through R15 of the specified memory word are destroyed.
Store Address	*STA*	340	This instruction option which deals with 16-bit addresses is applicable for AN/FSQ-8 operation only. The contents of bits RS through R15 of the right A register replace the right half-word contained in the memory location specified by the address part of the instruction and the selected index register. The contents of the A register and the left half-word in the specified right A memory location are not changed, but the original contents of the right half-word in the specified memory location are destroyed. This instruction usually follows a branch-class instruction because, as a result of a successful branch, the contents of the program counter are transferred to the right A register.
	STA 10	341	This instruction option which deals with 17-bit addresses is applicable for AN/FSQ-7 operation only. The contents of bits LS and RS through R15 of the right A register replace bits LS and RS through R15 of the memory word specified by the address part of the instruction and the selected index register. The contents of the right A register and bits L1 through L15 of the specified memory word are not changed, but the original contents of bits LS and RS through R15 of the specified memory word are destroyed. This instruction usually follows a branch-class instruction because, as a result of a successful branch, the contents of the program counter are transferred to the right A register.

TABLE 1–8. STORE CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
Add One Right	*AOR*	344	This instruction option is used when it is desired to use a memory register as a counter. During execution of this instruction, the memory word specified by the address part of the instruction and the selected index register is transferred to the arithmetic units, where the number contained in the right half-word is increased by a factor of 1. The modified number is then stored in the right half-word of the original memory location. The left-half memory word is not changed by this instruction. Execution of this instruction can cause a right overflow.
	AOR 10	345	This instruction option is used for instruction address modification. During execution of this instruction, the memory word (specified by the address part of the instruction and the selected index register) is transferred to the arithmetic units, where the number contained in bits LS and RS through R15 is increased by a factor of 1. The modified number is then stored in bits LS and RS through R15 of the specified location. Bits L1 through L15 of the original memory word are not changed by the instruction. Execution of this instruction can cause a left overflow.
Exchange	*ECH*	350	The contents of the left and right accumulator registers are stored in the memory location specified by the address part of the instruction and the selected index register. The accumulator registers are then cleared and the original contents of the specified memory location, which were temporarily stored in the A registers, are added to the accumulator registers.
Deposit	*DEP*	360	During the execution of this instruction, the contents of certain bits of the memory word (specified by the address part of the instruction and the selected index register) are replaced by the contents of the associated bits in the left and right accumulator registers. Since the contents of the B register determine which bits of the memory word are to be affected, the B registers must be loaded with the control mask prior to the execution of this instruction. Each bit in the B registers which contains a 1 designates that the corresponding accumulator register bit will replace the corresponding bit in the selected memory word.

5.5 SHIFT CLASS

Eight instructions are contained within the shift class, each of which involves shifting the contents of the accumulator registers or the combined accumulator and B registers to the left or right. The number of shifts to be performed during the execution of each instruction is specified by bits R10 through R15 of the instruction word. These six bits are loaded into the step counter, which keeps a count of the number of shifts to be performed and synchronizes the initiation and termination of the arithmetic pause. Although the indexing controls are conditioned during the execution of any shift class instruction, indexing is meaningless for this class. Each of these instructions requires a PT cycle and an arithmetic pause (if more than six shifts are to be performed) for its execution. The shift-class instructions and their functions are listed in table 1–9.

TABLE 1-9. SHIFT CLASS INSTRUCTIONS

NAME	SYMBOL	OCTAL CODE	FUNCTION
Dual Shift Left	*DSL*	400	The left accumulator and left B register and the right accumulator and right B register are formed into two separate 31-bit shifting registers, with connections established so that the contents of each bit position may be shifted to the next left bit position. The sign bit of each B register is connected to bit 15 of the corresponding accumulator register. As each shift is performed, the contents of accumulator bits L1 and R1 are lost and the contents of B register bits L15 and R15 are replaced by the contents of the associated accumulator sign bits. The contents of the accumulator sign bits are not affected by this instruction.
Dual Shift Right	*DSR*	404	The left accumulator and left B register and the right accumulator and right B registers are formed into two separate 31-bit shifting registers, with connections established so that the contents of each bit position may be shifted to the next right bit position. Bit 15 of each accumulator register is connected to the sign bit of the corresponding B register. As each shift is performed, the contents of B register bits L15 and R15 are lost. The contents of the accumulator sign bits are not affected by this instruction.
Shift Accumulators Left	*ASL*	420	The left and right accumulator registers are treated as two separate 15-bit shifting registers, with connections established so that the contents of each bit position may be shifted to the next left bit position. As each shift is performed, the contents of accumulator bits L1 and R1 are lost and the contents of accumulator bits L15 and R15 are replaced by the contents of the associated accumulator sign bits. The contents of the accumulator sign bits are not affected by this instruction.
Shift Accumulator Right	*ASR*	424	The left and right accumulator registers are treated as two separate 15-bit shifting registers, with connections established so that the contents of each bit position may be shifted to the next right bit position. As each shift is performed, the contents of accumulator bits L15 and R15 are lost. The contents of the accumulator sign bits are not affected by this instruction.
Left Element Shift Right	*LSR*	440	The left accumulator and left B register form a single 31-bit shifting register, with connections established so that the contents of each bit position may be shifted to the next right bit position. Bit 15 of the left accumulator register is connected to the sign bit of the left B register. As each shift is performed, the contents of left B register bit 15 are lost. The contents of the left accumulator sign bit are not affected by this instruction.
Right Element Shift Right	*RSR*	444	The right accumulator and right B register form a single 31-bit shifting register, with connections established so that the contents of each bit position may be shifted to the next right bit position. Bit 15 of the right accumulator register is connected to the sign bit of the right B register. As each shift is performed, the contents of right B register bit 15 are lost. The contents of the right accumulator sign bit are not affected by this instruction.

TABLE 1–9. SHIFT CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
Dual Cycle Left	*DCL*	460	The left accumulator and left B register and the right accumulator and right B register are formed into two separate 32-bit shifting rings, with connections established so that the contents of each bit position may be shifted to the next bit position. The sign bit of each B register is connected to bit 15 of the associated accumulator, and the sign bit of each accumulator register is connected to bit 15 of the associated B register.
Full Cycle Left	*FCL*	470	The left and right accumulator registers are formed into a single 32-bit shifting ring, with connections established so that the contents of each bit position may be shifted to the next left bit position. The sign bit of the right accumulator register is connected to bit 15 of the left accumulator register, and the sign bit of the left accumulator register is connected to bit 15 of the right accumulator register. This instruction can be used to cause the contents of the left and right accumulator registers to be interchanged by specifying $20_{(8)}$ shifts.

5.6 BRANCH CLASS

Six instructions are grouped within this class, all of which are capable of altering the usual sequential execution of the stored program instructions. In normal operation, the program counter in the program element specifies the memory address from which the next instruction is to be obtained. However, when certain conditions exist in the equipment, it may be desirable to execute a different sequence of instructions stored in a different group of memory locations. Branch-class instructions provide means of sensing various conditions in the computer which might indicate that the program sequence is to be altered or that a branch of program control is to be performed. To accomplish this, each branch-class instruction senses to determine whether a particular condition is present. If the specific condition is absent, the instruction has no effect and the program continues sequentially. However, if the specified condition is found to be present, a branch of program control is executed. During a branch operation, the contents of the program counter, which is numerically 1 greater than the address from which the branch-class instruction was obtained, are transferred to the right A register in the arithmetic element. The contents of the address register, which specify the memory address from which the next instruction is to be obtained, are then transferred to the cleared program counter and the memory address registers. Selection of a specified memory device causes the next instruction to be obtained from the memory address specified by the address part of the branch instruction and succeeding instructions to be obtained from the next sequential addresses. The original contents of the program counter, which are now in the right A register, may be preserved in the memory element by a *Store Address* instruction. Indexing is not possible in this class of instructions. All instructions in this class, with the exception of *BSN* and *BFZ*, require only a PT cycle for their execution; the *BSN* and *BFZ* instructions require a PT cycle and an OT cycle. The branch-class instructions and their functions are listed in table 1–10.

TABLE 1–10. BRANCH CLASS INSTRUCTIONS

NAME	SYMBOL	OCTAL CODE	FUNCTION
Branch and Index	*BPX*	51– –	This instruction may be used to provide either a conditional or an unconditional branch of program control, depending upon the contents of index selection bits L1 through L3 of the instruction word. If bits L1 through L3 specify one of the four physical index registers (IX 1, IX 2, IX 4, or IX 5), a conditional branch of control is implied and the sign bit of the selected index register will determine what action will be taken. If the sign bit is positive (contains a 0), the contents of index interval bits L10 through L15 of the instruction word

TABLE 1-10. BRANCH CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
			are subtracted from the contents of the selected index register, and the computer branches program control to the instruction contained in the memory address specified by the address part of the instruction. However, if the sign bit of the selected index register is negative (contains a 1), the conditions for branching are not met, the contents of the selected index register are not modified, and the program continues with the next sequential instruction. If bits L1 through L3 of the instruction word specify that an index register is not to be sensed, or that the right accumulator register is selected as index register 3, an unconditional branch of program control will result. In this event, the contents of index interval bits L10 through L15 of the instruction word are not used, and the computer branches program control to the instruction contained in the memory location specified by the address part of this instruction. If bits L1 through L3 of the instruction word specified index register 6 or 7 (which do not exist), the instruction is effectively voided and the program continues with the next sequential instruction.
Branch on Sense	*BSN*	52- -	This instruction enables the computer to determine the status of any one of a number of control circuits which are defined as sense units. During the execution of this instruction, the condition of the selected sense unit, as specified by the contents of index interval bits L10 through L15 of the instruction word, is sensed to determine what action will be taken. If the specified condition is present, a branch of program control is executed to the instruction contained in the memory location specified by the address part of the instruction. If the specified condition is not present, the instruction is ineffective and the program continues with the next sequential instruction.
Branch on Full Zero	*BFZ*	540	If the contents of the left and right accumulator registers are both zero, execution of this instruction will cause the computer to branch program control to the instruction contained in the memory location specified by the address part of this instruction. It should be noted that the term zero implies either positive or negative zero. Thus, a branch of control will be executed if the contents of both accumulator registers are either positive or negative zero, or if one accumulator register contains a positive zero and the other contains a negative zero. If the condition for causing a branch of control does not exist, this instruction is ineffective and the program continues with the next sequential instruction.
Branch on Full Minus	*BFM*	544	If the sign bits of the numbers contained in the left and right accumulator registers are both negative, execution of this instruction will cause the computer to branch program control to the instruction contained in the memory location specified by the address part of the instruction. If the condition for causing a branch of control does not exist, this instruction is ineffective and the program continues with the next sequential instruction.

TABLE 1–10. BRANCH CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
Branch on Left Minus	*BLM*	550	If the sign bit of the number contained in the left accumulator register is negative, execution of this instruction will cause the computer to branch program control to the instruction contained in the memory location specified by the address part of the instruction. If the condition for causing a branch of control does not exist, this instruction is ineffective and the program continues with the next sequential instruction.
Branch on Right Minus	*BRM*	554	If the sign bit of the number contained in the right accumulator register is negative, execution of this instruction will cause the computer to branch program control to the instruction contained in the memory location specified by the address part of the instruction. If the condition for causing a branch of control does not exist, this instruction is ineffective and the program continues with the next sequential instruction.

5.7 INPUT-OUTPUT CLASS

The five instructions grouped within the IO class deal with the programming of information transfers between the computer and other equipment in the AN/FSQ-7. Each instruction in the IO class is dependent upon the status of the IO interlock in the instruction control element for its execution. The IO interlock is turned on whenever the transfer of words into or out of memory is initiated, and turned off after all IO transfers are completed. Since the execution of any IO class instruction can interfere with the execution of a previously programmed IO operation, an IO class instruction is not executed if the IO interlock is on. Thus, if an IO class instruction appears in the program when the IO interlock is on, its execution and the execution of succeeding instructions are delayed until the IO interlock goes off. This delay is called an IO pause. If the IO interlock is off, an IO class instruction is executed as soon as it appears in the program. Although the indexing controls are conditioned during the execution of an IO class instruction, indexing is meaningless for the execution of the *SEL* instruction. All instructions, with the exception of *SDR* and *SEL*, require only a PT cycle for their execution; the *SDR* and *SEL* instructions require a PT cycle and an OT cycle. The IO class instructions and their functions are listed in table 1–11.

TABLE 1–11. INPUT-OUTPUT CLASS INSTRUCTIONS

NAME	SYMBOL	OCTAL CODE	FUNCTION
Load IO Address Counter	*LDC*	600	The IO address counter is cleared, and the number specified by the address part of the instruction and the selected index register is transferred to the IO address counter. The contents of this counter specify the starting memory address that will be used in the subsequent IO operation. As each data word is transferred, the contents of this counter will be increased by 1.
Select Drum	*SDR*	61– –	Execution of this instruction deselects the previously selected IO device and sets the computer and drum selection and transfer controls for the subsequent drum IO operation. Bit R1 of the instruction word specifies whether the main drum or the auxiliary Drum System is selected, and bits L10 through L15 specify which field in the selected Drum System is selected. If an addressable drum field is selected, the address part of the instruction and the selected index register specify the starting drum address. If a status drum field is to be read by an identity code, the address part of the instruction specifies the

TABLE 1–11. INPUT-OUTPUT CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
			identification code to be used. In either case, the pertinent information is transferred to the drum control register to provide the desired control.
Select	*SEL*	62– –	Execution of this instruction deselects the previously selected IO device and sets the computer and IO unit selection and transfer controls (with the exception of the drum controls which are set by the *Select Drum* instruction) for the subsequent IO operation. Bits L10 through L15 of the instruction word are used to specify which of the IO units is selected. The address part of this instruction is meaningless.
Read	*RDS*	670	Execution of this instruction sets the IO interlock and initiates the transfer of data from the selected IO device according to the controls set up by the *Load IO Address Counter* and *Select* or *Select Drum* instructions. If an addressable drum field had been previously selected, and an interleave mode of operation is required, bits L13 through L15 of the instruction word specify which mode of interleaving (by 8, by 16, or by 64) is to be used. The maximum number of words to be transferred during the subsequent IO operation is specified by the address part of the instruction and the selected index register. The complement of this number is stored in the IO word counter and, as each IO word is transferred, the contents of this counter are increased by 1. The IO operation is terminated and the IO interlock is cleared by either an IO-word-counter-end-carry pulse or by a disconnect signal from the selected input unit.
Write	*WRT*	674	Execution of this instruction sets the IO interlock and initiates the transfer of data to the selected IO device according to the controls set up by the *Load IO Address Counter* and the *Select* or *Select Drum* instructions. If an addressable drum field has been previously selected and an interleave mode of operation is required, bits L13 through L15 will specify which mode of interleaving (by 8, by 16, or by 64) is to be used. The maximum number of words to be transferred during the subsequent IO operation is specified by the address part of the instruction and the selected index register. The complement of this number is stored in the IO word counter and, as each IO word is transferred, the contents of this counter are increased by 1. The IO operation is terminated and the IO interlock is cleared by either an IO-word-counter-end-carry pulse or by a disconnect signal from the selected output unit.

5.8 RESET CLASS

Each of the three instructions of this class involves loading the selected index registers of the program element with a specific numerical value. These instructions are used to set up the computer for the performance of iterative program loops. Each of the three instructions is completely executed in a PT cycle. Indexing is not possible during the execution of an instruction in this class. The *XAC* instruction can be controlled to provide either dual or 17-bit arithmetic operation. The reset-class instructions and their functions are listed in table 1–12.

TABLE 1–12. RESET CLASS INSTRUCTIONS

NAME	SYMBOL	OCTAL CODE	FUNCTION
Reset Index Register	*XIN*	754	The contents of the index register specified by index selection bits L1 through L3 of the instruction word are replaced by the number contained in the address part of the instruction. The right accumulator may not be used as an index register for this instruction.
Reset Index Register from Right Accumulator	*XAC*	764	This instruction option is applicable for AN/FSQ-8 operation only. The contents of the index register specified by index selection bits L1 through L3 of the instruction word are replaced by the number contained in the right accumulator register. The address part of this instruction is meaningless.
	XAC 10	765	This instruction option is applicable for AN/FSQ-7 operation only. The contents of the index register specified by index selection bits L1 through L3 of the instruction word are replaced by the number contained in bits LS and RS through R15 of the accumulator registers. The address part of this instruction is meaningless.
Add Index Register	*ADX*	770	The contents of the index register specified by index selection bits L1 through L3 of the instruction word are added to the address part of the instruction. The sum is then placed in the right A register. If bits L1 through L3 specify that an index register is not selected, this instruction will cause the address part of the instruction to be placed in the right A register.

5.9 MISCELLANEOUS CLASS

The miscellaneous class contains the 16 instructions which, because of peculiarities of timing or operation, do not properly fit into any of the other classes. Although the indexing controls are conditioned during the execution of any miscellaneous class instruction, indexing is meaningless during the execution of the *HLT, PER, CSW,* and *SLR* instructions. The *CSW* instruction is the only instruction in this class that can be controlled to provide either dual or 17-bit arithmetic operation. The miscellaneous class instructions and their functions are listed in table 1–13. Information concerning the machine cycles required for the execution of these instructions is included in the function of each instruction.

TABLE 1–13. MISCELLANEOUS CLASS INSTRUCTIONS

NAME	SYMBOL	OCTAL CODE	FUNCTION
Program Halt	*HLT*	000	This instruction is used to stop computer operations. If an IO operation is in process during the execution of this instruction (that is, if the IO interlock is on), execution of this instruction will result in an IO pause and the instruction will not be completed until the IO operation is terminated. The address part of this instruction is meaningless. This instruction requires a PT and an OT cycle for its execution. To resume computer operation from the next sequential program instruction, manual intervention is necessary in that the operator must depress the PROGRAM CONTINUE pushbutton.
Extract	*ETR*	004	During the execution of this instruction, the contents of the accumulator registers are logically multiplied by the contents of the memory word specified by the address part of the instruction and the selected index register. As a result, individual

TABLE 1–13. MISCELLANEOUS CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
			accumulator bits will contain a 1 only if these bits initially contained a 1 and the corresponding bits of the memory word also contained a 1; all other accumulator bits will contain 0's. The contents of the specified memory word are not changed by this instruction. This instruction requires a PT and an OT cycle for its execution.
Operate	*PER*	01	This instruction enables the computer to perform an operation on any one of a number of control circuits which are defined as operate units. During the execution of this instruction, a control pulse is applied to the operate unit specified by the contents of bits L10 through L15 of the instruction word. The address part of this instruction is meaningless. This instruction requires a PT and an OT cycle for its execution.
Clear and Subtract Word Counter	*CSW*	020	This instruction option is applicable for AN/FSQ-8 operation only. The instruction is used primarily to determine the number of words that have been transferred to or from the main Drum System during a previously programmed IO operation when the status or identity mode of word transfer had been specified. In preparing the computer for this type of word transfer, the IO word counter is initially set to the complement of $0.20000_{(8)}$ and, as each IO word is transferred, a 1 is added to the contents of this counter. During the execution of this instruction, the contents of the IO word counter are transferred to the right accumulator register so that they can be mathematically manipulated to determine the number of words that have been transferred during the IO operation. Since execution of this instruction is not affected by the status of the IO interlock, this instruction may be used while the IO operation is still in process. The address part of this instruction is meaningless. This instruction requires a PT cycle for its execution.
	CSW 10	021	This instruction option is applicable for AN/FSQ-7 operation only. The instruction is primarily used to determine the number of words that have been transferred to or from the main Drum System during a previously programmed IO operation when the status or identity mode of word transfer had been specified. In preparing the computer for this type of word transfer, the IO word counter is initially set to the complement of $0.20000_{(8)}$ and, as each IO word is transferred, a 1 is added to the contents of this counter. During the execution of this instruction, the contents of the IO word counter are transferred to bits LS and RS through R15 of the accumulator registers so that they can be mathematically manipulated to determine the number of words that have been transferred during the IO operation. Since execution of this instruction is not affected by the status of the IO interlock, this instruction may be used while the IO operation is still in process. The address part of this instruction is meaningless. This instruction requires a PT cycle for its execution.

TABLE 1–13. MISCELLANEOUS CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
Shift Left and Round	*SLR*	*024*	This instruction is used after a multiply or a divide instruction when it is desired to round off the left and right products or quotients to the 15 most significant bits plus sign. For the shift operation, the left accumulator and left B register and the right accumulator and right B register are formed into two separate 31-bit shifting registers, with connections established so that the contents of each bit position may be shifted to the next left bit position. The sign bit of each B register is connected to bit 15 of the corresponding accumulator register. As each shift is performed, the contents of accumulator bits L1 and R1 are lost and the contents of B register bits L15 and R15 are replaced by the contents of the associated accumulator sign bit. The contents of the accumulator sign bits are not affected by the shift operation. The number of shifts to be performed is specified by the address part of the instruction. The shift left operation (if required) is followed by the roundoff operation, in which first the left and right products or quotients are put into positive form and then a 1 is added to the contents of the accumulator registers if the associated B register sign bit contains a 1. Both B registers are cleared at the completion of this instruction. Execution of this instruction can cause an overflow. This instruction requires a PT cycle and an arithmetic pause (if any shifts are required) for its execution.
Load B Registers	*LDB*	030	The contents of the left and right B register are replaced by the contents of the memory word specified by the address part of this instruction and the selected index register. This instruction requires a PT and an OT cycle for its execution.
Compare Masked Bits	*CMM*	040	This instruction is used to compare specific bits of the accumulator registers with the corresponding bits of the memory word specified by the address part of the instruction and the selected index register. Prior to execution of this instruction, the B registers must be loaded with a mask to specify which bits are to be compared. During the execution of the *CMM* instruction, the accumulator registers are complemented and the contents of the B registers are transferred to the associated A registers. The contents of the accumulator registers are then logically multiplied by the contents of the A registers so that the unmasked accumulator bits will all contain 0's. The A registers are then complemented, and the contents of the specified memory word are logically added to the A registers, with the result that the unmasked A register bits will contain 1's. A comparison is then made between the contents of each A register and its associated accumulator register. If the masked bits do not compare, the program counter is stepped once more, in addition to the normal stepping, so that the next sequential program instruction will be skipped. The accumulator registers are then recomplemented so that the final contents of the masked bits will be identical with the original contents. The unmasked accumulator bits will contain 1's. This instruction requires a PT and an OT cycle for its execution.

TABLE 1–13. MISCELLANEOUS CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
Compare Difference Masked Bits	*CDM*	041	This instruction is used to compare, and to obtain the difference between, specific bits of the accumulator registers and the corresponding bits of the memory word specified by the address part of the instruction and the selected index register. Prior to execution of this instruction, the B registers must be loaded with a mask to specify which bits are to be compared. During the execution of the *CDM* instruction, the accumulator registers are complemented and the contents of the B registers are transferred to the associated A registers. The contents of the accumulator registers are then logically multiplied by the contents of the A registers so that the unmasked accumulator bits will contain 0's. The A registers are then complemented, and the contents of the specified memory word are logically added to the A registers, with the result that the unmasked A register bits will contain 1's. A comparison is then made between the contents of each A register and its associated accumulator register. If the masked bits do not compare, the program counter is stepped once more, in addition to the normal stepping, so that the next sequential program instruction will be skipped. The contents of the A registers are then added to the contents of the associated accumulator registers. The final contents of each accumulator register will be in negative form (sign bit = 1) if the original masked bit contents were equal to or greater than the contents of the associated A register bits. The final contents will be in positive form if the original masked bit contents were less than the contents of the associated A register bits. Because of the possibility of end-carry, overflow, and carry-into-the-sign-bit position, the above general rule about the final contents of the accumulator registers is not necessarily valid if the sign bits are included in the mask; each case must be evaluated separately. This instruction requires a PT and an OT cycle for its execution.
Compare Right Half Word	*CMR*	042	This instruction is used to compare the contents of the right accumulator register and the right-half memory word specified by the address part of the instruction and the selected index register. During the execution of this instruction, the accumulator registers are complemented and the selected memory word is transferred to the A registers. A comparison is then made between the contents of the right A register and right accumulator register. If the right half-words do not compare, the program counter is stepped once more, in addition to the normal stepping, so that the next sequential program instruction will be skipped. The accumulator registers are then recomplemented so that their final contents will be identical with the original contents. This instruction requires a PT and an OT cycle for its execution.
Compare Difference Right Half Word	*CDR*	043	This instruction is used to compare, and to obtain the difference between, the contents of the right accumulator register and the right-half memory word specified by the address part of the instruction and the selected index register. During the execution of this instruction, the accumulator registers are

TABLE 1–13. MISCELLANEOUS CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
			complemented and the selected memory word is transferred to the A registers. A comparison is then made between the contents of the right A register and the right accumulator register. If the right half-words do not compare, the program counter is stepped once more, in addition to the normal stepping, so that the next sequential program instruction will be skipped. The contents of the A registers are then added to the contents of the associated accumulator registers to compute the difference between the words. The final contents of each accumulator register will be in negative form (sign bit = 1) if the original contents were equal to or greater than the contents of the associated A register, or in positive form if the original contents were less than the contents of the associated A register. Execution of this instruction can cause overflow. This instruction requires a PT and an OT cycle for its execution.
Compare Left Half Word	*CML*	044	This instruction is used to compare the contents of the left accumulator register and the left-half memory word specified by the address part of the instruction and the selected index register. During the execution of this instruction, the accumulator registers are complemented and the selected memory word is transferred to the A registers. A comparison is then made between the contents of the left A register and left accumulator register. If the left half-words do not compare, the program counter is stepped once more, in addition to the normal stepping, so that the next sequential program instruction will be skipped. The accumulator registers are then recomplemented so that their final contents will be identical with the original contents. This instruction requires a PT and an OT cycle for its execution.
Compare Difference Left Half Word	*CDL*	045	This instruction is used to compare, and to obtain the difference between, the contents of the left accumulator register and the left-half memory word specified by the address part of the instruction and the selected index register. During the execution of this instruction, the accumulator registers are complemented and the selected memory word is transferred to the A registers. A comparison is then made between the contents of the left A register and the left accumulator register. If the left half-words do not compare, the program counter is stepped once more, in addition to the normal stepping, so that the next sequential program instruction will be skipped. The contents of the A registers are then added to the contents of the associated accumulator registers to compute the difference between the words. The final contents of each accumulator register will be in negative form (sign bit = 1) if the original contents were equal to or greater than the contents of the associated A register, or in positive form if the original contents were less than the contents of the associated A register. Execution of this instruction can cause overflow. This instruction requires a PT and an OT cycle for its execution.

TABLE 1-13. MISCELLANEOUS CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
Compare Full Word	*CMF*	046	This instruction is used to compare the contents of the accumulator registers and the memory word specified by the address part of the instruction and the selected index register. During the execution of this instruction, the accumulator registers are complemented and the selected memory word is transferred to the A registers. A comparison is then made between the contents of each A register and its associated accumulator register. If the two words do not compare, the program counter is stepped once more, in addition to the normal stepping, so that the next sequential program instruction will be skipped. The accumulator registers are then recomplemented so that their final contents will be identical with the original contents. This instruction requires a PT and an OT cycle for its execution.
Compare Difference Full Word	*CDF*	047	This instruction is used to compare, and to obtain the difference between, the contents of the accumulator registers and the memory word specified by the address part of the instruction and the selected index register. During the execution of this instruction, the accumulator registers are complemented and the selected memory word is transferred to the A registers. A comparison is then made between the contents of each A register and its associated accumulator register. If the two words do not compare, the program counter is stepped once more, in addition to the normal stepping, so that the next sequential program instruction will be skipped. The contents of the A registers are then added to the contents of the associated accumulator registers to compute the difference between the two words. The final contents of each accumulator register will be in negative form (sign bit = 1) if the original contents were equal to or greater than the contents of the associated A register, or in positive form if the original contents were less than the contents of the associated A register. Execution of this instruction can cause overflow. This instruction requires a PT and an OT cycle for its execution.
Test One Bit	*TOB*	050 — —	This instruction is used to sense the status of a particular bit of the 32-bit memory word specified by the address part of the instruction and the selected index register. The content of the particular bit, which is specified by bits L11 through L15 of the instruction word, is used to determine whether the next sequential program instruction is to be executed or skipped. If the selected bit contains a 1, the program counter is stepped once more, in addition to the normal stepping, so that the next sequential program instruction will be skipped; if the selected bit contains a 0, the program counter will be stepped normally and the next program instruction will be executed sequentially. This instruction requires a PT and an OT cycle for its execution.
Test Two Bits	*TTB*	054 — —	This instruction is used to sense the status of two adjacent bits of the 32-bit memory word specified by the address part of the instruction and the selected index register. The second of

TABLE 1–13. MISCELLANEOUS CLASS INSTRUCTIONS (cont'd)

NAME	SYMBOL	OCTAL CODE	FUNCTION
			the two bits to be tested is specified by bits L11 through L15 of the instruction word; the first bit to be tested is always situated adjacent to, and to the left of, the selected bit except when either of the sign bits is specified. In this latter case, the *Test Two Bits (TTB)* instruction degenerates into a *Test One Bit (TOB)* instruction and only the specified bit is tested. During the execution of the *TTB* instruction, the contents of the particular bits are used to determine how many of the subsequent program instructions will be skipped. If the two bits under test contain: 00 – The program counter is stepped normally, and the next program instruction is executed sequentially. 01 – The program counter is stepped by 1 in addition to the normal stepping, and the next sequential instruction is skipped. 10 – The program counter is stepped by 2 in addition to the normal stepping, and the next two sequential instructions are skipped. 11 – The program counter is stepped by 3 in addition to the normal stepping, and the next three sequential instructions are skipped. This instruction requires a PT and an OT cycle for its execution.

PART 2

INSTRUCTION CONTROL ELEMENT

CHAPTER 1

INTRODUCTION

1.1 GENERAL

The instruction control element of the Central Computer System of AN/FSQ-7 controls and guides most of the operations of the computer. It does this by generating and distributing command pulses for the execution of all the instructions the computer is capable of performing.

The instruction control element operates in response to a program which has been stored in consecutive addresses of the memory element. Before being loaded into the Central Computer System, the program is punched on IBM binary cards. From the cards, it is loaded into core memory by means of the card reader. The program may be used immediately, or it may be taken from core memory and stored in auxiliary memory drum fields or on magnetic tape for future use. Once the program has been stored in auxiliary memory drum fields, or on magnetic tape, it can be reloaded into the core memory element at the discretion of the operator.

The Central Computer System can execute 59 distinct instructions, which are grouped into eight classes. An instruction class consists of instructions that are similar in their manner of execution. The classes of instructions and their variations are coded in binary notation. To select an instruction, it is necessary to designate both codes.

In the execution of the 59 instructions, the computer is directed by 162 identified commands. A complete list of commands, their functions, and logic reference numbers are given in table 2–1. Throughout the text, commands are generally referred to by number; occasionally, in the interests of clarity, the command name is also included.

TABLE 2–1. COMMAND PULSE ANALYSIS

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
A	*Inhibit sample No. 2*	0.1.4 (A5)	4DTG3	Store-class common command.	Conditional: Set the sample gate generator flip-flop in Memory 2 during OTB cycle.
A6	*Clock register to right memory buffer register*	0.2.6 (A5)	4DTA6	Occurs in *CAC* instruction.	Conditional: Transfer occurs during OTA cycle.
C	*Clear left test register*	0.1.3 (B2)	4DTC7	Store-class common command.	Conditional: Test register is cleared during OTB cycle if test memory is selected.
D	*Left memory buffer register to left test register*	0.1.1 (E9)	4DTA3	Store-class common command.	Conditional: Transfer occurs during OTB cycle if test memory is selected.
F	*Test memory to memory buffer register*	0.1.3 (A3)	4DTB4	Common to all machine cycles when test memory is selected.	Conditional: If test memory is selected, transfer left test register, right test register, and test memory

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
					readout control circuits to memory buffer register.
G	*Clear right test register*	0.1.3 (B2)	4DTC7	Store-class common command.	Conditional: Test register is cleared during OTB cycle if test memory is selected.
H	*Right memory buffer register to right test register*	0.1.1 (E9)	4DTA3	Store-class common command.	Conditional: Transfer occurs during OTB cycle if test memory is selected.
1	*Left accumulator register to left B register*	0.5.1–2 (E19)	4FYA6	Occurs in *DIM*, *MUL*, and *TMU* instructions.	
2	*Left accumulator register bits (2-15) to bits (1-14)*	0.5.1–2 (E21)	4GYC6	Add-class common command; also occurs in *ECH*, *DCL*, *FCL*, *DSL*, *ASL*, *BFZ*, and *SLR* instructions.	
4	*Left accumulator register bit 1 to bit S*	0.5.1–2 (E21)	4GYH6	Add-class common command; also occurs in *ECH*, *DCL*, *FCL*, and *BFZ* instructions.	
5	*Left accumulator register bit S to left B register bit 15*	0.5.1–2 (D19)	4FYB4	Occurs in *DCL*, *DSL*, and *SLR* instructions.	
6	*Left accumulator register bit S to left accumulator register bit 15*	0.5.1–2 (D19)	4GWH7	Occurs in *ASL* instruction.	
7	*Left accumulator register bit 15 to left B register bit S*	0.5.1–2 (D1)	4FTC7	Occurs in *LSR* and *DSR* instructions.	
9	*Left correct sign*	0.5.1–2 (C21)	4GWH3	Multiply-class common command.	Conditional: Sense sign control flip-flop; if set, clear sign control flip-flop, complement left B register, and complement left accumulator register.
10	*Clear left accumulator register*	0.5.1–2 (B21)	4FUJ3	Occurs in *CAD*, *CSU*, *CAM*, *MUL*, *TMU*, and *ECH* instructions.	

TABLE 2-1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
11	*Correct left remainder*	0.5.1 (C5)	4EPB4	Occurs in *DVD* and *TVD* instructions.	Conditional: Sense bit S of left A register; if cleared, pulse carry 0 line of left 15 adder, and transfer left B register bit S to left B register S storage.
12	*Left accumulator register bit S to right accumulator register bit 15*	0.5.1-2 (D19)	4FYA3	Occurs in *FCL* instruction.	
13	*Make left accumulator register positive*	0.5.1-2 (B21)	4FUJ3	Occurs in *DIM*, *MUL*, *TMU*, and *BFZ* instructions.	Conditional: Sense bit S of accumulator register; if set, complement sign control flip-flop and left accumulator register.
14	*Right end-carry after add 1*	0.3.1 (E3)	4FGF2	Occurs in *AOR* instruction.	Conditional: Sense carry storage flip-flop; if set, clear carry storage flip-flop, and set right accumulator register bit 15.
16	*Make left accumulator B register positive*	0.5.1-2 (D21)	4FVJ1	Occurs in *DVD*, *TDV*, and *SLR* instructions.	Conditional: Sense bit S of left accumulator register; if set, complement sign control flip-flop, left accumulator register, and left B register.
17	*Left accumulator register to left memory buffer register*	0.5.1-2 (E19)	4FWG3	Occurs in *FST*, *LST*, *ECH*, and *DEP* instructions.	
18	*Ripple left accumulator register right*	0.5.1-2 (D1)	4FTA3	Occurs in *LSR*, *ASR*, and *DSR* instructions.	
19	*Complement left accumulator register*	0.5.1-2 (B21)	4FUC6	Occurs in *DEP* and *BFZ* instructions.	
20	*Make right accumulator-B register positive*	0.5.2-2 (C22)	4FVF6	Occurs in *DVD*, *TDV*, and *SLR* instructions.	Conditional: Sense bit S of right accumulator register; if set, complement sign control flip-flop, right accumulator register, and right B register.
21	*Clear left A register*	0.5.1 (B4)	4EMH6	Add-class, multiply-class, and branch-class common command; also occurs in *RST*, *STA*, *AOR*, *ECH*, *DEP*, *LST*, *SLR*, and *ETR* instructions.	

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
22	*Make left A register positive*	0.5.1 (D5)	4ELC7	Multiply-class common command; also occurs in *CAM* and *DIM* instructions.	Conditional: Sense bit S of left A register; if set, complement sign control flip-flop and left A register.
23	*Left A register to left memory buffer register*	0.5.1 (D5)	4ELA6	Occurs in *RST*, *STA*, and *AOR* instructions.	If 17-bit operation is specified, transfer involves bits L1-L15 only.
25	*Left logical multiply*	0.5.1 (D4)	4EMJ1	Occurs in *DEP* and *ETR* instructions.	Transfer 0 side of left A register to left accumulator register.
26	*Complement left A register*	0.5.1 (B5)	4ELH3	Occurs in *CSU*, *SUB*, *TSU*, and *DIM* instructions.	
31	*Clear memory 2 controls*	0.1.4 (B5)	4DVF6	Common command.	Clear memory 2 address register and sample gate generator flip-flop.
31A	*Clear test memory address register*	0.1.3 (A16)	4DVE6	Common command.	
33	*Inhibit sample No. 1*	0.1.4 (A5)	4DTG3	Store-class common command.	Conditional: Set sample gate generator flip-flop in memory 1 during OTB cycle.
39	*Left memory buffer register to left B register*	0.1.1 (D9)	4DXA6	Occurs in *LDB* instruction.	
40	*Right memory buffer register to right B register*	0.1.1 (D9)	4DXA6	Occurs in *LDB* instruction.	
41	*Clear left memory buffer register*	0.1.1 (A9)	4DYJ3	Common command.	Clear parity write flip-flop, parity check flip-flop, and left memory buffer register.
42	*Left memory buffer register to operation register*	0.1.1 (D9)	4DXH7	Common command.	Transfer involves bits L1-L15 only.
43	*Left memory buffer register to left A register*	0.1.1 (D9)	4DXG3	Add-class, multiply-class, and store-class common command; also occurs in *ETR* instruction.	
44	*Left memory buffer register to right A register*	0.1.1 (D8)	4DXH3	Occurs in *TAD*, *TSU*, *TMU*, and *TDV* instructions.	

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
47	*Memory parity count*	0.1.2 (C1)	5EUC6	Common command.	Set parity check flip-flop, and initiate a parity count on memory buffer register.
51	*Right memory buffer register to right A register*	0.1.2 (D8)	4DXE7	Occurs in *CAD, ADD, CSU, SUB, CAM, DIM, MUL, DVD, AOR, ECH, LST, DEP,* and *ETR* instructions.	
52	*Right memory buffer register to address register*	0.1.2 (D8)	4DXB4	Common command.	Transfer bits LS, RS-R15 to address register and bits R10-R15 to step counter.
53	*Clear right memory buffer register*	0.1.2 (B8)	4DYJ1	Common command.	
55	*Parity check*	0.1.1 (C9)	4DXA3	Common command.	If core memory is selected, sense parity check flip-flop; if set, set memory parity error and error indicator flip-flops. If autobranch flip-flop is set and computer is active, set alarm 1 indicator in other computer. Sense MEMORY PARITY switch. If it is active and autobranch flip-flop is set: store parity word in test register, execute alarm disconnect, and sense STOP-BRANCH switch. If set to branch, execute alarm branch (that is, clear right A register, execute an alarm branch delayed control clear, transfer program counter to right A register, and set program counter to $3.77770_{(8)}$). Computer will resume operation from test memory address.
60	*Left accumulator register conditional shift left*	0.5.1–2 (A18)	4ERE7	Add-class common command; also occurs in *DVD, TDV,* and *SLR* instructions.	Conditional: Sense carry storage flip-flop; if set, clear carry storage flip-flop, and shift left accumulator register bits 1-15

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
					to bits S-14, shift left B register bit S to left accumulator register bit 15, and transfer left B register S storage flip-flop to left B register bit S.
61	*Make left accumulator register and left A register signs unlike*	0.5.1–2 (A17)	4ERB4	Occurs in *DVD* and *TDV* instructions.	Conditional: Sense left sign adder; if signs are alike, complement left A register.
62	*Left carry 1*	0.5.1–2 (A1)	4ESJ1	Occurs in *BFZ* instruction.	Pulse left 15 adder, and transfer left B register bit S to left B register S storage flip-flop.
63	*Left end-carry*	0.5.1–2 (A17)	4ERA6	Add-class common command.	Conditional: Sense carry storage flip-flop. If set, sense for 16- or 17-bit operation. If 16-bit operation, pulse carry 1 line of left 15 adder, and transfer left B register bit S to left B register S storage. If 17-bit operation, pulse carry 0 line of left 15 adder, transfer left B register bit S to left B register sign storage, pulse carry 1 line of right 15 adder, and transfer right B register bit S to right B register sign storage.
64	*Left carry 0*	0.5.1–2 (A1)	4ESJ3	Add-class common command; also occurs in *ECH* instruction.	Pulse left 15 adder, and transfer left B register bit S to left B register S storage flip-flop.
66	*Record left overflow*	0.5.1–2 (B22)	4ERA3	Common command.	Clear sign control flip-flop, and sense and clear auxiliary overflow flip-flop. If auxiliary overflow flip-flop is set, sense index interval register bit 13. If bit 13 contains a 0, set left overflow alarm and left overflow alarm indicator flip-flops, and sense index interval register bit 14. If bit 14 contains a 1, sense the OVERFLOW

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
					switch; if it is set, sense auto-branch control flip-flop; if set, sense STOP-BRANCH switch. If this switch is set, execute an alarm branch; that is, clear right A register, execute an alarm branch delayed control clear, transfer program counter to right A register, and set program counter to $3.77770_{(8)}$. Computer will resume operation from test memory address.
67	*Complement left divide connect flip-flop*	0.5.1–2 (A19)	4ERC7	Occurs in *DVD* and *TDV* instructions.	Clear carry storage flip-flop and auxiliary overflow flip-flop, and complement divide connect flip-flop.
69	*Right carry 1*	0.5.2–2 (A1)	4ESH6	Occurs in *AOR* and *BFZ* instructions.	Pulse right 15 adder, and transfer right B register bit S to right B register S storage flip-flop.
71	*Address register to memory address registers*	0.4.1 (C11)	4EHC7	Add-class, multiply-class, and store-class common command; also occurs in *BSN, SEL, SDR, ETR, HLT, PER,* and *LDB* instructions.	Sense bit LS of address register: 1. If LS = 0: start memory 1, set parity check-start test memory flip-flop, and transfer address register to core memory address registers 1 and 2 and to test memory address register. 2. If address register LS = 1: start either memory 2 or test memory, depending on bits RS through R11 of address word. If any are 0, start memory 2; if all bits are 1, start test memory. Set parity check-start test memory flip-flop, and transfer address register to core memory address registers 1

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
					and 2 and to test memory address register.
72	*Address register to IO address counter*	0.4.1 (C14)	4EHH3	Occurs in *LDC* instruction.	
73	*Subtract 1 from step counter*	0.5.3 (D2)	4FPA6	Shift-class common command; also occurs in *MUL*, *TMU*, and *SLR* instructions.	
74	*Set step counter to 17*	0.5.3 (B2)	4FMH3	Occurs in *DVD* and *TDV* instructions.	
75	*Set step counter to 15*	0.5.3 (B3)	4FMG3	Occurs in *MUL* and *TMU* instructions.	
76	*Address register to right A register*	0.4.1 (C14)	4EHJ3	Occurs in *ADX* instruction.	
77	*Clear address register*	0.4.1 (B13)	4EJC6	Common command.	
	Clear step counter	0.5.3 (B2)	4FMH7	Common command.	Clear step counter and divide time pulse distributor.
78	*Address register to index register No. 2*	0.4.1 (E12)	4EKH3	Occurs in *BPX*, *XIN*, and *XAC* instructions.	
79	*Address register to index register No. 1*	0.4.1 (E12)	4EKG3	Occurs in *BPX*, *XIN*, and *XAC* instructions.	
80	*Partial quotient*	0.5.3 (C2)	4FRA6	Occurs in *DVD* and *TDV* instructions.	Step divide time pulse distributor, and generate divide time pulses. DTP 0 pulse will result in following action: Complement left A register if left A register sign and left accumulator sign are alike, shift left accumulator register bit 1 to bit S, shift left accumulator register bits 2-15 to bits 1-14, shift left B register bit S to left accumulator register bit 15, and shift left B register bits 1-15 to bits 1-14. Same action will be performed in right arithmetic unit.

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
					DTP 1 pulse will transfer left B register bit S to left B register S storage and will sense left A register sign bit. If this bit contains a 0, a carry 0 pulse will be developed. If this bit contains a 1, a carry 1 pulse will be developed. These carry pulses will be applied to left 15 adder. Same action will be performed in right arithmetic unit. DTP 2 pulse does not perform any operation. DTP 3 pulse subtracts 1 from step counter. DTP 4 pulse shifts left accumulator register bit 1 to bit S, left accumulator register bits 2-15 to 1-14, left B register bit S to left accumulator register bit 15, and left B register S storage to left B register bit S. Same action is performed in right arithmetic unit.
81	*Left B register bit S to left accumulator register bit 15*	0.5.1–3 (E6)	4FJH6	Add-class common command; also occurs in *ECH*, *DCL*, *DSL*, *BFZ*, and *SLR* instructions.	
82	*Left B register bits (1-15) to bits (S-14)*	0.5.1–3 (D7)	4FHB4	Occurs in *DCL*, *DSL*, and *SLR* instructions.	
83	*Left partial product*	0.5.1–3 (C8)	4FLB4	Occurs in *MUL* and *TMU* instructions.	Shift left B register bits S-14 to bits 1-15, and sense left B register bit 15 flip-flop. If this flip-flop contains a 1, pulse left carry 0 line of bit 15 adder; if this flip-flop contains a 0, shift left accumulator register bits S-14 to 1-15 and shift bit 15 to left B register sign.

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
84	*Clear left B register*	0.5.1–3 (A7)	4FKJ1	Occurs in *DIM*, *MUL*, *TMU*, *LDB*, and *SLR* instructions.	
85	*Left B register bits (S-14) to bits (1-15)*	0.5.1–3 (D6)	4FJF6	Occurs in *LSR* and *DSR* instructions.	
87	*Left round off*	0.5.1–3 (E7)	4FHA3	Occurs in *SLR* instruction.	Sense bit S of left B register; if set, set carry storage flip-flop and pulse carry 1 line to left 15 adder, and transfer left B register bit S to left B register S storage.
88	*Left B register to left A register*	0.5.1–3 (D6)	4FJJ1	Occurs in *ADB* and *DEP* instructions.	
89	*Left B register S storage flip-flop to left B register bit S*	0.5.1–3 (B7)	4FKC6	Add-class common command; also occurs in *ECH* and *BFZ* instructions.	
91	*Program counter to memory address registers*	0.4.1 (A4)	4EHB4	Common command.	If branch flip-flop is cleared, sense program counter bit LS: 1. If LS = 0: start memory 1, set parity check-start test memory flip-flop, and transfer program counter to core memory address registers 1 and 2 and to test memory address register. 2. If LS = 1: start either core memory 2 or test memory, depending on bits RS through R11 of address word. If any are 0, start memory 2; if all bits are 1, start test memory. Set parity check-start test memory flip-flop, and transfer program counter to core memory address registers 1 and 2 and to test memory address register.
92	*Add 1 to program counter*	0.4.1 (B1)	4EHA3	Common command.	

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
93	*Program counter to right A register*	0.4.1 (C5)	4EKE7	Occurs in *BSN*, *BFM*, *BRM*, *BPX*, and *BLM* instructions.	Conditional: If branch flip-flop is set, transfer program counter to right A register, and clear program counter.
95	*Address register to index register No. 4*	0.4.1 (E12)	4EKH7	Occurs in *BPX*, *XIN*, and *XAC* instructions.	
96	*Address register to index register No. 5*	0.4.1 (E10)	4EKJ3	Occurs in *BPX*, *XIN*, and *XAC* instructions.	
97	*Clear bit L10 storage flip-flop*	0.6.2 (A8)	4FLJ3	Occurs during *TOB* and *TTB* instructions.	
98	*Transfer left or right memory buffer to TOB, TTB gates*	0.1.1 (E7)	4EJJ3	Occurs during *TOB* and *TTB* instructions.	On transfer of right or left memory buffer to TOB and TTB gates: Sense L10 bit storage flip-flop. If it contains a 1, step program counter by 2; if it contains a 0, step program counter by 1.
99	*Check for compare*	0.6.2 (E4)	4JPH7	Occurs in *CMM*, *CDM*, *CMR*, *CDR*, *CML*, *CDL*, *CMF*, and *CDF* instructions.	Check left and/or right half-word for compare as specified by compare instruction being executed.
100	*Address register to memory address registers*	0.4.1 (D12)	4EKA6	Branch class common command.	Conditional: If branch flip-flop is set, sense bit LS of address register: 1. If LS = 0: start memory 1, set parity check-start test memory flip-flop, and transfer address register to core memory address registers 1 and 2 and to test memory address register. 2. If address register LS = 1: start either memory 2 or test memory, depending on bits RS through R11 of address word. If any are 0, start memory 2; if all bits are 1, start test

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
					memory. Set parity check-start test memory flip-flop, and transfer address register to core memory address registers 1 and 2 and to test memory address register.
101	*Clear operations register*	0.2.4 (A5)	4FFA6	Common command.	Clear operations register and index interval register.
102	*Set bits L10 and L11 of index interval register to 1's*	0.6.1 (B14)	4FLG3	Occurs during *TOB* and *TTB* instructions.	
103	*Index interval complement to address register*	0.4.1 (A5)	4FDE7	Occurs in *BPX* instruction.	Set address register bits S-9 to 1, and transfer index interval register bits from 0 side to set bits 10-15 to 1 side.
104	*OT9 PER*	0.7.5 (A7)	4FDC7	Occurs in *PER* instruction.	Sense all operate controls, and execute operation specified.
113	*Clear index register No. 1*	0.4.2 (E5)	4EGE7	Occurs in *BPX*, *XIN*, and *XAC* instructions.	
114	*Index register No. 1 to address register*	0.4.2 (E4)	4EFC6	Add-class, multiply-class, store-class, IO-class, and miscellaneous-class common command; also occurs in *BPX* instruction.	Add index register contents to address register through index adders.
115	*Index register No. 2 to address register*	0.4.2 (D4)	4EFF6	Add-class, multiply-class, store-class, IO-class, and miscellaneous-class common command; also occurs in *BPX* instruction.	Add index register contents to address register through index adders.
116	*Clear index register No. 2*	0.4.2 (C5)	4EGG3	Occurs in *BPX*, *XIN*, and *XAC* instructions.	
117	*Index register No. 4 to address register*	0.4.2 (C4)	4EFJ1	Add-class, multiply-class, store-class,	Add index register contents to address register through index adders.

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
				IO-class, and miscellaneous-class common command; also occurs in *BPX* instruction.	
118	*Clear index register No. 4*	0.4.2 (B5)	4EGH3	Occurs in *BPX*, *XIN*, and *XAC* instructions.	
119	*Test index register No. 4 bit S for 0*	0.4.2 (B5)	4EGB4	Occurs in *BPX* instruction.	Sense bit S of index register 4; if cleared, set branch flip-flop.
121	*Right accumulator register to right memory buffer register*	0.5.2–2 (D19)	4FVC6	Occurs in *FST*, *RST*. *AOR*, *ECH*, and *DEP* instructions.	
123	*Ripple right accumulator register right*	0.5.2–2 (D1)	4FYH3	Occurs in *RSR*, *ASR*, and *DSR* instructions.	Ripple shift right accumulator register bits S-14 to bits 1-15.
124	*Index register No. 5 to address register*	0.4.2 (A5)	4EFJ3	Add-class, multiply-class, store-class, IO-class, and miscellaneous-class common command; also occurs in *BPX* instruction.	Add index register contents to address register through index adders.
125	*Clear index register No. 5*	0.4.2 (A5)	4EGH7	Occurs in *BPX*, *XIN*, and *XAC* instructions.	
126	*Test index register No. 5 bit S for 0*	0.4.2 (A5)	4EGJ3	Occurs in *BPX* instruction.	Sense bit S of index register 5; if cleared, set branch flip-flop.
131	*Set PT-OT flip-flop to OT*	0.3.1 (B3)	4FMA6	Add-class, multiply-class, and store-class common command; also occurs in *BSN*, *BFZ*, *SEL*, *SDR* instructions and all miscellaneous-class instructions except *CSW* and *SLR*.	
132	*Set A-B flip-flop to B*	0.3.1 (B3)	4FMB4	Store-class common command.	

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
134	*Set pause flip-flop*	0.2.2 (A8)	4FPG3	Multiply-class, shift-class, and IO-class common command; also occurs in *SLR* and *HLT* instructions.	Conditional: During execution of multiply-class, shift-class, or *SLR* instructions, set pause flip-flop if step counter content is greater than 6. During IO class or *HLT* instructions, set pause flip-flop if IO interlock is set.
138	*Set 2-mc set-sync flip-flop*	0.5.3 (D7)	4FNF6	Multiply-class and shift-class common command; also occurs in *SLR* instruction.	The 2-mc operate flip-flop is set 0.5 μsec later by a 2-mc control pulse.
140	*Sense for branch condition met*	0.7.4 (D3)	4FDH3	Occurs in *BSN* instruction.	If branch condition is met, set branch flip-flop.
144	*Clear left IO register*	0.7.1 (C8)	4EWE7	Occurs in *RDS* and *WRT* instructions.	If the IO register is not selected as an input device.
146	*Clear drum control register*	0.7.2 (D12)	4EWF7	Occurs in *SEL* and *SDR* instructions.	
147	*Clear right IO register*	0.7.1 (C8)	4EWE7	Occurs in *RDS* and *WRT* instructions.	If the IO register is not selected as an input device.
148	*Clear IO address counter*	0.4.1 (E1)	4EWA3	Occurs in *LDC* instruction.	
149	*Clear IO word counter*	0.7.3 (B5)	4EWA7	Occurs in *RDS* and *WRT* instructions.	Clear IO word counter, and sense single-pulse flip-flop; if set, set continue set-sync flip-flop and instruction step flip-flop.
151	*Address register to drum control register*	0.4.1 (D13)	4EHH7	Occurs in *SEL* and *SDR* instructions.	
152	*Address register complement to IO word counter*	0.4.1 (C13)	4EHG3	Occurs in *RDS* and *WRT* instructions.	Transfer address register bits S-15 from 0 side to set IO word counter bits S-15 to 1 side.
153	*IO word counter to right accumulator register*	0.7.3 (B5)	4EWH1	Occurs in *CSW* instruction.	
154	*Address register to program counter*	0.4.1 (D13)	4EHA6	Branch-class common command.	Conditional: If branch flip-flop is set, transfer address register to program counter.
155	*Deselect pulse*	0.7.5 (A6)	4FEA7	Occurs in *SEL* and *SDR* instructions.	Reset all computer IO unit selection and transfer control flip-flops and field selection registers in main and auxiliary drum units.

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
156	*PT5 select pulse*	0.7.5 (A7)	4FDB4	Occurs in *SEL* instruction.	Set computer IO unit selection and transfer control circuits as specified by *PERSELBSN* matrix.
157	*Reset CSW control circuits*	0.7.3 (A8)	4FTJ3	Occurs in *CSW* instruction.	
161	*Clear PT-OT flip-flop to PT*	0.3.1 (B4)	4FPA3	Add-class, multiply-class, and store-class common command; also occurs in *SDR, SEL, BFZ, BSN* instructions and all miscellaneous-class instructions except *CSW* and *SLR.*	
162	*Test bit S of left and right accumulator register for 1*	0.5.2–2 (D19)	4GYF6	Occurs in *BFZ* and *BFM* instructions.	Sense right accumulator register bit S for 1; if set, sense left accumulator register bit S for 1; if set, set branch flip-flop.
163	*Clear branch flip-flop*	0.3.1 (B2)	4FME7	Branch-class common command.	
164	*Test index register No. 1 bit S for 0*	0.4.2 (E5)	4EGA6	Occurs in *BPX* instruction.	Sense bit S of index register 1; if cleared, set branch flip-flop.
165	*Test left accumulator register bit S for 1*	0.5.1–2 (D21)	4FYJ3	Occurs in *BLM* instruction.	Sense left accumulator register bit S for 1; if set, set branch flip-flop.
166	*Test right accumulator register bit S for 1*	0.5.2–2 (D22)	4FYG3	Occurs in *BRM* instruction.	Sense right accumulator register bit S for 1; if set, set branch flip-flop.
167	*Clear A-B flip-flop to A*	0.3.1 (B4)	4FPC7	Store-class common command.	
170	*Set branch flip-flop*	0.3.1 (B4)	4FPH3	Occurs in *BPX* instruction.	Conditional: If IX_0 or IX_3 is selected.
174	*Test index register No. 2 bit S for 0*	0.4.2 (D5)	4EGA3	Occurs in *BPX* instruction.	Sense bit S of index register 2; if cleared, set branch flip-flop.
180	*PT6 on read*	0.7.3 (C8)	4FDH7	Occurs in *RDS* instruction.	Sense bits 13, 14, and 15 of index interval register; if set, set specified interleave flip-flop. Also sense all IO unit selec-

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
					tion circuits to set required IO transfer control circuits, and send a start signal to selected IO unit.
182	*PT6 on write*	0.7.3 (C8)	4FDJ3	Occurs in *WRT* instruction.	Sense bits 13, 14, and 15 of index interval register; if set, set specified interleave flip-flop. Also sense all IO unit selection circuits to set required IO transfer control circuits, and send a start signal to selected IO unit.
201	*Right accumulator register to right B register*	0.5.1–2 (E19)	4FYA6	Occurs in *DIM*, *MUL*, and *TMU* instructions.	
202	*Right accumulator register bits (2-15) to bits (1-14)*	0.5.2–2 (D21)	4GXC6	Add-class common command; also occurs in *AOR*, *ECH*, *DCL*, *FCL*, *DSL*, *ASL*, *BFZ*, and *SLR* instructions.	
204	*Right accumulator register bit 1 to bit S*	0.5.2–2 (D21)	4GXH6	Add-class common command; also occurs in *AOR*, *ECH*, *DCL*, *FCL*, and *BFZ* instructions.	
205	*Right accumulator register bit S to right B register bit 15*	0.5.1–2 (D19)	4FYB4	Occurs in *DCL*, *DSL*, and *SLR* instructions.	
206	*Right accumulator register bit S to bit 15*	0.5.1–2 (D19)	4GWH7	Occurs in *ASL* instruction.	
207	*Right accumulator register bit 15 to right B register bit S*	0.5.2–2 (D1)	4FYH7	Occurs in *RSR* and *DSR* instructions.	
209	*Right correct sign*	0.5.2–2 (C22)	4GXA7	Multiply-class common command.	Conditional: Sense sign control flip-flop; if set, clear sign control flip-flop, complement right B register, and complement right accumulator register.

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
210	*Clear right accumulator register*	0.5.2–2 (B21)	4FVH6	Occurs in *CAD*, *CSU*, *CAM*, *CSW*, *MUL*, *TMU*, *AOR*, and *ECH* instructions.	
211	*Correct right remainder*	0.5.1 (C5)	4EPB4	Occurs in *DVD* and *TDV* instructions.	Conditional: Sense bit S of right A register; if cleared, pulse carry 0 line of right 15 adder, and transfer bit S to the right B register S storage.
212	*Right accumulator register to address register*	0.5.2–2 (E21)	4FVJ3	Add-class, multiply-class, store-class, IO-class, and miscellaneous-class common command; also occurs in *BPX* instruction.	Add right accumulator register to address register through index adders.
213	*Make right accumulator register positive*	0.5.2–2 (A22)	4FUH6	Occurs in *DIM*, *MUL*, *TMU*, and *BFZ* instructions.	Conditional: Sense bit S of accumulator register; if set, complement sign control flip-flop, and complement right accumulator register.
214	*Left round off sign*	0.5.1–2 (A22)	4FTB4	Occurs in *BFZ* and *SLR* instructions.	Clear carry storage flip-flop, and sense sign control flip-flop; if set, clear sign control flip-flop, and complement accumulator register.
216	*Right round off sign*	0.5.1–2 (A22) 0.5.2–2 (A22)	4FTB4 4FUG2	Occurs in *BFZ* and *SLR* instructions.	Clear carry storage flip-flop, and sense sign control flip-flop; if set, clear sign control flip-flop, and complement accumulator register.
217	*Right accumulator register bit S to left accumulator register bit 15*	0.5.1–2 (D19)	4FYA3	Occurs in *FCL* instruction.	
219	*Complement right accumulator register*	0.5.2–2 (A22)	4FNC6	Occurs in *DEP* and *BFZ* instructions.	
225	*Right logical multiply*	0.5.2 (D7)	4EMF6	Occurs in *DEP* and *ETR* instructions.	Transfer 0 side of right A register to right accumulator register.

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
226	*Complement right A register*	0.5.1 (B5)	4ELH3	Occurs in *CSU*, *SUB*, *TSU*, and *DIM* instructions.	
228	*Right A register to right memory buffer register*	0.5.2 (D7)	4ELE7	Occurs in *LST* and *STA* instructions.	
229	*Make right A register positive*	0.5.1 (D5)	4ELC7	Multiply-class, common command; also occurs in *CAM* and *DIM* instructions.	Conditional: Sense bit S of right A register; if set, complement sign control flip-flop, and complement right A register.
230	*Clear right A register*	0.5.2 (B6)	4EMC6	Add-class, multiply-class, and branch-class common command; also occurs in *DEP*, *ADX*, *SLR*, *ADB*, *ECH*, *LST*, *AOR*, *RST*, and *ETR* instructions.	
260	*Right accumulator register conditional shift left*	0.5.1–2 (A18)	4ERE7	Add-class common command; also occurs in *DVD*, *TDV*, and *SLR* instructions.	Conditional: Sense carry storage flip-flop; if set, clear carry storage flip-flop, and shift right accumulator register bits 1-15 to bits S-14, shift right B register bit S to right accumulator register bit 15, and transfer right B register S storage flip-flop to right B register bit S.
261	*Complement right A register*	0.5.1–2 (A18)	4ERB4	Occurs in *DVD* and *TDV* instructions.	Conditional: Sense right sign adder; if signs are alike, complement right A register.
263	*Right end-carry*	0.5.1–2 (A18)	4ERA6	Add-class common command.	Conditional: Sense carry storage flip-flop; if set, pulse carry 1 line of right 15 adder, and transfer left B register bit S to left B register S storage.
264	*Right carry 0*	0.5.2–2 (A1)	4ETA7	Add-class common command; also occurs in *ECH* instruction.	Pulse right 15 adder, and transfer right B register bit S to right B register S storage flip-flop.

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
266	*Record right overflow*	0.5.1–2 (B22)	4ERA3	Common command.	Clear sign control flip-flop, and sense and clear auxiliary overflow flip-flop. If auxiliary overflow flip-flop is set, sense index interval register bit 13. If bit 13 contains a 0, set right overflow alarm and right overflow alarm indicator flip-flops, and sense index interval register bit 15. If bit 15 contains a 1, sense OVERFLOW switch; if it is set and auto-branch control flip-flop is set, sense STOP-BRANCH switch. If this switch is set, execute an alarm branch; that is, clear right A register, execute an alarm branch delayed control clear, transfer program counter to right A register, and set program counter to $3.77770_{(8)}$. Computer will resume operation from test memory address.
267	*Complement right divide connect flip-flop*	0.5.1–2 (A17)	4ERC7	Occurs in *DVD* and *TDV* instructions.	Clear carry storage flip-flop and auxiliary overflow flip-flop, and complement divide connect flip-flop.
270	*Clear continue flip-flop*	0.2.2 (B4)	4FPJ3	Occurs in *HLT* instruction.	Conditional: Sense load flip-flop. If cleared, sense CPC-IO flip-flop; if cleared, clear continue flip-flop.
281	*Right B register bit S to right accumulator register bit 15*	0.5.2–3 (E6)	4FJC6	Add-class common command; also occurs in *ECH*, *DCL*, *DSL*, *BFZ*, and *SLR* instructions.	
282	*Right B register bits (1-15) to bits (S-14)*	0.5.2–3 (D7)	4FJG2	Occurs in *DCL*, *DSL*, and *SLR* instructions.	

TABLE 2–1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
283	*Right partial product*	0.5.2–3 (C7)	4FKA2	Occurs in *MUL* and *TMU* instructions.	Shift right B register bits S-14 to bits 1-15, and sense right B register bit 15 flip-flop. If this flip-flop contains a 1, pulse right carry line of bit 15 adder; if this flip-flop contains a 0, shift right accumulator register bits S-14 to 1-15 and shift bit 15 to right B register sign.
284	*Clear right B register*	0.5.2–3 (A7)	4FKF6	Occurs in *DIM*, *MUL*, *TMU*, *SLR*, and *LDB* instructions.	
285	*Right B register bits (S-14) to bits (1-15)*	0.5.2–3 (D7)	4FJJ3	Occurs in *RSR* and *DSR* instructions.	
287	*Right round off*	0.5.2–3 (E6)	4FJA7	Occurs in *SLR* instruction.	Sense bit S of right B register; if set, set carry storage flip-flop and pulse carry 1 line to right 15 adder, and transfer left B register bit S to left B register S storage.
288	*Right B register to right A register*	0.5.2–3 (C7)	4FKJ3	Occurs in *ADB* and *DEP* instructions.	
289	*Right B register S storage flip-flop to right B register bit S*	0.5.2–3 (B7)	4FKH6	Add-class common command; also occurs in *ECH*, *AOR*, and *BFZ* instructions.	
290	*Sense IO word counter equal to positive 0*	0.7.3 (D5)	4EWC7	Occurs in *RDS* and *WRT* instructions.	Set sense word counter, disconnect drum control flip-flops, and step IO word counter. If an IO word counter end-carry pulse is generated, clear write drums flip-flop and sense read-write zero tapes and card machines flip-flop; if set, clear IO interlock and unlock address counter.
294	*Set IO interlock on*	0.3.1 (B2)	4FDG3	Occurs in *RDS* and *WRT* instructions.	Set both IO interlock flip-flops and word counter status flip-flop, and clear

TABLE 2-1. COMMAND PULSE ANALYSIS (cont'd)

COMMAND NUMBER	COMMAND NAME	ORIGIN		COMMAND INCIDENCE	REMARKS
		LOGIC	PU		
					interleave control flip-flop, the interleave by 8, 16, and 64 flip-flops, the IO register status flip-flop, the accept flip-flop, the read flip-flop, and the write flip-flop, and clear the left and right IO buffer registers. This pulse is also used to sense if both the tape operate flip-flop and the drum operate flip-flop are cleared. If this condition exists, the parity check control flip-flop is cleared.
321	*PT11 on sense*	0.7.8 (C4)	3FDA6	Occurs in *BSN* instruction.	Sense the tape operate flip-flop; if set, sense TAPE TEST-OPERATE switch. If the tape element is under computer control, sense for tape-unit-prepared and tape-unit-ready status. If these conditions exist, clear tapes not prepared and tapes not ready alarm flip-flops.
325	*Select pulse from drums*	0.7.7 (B9)	4FFA3	Occurs in *SDR* instruction.	Set drum operation flip-flop, and sense address register R1 flip-flop. If this flip-flop is set, transfer index interval register to auxiliary drum field selection register. If this flip-flop is cleared, transfer index interval register to main drum field selection register and sense *PERSELBSN* matrix outputs to set address mode flip-flop or one of the five identity mode flip-flops if one of these modes of control is selected.

1.2 BLOCK DIAGRAM ANALYSIS

The functional arrangement of the instruction control element circuits is shown in figure 2–1. The instruction decoder receives coded instruction pulses from the memory element by way of the memory buffer register. As a result of decoding by the various control and instruction matrices, a +10V level is applied to the suppressor grids of the command generators required for the execution of the instruction, and a −30V level to the suppressor grids of the remaining command generators. A command generator to which the +10V level has been applied is said to be conditioned; one to which the −30V level has been applied is called deconditioned. Simultaneously with the generation and routing of these levels, the pulse generation and control section produces three types of repetitive timing pulses of 0.1-μsec duration. Specific timing pulses are applied to the control grids of individual command generators. If a command generator is conditioned when it is examined by a timing pulse, a command pulse is generated.

Figure 2–1 further shows that the pulse generation and control section, in addition to supplying timing pulses to the command generators, supplies timing pulses to the IO element also.

The primary function of the instruction control element is to decode coded instructions which it receives from core memory by way of the memory buffer register and to issue appropriate commands in response. To accomplish this, the instruction control element generates standard levels and pulses. The levels are applied to command generators (gate tubes) to condition them.

The pulses generated by the instruction control element originate in a 2-mc crystal-controlled oscillator. The sine-wave output of the oscillator is clipped and shaped into positive pulses ranging from 20V to 40V above ground level in amplitude, 0.1 μsec wide at the base, and having a 0.5-μsec repetition time. The pulses are distributed among a number of circuits, in which they perform four functions (table 2–2).

TABLE 2–2. PULSES GENERATED BY INSTRUCTION CONTROL ELEMENT

NAME	ABBREVIATION	REPETITION RATE
2-megacycle	2 mc	0.5 μsec
Instruction pulse	IP	6.0 μsec
Time pulse	TP	6.0 μsec
Divide time pulse	—	2.5 μsec
Command pulse	cp	—

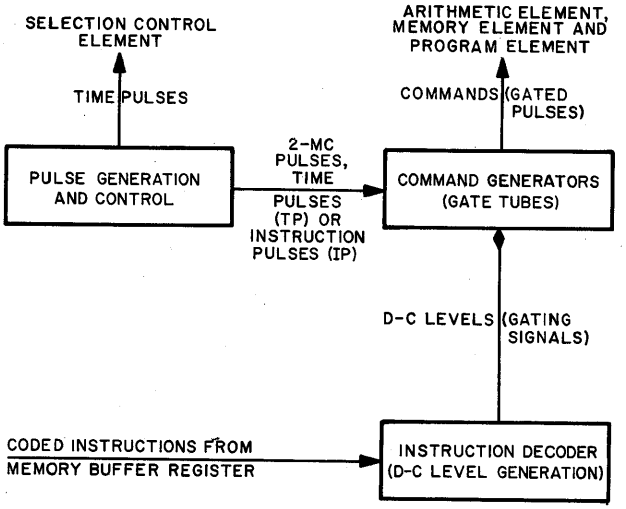


Figure 2–1. Instruction Control Element, Simplified Block Diagram

Instruction pulses, time pulses, and divide time pulses are identical in origin, shape, and amplitude, but differ in timing (as shown in table 2–2). All these pulses are used for the internal operation of the Central Computer System.

A command pulse (or command) is formed when a conditioned command generator tube is examined by any of these pulses. Command pulses are identical with other pulses in amplitude and shape; however, they have no repetition rate. During computer operation, the binary-coded instructions stored in core memory are sequentially transferred to a control register. The contents of this register are decoded by control matrices. The product of the decoding process is a pattern of d-c levels, the configuration of the pattern depending on the nature of the instruction. Finally, the d-c levels condition command generators which issue command pulses when examined as described above.

The commands issued by the command generator tubes either activate other parts of the instruction control element or are distributed among the other elements of the Central Computer System. The majority of the commands are transmitted to the arithmetic element, where they initiate and control arithmetic operations. Other commands are:

a. Applied to the program element to co-ordinate the overall operation of the Central Computer System.
b. Routed to the IO element to set up controls which govern the transfer of information into or out of the IO devices.
c. Applied to memory element to initiate the transfer of information between memory and the Central Computer.

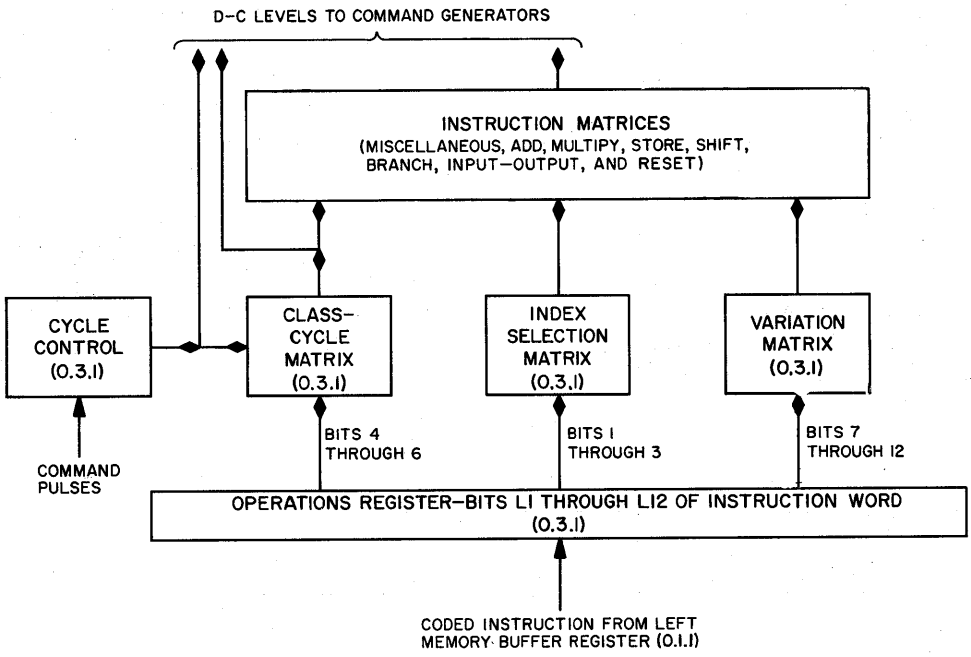


Figure 2–2. Instruction Decoder, Simplified Block Diagram

1.2.1 Instruction Decoding

Figure 2–2 presents the instruction decoder of figure 2–1 in somewhat greater detail. The decoder consists of an operations register, a cycle control, three control matrices, and eight instruction matrices.

Assume that a coded instruction has been placed in the operations register, which is composed of 12 flip-flops. This coded instruction corresponds to bits L1 through L12 of the instruction word. If the instruction is indexable, bits L1 through L3, which are decoded by the index selector matrix, specify which of five available index registers will be used. If the instruction belongs to the reset class, or if it is *Branch and Index* (branch class), the index selector matrix determines the index register whose contents are to be reset or modified.

Bits L4 through L6 are routed to the class-cycle matrix. There they are combined with information from two of the eight flip-flops of the cycle control, the PT-OT flip-flop and the A-B flip-flop. These two flip-flops specify whether a PT cycle or an OT cycle is in process. If the PT-OT flip-flop specifies that an OT cycle is in process, the A-B flip-flop will further specify whether the cycle is an OTA cycle or an OTB cycle (fig. 2–3). Each output line of this matrix conditions different sets of command generators, which constitute the common command generators used at different times during a computer cycle. This matrix also supplies d-c levels to the eight instruction matrices.

The outputs of bits L7 through L12 of the operations register are applied to the variation matrix to specify which variation within the selected class has been selected.

The outputs of the three control matrices go to the instruction matrices, as shown in figure 2–2. There are eight of these, one for each class of instruction. Each instruction matrix responds to levels received from the control matrices and from the other instruction matrices, to condition the unique command generators required for the execution of the indicated instruction.

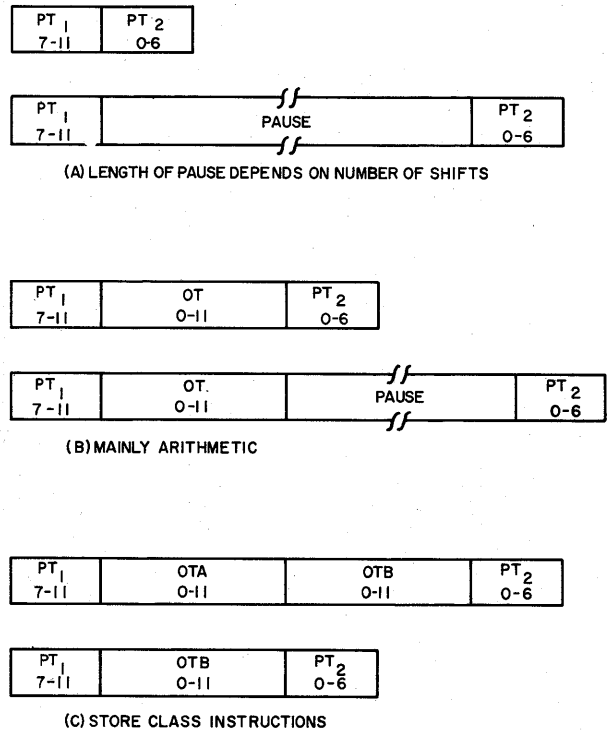


Figure 2–3. Machine Cycles

As a result, three groups of command generators are conditioned during the execution of any instruction. The total output indicated in figure 2–2 includes not only individual d-c levels corresponding to individual instructions, but also d-c levels derived from the class-cycle matrix which apply to all instructions of a given class and cycle, and those derived from the cycle control which apply to the type of operating cycle in process.

1.2.2 Pulse Generation and Control

The pulse generation and control section shown in figure 2–1 is represented in greater detail in figure 2–4. The oscillator section shown in figure 2–4 is the source of all pulses used by the instruction control element. Its 2-mc sine-wave output is rectified, clipped, and shaped into positive pulses of 0.1-μsec duration, occurring at 0.5-μsec intervals. The pulses are then passed on to the time pulse distributor control.

The output of the time pulse distributor control (fig. 2–5) is divided into two channels, designated instruction pulse driver and time pulse driver. These two output lines are routed to the time pulse distributor, which distributes the time-pulse-driver pulses to 12 output lines (labeled TP 0 through TP 11).

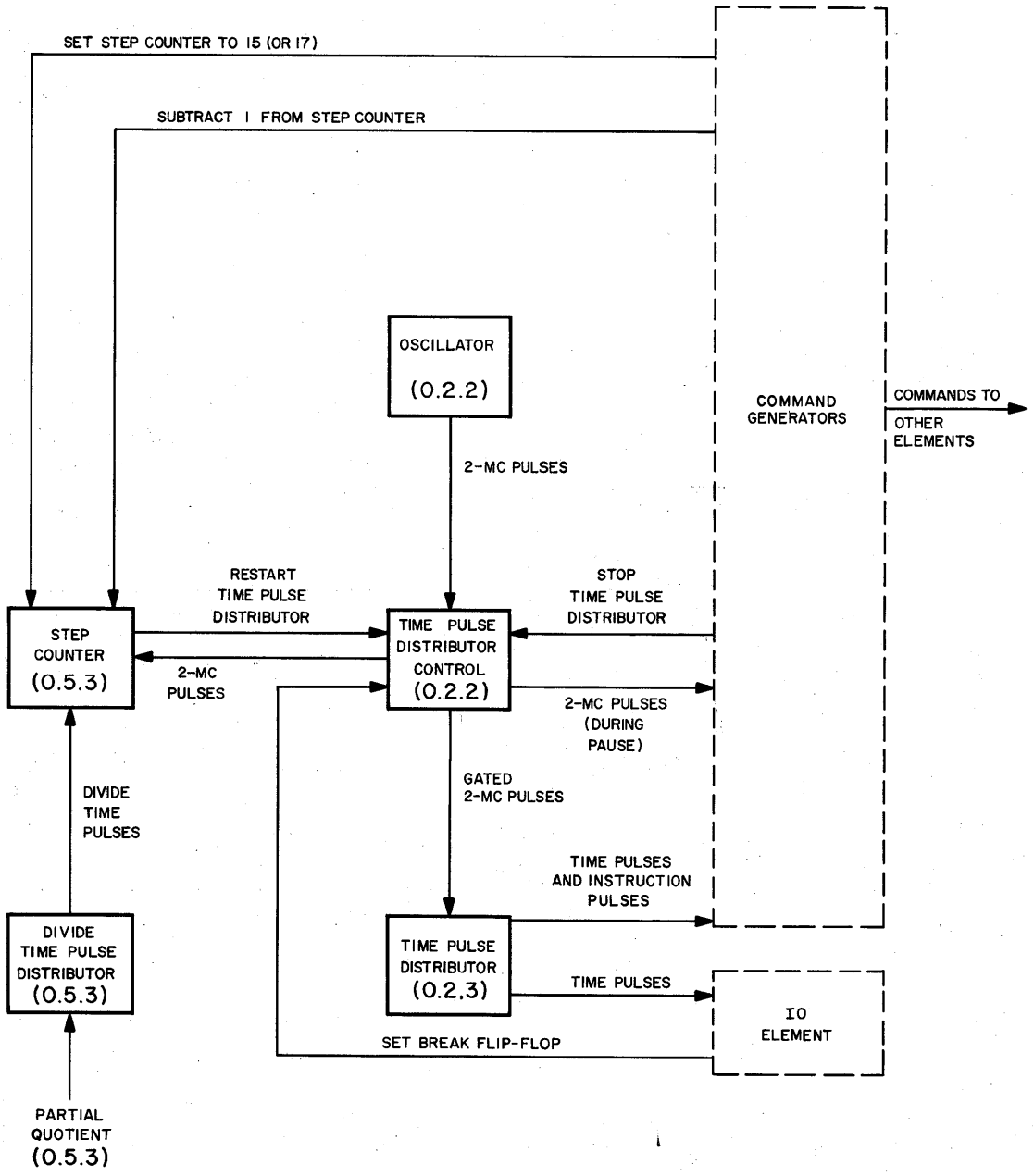


Figure 2–4. Pulse Generation and Control, Simplified Block Diagram

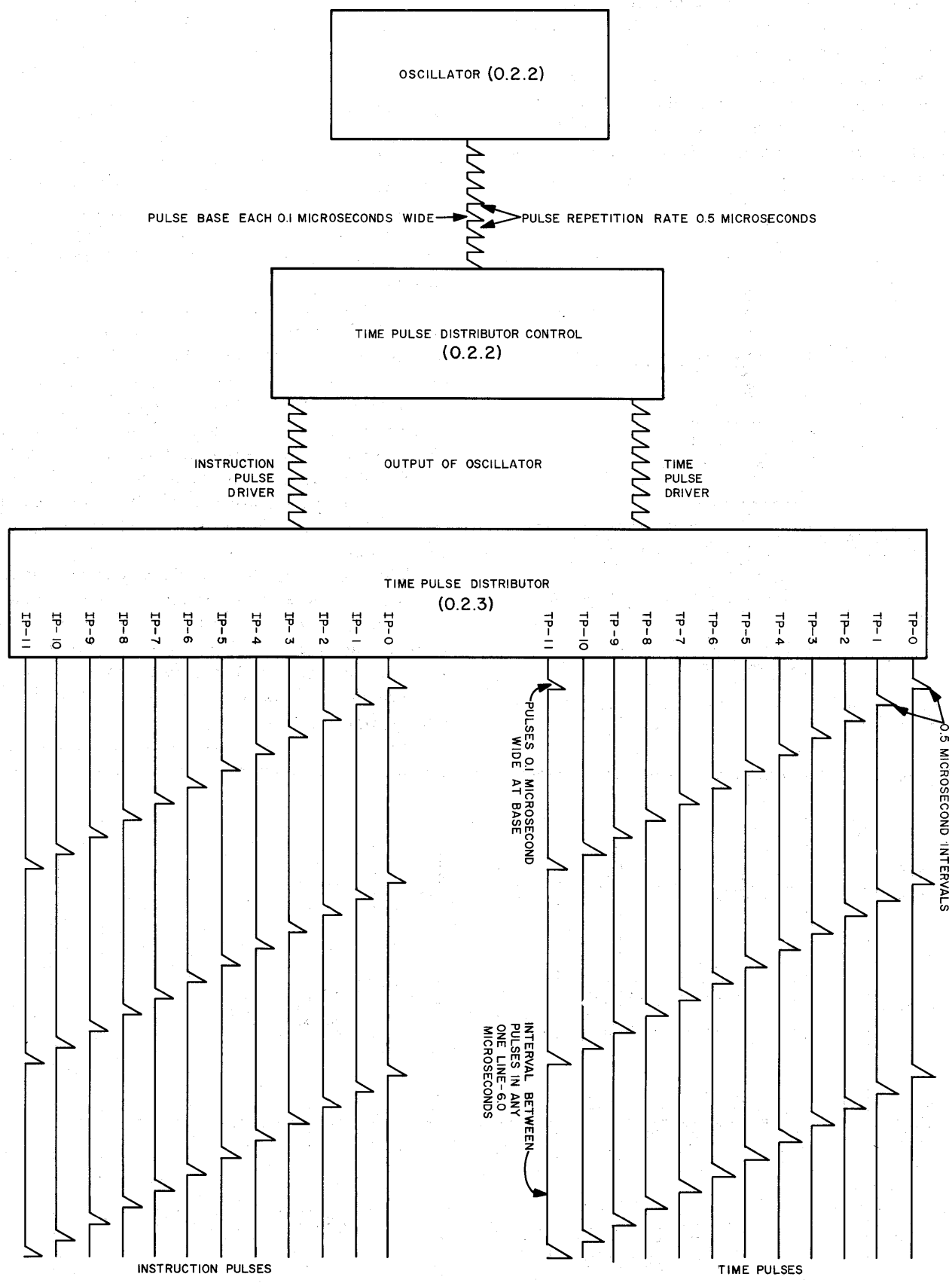


Figure 2-5. Generation of Pulses

The conversion of the single-line output of the instruction pulse driver into 12 outputs (IP 0 through IP 11) is similarly accomplished.

The pulse output of each line occurs 0.5 μsec before or after the pulse output of the adjacent line. For example, TP 5 occurs 0.5 μsec before TP 6 and 0.5 μsec after TP 4. The output pulses on each line are 0.1 μsec wide at the base and occur at 6.0-μsec intervals.

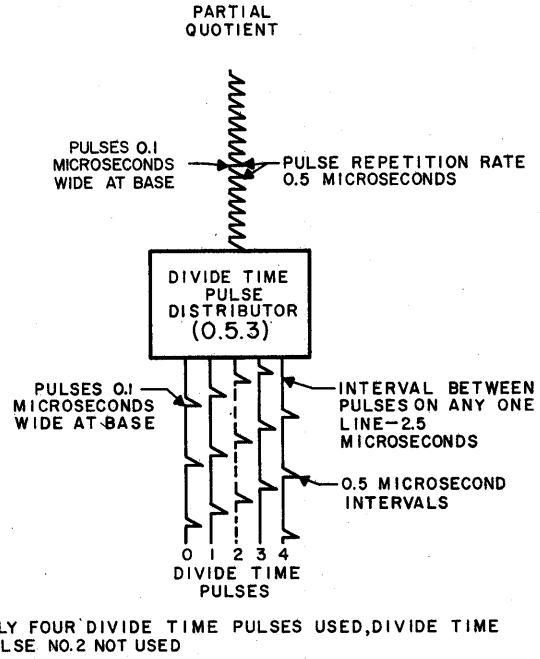


Figure 2–6. Generation of Divide Time Pulses

Figure 2–6 shows the action of the divide time pulse distributor. The pulses shown are used in the divide operation. They occur at 2.5-μsec intervals in each line.

1.3 TIMING

Timing of the Central Computer System is accomplished by means of time pulses (TP's) and instruction pulses (IP's), shown in figure 2–5. There are two general types of computer time cycles: one, called a memory cycle, is used to time memory operations; the other, called a machine cycle, is used to time all other computer operations.

Every machine cycle has a memory cycle associated with it and synchronized with it by the action of the time pulses. The operations that can be performed in the memory element during each specific portion of the memory cycle are defined and explained in figure 2–7.

The machine cycle has five variations: program time, operate time A, operate time B, break-in, and breakout. The first three are used for the internal operation of the computer and are discussed below. Break-in and breakout cycles refer to IO operations and are discussed in detail in Part 6.

The use of the three machine-cycle variations associated with internal operation of the computer is shown in figure 2–3. Central Computer System instructions are divided into eight classes according to their similarity. In figure 2–3, the instructions are classified into groups A, B, and C, according to the machine cycles required for their execution.

In analyzing the operations performed during the execution of any instruction, the instruction cycle is con-

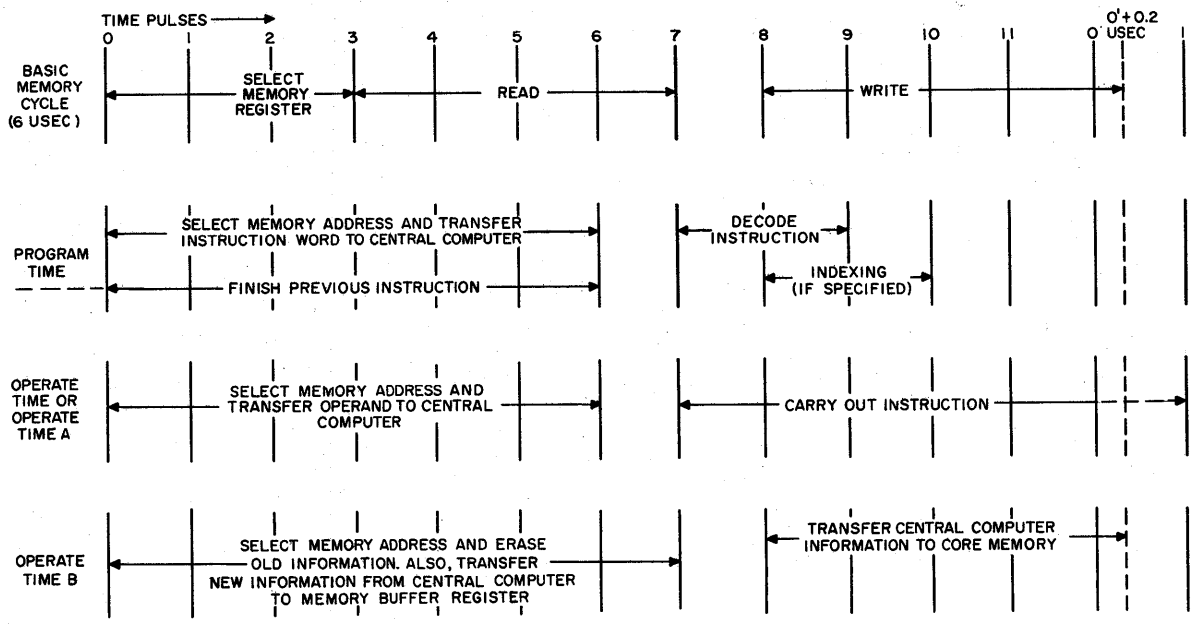


Figure 2–7. Memory and Machine Cycles

sidered as beginning at PT 7 with the decoding of the instruction and ending at PT 6 of the succeeding repetitive series. The interval PT 0 to PT 6, which completes the instruction in process, is also used to bring the next instruction out of core memory. This performance of two different operations during the same time interval is possible because a transfer of instructions out of memory never involves circuits used to complete previous operations.

Part A of figure 2–3 applies to those instructions which do not require an operand. The upper configuration of part A applies to instructions (such as those of the reset class) that do not need more than 6 μsec for their completion. The lower configuration (PT-Pause-PT) applies to instructions (such as those of the shift class) that may require more than 6 μsec for their completion. The additional time during which repetitive operations are performed is provided by the arithmetic pause. (The pause is not necessary and does not occur if fewer than six shifts are involved.)

Part B of figure 2–3 applies to instructions which involve an operand. The upper configuration (PT-OTA-PT) is associated with instructions (such as those of the add class) which can be completed in a 6-μsec operate-time cycle. The lower configuration (PT-OTA-Pause-PT) corresponds to instructions (such as those of the multiply class) which need both an operand and more than 6 μsec of OT time in which to operate upon it. The length of the pause is governed by the step counter and depends on the number of iterative operations to be performed. (Refer to 1.3.1.)

Part C of figure 2–3 applies to store-class instructions. The upper configuration (PT-OTA-OTB) applies to those store processes in which one half-word in the memory register is to be replaced while the other half-word must be retained. The full word is read out of memory into the Central Computer System during the OTA portion of the total cycle. During this same interval, a new half-word is prepared by the computer. The OTB interval is used to store in core memory the new composite word consisting of the new half-word and the unaltered half of the old word.

The lower configuration of figure 2–3, part C, corresponds to the operation of storing a full word in core memory. Since the problem of saving an existing half-word does not arise, the OTA interval is not needed, and the configuration is PT-OTB-PT.

Whether the computer uses machine cycle A, B, or C to execute a particular instruction is determined by the cycle control in conjunction with the time pulse distributor control (figs. 2–2 and 2–4). Different circuits are selected for different cycles or combinations of cycles. Separate sets of command generators are conditioned for PT, for OTA, and for OTB cycles. When, for example, IP 1 occurs during PT time, a certain set of command generators is conditioned and a certain set of commands issued; however, when IP 1 occurs during OTA time, a different set of command generators is conditioned and a different group of commands issued; when IP 1 occurs during OTB time, a third set of command generators acts and a third combination of commands results.

1.3.1 Pause Conditions

During normal computer operation (execution of stored program), a temporary pause in machine cycle timing will occur as a result of the execution of certain instructions. These instructions, namely, the multiply and shift class instructions and the *SLR* instruction, each require a number of repetitive operations, not all of which can be performed within the normal machine cycle timing. As a result, during the execution of these instructions, a temporary arithmetic pause condition is provided (between machine cycles) during which the excess number of repetitive operations is performed.

During the execution of any of these instructions, the step counter, which is used to control the execution of the repetitive operations and the duration of the arithmetic pause (if required), is set to indicate how many repetitive steps are to be performed. In the case of a shift instruction, the number of shifts to be performed is set into the step counter when the instruction word is transferred from the memory element. For the multiply class instructions, the step counter is set by the generation of an appropriate command.

For each of these instructions, the required repetitive operations are performed by generating and gating 2-mc operate pulses to produce the required command pulses. Generation of the 2-mc operate pulses is initiated by a command if the step counter contents are greater than zero. The step counter contents are also sensed to determine whether an arithmetic pause is required. The need for an arithmetic pause is actually a function of both, the instruction being executed and the number of repetitive operations required. If an arithmetic pause is required, the time pulse distributor is stopped at the beginning of a machine cycle. During the execution of the repetitive steps, the step counter contents are reduced by 1 for each required operation. The step counter contents are continually sensed (step counter status levels) by 2-mc control pulses to determine when the arithmetic pause is to be terminated and when the generation of the 2-mc operate pulses is to be halted.

During normal computer operation, a second type of pause in machine cycle timing can be produced. This condition, called an IO pause, is of indefinite duration and is produced if the IO interlock is set and either an *HLT* instruction or an IO class instruction is being executed. The IO pause condition indicates that the instruc-

tion being executed would disturb the IO operation in process. The IO pause condition is terminated when the previously programmed IO operation is terminated.

1.3.2 Break Cycle Generation

During the execution of the stored program, any one of the IO devices associated with the computer can be selected to transfer information between itself and the memory element. Since the IO devices are asynchronous, they inform the computer (by means of a break request) whenever they are ready to transfer a data word into or out of memory. This break request condition is sensed for at the end of every machine cycle, and, if it exists, the next memory cycle is assigned as a break cycle. During the break cycle, one memory word is transferred into or out of the memory element. During the break cycle, the time pulse distributor control inhibits the generation of instruction pulses to temporarily stop (for one memory cycle) the internal operation of the computer.

If a break request is received from an IO device while an arithmetic or IO pause is in process, the time pulses, which were interrupted by the pause condition, are restored temporarily so that the break cycle can be executed. If the break cycle was executed during the arithmetic pause, the internal repetitive operations continue uninterrupted; when the break cycle is completed, the time pulses are again suppressed until the end of the arithmetic pause.

1.4 MANUAL CONTROLS

The manual control circuits of the Central Computer System, part of which are physically located in the instruction control unit, are discussed in Part 8.

CHAPTER 2

INSTRUCTION DECODING AND COMMAND PULSE GENERATION

Instruction words supplied to the control and instruction matrices of the instruction control element are converted into appropriate d-c levels by the matrices and applied to the suppressor grids of command generators. Whenever the d-c level for a specific command generator (specified by an instruction) is made positive, that gate tube is said to be conditioned, and when an instruction pulse is applied to its control grid, an output pulse is generated. Such output pulses are called command pulses and are the means by which instructions are executed.

2.1 INSTRUCTION WORDS

As shown in figure 2–8, an instruction word consists of two half-words that are obtained simultaneously from core memory. The left half-word contains the coded information which specifies a particular instruction; the right half-word, if used by the instruction, contains either the address of data called for by the instruction or data which completes the details of the instruction.

Specific portions of the instruction word are transferred to the operations register, the index interval register, and the address registers. Bits L1 through L12 are transferred to the operations register, bits L10 through L15 are transferred to the index interval register (part of the selection element), and bits LS and RS through R15 are transferred to the address register (part of the program element). The three registers (and the respective control elements of which they are a part) function together to completely decode the operating program instructions.

The operations register contains that part of the instruction represented by bits L1 through L12. Bits L1 through L3 are the index indicators. If used, they designate any one of the six indexing possibilities listed in table 2–3.

Bits L4 through L6 specify which of the eight instruction classes of instructions is selected. For most instructions, bits L7 through L10 specify one of the variations within the selected class; exceptions are listed below:

a. For instructions using the index interval, bits L7 through L9 completely identify the selected variation and bits L10 through L15 are used to specify which one of several devices is to be operated on.
b. For compare instructions, bits L7 through L9 specify that a logical comparison is to be made and bits L10 through L12 specify which specific compare instruction is to be executed.
c. For the *TOB* and *TTB* instructions, bits L7 through L9 specify that a test instruction is to be executed, bit L10 determines which of the two variations is selected, bit L11 specifies whether the left or right half-word is to be tested, and bits L12 through L15 specify which of 16 bits of the specified left or right half-word is (are) to be tested.

Bit L12 may be used (optionally) to provide for 17-bit operation during *XAC, AOR, RST, CSW, STA,* and all add class instructions.

Bits L13 through L15 are transferred directly to the index interval register and, in conjunction with bits L10 through L12, comprise the index interval bits used for control during the execution of the *BPX, BSN, PER, SEL, SDR, TOB,* or *TTB* instructions. If not employed as index interval bits, bits L13 through L15 may be used as follows:

a. L13 is used to suppress overflow alarm arising from arithmetic modification of addresses.
b. L14 (control of left overflow) and L15 (control of right overflow) are used to determine whether any action is to be taken as a result of overflow on instructions that can cause overflow.
c. L13, L14, and L15 may be used singly during the execution of the *RDS* or *WRT* instruction to specify an interleaving mode of 8, 16, or 64.

TABLE 2–3. INDEXING CODE

INDEX REGISTER	BINARY CODE
Index Register 1	001
Index Register 2	010
Index Register 4	100
Index Register 5	101
Index Register 3 (right accumulator)	011
None	000

Bits LS and RS through R15 select a specific memory device and designate the desired address within the selected device.

2.2 INSTRUCTION DECODING EQUIPMENT

The following units are used for instruction decoding:

a. Operations register (0.3.1) bits L1-L12
b. Cycle control (0.3.1)
c. Class-cycle matrix (0.3.1)
d. Variation matrix (0.3.1)
e. Index selector matrix (0.3.1)
f. Instruction matrices (0.3.2)

The six units in the above list receive their information from the left memory buffer register during the PT cycle of the instruction and select the class and specific variation.

2.2.1 Operations Register

Bits L1 through L12 of an instruction word are transferred from the left memory buffer register to the operations register during the PT cycle. Ten of the 12 flip-flops of the operations register control the AND circuits in the class-cycle matrix, the variation matrix, and the index selector matrix. The AND circuits, in the class-cycle matrix, are further controlled by the cycle control.

2.2.2 Cycle Control

The cycle control employs six flip-flops to carry out four functions. Two of these flip-flops, the PT-OT flip-

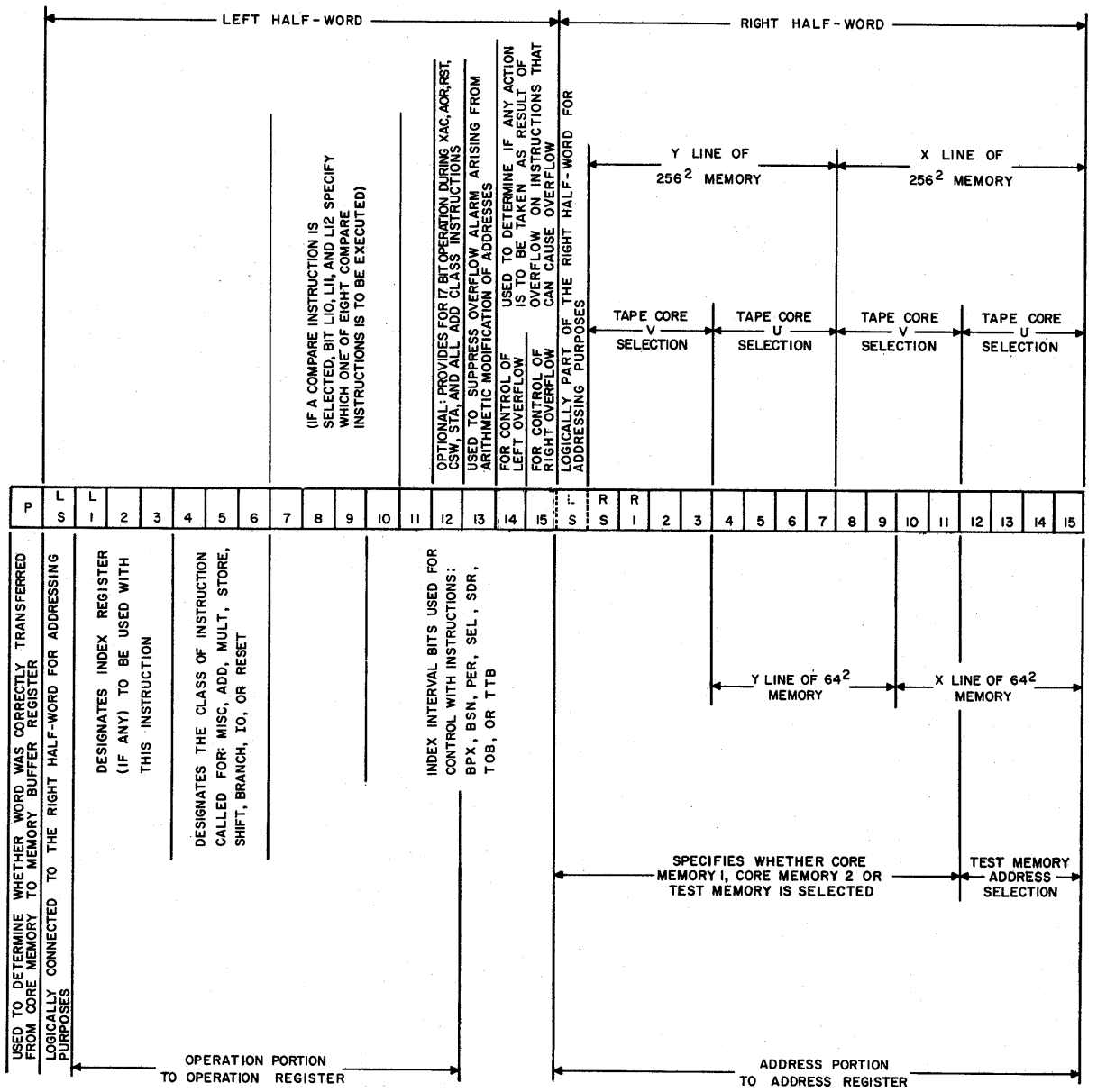


Figure 2–8. Instruction Word Analysis

flop and the A-B flip-flop (0.3.1), select the type of machine cycle to be performed, as directed by command generators. The PT-OT flip-flop determines whether the machine cycle will be a program time cycle or an operate time cycle. If it is to be an operate time cycle, the A-B flip-flop determines whether it shall be OTA or OTB.

The outputs of the PT-OT flip-flop and the A-B flip-flop are also used to condition command generators directly. Command generators conditioned directly are known as common-command generators. (Common commands are discussed in 2.3.)

The remaining four flip-flops of the cycle control are not used for instruction decoding, but to effect timing control. The branch flip-flop (0.3.1) is set to the 1 state by command 170 if the condition for a branch of program control, as specified by a branch instruction, has been met. The IO interlock flip-flop (logic 0.3.1), which is set at the initiation of an IO transfer operation, is used to indicate that an IO operation is in process. Whenever an IO instruction is being executed, the IO interlock flip-flop is sensed to determine whether the program is to continue, or whether an IO pause will result. This control prevents the Central Computer System from executing an IO instruction while a previously programmed IO operation is still in process.

The CSW control flip-flop and the CSW gate flip-flop (0.7.3) determine the time at which the contents of the IO word counter are transferred to the right accumulator register. Normally, the contents of the counter are transferred at OT 1 of the *CSW (Clear and Subtract Word Counter)* instruction; however, if the IO word counter is being stepped at OT 1, the CSW control and gate flip-flop delays the transfer so that it occurs at OT 5 instead. The operation of the two flip-flops is discussed in detail in Part 3.

2.2.3 Class-Cycle Matrix

The class-cycle matrix (0.3.1) combines the information contained in the class-selecting portion (bits L4, L5, and L6) of the operations register with the outputs of the PT-OT flip-flop and the A-B flip-flop to provide 17 combinations of class and cycle output levels. This combining of information is accomplished by using standard AND circuits.

Reference to logic 0.3.1 shows that the 17 outputs of the class-cycle matrix are numbered according to the class and cycle they represent. The 3-digit number associated with each line corresponds to bits L4, L5, and L6 of the instruction word. If, for example, bit L4 of the instruction word is 0, bit L5 is 1, and bit L6 is 1, the three together, 011, designate the store class of instruction. Thus, if a store-class instruction is selected, the 0 side output of the bit 4 flip-flop is at +10V, and the 1 side outputs of the bit 5 and bit 6 flip-flops are at +10V.

The combination 011 then appears at the inputs to three AND circuits: store OTA, store OTB, and store PT. If, at the same time, the PT-OT flip-flop is in the PT state (0 side at +10V), all four inputs to the AND circuit associated with the store PT output are at +10V; therefore, its output line is at +10V. However, if the PT-OT flip-flop is in the OT state (1 side output at +10V), the store OTA and store OTB AND circuits have four of their five inputs at +10V. The condition of the A-B flip-flop then determines which of the output lines (OTA or OTB) will be energized. Each of the other output lines will be similarly conditioned when all the inputs to the associated AND circuits are at +10V.

It is noted that two classes of instructions, multiply and miscellaneous, each have three output lines. The three lines for the multiply class are designated on the logic as MULT, MULT-PT, and MULT-OT. During the execution of instructions of this class, two of the three lines are always energized simultaneously: MULT and either MULT-PT or MULT-OT. The MULP-PT and MULT-OT levels are used to provide the controlling level during the appropriate cycle, and the MULT level is used to provide the controlling level during the arithmetic pause portion of this instruction cycle. Essentially the same thing is true of the three miscellaneous class outputs because of the *SLR* instruction.

Note, also, that the shift and reset classes of instructions are represented by only one output line each. These instructions do not require an OT cycle, and, in their case, no cycle information is necessary.

The AND circuits of the class-cycle matrix combine the information contained in bits L4, L5, and L6 of the instruction word (except as noted above) with information from the cycle control, with the result that one or more output lines are selected (as necessary) for executing the required instruction.

Some of the outputs of the class-cycle matrix have two destinations. All of them are combined with the outputs of the variation matrix in the AND circuits of the instruction matrices. In addition, some of them are used to condition class-common command generators.

A class-common command is one that is common to all instructions of a given class. For example, command 43, *left memory buffer register to left A register,* is common to all variations of add class instructions (occurs at IP 7). Those outputs of the class-cycle matrix which condition class-common command generators do so directly or through OR circuits; they are not mixed with other outputs in the instruction matrix AND circuits.

2.2.4 Variation Matrix

The variation matrix (0.3.1) decodes bits L7 through L12 of the instruction word to condition one output line specifying which variation of the selected class of instructions is desired. Outputs of the variation matrix, the class-cycle matrix, and the index selector matrix are combined in the AND circuits of the appropriate instruction matrices; the outputs are d-c levels that condition command generators.

The action of the variation matrix is similar to that of the class-cycle matrix described in 2.2.3. As noted in logic 0.3.1, the majority of the variation matrix AND circuits are supplied by the outputs of bits L7 through L10 of the operations register. The four flip-flops used correspond to the four bits to be decoded. When all the inputs to any AND circuit of the variation matrix are at +10V, the output of that AND circuit is a +10V level.

Variations that use an index interval have a 3-input, rather than a 4-input, AND circuit. For these instructions, bit L10 is not used as part of the variation code but functions, instead, as part of the index interval code. Therefore, while 16 of the 20 output lines supplying instruction variations are identified by 4-digit numbers, corresponding to bits L7-L10, four output lines are identified by 3-digit numbers, corresponding to bits L7-L9. These four lines specify seven variations that utilize the index interval (see table 2–4).

Bit R1 of the *Select Drums* instruction word specifies whether the main drums or the auxiliary drums are to be selected. Bit R1 accomplishes this selection by means of two gate tubes associated with the R1 bit flip-flop of the address register. If this flip-flop is in the 0 state, it conditions one of the gate tubes, thereby causing the contents of the index interval register to be transferred to the main drum decoder; if in the 1 state, the flip-flop conditions the other gate tube and the contents of the index interval register are transferred to the auxiliary drum decoder.

2.2.5 Index Selector Matrix

The Central Computer System contains four index registers (Nos. 1, 2, 4, and 5), which may be used in the process of instruction indexing. In addition, the right accumulator register of the arithmetic element is also used as an index register, under certain circumstances, and functions as index register 3.

Instruction indexing is a process which automatically modifies the address part of an instruction without changing the basic instruction as it is stored in the core memory. Not all instructions are indexable. If the indexing process is to be used, the address portion of the selected instruction is modified immediately after the instruction is decoded.

The index selector matrix (0.3.1) decodes index selection bits L1 through L3 of the operations register to determine which index register, if any, is to be used during the execution of the instruction.

The index selector matrix may select an index register for any *one of three purposes:* the contents of the index register may be added to the address register during the execution of an indexable instruction; the index register may be reset to a new number during reset-class instructions; or the contents of the index register may be inspected and possibly modified during the execution of a *Branch and Index (BPX)* instruction.

Selection of an index register is effected by the outputs of the index selector matrix. There are six possible outputs: IV_0 through IX_5. When no index register is specified for the instruction in process, IX_0 is selected.

Index selector matrix outputs IX_1, IX_2, IX_4, or IX_5 condition the command generator associated with the selected index register (logic 0.3.1). If bit L4 or L6 of the instruction word is 0, the command generator is examined by an IP 8 delayed pulse (IP 9 for the branch and reset classes of instruction) during the PT portion of the instruction cycle. The IP 8 delayed pulse is passed by the command generator to become a command pulse, which causes the selected index register to be added to the address register (thus modifying the address).

Index selector matrix outputs IX_1, IX_2, IX_4, or IX_5 are also fed to the reset class and branch class instruction matrices, applied to AND circuits, and combined with other levels to produce output levels which are fed to the command generators in the index registers. During the execution of the *XIN, XAC,* and *BPX* instruction, the command generators are examined by a TP 4 which is passed by the selected (conditioned) command generator to become a command pulse which resets (clears) the selected index register. The modified address may then be transferred from the address register to the selected index register.

TABLE 2–4. INSTRUCTIONS THAT UTILIZE INDEX INTERVAL

CLASS	VARIATION	INSTRUCTION ABBREVIATION	BINARY CODE (CLASS AND VARIATION)
Miscellaneous	*Operate*	*PER*	000 001-
Branch	*Branch and Index*	*BPX*	101 001-
Input-output	*Select Drums*	*SDR*	110 001-
Branch	*Branch and Sense*	*BSN*	101 010-
Input-output	*Select*	*SEL*	110 010-
Miscellaneous	*Test 1 Bit*	*TOB*	000 101 0-
Miscellaneous	*Test 2 Bits*	*TTB*	000 101 1-

2.2.6 Instruction Matrices

The outputs of the class-cycle, variation, and index selector matrices are fed to the instruction matrix (logic 0.3.2), where they are combined to produce the voltage levels needed to execute the selected instruction. These levels are applied to condition the necessary command generators. When the conditioned command generators are sensed by pulses from the time pulse distributor, they allow the pulses to emerge as commands. The commands are issued in proper sequence and at the proper time to accomplish the objective of the selected instruction. The same process is followed for all 59 instructions which the Central Computer is capable of performing.

In many cases, several different instructions require the same command at exactly the same pulse time. In this case, one command generator fed by an OR circuit is provided. The several d-c instruction lines are connected to the OR circuit so that any one of the inputs can condition the selected command generator.

2.3 COMMON COMMANDS

There are 15 commands that are common to all eight classes of instructions, and they occur in every PT cycle (also in every OT cycle, if there is one). These common commands, with their logic numbers, are listed in table 2–5.

All instruction cycles begin at PT 7 with common commands 42, 47, 52, and 92. Command 42 transfers the contents of the left memory buffer register to the operations register. Command 47 initiates the parity count. Command 52 transfers the contents of the right memory buffer register to the address register and bits R10-R15 of the right memory buffer register to the step counter. Command 92 adds 1 to the program counter. No common commands occur at PT 8 or PT 9. At PT 10, there is a parity check (command 55). If the execution of the instruction requires an OT cycle, command 31 clears Memory 2 controls. At OT 0 delayed, command 71 transfers the contents of the address register to the memory address registers and starts the selected memory device. At OT 1, commands 41 and 53 clear the left and right memory buffer registers, respectively. At OT 6, command 31A clears the test memory address register. At OT 7, a parity count (command 47) is initiated, followed by a parity check (command 55) at OT 10. At PT 0, command 31 again clears Memory 2 controls. At PT 0 delayed, command 91 transfers the contents of the program counter to the memory address registers and starts the selected memory device. At PT 1, commands 41 and 53 clear the left and right memory buffer register. At PT 6, command 31A again clears the test memory address register. Command 77 clears the address register, and command 101 clears the operations register. Commands 66 *(record left overflow)* and 266 *(record right overflow)* are also considered common commands even though they do not take effect during every instruction. These commands always occur at PT 6, but are effective only during the seven instructions that can cause overflow: *Shift Left and Round (SLR), Add (ADD), Twin and Add (TAD), Add B Registers to Accumulator Registers (ADB), Subtract (SUB), Twin and Subtract (TSU),* and *Right Add One (AOR).*

2.4 INSTRUCTION CLASSES

As previously stated, there are eight instruction classes and several variations within each class. Command sequence charts for all classes and variations appear in figures 2–9 through 2–14 and figures 2–15 and 2–16 (foldouts). Detailed analyses of all instructions may be found in other parts of this manual as listed below:

a. Part 3 – All instructions of the add, multiply, shift, and store classes; the right compare instructions; the *BLM, BRM, BFZ, BFM, ETR, CSW, SLR,* and *LDB* instructions.
b. Part 4 – All reset class instructions and the *BPX* instruction.
c. Part 5 – The *BSN, PER, TOB, TTB, SEL,* and *SDR* instructions.
d. Part 6 – The *LDC, RDS,* and *WRT* instructions.

TABLE 2–5. COMMON COMMANDS

COMMAND NO.	COMMAND NAME	LOGIC NO.
31	*Clear memory No. 2 controls*	0.1.4
31A	*Clear test memory address register*	0.1.3
41	*Clear left memory buffer register*	0.1.1
42	*Left memory buffer register to operations register*	0.1.1
47	*Memory parity count*	0.1.2
52	*Right memory buffer register to address register and set step counter*	0.1.2
53	*Clear right memory buffer register*	0.1.2
55	*Parity check*	0.1.1
66	*Record left overflow*	0.5.1-2
77	*Clear address register*	0.4.1
	Clear step counter	0.5.3
91	*Program counter to memory address register*	0.4.1
92	*Add 1 to program counter*	0.4.1
101	*Clear operations register*	0.3.1
266	*Record right overflow*	0.5.2-2

	PT					OT												PT						
	7	8	9	10	11	0	1	2	3	4	5	6	7	8	9	10	11	0	1	2	3	4	5	6
COMMON COMMANDS	42 47 52 92			55		31	41 53					31	47			55		31	41 53 91					31 66* 77 101 266*
ADD-CLASS COMMON COMMANDS			114 115 117 124 212		131		21 71 230						43			64 264	161		2 4 21 81 89 202 204 230 281 289	63 263			60 260	
ADD (ADD)(104)													51											
CLEAR AND ADD (CAD)(100)												10 210	51											
TWIN AND ADD (TAD)(110)													44											
ADD B REGISTER (ADB)(114)														21 230	88 288									
SUBTRACT (SUB)(134)													51		26 226									
CLEAR AND SUBTRACT (CSU)(130)												10 210	51		26 226									
TWIN AND SUBTRACT (TSU)(140)													44		26 226									
CLEAR AND ADD MAGNITUDES (CAM)(160)												10 210	51	22 229										
DIFFERENCE MAGNITUDES (DIM)(164)												84 284	51	1 22 201 229	13 26 213 226									
CLEAR AND ADD CLOCK (CAC)(170)									A6			10 210	51											

* AFFECTS ONLY ADD, TAD, ADB, SUB AND TSU INSTRUCTIONS

Figure 2–9. Add-Class Instructions, Command Sequence Chart

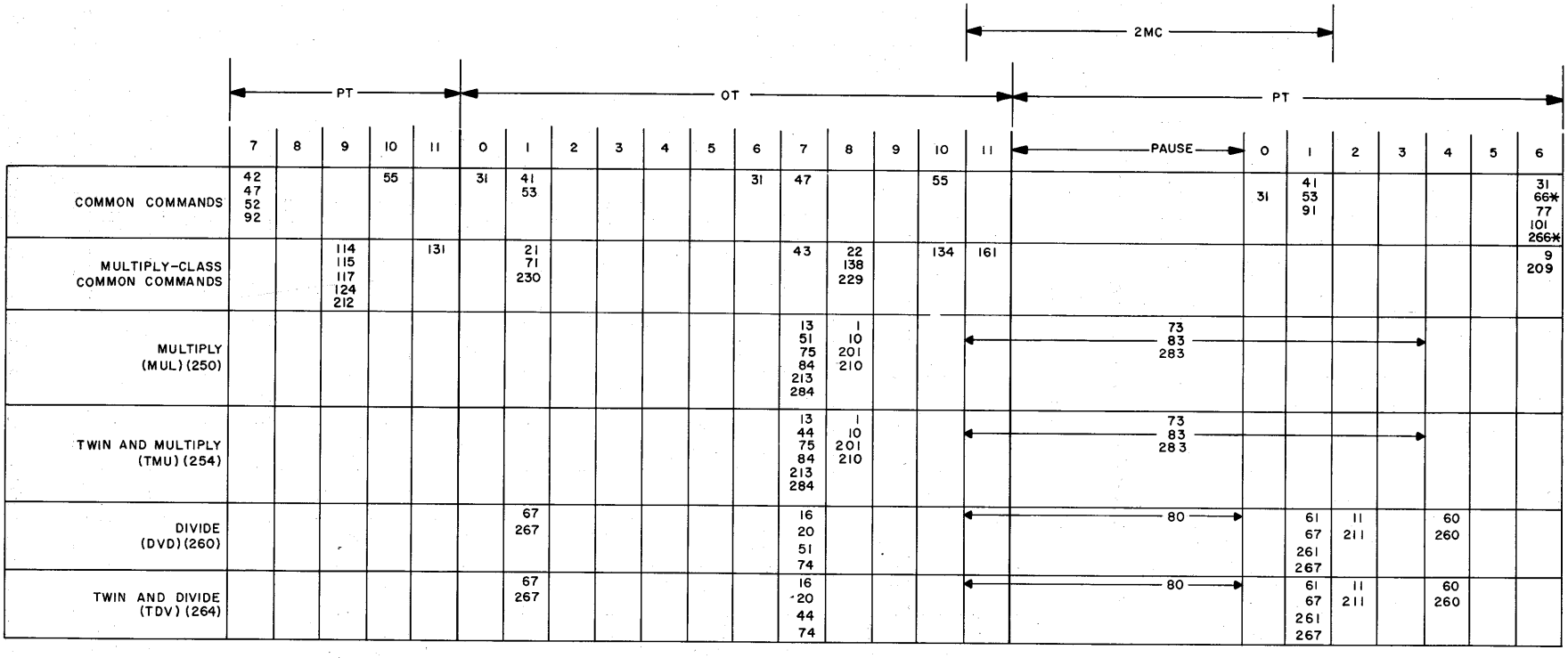


Figure 2—10. Multiply-Class Instructions, Command Sequence Chart

PT

	7	8	9	10	11	0	1	2	3	4	5	6
COMMON COMMANDS	42 47 52 92			55		31	41 53 91					31 66* 77 101 266*
ADD INDEX REGISTER (ADX) (770)			114 115 117 124 212						230		76	
RESET INDEX REGISTER (XIN) (754)										113 116 118 125		78 79 95 96
RESET INDEX REGISTER FROM RIGHT ACCUMULATOR (XAC) (764)			77	212						113 116 118 125		78 79 95 96

* NO EFFECT IN THIS CLASS OF INSTRUCTIONS

Figure 2–11. Reset-Class Instructions, Command Sequence Chart

2MC

PT

	7	8	9	10	11	PAUSE	0	1	2	3	4	5	6
COMMON COMMANDS	42 47 52 92			55		31		41 53 91					31 66* 77 101 266*
SHIFT-CLASS COMMON COMMANDS			138	134		73							
CYCLE ACCUMULATORS LEFT (FCL)(470)						2 4 12 202 204 217							
CYCLE LEFT (DCL)(460)						2 4 5 81 82 202 204 205 281 282							
RIGHT ELEMENT SHIFT RIGHT (RSR)(444)						123 207 285							
LEFT ELEMENT SHIFT RIGHT (LSR)(440)						7 18 85							
SHIFT ACCUMULATORS RIGHT (ASR)(424)						18 123							
SHIFT RIGHT (DSR)(404)						7 18 85 123 207 285							
SHIFT LEFT (DSL)(400)						2 5 81 82 202 205 281 282							
SHIFT ACCUMULATORS LEFT (ASL)(420)						2 6 202 206							

* NO EFFECT IN THIS CLASS OF INSTRUCTIONS

Figure 2–12. Shift-Class Instructions, Command Sequence Chart

	PT					OT											PT						
	7	8	9	10	11	0	1	2	3	4	5	6	7	8	9	10	11	0	1	2	3	4	5	6
COMMON COMMANDS	42 47 52 92			55		31	41 53					31	47			55		31	41 53 91					31 66 77 101 266
IO-CLASS COMMON COMMANDS			114 115 117 124 212	134																				
SELECT (SEL)(62—)					131		71				155						161			146	151		156	
SELECT DRUMS (SDR)(61—)					131		71				155						161			146	151		325	
LOAD IO ADDRESS COUNTER (LDC)(600)						← NO OT →														148	72			
READ (RDS)(670)						← NO OT →													149 294	144 147 152	290			180
WRITE (WRT)(674)						← NO OT →													149 294	144 147 152	290			182

Figure 2—13. IO-Class Instructions, Command Sequence Chart

	PT					OT											PT						
	7	8	9	10	11	0	1	2	3	4	5	6	7	8	9	10	11	0	1	2	3	4	5	6
COMMON COMMANDS	42 47 52 92			55		31	41 53					31	47			55		31	41 53					31 66* 77 101 266*
BRANCH-CLASS COMMON COMMANDS			21 230															154	100					163
BRANCH ON FULL MINUS (BFM)(554)			162		93	NO OT																		
BRANCH ON LEFT MINUS (BLM)(550)			165		93	NO OT																		
BRANCH ON RIGHT MINUS (BRM)(554)			166		93	NO OT																		
BRANCH AND SENSE (BSN)(52–)					131 321		71								140		93 161							
BRANCH ON FULL ZERO (BFZ)(540)					131		13 71 213	19 219	62 69			2 4 81 89 202 204 281 289	19 219	62 69 162			9 4 81 89 93 161 202 204 281 289							214 216
BRANCH AND INDEX (BPX)(51–)			119 126 164 170 174		93	NO OT													77	103	114 115 117 124	113 116 118 125		78 79 95 96

* NO EFFECT ON THIS CLASS OF INSTRUCTIONS

Figure 2–14. Branch-Class Instructions, Command Sequence Chart

CHAPTER 3
PULSE GENERATION AND CONTROL

3.1 GENERAL

The instruction control element furnishes two types of pulses to the Central Computer System: repetitive and nonrepetitive. Among the repetitive pulses are the 2-mc control pulses, 2-mc operate pulses, instruction pulses (IP's), time pulses (TP's), and divide time pulses (DVTP's). The nonrepetitive pulses include command pulses (CP's) and certain other pulses, BO and BI, which function as command pulses although they are not so designated.

The 2-mc control pulses are generated continuously during every type of computer operation. All other repetitive and nonrepetitive pulses are generated or inhibited as required by the program to be executed.

All pulses are 0.1 μsec in width and between 20V and 40V in amplitude. The TP driver supplies pulses, occurring at a 2-mc rate, to the time pulse distributor (logic 0.2.3). The pulses are then distributed on 12 TP lines; the repetition time in each line is 6.0 μsec. The IP's are similarly derived and distributed on 12 lines; the repetition time in each line is again 6.0 μsec. The DVTP's are distributed on three lines; the repetition time in each line is 2.5 μsec.

Nonrepetitive pulses have no specific repetition rate; they occur when and as required by the program to be executed.

3.2 BLOCK DIAGRAM ANALYSIS

The time pulse distributor converts the 2-mc control pulses into time pulses. As noted in logic 0.2.3, successive 2-mc control pulses on the TP driver line are sequentially routed to 12 time pulse output lines. The routing is accomplished through 12 flip-flops, of which only one is in the set condition at any particular time. A 2-mc control pulse enters the circuit and, sensing the flip-flops, finds only one is set and passes through the gate conditioned by the set flip-flop to the associated output line. Simultaneously, the pulse clears the set flip-flop and sets the next one. The succeeding entering pulse, finding the next flip-flop the only one set, passes through the associated gate to the next output line. The pulse that senses the last flip-flop of the circuit clears that flip-flop and resets the first; thus the process repeats itself indefinitely.

The time pulse distributor also converts 2-mc control pulses into instruction pulses by conditioning a second set of gate tubes.

The divide time pulse distributor has five flip-flops and three output lines. The divide time pulse distributor operates in the same manner as the time pulse distributors, except that it distributes 2-mc operate pulses instead of 2-mc control pulses.

The primary function of the time pulses is to provide a timing reference for the Central Computer System. Certain pulses are sent to the core memory element to initiate memory cycles; others are sent to the IO element, where they are converted into BI and BO pulses.

Instruction pulses are used to sample the command generators in the instruction control element. The command generators are conditioned by d-c levels; when a conditioned command generator is sampled by an instruction pulse, the instruction pulse is converted into a command pulse.

Time pulses and instruction pulses are inhibited during an arithmetic pause. Two-mc operate pulses are generated and used throughout the pause.

Divide time pulses are generated from the 2-mc operate pulses during a *Divide* or *Twin and Divide* instruction. The 2.5-μsec repetition time of the divide time pulses is required for the execution of the divide operation.

The units comprising the pulse generation and control section of the instruction control element and the interconnections of the units are shown in figure 2–17.

3.3 FUNCTION OF VARIOUS PULSES

The instruction control element supplies the following pulses to the Central Computer System:

a. 2-mc control pulses
b. Time pulses
c. Instruction pulses
d. 2-mc operate pulses
e. Divide time pulses
f. Command pulses
g. BI pulses
h. BO pulses

The several pulses are used as follows:

a. The 2-mc control pulses synchronize, control, and co-ordinate computer operations.

b. Time pulses are generally used to provide a timing reference for the Central Computer System. Some of the time pulses are used to generate commands which are common to all instructions that the Central Computer System can perform. All 12 time pulses are also routed to the IO element, where they are converted into BI and BO pulses whenever a break cycle occurs.

c. Instruction pulses sample the command generators that are not sampled by time pulses during internal computer operations. The command generators which have been conditioned by the decoded instructions then produce command pulses.

d. The 2-mc operate pulses are used to carry out the repetitive operations required by some instructions. Whenever an instruction requires more time for execution than a normal operating cycle allows, an arithmetic pause is generated to permit its completion. During an arithmetic pause, the 2-mc operate pulses continue to carry forward the execution of the instruction, but the time pulses and instruction pulses are inhibited until the end of the pause. Regardless of when the pause terminates, the 2-mc operate pulses continue until the repetitive operation is complete.

All multiply instructions, shift instructions that involve more than six shifts, and a *Shift Left and Round* instruction that involves more than one shift require an arithmetic pause.

Another pause condition, distinct from the arithmetic pause, is the IO pause. An IO pause results from the introduction of a new IO instruction while a previous IO operation is still in process, or from the execution of the *HLT* instruction while an IO operation is in process. During an IO pause, internal computer pulses (except 2-mc control pulses) are not generated; therefore, no internal computer operation is possible. During an IO pause condition, computer time is lost until the IO pause is terminated.

e. Divide time pulses are used only with *Divide* and *Twin and Divide* instructions. In the execution of the multiply instructions, the 2-mc operate

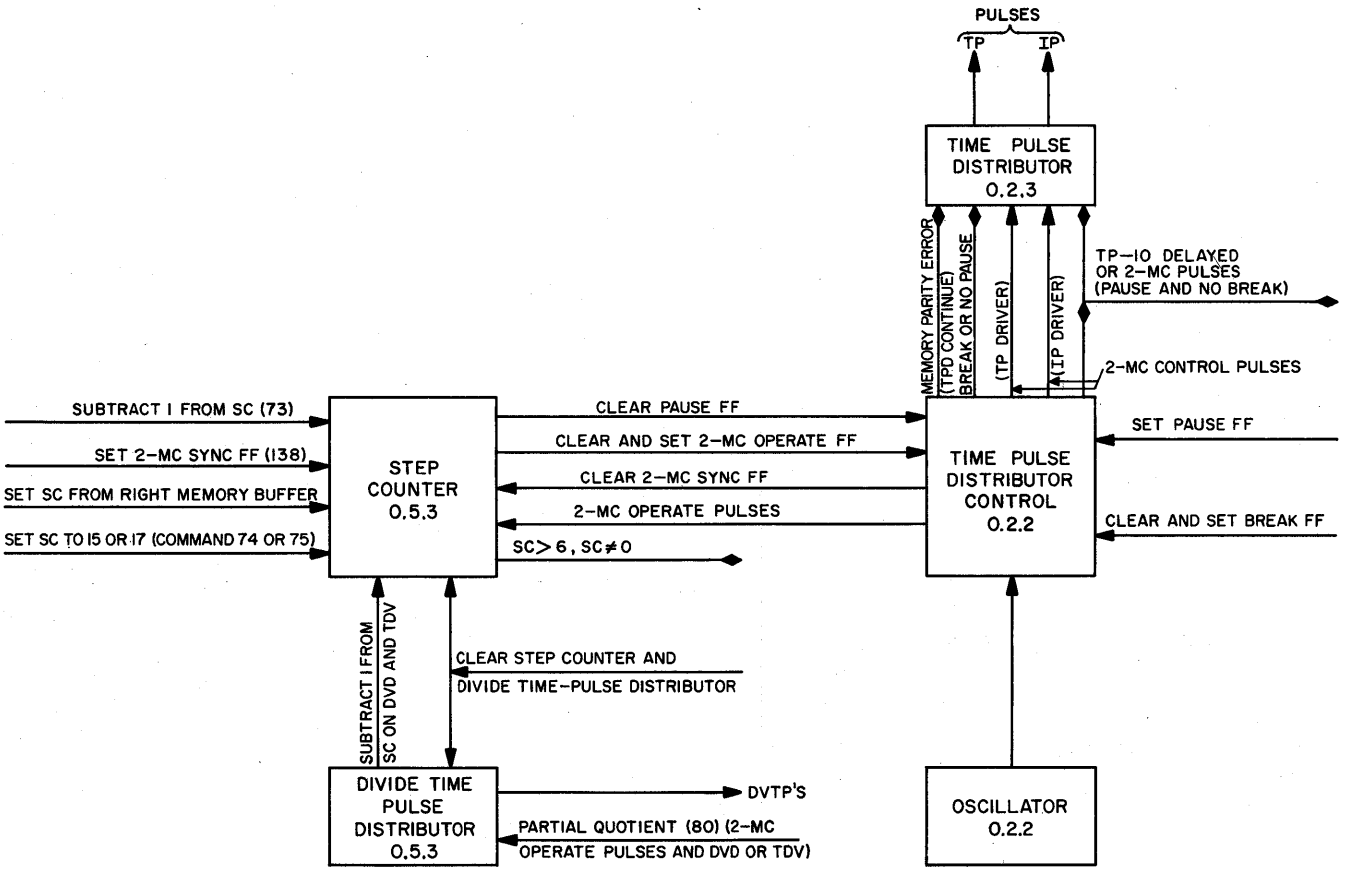


Figure 2–17. Pulse Generation and Control, Logic Block Diagram

pulses are generated to perform the repetitive operations required. However, because of the complexity involved in the execution of the divide operations of multiply instructions, the 2-mc operate pulses are not used directly. Instead, they are fed to the divide time pulse distributor, which converts them into divide time pulses. Divide time pulses occur at a repetition time of 2.5 μsec in each output line; however, the time interval between successive divide time pulses is 0.5 μsec.

f. Command pulses (commands) are of standard amplitude and width but have no specific repetition rate. Commands are generated whenever a TP, DVTP, IP, or 2-mc pulse senses a conditioned command generator.

g. BI pulses are used in IO operations when information from an input device is to be stored in the memory element.

h. BO pulses are used in IO operations when information from the memory element is to be transferred to an output device.

The instruction control element does not generate BI and BO pulses, but supplies time pulses during break cycles. The time pulses are converted into BI and BO pulses, as required, in the IO element.

As noted in logic 0.2.3, pulses from the oscillator are fed to the time pulse distributor control, where they are divided into two parallel lines designated time pulse driver and instruction pulse driver. The two outputs are then fed to the time pulse distributor. In the time pulse distributor, the time pulse driver input is routed through a system of 12 output lines to produce time pulses. Only one line is open at any given time, and, in the process of traversing the only open line, the pulse closes that line and opens the next. The succeeding pulse traverses the newly opened line, closes it, and opens the next. After all 12 outputs have been pulsed in this manner, the process is repeated.

The time pulse distributor also receives an input from the IP driver and distributes it among 12 output lines. The process for producing instruction pulses is identical with that used for the TP outputs.

Divide time pulses are required for execution of *Divide* and *Twin and Divide* instructions. Each division requires 16 partial quotient operations, and each partial quotient needs five divide time pulses. The divide time pulse distributor receives an input consisting of 2-mc operate pulses and distributes it among three output lines. Once again, the process is identical with that used in the production of time pulses.

3.4 TIME PULSE DISTRIBUTOR CONTROL

In the time pulse distributor control (logic 0.2.2), the pulses derived from the crystal-controlled oscillator are distributed as directed by signals received from other units, including manual controls, which reflect the current status of the Central Computer System. The following discussion will apply to automatic control as a result of program execution. The effects of manual controls (pushbuttons on the maintenance console) on these same circuits are discussed in Part 8.

The time pulse distributor control routes 2-mc pulses to the time pulse distributor for conversion into time pulses and instruction pulses. These pulses are in turn converted into command pulses to perform the required operations specified by the program instructions. During the execution of multiply and shift instructions, repetitive operations must be performed. During the execution of these instructions, the repetitive operations are executed by the repetitive generation of a single command pulse. The number of repetitive operations to be performed is indicated and controlled by the step counter.

When an arithmetic pause is required, the time pulse distributor control temporarily terminates the generation of time and instruction pulses at the end of a machine cycle. Since all repetitive operations are controlled by 2-mc operate pulses, only these pulses are generated during an arithmetic pause. At the end of the pause, the 2-mc operate pulses are terminated and time pulses and instruction pulses are again generated.

3.4.1 Time Pulse Distributor Control Flip-Flop

The time pulse distributor control flip-flop is normally set during computer action. When set, it allows pulses from the 2-mc oscillator to pass through a gate (logic 0.2.2, GTI, 4BJ) to become TP driver pulses and TPD-on pulses. The latter pulses are applied to the step counter to sense the 2-mc sync, clear pause delay sync, and clear pause delay flip-flops. The TPD-on pulses also become IP driver pulses and sense the pause flip-flop for a pause-and-no-break condition. The operation described above continues until the TPDC flip-flop is cleared to the 0 state.

3.4.2 Two-MC Operate Flip-Flop

The 2-mc operate flip-flop is set to the 1 state by command 138 during MULT-OT, SHIFT and SC $\neq$ 0, or *SLR*-PT and SC $\neq$ 0. These instructions involve repetitive operations and, therefore, require 2-mc operate pulses. During the execution of an instruction, the 2-mc operate flip-flop may be cleared to the 0 state by the output of either the SC $<$ 2 or the SC $<$ 4 line, depending on the original contents of the step counter.

During the execution of instructions in which the step counter is not set higher than 1, the gate controlled by the SC < 4 line is never conditional; therefore, the SC < 2 line acts to interrupt the 2-mc operate pulses as soon as one 2-mc operate pulse has passed through the gate. For instructions in which the SC < 4 line is energized, the SC < 4 output clears the 2-mc operate flip-flop. Because of delays in this action, two additional 2-mc operate pulses are allowed to pass; therefore, the 2-mc operate pulses are not interrupted until the step counter is at 0 (or 1 in a divide operation).

3.4.3 Break and Pause Flip-Flops

The outputs of the break and pause flip-flops are arranged so that four operational set-and-clear combinations are possible. If, in the course of Central Computer internal operation, an arithmetic pause is specified by the instruction, the pause flip-flop is set as the instruction requires; if a break request occurs, the break flip-flop is set at the end of the cycle then in progress.

During a no-break no-pause condition, both the TP driver and IP driver outputs supply pulses to the time pulse distributor. During this time, the break or no-pause output level is applied to the AND circuit associated with the TPD 0 stage to allow the time pulse distributor to continue supplying time pulses and instruction pulses to the Central Computer System. During the resultant internal operation of the computer, the break-request flip-flop is sensed at TP 10 delayed of each memory cycle to determine whether the next memory cycle is to be associated with either a BI or BO cycle.

During a break-but-no-pause condition, the IP driver output is inhibited to terminate the internal operations of the computer. However, since the break-or-no-pause line remains at the +10V level, the time pulse distributor will continue to function to develop time pulses.

When a pause-but-no-break condition occurs, the generation of time pulses and instruction pulses stops because the break-or-no-pause output line is de-energized, causing the time pulse distributor to be deconditioned. Under this operating condition, the break request flip-flop is sensed at a 2-mc rate.

When a break-and-pause condition exists, the IP driver output line is inactive, but the break-or-no-pause level to the TPD 0 stage is energized to keep the time pulse distributor active and functioning. The break-request flip-flop is examined by the sense-break-request output at TP 10 delayed.

3.4.4 Continue Flip-Flop

When set, the continue flip-flop conditions the AND circuit associated with the 1 side output of the TPD 7 flip-flop. The effects of the continue flip-flop outputs are discussed in 3.5 in connection with the time pulse distributor.

3.5 TIME PULSE DISTRIBUTOR

3.5.1 General

The function of the time pulse distributor (logic 0.2.3) is to convert 2-mc control pulses into time pulses and instruction pulses under the control of the time pulse distributor control. Time pulses are produced by distributing the 2-mc pulses (0.5-μsec repetition time) among 12 time pulse lines; the pulse repetition time in each TP line is 6.0 μsec. Instruction pulses are similarly produced.

The oscillator provides 2-mc control pulses which are routed to the time pulse distributor control. The time pulse distributor control determines whether the time pulse distributor will produce time pulses and instruction pulses, time pulses only, or no pulses of any kind.

Control of the time pulse distributor is exercised by the pause, break, continue, and alarm-stop flip-flops. The outputs of the pause, break, and alarm-stop flip-flops affect the AND circuit associated with the set side of the TPD 0 flip-flop. The output of the continue flip-flop affects the AND circuit associated with the set side of the TPD 7 flip-flop.

3.5.2 Time Pulse Generation

The discussion of the generation of time pulses refers to logic 0.2.3 (time pulse distributor) and assumes that the TPD 0 flip-flop is set. It also assumes a break-or-no-pause condition and no memory parity error. These assumptions are made to fully condition the AND circuit associated with the TPD 0 flip-flop. When the AND circuit is fully conditioned, its output level conditions the gate associated with the TPD 0 flip-flop. When the conditioned gate is examined by a pulse from the TP driver, it allows the pulse to pass and become a TP 0 pulse. The pulse also clears the TPD 0 flip-flop and sets the TPD 1 flip-flop, which conditions its associated gate (TP 1 gate). When the next pulse from the TP driver examines the 12 TP gates, it finds that only the TP 1 gate is conditioned and passes through that gate to become the TP 1 pulse, which clears the TPD 1 flip-flop and sets the TPD 2 flip-flop. If the continue flip-flop is on (supplying a conditioning level to the AND circuit associated with the TPD 7 flip-flop), the process continues, in sequence, until the TP 11 pulse is generated. The output of the TP 11 gate becomes the TP 11 pulse, clears the TPD 11 flip-flop, and sets the TPD 0 flip-flop. If all assumed conditions are met at this point in each sequence, the process will be repeated indefinitely. It is well to note that this sequence occurs in normal computer operations and that altered conditions will affect this sequence. These are discussed in succeeding paragraphs.

If a pause-and-no-break condition occurs at the same time the TPD 0 flip-flop is set, no time pulses will

be generated. This situation exists because the AND circuit which normally applies a conditioning level to the TP 0 gate is not fully conditioned. Therefore, the TP 0 gate will not pass the pulse received from the TP driver line, the TPD 0 flip-flop will not be cleared, the TPD 1 flip-flop will not be set, and, since no other gate is conditioned, the generation of time pulses is interrupted. This condition will exist until the pause is terminated or a break (during the pause) is instituted. When either of the above conditions arises, the TPD 0 AND circuit will be fully conditioned and its output level will condition the TP 0 gate, allowing the generation of time pulses to resume when the TPD receives the next pulse from the TP driver line.

During normal computer operations, the AND circuit associated with the TPD 7 flip-flop is conditioned by the set output of the continue flip-flop. If the continue flip-flop is cleared to terminate the operation of the computer, the generation of time pulses will be interrupted at TP 6. Clearing of the continue flip-flop may be caused or accomplished by any of the following operations:

a. Execution of the *Program Stop (HLT)* instruction
b. Alarm Stop
c. Depression of the PROGRAM STOP pushbutton located on the maintenance console.
d. Depression of one of the ACTIVE pushbuttons (panels A and B of duplex switching console) to effect switch-over from one computer (A or B) to the other
e. An operation involving the cyclic program counter (CPC) (See Part 8.)

Time pulse generation will be resumed when the continue flip-flop is again permanently set to the 1 state; this occurs when the computer is placed in continuous operation (under program control). Continuous operation of the computer is initiated by depressing one of the following manual control pushbuttons:

a. START FROM TEST MEMORY
b. LOAD FROM CARD READER
c. LOAD FROM AM DRUMS
d. PROGRAM CONTINUE

During manual operation of the computer (for maintenance purposes), the generation of time pulses and instruction pulses can be initiated and terminated by use of one of three pushbuttons. Depression of the MEMORY CYCLE, INST STEP, or SINGLE PULSE pushbuttons sets the continue and TPD control flip-flops to allow the desired memory cycle, instruction cycle, or single-pulse operation to be executed. In each case, the actuated pushbutton also sets up control so that the continue and TPD control flip-flops will be cleared after the desired operation is completed.

3.5.3 Instruction Pulse Generation

Instruction pulses are generated in exactly the same manner as time pulses. Any interruption in the generation of time pulses also interrupts the generation of instruction pulses because they are produced under TP control. However, instruction pulses may be interrupted, without interrupting time pulses, by inhibiting the IP driver line.

3.6 DIVIDE TIME PULSE DISTRIBUTOR

The purpose of the divide time pulse distributor (logic 0.5.3) is to convert 2-mc operate pulses into divide time pulses occurring every 2.5 μsec in each DVTP output line. The divide time pulses are used to perform the 16 repetitive operations (partial quotients) which are required by *Divide (DVD)* and *Twin and Divide (TDV)* instructions.

The divide time pulse distributor consists of five flip-flops (DVTPD 0 through DVTPD 4). The action of these flip-flops in producing divide time pulses is exactly the same as that of the TPD flip-flops in producing time pulses and instruction pulses. Only three DVTP output lines (DVTP 0, DVTP 1, and DVTP 4) are used for the divide operation. The output of the DVTPD 2 flip-flop is used to provide for proper sequencing and for initiating the termination of the required arithmetic pause. The output of the DVTPD 3 flip-flop is used to substract one from the step counter for each DVTPD cycle, thus keeping a count of the number of DVTPD cycles that remain to be performed.

A *Divide* or *Twin and Divide* instruction is executed under the control of the divide time pulse distributor as follows: command 138 (set 2-mc set-sync flip-flop) is generated at IP 8, and 2-mc operation is commenced immediately after the 2-mc operate flip-flop is set. The 2-mc operate pulses are applied to the divide time pulse distributor (command 80, partial quotient). At IP 10, command 134 (set pause flip-flop) is generated and, since the step counter was previously set to 17, a pause is initiated.

When the step counter contains a count of less than 4, the SC $<$ 4 line conditions a gate which is examined when the step counter is being stepped from 2 to 1. The output pulse of the gate initiates action to stop the generation of 2-mc operate pulses and to clear the pause flip-flop. Because of circuit delays, interruption of the 2-mc operate pulses does not occur until the step counter reads 1 and the pause is terminated 0.5 μsec after 2-mc operate pulse generation ceases.

3.7 STEP COUNTER

3.7.1 General

The purpose of the step counter (logic 0.5.3) is to keep a count of the number of repetitive operations required by some instructions. It also periodically reports the current status of the count to the Central

Computer System and to the flip-flops which start, stop, and control the repetitive operations.

All instructions requiring repetitive operations which are activated by 2-mc operate pulses, either directly or through the divide time pulse distributor, must employ the step counter. Instructions utilizing the step counter are:

a. All multiply class
b. All shift class
c. *Shift Left and Round (SLR)*

When one of the above instructions is to be executed, the step counter is loaded with a number which designates the number of repetitive operations to be performed. For all shift instructions, the number of shifts to be performed is specified by bits R10-R15 of the instruction word, which is loaded into the step counter at PT 7 of every instruction. For the multiply instructions, this content is changed by either command 75, which sets the step counter to 15 for the *MUL* and *TMU* instructions, or command 74, which sets the step counter to 17 for the *DVD* and *TDR* instructions. Once the counter has been set, the number is reduced by 1 each time a repetitive operation is performed.

3.7.2 Step Counter Operation

During a shift instruction or the *SLR* instruction, conditional command 138 is generated at PT 9 to set the 2-mc operate flip-flop. The command is generated in every case, but the 2-mc operate flip-flop is set only if the step counter contents are greater than zero. At PT 10, command 134 is generated to set the pause flip-flop. This action is conditional and depends on the number of shifts to be performed and the type of shift instruction specified. If a shift instruction is being executed and more than six shifts are specified, the pause flip-flop is set. However, if six or less than six shifts are specified, a pause is not necessary and the flip-flop is not set. During the *SLR* instruction, the pause flip-flop is set if the step counter contents are greater than zero.

During multiply instructions, command 138 is generated at OT 8 and 2-mc operation commences at OT 10. Command 134 is generated at OT 10 and, since the step counter was previously set to 15 *(MUL* and *TMU)* or 17 *(DVD* and *TDV)*, an arithmetic pause occurs after the OT cycle is completed (OT 11).

The Central Computer System must be informed of the current status of the counter at all times so that it may synchronize the operations necessary to execute any of these instructions. Accordingly, the step counter furnishes the proper information relative to the number held in the counter. This information is supplied by one of the following signals (control levels):

a. SC $\neq 0$
b. SC > 6
c. SC < 8
d. SC < 4
e. SC < 2

The functions of the step counter control levels appear in table 2–6.

The SC $\neq 0$ status is detected by an OR circuit which receives inputs from the 1 side of each of the six counting flip-flops. If any of the flip-flops contains a 1, the number held in the counter is greater than, and therefore not equal to, 0, so the SC $\neq 0$ level is developed. The SC $\neq 0$ level conditions a gate tube which is sampled by command 138. The output of the gate sets the 2-mc operate flip-flop, thus starting the generation of 2-mc operate pulses.

The SC > 6 status is detected by a circuit composed of an AND circuit and an OR circuit. If any of the three higher-order bits (8, 16, or 32) contains a 1, or if all the lower-order bits (1, 2, and 4) contain 1's, the total number in the counter is greater than 6. The step counter does not make direct use of this information, but routes it to the shift-class instruction matrix.

The SC < 8 status is detected by an AND circuit and conditions a gate. The number held by the step counter is less than 8 when all three of the higher-order flip-flops contain 0's. The step counter makes direct use of this information to control further progress of the instruction. The exact details of the control exercised depend upon the nature of the instruction. (Refer to table 2–6.)

The SC < 4 status is detected by an AND circuit and conditions a gate tube. The SC < 4 level occurs when the four higher-order bits contain 0's. This level is also used to control the further progress of the instruction. Again, the details of the control exercised depend upon the nature of the instruction. (See table 2–6.)

The SC < 2 status is detected by an AND circuit and conditions a gate. This status exists when the five higher-order bits all contain 0's. It is used to control the progress of the instruction. (Refer to table 2–6.)

To complete the execution of instructions controlled by the step counter, regardless of class, control levels SC < 8, SC < 4, and SC < 2 perform the following functions: terminate the pause, if there is one, and stop the 2-mc operate pulses. However, these control levels do not always perform these functions in the same order, nor at the same time. For example, in the *SLR, DVD,* and *TDV* instructions, the 2-mc operate flip-flop is cleared before the pause flip-flop is cleared; in the *MUL* and *TMU* instructions and in all shift instructions, the pause flip-flop is cleared first.

TABLE 2–6. FUNCTION OF STEP COUNTER CONTROL LEVELS

INSTRUCTION	SC ≠ 0	SC > 6	SC < 8	SC < 4	SC < 2
Shift class — More than one shift specified.	Controls generation of command 138 to set 2-mc sync flip-flop.	Controls generation of command 134 to set pause flip-flop if more than six shifts specified.	Controls operation to clear pause flip-flop if more than six shifts were specified. The control signal to clear pause flip-flop is generated when step counter is stepped from 7 to 6; because of delays, however, flip-flop is not cleared until step counter contents are equal to 4.	Controls operation to clear 2-mc operate flip-flop. The control pulse to clear this flip-flop is generated when step counter is stepped from 2 to 1; because of delays, however, 2-mc operate flip-flop is not cleared until step counter is equal to 0.	No effect.
Shift class — Only one shift specified.	Controls generation of command 138 to set 2-mc sync flip-flop.	Condition does not exist.	No effect.	No effect.	Controls complementing of 2-mc operate flip-flop. This flip-flop is first complemented (to set status) to generate one 2-mc operate pulse, after which it is complemented again (to clear status) to terminate 2-mc operation.
MUL, TMU	Not applicable (command 138 is generated at OT 8 to set 2-mc sync flip-flop).	Not applicable (command 134 is generated at OT 10 to set pause flip-flop).	Controls operation to clear pause flip-flop. The control signal to clear pause flip-flop is generated when step counter is stepped from 7 to 6; because of delays, however, flip-flop is not cleared until step counter contents are equal to 4.	Controls operation to clear 2-mc operate flip-flop. The control pulse to clear this flip-flop is generated when step counter is stepped from 2 to 1; because of delays, however, flip-flop is not cleared until step counter is equal to 0.	No effect.

TABLE 2–6. FUNCTION OF STEP COUNTER CONTROL LEVELS (cont'd)

INSTRUCTION	SC ≠ 0	SC > 6	SC < 8	SC < 4	SC < 2
DVD, TDV	Not applicable (command 138 is generated at OT 8 to set 2-mc sync flip-flop).	Not applicable (command 134 is generated at OT 10 to set pause flip-flop).	No effect.	Controls operation to clear pause and 2-mc operate flip-flops. The control pulse to clear pause flip-flop is generated by gating a DVTP 2 pulse. When the SC is stepped from 3 to 2 (by a DVTP 3 pulse), divide clear pause delay flip-flop is set to condition a gate to pass next DVTP 2 pulse, which initiates clearing of pause flip-flop. Because of delays, however, pause flip-flop is not cleared until 1.5 μsec later (0.5 μsec after next DVTP 4 pulse). Next DVTP 3 pulse steps SC from 2 to 1. This pulse is used to initiate clearing of 2-mc operate flip-flop; because of delays, however, this flip-flop is not cleared until 0.5 μsec after next DVTP 4 pulse is generated. As a result, 2-mc operate and pause flip-flops are cleared at same time. The SC contains 1 at end of divide operation.	No effect.

TABLE 2–6. FUNCTION OF STEP COUNTER CONTROL LEVELS (cont'd)

INSTRUCTION	SC $\neq 0$	SC > 6	SC < 8	SC < 4	SC < 2
SLR – More than two shifts specified.	Controls generation of commands 138 and 134 to set 2-mc sync and pause flip-flops.	No effect.	No effect.	Controls operation to clear pause and 2-mc operate flip-flops. The control signal used to clear pause flip-flop is generated when SC is stepped from 3 to 2, while control signal to clear 2-mc operate flip-flop is generated when SC is stepped from 2 to 1. Because of delays, flip-flops are not cleared until SC contains 0.	No effect.
SLR – Two shifts specified	Controls generation of commands 138 and 134 to set 2-mc sync and pause flip-flops.	Condition does not exist.	No effect.	Controls operation to clear 2-mc operate and pause flip-flops. The control signal to clear 2-mc operate flip-flops is generated when SC is stepped from 2 to 1; because of delays, however, flip-flop is not cleared until SC contains 0. The control signal to clear pause flip-flop is generated when SC is stepped from 1 to 0; however, pause flip-flop is not cleared until 1.0 μsec after SC contains 0.	No effect.
SLR – One shift specified	Controls generation of commands 138 and 134 to set 2-mc sync and pause flip-flops.	Condition does not exist.	No effect.	No effect.	Controls complementing of 2-mc operate flip-flop and clearing of pause flip-flop.

TABLE 2–6. FUNCTION OF STEP COUNTER CONTROL LEVELS (cont'd)

INSTRUCTION	SC ≠ 0	SC > 6	SC < 8	SC < 4	SC < 2
					The control pulse to clear pause flip-flop is generated when 2-mc operate flip flop is first complemented to generate one 2-mc operate pulse. After 2-mc operate pulse is generated, 2-mc operate flip-flop is complemented again to terminate 2-mc operation. Because of delays, pause flip-flop is not cleared until 0.5 μsec after SC contains 0.

PART 3
ARITHMETIC ELEMENT

CHAPTER 1
INTRODUCTION

1.1 GENERAL

The arithmetic element of the Central Computer System is composed of two identical groups of circuits, each of which contains a 16-bit A register, a B register, an accumulator register, a binary adder, and a compare circuit. The individual registers and circuits are identified by the prefixed designation left or right to indicate the group in which the circuits are located.

The logic block diagram in figure 3–1 shows the interrelationship of the registers and adders and also shows the information transfer paths between the arithmetic element and the memory and program elements. To simplify the drawing, the command pulses from the instruction control element are not shown. The memory storage devices and the memory buffer registers (MBR) are included to show the association of the registers in each group with the corresponding memory half-word.

As noted in the figure, only the right arithmetic registers are provided with information transfer paths to and/or from the program and IO elements. These paths are associated with the right A register and the right accumulator register and are the means by which these registers can be utilized for the temporary storage of program information. The left and right accumulator registers are interconnected to enable the exchange of information between the two registers during certain shift-class instructions. The left MBR is also connected to the right A register during the execution of twin type instructions (*TAD*, *TSU*, *TMU*, and *TDV*) wherein the operand in the left MBR is used in both the left and right portions of the element.

For most arithmetic instructions, the two portions of the arithmetic element operate separately to accommodate the two 16-bit half-words. However, for those instructions that require a 17-bit operand, the element operates as a single unit (the left and right portions operating together). For 17-bit operation, the 16-bit right adder is connected to the bit LS adder to form a 17-bit adder. (The remaining 15 bits are not used.)

1.2 ARITHMETIC ELEMENT FUNCTIONS

The arithmetic element is used for computations involving addition, subtraction, multiplication, and division, for comparison or modification of selected information within a specified memory word, and for temporary storage of program information. Each of these operations is accomplished by sequenced command pulses from the instruction control element.

During all arithmetic operations, the A register, the adder, and the accumulator register form a functional unit. The adder combines operands from the A register and the accumulator register and stores the results in the accumulator register. After an arithmetic operation, the accumulator register contains some form of the information listed in the last column of table 3–1.

All arithmetic operations involve variations of the add process. During the add operation, a command pulse to the carry-0 line of the rightmost adder circuit starts a ripple through the adder circuit, and the sum appears in the accumulator register shifted one bit position to

TABLE 3–1. INFORMATION CONTENTS OF ACCUMULATOR REGISTER

ARITHMETIC OPERATION	INITIAL CONTENTS	FINAL CONTENTS
Addition	Augend	Sum
Subtraction	Minuend	Difference
Multiplication	Multiplier	Sign bit plus the 15 most significant bits of the product
Division	Sign bit and the 15 most significant bits of the dividend	Sign bit of the quotient and magnitude of the remainder

the right. This is the synchronous-add-and-shift process and occurs in all basic arithmetic operations. The inherent shift right, a distinguishing feature of this process, is important to the multiplication operation because it permits addition to progress while the partial products are being generated. This is a time-saving device for multiplication, and the correctional shift left that is necessary for addition and subtraction does not increase the execution time of the latter instruction. As a result of the inherent shift right that is produced during addition, subtraction, and division, the sum bit of the bit 15 adder is temporarily stored in the B register sign bit

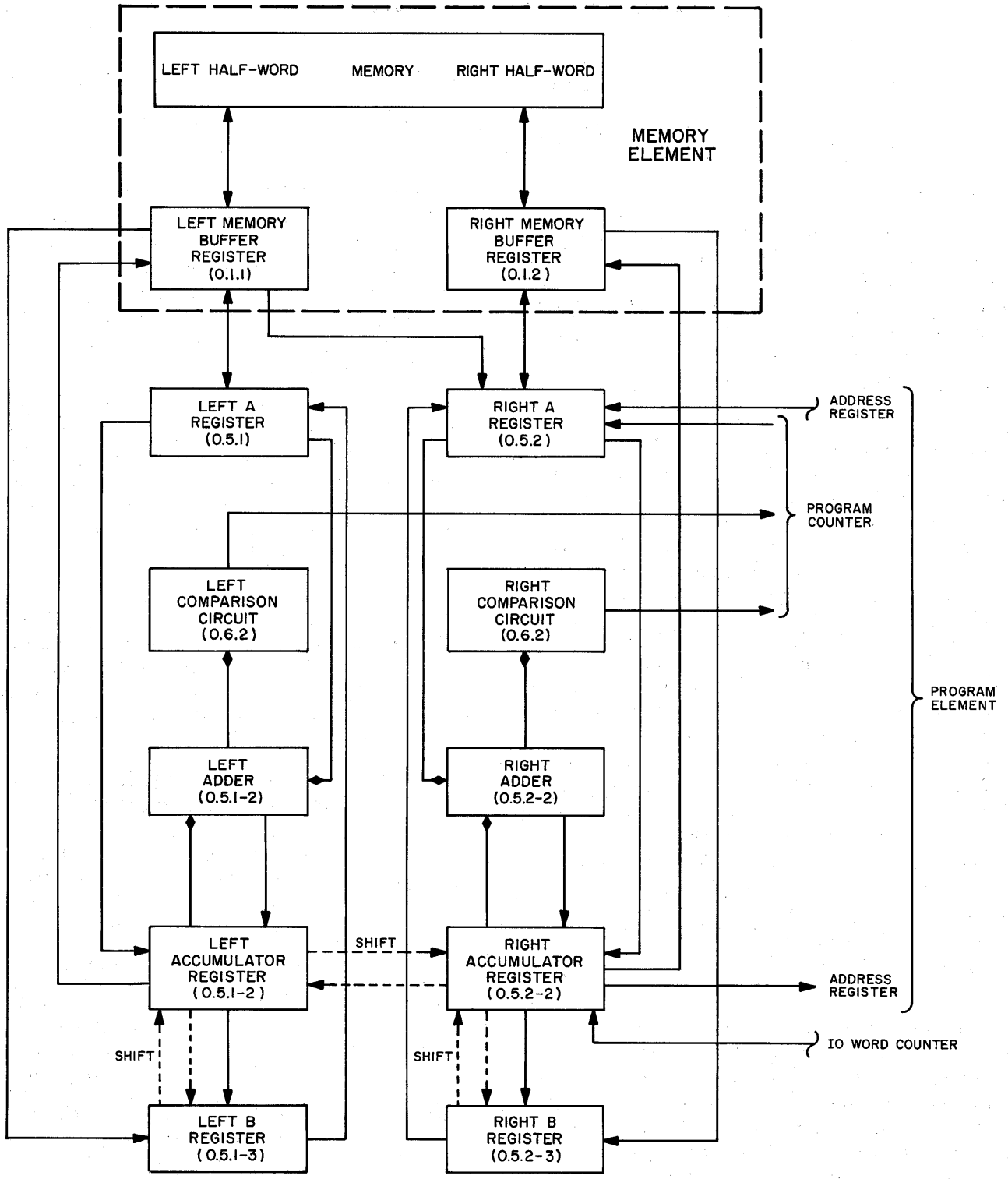


Figure 3–1. Arithmetic Element, Logic Block Diagram

position. During such an operation, the contents of the B register sign bit are temporarily stored in the B register sign storage flip-flop. As a result of the correctional shift left, the contents of these two bits are reset to their original status.

Subtraction is accomplished by complementing the subtrahend, which is held in the A register, and then executing the add operation.

Multiplication is accomplished by generating and adding partial products. Each successive partial product is generated by investigating the least significant bit of the multiplier (in the B register) to determine whether an add operation is required. If the bit is a 1, the multiplicand (in the A register) is added to the contents of the accumulator register; if the bit is a 0, no addition is necessary because the partial product is 0. In either case, the combined accumulator register and B register contents are shifted one bit position to the right (by the inherent shift right and/or a direct command) before the next partial product is generated. Shifting the B register to the right destroys the multiplier bit just used and replaces it with the next most significant multiplier bit; shifting the accumulator register to the right properly aligns the bit positions of that register with the bit positions of the next partial product to be added.

Division consists of a series of subtract operations. The divisor (in the A register) is subtracted first from the dividend (in the combined accumulator register and B register) and then, in subsequent steps, from each successive remainder. If the subtraction results in a positive remainder, a quotient of 1 is recorded; if the subtraction results in a negative remainder, a quotient of 0 is recorded. At the completion of the operation, the B register contains the quotient and the accumulator register contains the final remainder.

Missing From Original Document

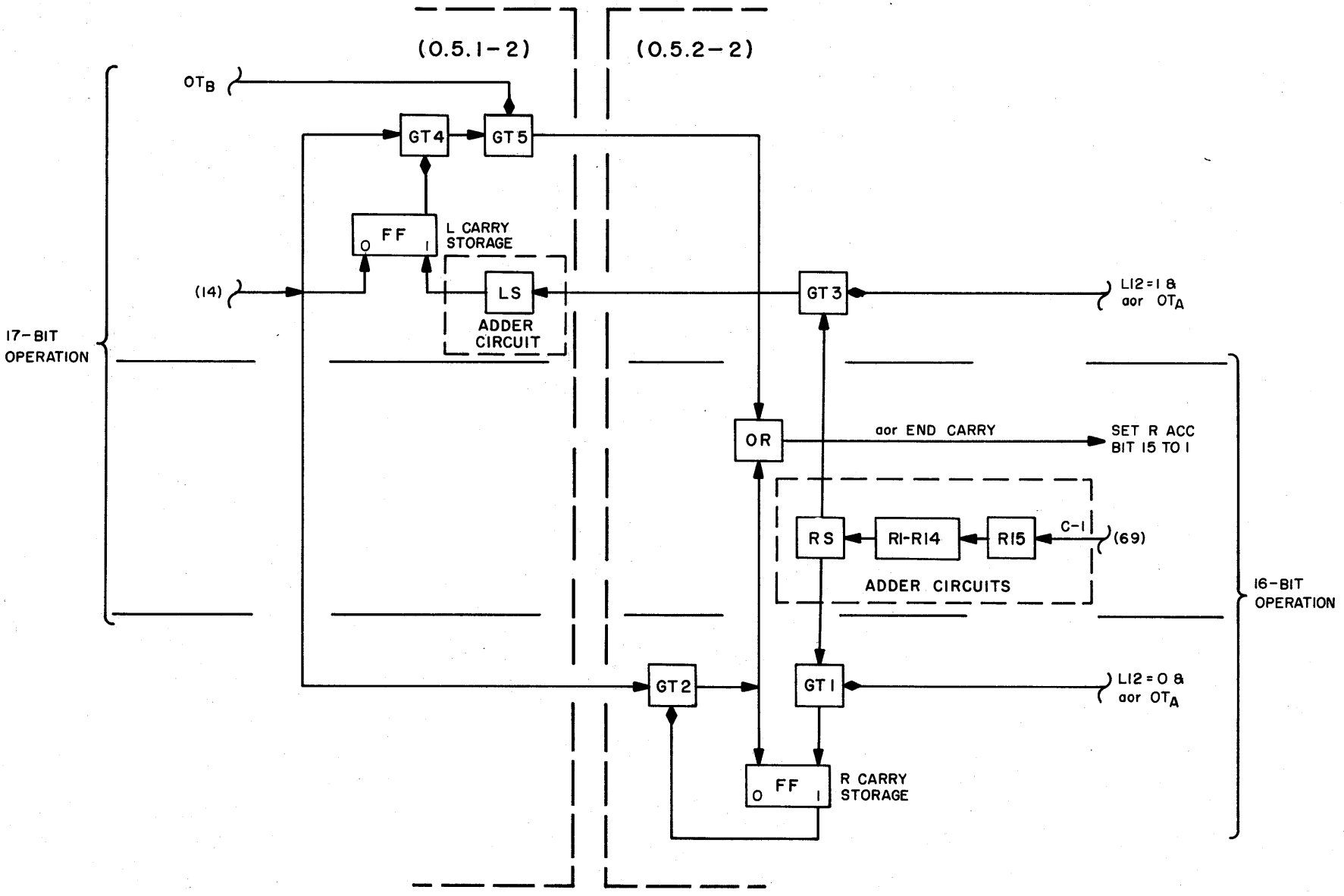


Figure 3–5. End-Carry Circuits for 16- and 17-Bit AOR Operation

Seventeen-bit *AOR* (L12 = 1) operation is also initiated by command 69; however, the carry 1 from the bit RS adder is connected through GT 3 to the bit LS adder. If an end-carry condition occurs, the carry 1 pulse from the bit LS adder will set the left carry storage flip-flop. After the correctional shift left has been accomplished, command 14 is generated to sense for a left end carry (a right end carry cannot occur). If a left end carry did occur, then the command 14 pulse will pass through GT 4. Since command 14 is only generated during an OTB cycle, GT 5 will be conditioned to pass the output of GT 4 to set accumulator bit R15 to a 1 and to clear the left carry storage flip-flop for the next operation.

2.2.1.4 Overflow

Since the computer is limited to operation with numbers less than unity, an overflow condition will occur when the sum or difference of two numbers is equal to or greater than unity. This condition is detected by the overflow circuit and indicates an error in the execution of a program.

An overflow condition can occur only if the carry pulse from the bit 1 adder circuit is unlike both the sign bit of the accumulator register and the sign bit of the A register. These three factors are combined in the sign bit adder (fig. 3–6) by using two AND circuits and two gate circuits to determine which, if either, of

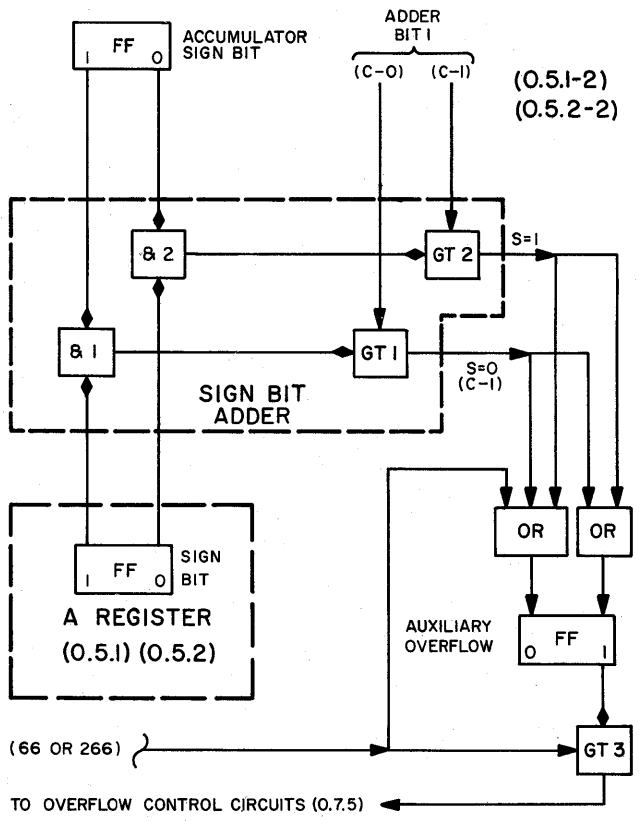


Figure 3–6. Overflow Circuit

the sign bit adder outputs (sum lines) will complement the auxiliary overflow flip-flop. This flip-flop is always in the cleared state before this operation begins so that the first complement pulse (from either GT 1 or GT 2) sets it to the 1 side. If an end-carry pulse is not produced by the sign bit adder, then an overflow condition does exist and the auxiliary overflow flip-flop will gate command 66 or 266 *(record overflow)* through GT 3 to set the overflow alarm flip-flop and sense the status of the alarm switches (Part 8).

The overflow process is illustrated numerically by the following example, in which $+\frac{5}{8}$ is added to $+\frac{3}{8}$:

+3/8	0.011	Augend, accumulator register
+5/8	0.101	Addend, A register
+8/8	1.000	Sum (meaningless because of overflow)

If the auxiliary overflow flip-flop is set by the sum = 0 pulse from GT 1, an end-carry condition will be generated. Under these conditions, the end-carry operation must be performed before the auxiliary overflow flip-flop is sensed by command 66 or 266. If the end-carry operation causes the auxiliary overflow flip-flop to be complemented a second time (by the output from GT 2), the flip-flop will be in the 0 status when the *record overflow* command is received, and no overflow will be recorded. Thus, if two negative numbers are added, an end-carry condition will exist and the auxiliary overflow flip-flop may be set. If the auxiliary flip-flop is set by the initial add operation and subsequently cleared by the end-carry operation, then a false overflow condition was initially indicated. The sequence that produces a false overflow indication which is subsequently corrected is illustrated numerically by the following example:

−3/8 =	1.100	Augend, accumulator register
−4/8 =	1.011	Addend, A register
	(1) 0.111	Uncorrected sum (end carry and auxiliary flip-flops are set)
	0.000	New addend, A register
	0.001	End carry (carry =1 to bit 15 adder)
−7/8 =	1.000	Corrected sum (clear auxiliary overflow flip-flop)

2.2.1.5 Comparison Process

The comparison process is not an arithmetic operation, but it utilizes the bits-alike outputs from the adder circuits (fig. 3–7) to indicate the bit-by-bit comparison of words in the accumulator and A registers. The whole of each adder circuit is not used because, in the comparison process, it does not function as an adder. The por-

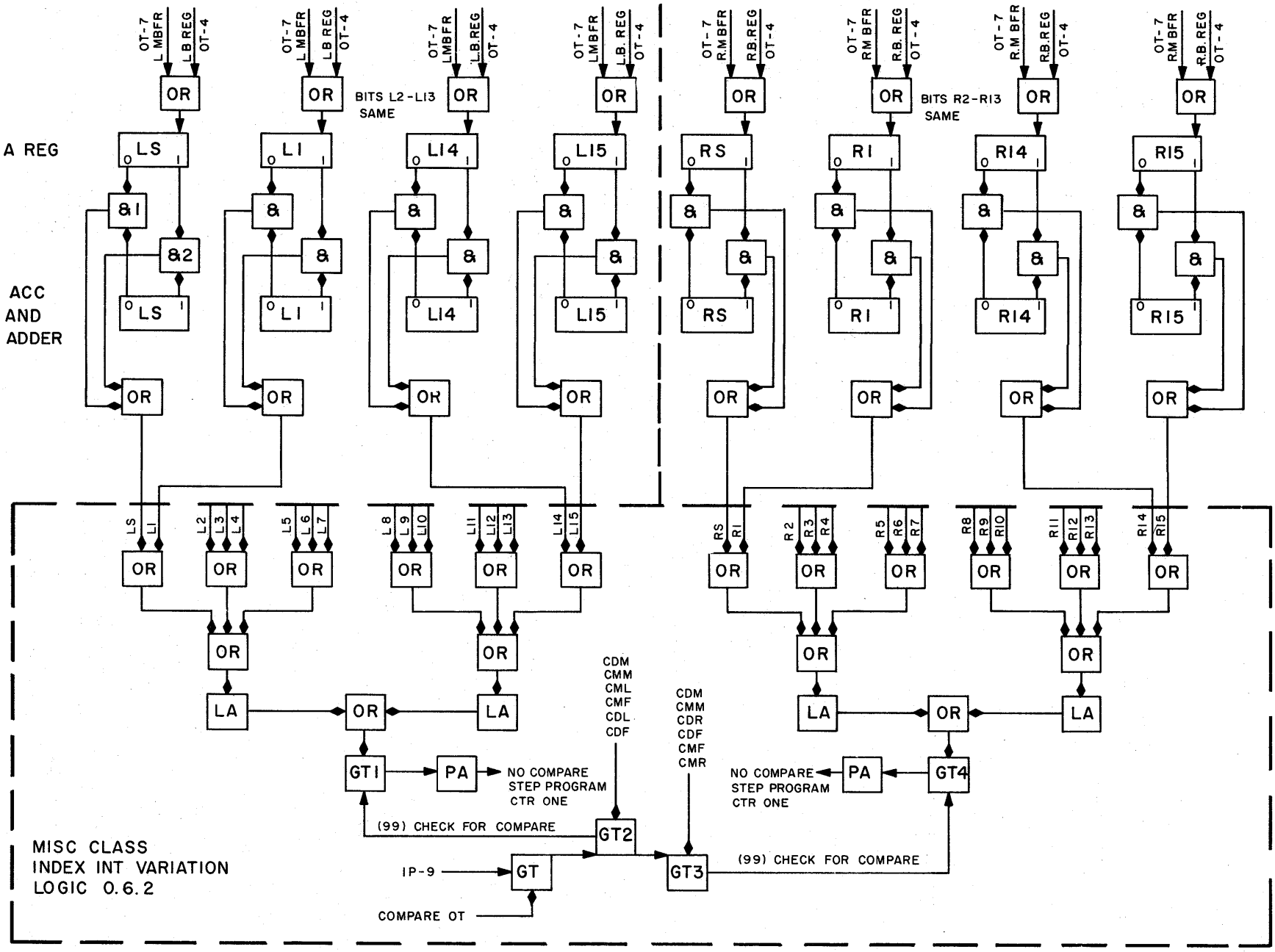


Figure 3–7. Compare Circuits

tion of each adder circuit that is used includes two AND circuits, which combine the corresponding 1 and 0 outputs from the accumulator and A registers, and an OR circuit, which joins the outputs from the two AND circuits.

The bits-alike output lines from the 16 bits of an adder are joined in a series of OR circuits so that a single line carries the outputs from several or from any one of the 16 bits to a gate. The comparison status is always present at this gate and, when a compare instruction is executed, this gate is sensed by a *check for compare* command from the instruction control element. This command is issued separately to the left and right compare circuits so that, if desired, the comparison may be limited to the left or to the right half-words. If either or both gates are conditioned when the *check for compare* command is issued, the resulting output steps the program counter.

This circuit provides a no-compare pulse when any of the corresponding bits of the accumulator and A registers successfully compare. Since a no-compare pulse is desired only when one or more bits of these registers do not compare, a negative approach is taken where the accumulator register is complemented before the comparison check is made. Thus, if the contents of these two registers do compare before the comparison is made, complementing the accumulator register will cause these two registers to be exactly opposite each other. As a result, the compare gate will be deconditioned when sensed by the *check for compare* command and a no-compare pulse will not be issued.

2.2.2 Arithmetic Operations

The addition process is the fundamental operation in the arithmetic element. The remaining basic arithmetic operations (subtraction, multiplication, and division) are made possible by variations of this process.

2.2.2.1 Addition and Subtraction

The addition process is initiated by applying a carry 0 pulse to the left and right bit 15 adder circuits. This pulse initiates a ripple addition (2.2.1.1 and fig. 3–3) wherein the carry pulse from each sequential bit adder is applied to the next more significant bit, and the sum pulse is used to set the next less significant bit of the accumulator. When the rippling process reaches the sign bits, the possibility of end carry and overflow is taken into account to set the end-carry and auxiliary overflow flip-flops. After the addition has been completed and all flip-flops have settled, the accumulator register is shifted one bit position to the left to properly align the sum. If an end carry has occurred, the carry 1 line of the bit 15 adder circuit is pulsed and a second addition operation is performed. If an overflow condition has occurred, the overflow control in the arithmetic element is sensed to generate an alarm pulse. The command sequence for addition is given in table 3–2.

In the subtraction process, the subtrahend (which is transferred into the A register) is first complemented before the addition process is initiated. This step is the only difference between addition and subtraction, and the subtraction process is completed by the sequence of commands given in table 3–2.

When 16-bit arithmetic operation is specified, the left and right arithmetic circuits function separately to simultaneously process two half-words. When 16-bit arithmetic addition or subtraction is completed, the individual sums or differences are contained in bits S through 15 of the respective accumulators. If 17-bit arithmetic operation is specified, the bit LS adder is connected to the bit RS adder and only the right adder performs a useful function. When 17-bit operation is specified, the desired results are contained in bit S of the left accumulator and bits S through 15 of the right accumulator.

2.2.2.2 Multiplication

Multiplication is accomplished by repeated conditional addition of the multiplicand to the contents of the accumulator, interspersed with a shift of the accumulator and B registers one bit position to the right. The additions in this process are conditioned and initiated by the 1 side output of the B register bit 15 flip-flop (fig. 3–8). The 0 side output from this bit causes the accumulator to ripple-shift to the right (without addition taking place) so that the partial product (in the accumulator) reflects a value consistent with the multiplier bits already used.

TABLE 3–2. COMMAND SEQUENCE FOR ADD PROCESS

COMMAND	FUNCTION
Carry 0	Initiates the add process.
Correctional shift left	Correctly aligns the sum in the accumulator after addition is complete.
Clear A register	Prevents adding the number in the A register to the sum a second time if an end carry is to be performed.
End carry	Senses for a carry 1 from the sign bit and corrects the sum accordingly.
Conditional shift left	Re-aligns the sum in the accumulator register if there has been an end carry. (This command is generated, but not executed, if there has been no end carry.)

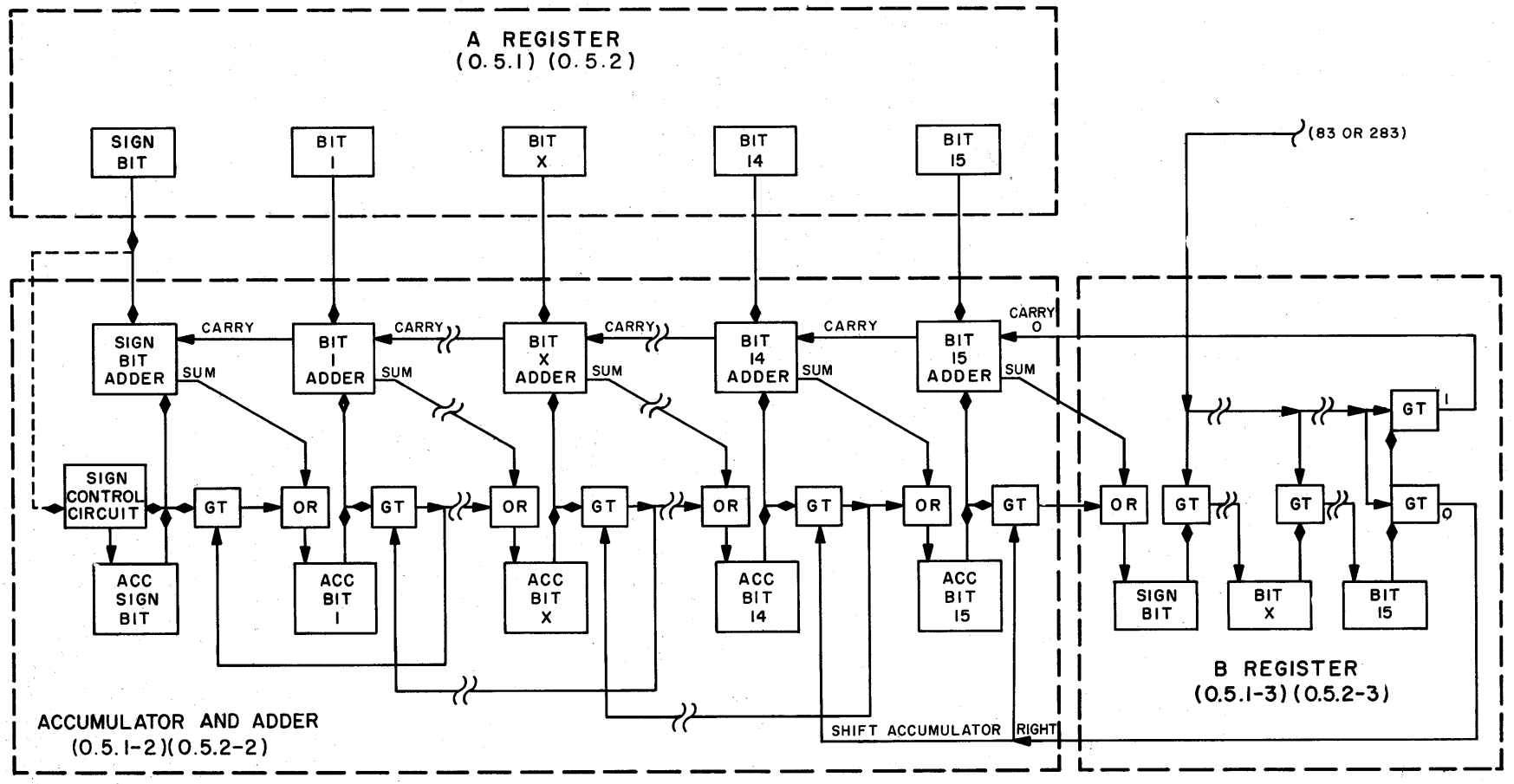


Figure 3—8. Multiplication Process

The B register is shifted one bit position to the right each time the bit 15 position is sensed. Each shift causes the least significant bit in the multiplier to be destroyed, in that the next multiplier bit is advanced to the bit 15 position. This shift also causes the B register sign bit position to be made available for the next more significant bit of the partial product.

At the end of the multiplication process, the product occupies bits S through 15 of the accumulator and bits S through 14 of the B register. If the product is to be utilized in a standard-length register, it is reduced to 15 magnitude bits plus sign by the *SLR (Shift Left and Round)* instruction (3.7.1). For this condition, the latter instruction will not execute any shifts but round off the 15 least significant bits (contained in B register bit S through 14). If the B register sign bit contained a 1, then a 1 is added to the contents of the associated accumulator register.

The multiplication process that occurs in the arithmetic element is shown numerically by the following example:

(1)	8/32 = 0.010	Multiplicand, A register
(2)	X 20/32 = 0.101	Multiplier, B register
(3)	0.000	Initial contents of accumulator
	8/32 = 0.010	1st multiplier is 1: (1) is added to (3)
(4)	4/32 = 0.0010	Shift right during addition halves number to form 1st partial product (4)
(5)	2/32 = 0.00010	2nd multiplier is 0: No addition; shift right halves number to form 2nd partial product (5)
	10/32 = 0.01010	3rd multiplier is 1: (1) is added to (5)
(6)	5/32 = 0.001010	Shift right during addition halves number to form final product (6)

Multiplication is performed only on positive numbers. If either or both of the numbers to be multiplied are negative, they must be complemented before the multiplication operation can be initiated. The complement operation is performed by the first two commands shown in table 3–3. These two commands sense the status of the sign bit of the associated register. If the sign bit contains a 1, the register and the sign control flip-flop in the sign control circuit (fig. 3–7) are both complemented. If both operands are initially positive, the sign control flip-flop remains in its cleared or 0 state. If one of the operands is negative, the sign control flip-flop is complemented once (to the 1 state); if both operands are negative, the sign control flip-flop is complemented twice (to the 0 state). At the end of the multiply operation, the status of the sign control flip-flop is sensed to determine whether the final product should remain in the positive form or whether it should be complemented to the negative form. If the sign control flip-flop contains a 1, the final product is negative, and the sign control flip-flop and the combined accumulator and B register are complemented.

2.2.2.3 Division

The complexity of the divide process is evidenced by the sequence of commands given in table 3–4. This process is the longest of the basic arithmetic operations, the generation of each partial quotient requiring 2.5 μsec. The partial quotient commands accomplish the division of one number by another by directing a series of interspersed subtractions and shifts of the dividend to the left. A simplified logic diagram of this process is shown in figure 3–9.

The first five commands in table 3–4 are preparatory. Note that both operands are made positive and that the sign control flip-flop is complemented each time one of the registers is complemented by command 16, 20, 22, or 229 (fig. 3–10). At the end of the divide operation, the status of the sign control flip-flop is sensed by command 9 or 209. If this flip-flop is set, the command pulse is passed to complement the sign control flip-flop and the associated accumulator and B register. For all divide instructions, the divisor must be larger than the dividend so that the quotient will be less than unity.

TABLE 3–3. COMMAND SEQUENCE FOR MULTIPLY PROCESS

COMMAND	FUNCTION
Make A register positive	Makes the multiplicand positive.
Make accumulator register positive	Makes the multiplier positive.
Clear B register	Prepares the B register for the transfer of the multiplier.
Accumulator register to B register	Places the multiplier in the B register.
Partial product	Performs a partial product operation which is repeated 14 times (that is, a total of 15 times).
Correct sign	Affixes the proper sign to the final product.

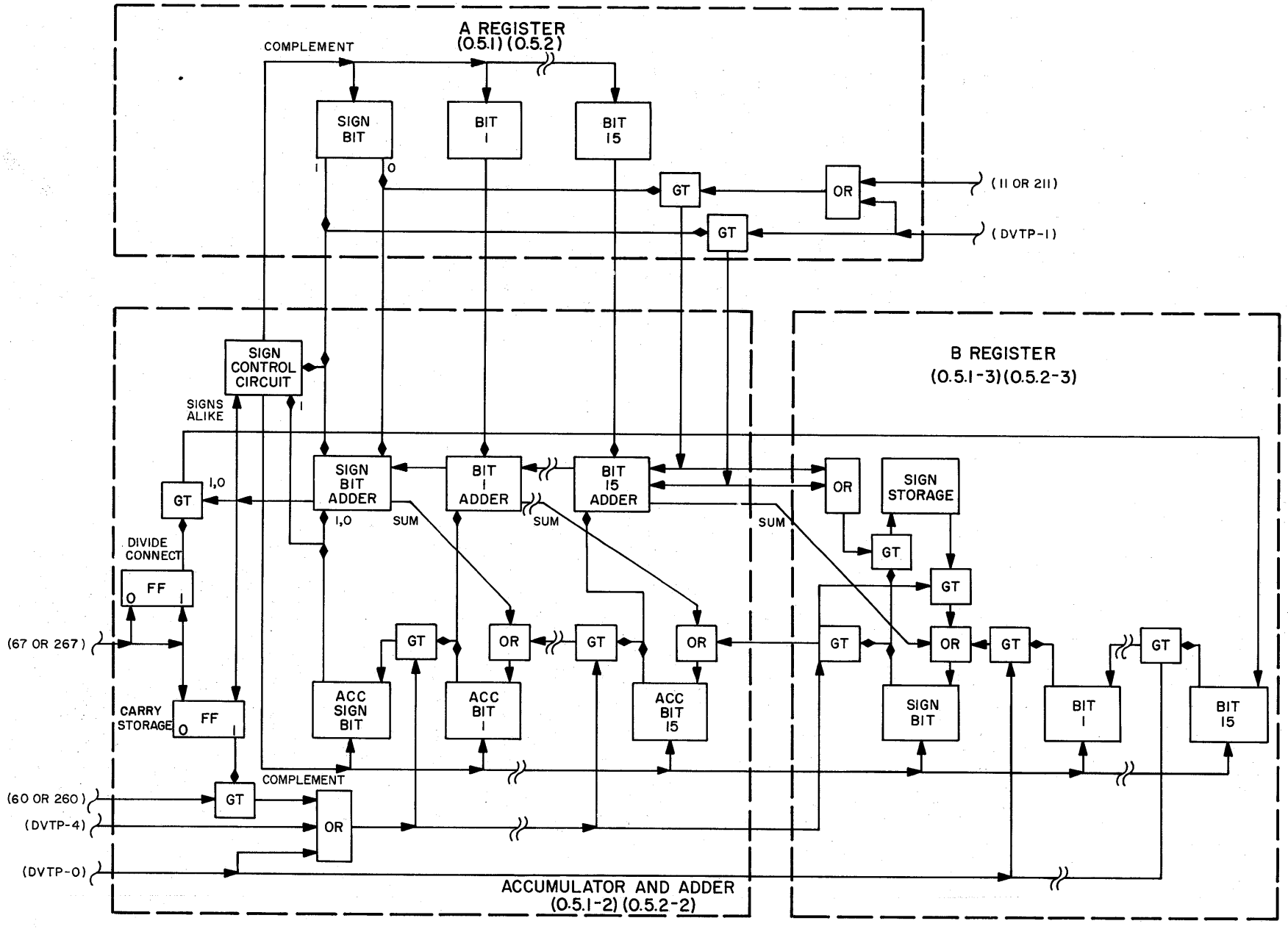


Figure 3–9. Division Process

TABLE 3–4. COMMAND SEQUENCE FOR DIVIDE PROCESS

COMMAND	NAME AND FUNCTION
21, 230	*Clear A registers:* Prepares the A register for the divisor.
67, 267	*Complement divide-connect flip-flops:* Connects the output of the sign bit adder to bit 15 of the B register.
16, 20	*Make accumulator and B registers positive:* Complements the accumulator, B register, and sign control flip-flop if the accumulator sign bit is 1.
43, 51	*Transfer memory buffer register to A registers:* Places divisor in A register.
22, 229	*Make A registers positive:* Complements the A register and sign control flip-flop if the A register sign bit is 1.
80 (repeated 80 times)	*Step divide time pulse distributor* (DTPD): Generates divide time pulses (DVTP's). The DTPD is cycled 16 times to produce 16 partial quotients (16 quotient bits).
	DVTP 0: Complements the A register if the signs of the A and the accumulator registers are alike. Shifts the combined accumulator-B register one bit position to the left to eliminate the first digit and double the value of the remaining digits in the dividend, and to make bit 15 of the B register available for the next partial quotient.
	DVTP 1: Transfers the contents of the B register sign bit into the B register sign storage flip-flop (for temporary storage), and senses the A register sign bit. If the latter bit contains a 0, pulses the carry 0 line of the bit 15 adder; if the bit contains a 1, pulses the carry 1 line of the bit 15 adder. If the remainder of this subtraction is positive, sets B register bit 15 to a 1; if the remainder is negative, sets B register bit 15 to a 0.
	DVTP 3: Subtracts 1 from the step counter.
	DVTP 4: Transfers the contents of the B register sign storage flip-flop into the B register sign bit, and shifts the accumulator register and B register sign bit one bit position to the left to correct for the inherent shift right that occurred during the add process. (The partial quotient command is repeated 15 more times.)
61, 261	*Make A register and accumulator register signs unlike:* If the signs of the operand are alike, complements the A register.
67, 267	*Complement the divide-connect flip-flop:* Disconnects the sign bit adder from B register bit 15 to permit a normal addition (if required) for correction of the remainder. Also clears the carry storage flip-flop.
11, 211	*Correct remainder:* If the A register sign bit is 0, pulses the carry 0 line of the bit 15 adder and transfers the B register sign bit into the B register sign storage flip-flop.
60, 260	*Accumulator conditional shift left:* If a carry 0 was generated by command 11 or 61 to correct the final remainder, then command 60 or 260 is generated to shift the accumulator register and the B register sign bit one bit position to the left, and to transfer the contents of the B register sign storage flip-flop to the B register sign bit.
9, 209	*Correct sign:* If the sign control flip-flop was complemented only once by command 16, 20, 22, or 229, then the accumulator and B register are complemented. The final sign, in the accumulator sign bit position, is the algebraic product of the original signs of the operands.

The divide process used in the arithmetic element consists of performing a trial subtraction and then determining the sign of the remainder. If the sign is positive, then the trial subtraction was successful and the quotient bit is set to 1. However, if the sign is negative, the subtraction was not successful and the quotient bit is set to 0. An unsuccessful subtraction (resulting in a negative remainder) must be nullified before the process can be continued. This could be accomplished by adding the divisor to the negative remainder to restore the dividend before another trial subtraction is performed; however, the correction process would require additional time. To speed up the operation, a different method (a nonrestoring process) is used in which the complement of the divisor is subtracted from the current remainder. In this method, a negative remainder is not corrected before the operation is continued. Instead, the current remainder is doubled and the positive form of the divisor is then used. This process is repeated until the 16 quotient bits are obtained. The details of the nonrestoring process are presented in the example given in table 3–5.

The sequence of DVTP pulses in table 3–5 is repeated only often enough to illustrate the manner in which quotient bits of 1 and 0 are generated. Although the DVTP 2 and DVTP 3 pulses occur during the DVTP sequence, they are omitted from this table because they affect only the divide time pulse distributor and the step counter in the instruction control element. The x's in the accumulator and B register columns of table 3–5 indicate bit positions made ready to receive new information.

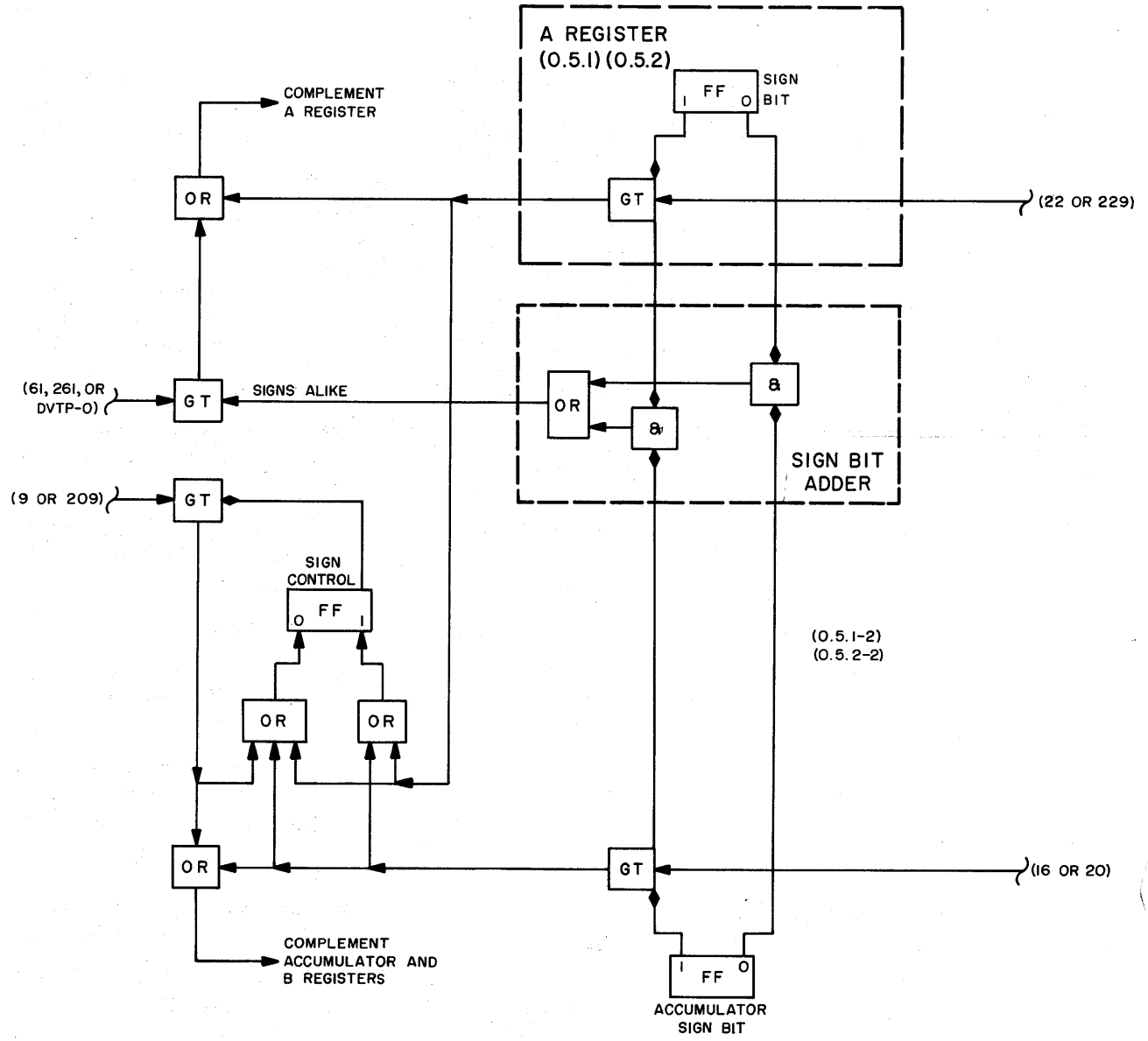


Figure 3–10. Sign Control for Division Process

Note that the last addition (step 15 in table 3–5) differs from the other adder sign bit carries in that it is not transferred to the B register bit 15 position. However, it does set the carry storage flip-flop, the output of which is gated by the next command (60 or 260) for a conditional shift left.

A division instruction is usually followed by an *SLR* instruction (3.7.1) to shift the quotient to the accumulator register and round off the least significant bit, which is in bit S of the B register after the shift operation. The remainder is lost when the *SLR* instruction is executed.

TABLE 3–5. APPLICATION OF *DVD* COMMANDS

COMMAND	A REGISTER	ACCUMULATOR	B REGISTER	REMARKS
21, 230; 67, 267; 16, 20; 43, 51; 22, 229	0.110		0.100 0.000	1. Preparation for divide process; divisor in A register, dividend in accumulator.
	DVTP 0	1.001	1.000 0.00x	2. Complement A register and shift accumulator B register one bit to the left.
	1		(1) x.001 0.001	3. Add 2's complement of divisor to dividend. Since the remainder is positive, set B register bit 15 to 1.
	4		0.010 0.001	4. Correctional shift left.
	DVTP 0	1.001	0.100 0.01x	5. Shift accumulator B register one bit to the left.
	1		(0) x.111 0.010	6. Add 2's complement of divisor to dividend. Since the remainder is negative, set B register bit 15 to 0.
	4		1.110 0.010	7. Correctional shift left.
	DVTP 0	0.110	1.100 0.10x	8. Complement A register and shift accumulator B register one bit to the left.
	1		(1) x.001 0.101	9. Add divisor to dividend. Since the remainder is positive, set B register bit 15 to 1.
	DVTP 4		0.010 0.101	10. Correctional shift left.
	DVTP 0	1.001	0.100 1.01x	11. Complement A register, and shift accumulator B register one bit to the left.
1			1 (0) x.111 0.010	12. Add 2's complement of divisor to dividend. Since the remainder is negative, set B register bit 15 to 0.
4			1.110 1.010	13. Correctional shift left.
61, 261; 67, 267;	0.110			14. Complement A register to prepare for correction of remainder.
11, 211			1 (1) x.010 0.010	15. Add divisor to dividend, and set carry storage flip-flop.
60, 260			0.100 1.010	16. Conditional shift left.

TABLE 3–5. APPLICATION OF *DVD* COMMANDS (cont'd)

COMMAND	A REGISTER	ACCUMULATOR	B REGISTER	REMARKS
9, 209			0.100 1.010	17. Correct sign.
	At the end of the division process, the sign of the quotient and remainder is contained in bit S of the accumulator register, the remainder is contained in accumulator register bits 1 through 15, and the quotient is contained in B register bits S through 15. If this example is followed by an *SLR* 3 instruction, the accumulator and B register contents appear as follows:			
			0.101 0.000	18. Quotient correctly positioned; remainder destroyed; quotient bit in B register sign bit position rounded off.

CHAPTER 3
INSTRUCTION ANALYSIS

3.1 GENERAL

The instructions that utilize the arithmetic element for their execution are analyzed in this chapter. There are 45 such instructions, and they are grouped in six classes as follows:

a. Add class: *ADD, SUB, TAD, TSU; CAD, CSU, CAM, CAC; DIM; ADB*
b. Multiply class: *MUL, TMU; DVD, TDV*
c. Shift class: *DCL, FCL; DSL, DSR, LSR, RSR; ASL, ASR*
d. Store class: *FST, LST, RST, STA; ECH; AOR; DEP*
e. Branch class: *BFZ; BFM, BLM, BRM*
f. Miscellaneous class: *SLR; CSW; ETR; LDB; CMF, CML, CMR; CDF, CDL, CDR; CMM, CDM.*

The command pulse sequence generated during the execution of any of these instructions is shown in figures 2–9, 2–10, 2–12, 2–14, 2–15, and 2–16. Reference should be made to the applicable figure whenever necessary.

3.2 ADD CLASS

All instructions in the add class are indexable, but indexing has no meaning for the *ADB* instruction and must not be used with the *CAC* instruction. Each instruction in this class requires a PT cycle and an OT cycle. The instruction cycle begins at PT 7, at which time command 42 (logic 0.1.1, D9) transfers the contents of the left MBR to the operation and index interval registers, command 52 (logic 0.1.2, D8) transfers the contents of the right MBR to the address register and the step counter, and command 92 (logic 0.4.1, B1) steps the program counter. At PT 8, the contents of the selected index register, if one is selected, are added to the contents of the address register. Command 131 (logic 0.3.1, B3) occurs at PT 11 to set the PT-OT flip-flop to OT. The operations that occur during the OT cycle and the remainder of the PT cycle vary according to the instruction being executed and are discussed in the following paragraphs.

3.2.1 *Add (ADD), Subtract (SUB), Twin and Add (TAD), Twin and Subtract (TSU)*

The *Add* and *Subtract* instructions differ in that the subtract process requires command 26 (logic 0.5.1, B5), which complements the contents of the A register just prior to the execution of the standard add process used in the arithmetic element. The twin and non-twin instructions differ in the manner of information transfer between the MBR's and the A registers. In a non-twin instruction, the left MBR supplies data to the left A register (command 43, logic 0.1.1, D9) and the right MBR supplies data to the right A register (command 51, logic 0.1.2, D8). In a twin instruction, command 51 is replaced by command 44 (logic 0.1.1, D8), which transfers the left MBR to the right A register, with the result that the same data is supplied to both the left and the right A registers. The contents of the right MBR are not used during the execution of twin instructions.

The following sequence is applicable to the four add-class instructions in this group. At OT 0, command 31 (logic 0.1.4, B5) clears MAR 2 (MAR 1 is cleared by the memory 1 control circuits during each memory cycle). At OT 0 delayed, commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's, and command 71 (logic 0.4.1, C11) transfers the contents of the address register to MAR 1, MAR 2, and TMAR and starts the selected memory device. At OT 1, commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6) clear the A registers. If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at OT 5. If either core memory was selected, the contents of the selected core memory register are transferred at OT 6.

At OT 6, the TMAR is cleared by command 31. At OT 7, commands 43 and 51 transfer the data in the MBR's to the A registers for non-twin instructions, or commands 43 and 44 transfer the data in the left MBR to both A registers for twin-type instructions. If a subtract instruction is in progress, the A registers are complemented at OT 9 by commands 26 and 226 (logic 0.5.2, B5).

Following these preliminary operations, the addition process is initiated at OT 10 by commands 64 (logic 0.5.1-2, A1) and 264 (logic 0.5.2-2, A1), each of which simultaneously transmits a carry 0 pulse to the associated bit 15 adder circuit and pulses two gates associated with the B register sign-bit flip-flop. The latter operation causes the contents of the associated B register sign-bit flip-flop to be transferred to the B register sign-storage flip-flop, thus preserving that information while the sum of the lowest-order bits in the associated accumulator and A registers is temporarily stored in the B register sign-bit flip-flop.

The addition process causes the sum to appear in the accumulator register, and in the sign bit of the B

register, shifted one bit position to the right. If the process results in a carry 1 from the sign-bit adder circuit, the end-carry pulse sets the carry-storage flip-flop. The output of the carry-storage flip-flop in turn conditions two gates in preparation for the *end-carry* and *conditional shift left* commands that occur later.

Command 161 (logic 0.3.1, B4), issued at OT 11, clears the PT-OT flip-flop to terminate the OT cycle and to start the next PT cycle. At PT 0, command 31 (logic 0.1.4, B5) clears MAR 2. At PT 0 delayed, command 91 (logic 0.4.1, A4) transfers the contents of the program counter to MAR 1, MAR 2, and TMAR and starts the selected memory device, and commands 41 and 53 clear the MBR's. At PT 1, commands 21 and 230 clear the A registers, and commands 2, 4, 81, 89, 202, 204, 281, and 289 correct the inherent shift right that occurred during the addition process. Commands 2 (logic 0.5.1-2, E21) and 202 (logic 0.5.2-2, D21) initiate broadside shifts which transfer the information in respective accumulator bits 2 through 15 one bit position to the left, commands 4 (logic 0.5.1-2, E21) and 204 (logic 0.5.2-2, D21) shift the contents of bit 1 of each accumulator into the corresponding sign bit position, commands 81 (logic 0.5.1-3, E6) and 281 (logic 0.5.2-3, E6) transfer the contents of the respective B register sign bits into bit 15 of the corresponding accumulator register, and commands 89 (logic 0.5.1-3, B7) and 289 (logic 0.5.2-3, B7) return the stored sign-bit information to the B register sign-bit positions.

After the execution of these commands, *end-carry* commands 63 and 263 (logic 0.5.1-2, A17) inspect the respective carry-storage flip-flops at PT 2. If a carry-storage flip-flop has been set by a carry 1 from the sign-bit adder circuit, the *end-carry* command is gated by the output of the carry-storage flip-flop to pulse the carry 1 line of the bit 15 adder circuit. If this command is executed, the sum in the accumulator register is increased by 1 and the sum bits undergo a second inherent shift right. If command 63 or 263 resulted in a shift right, the shift is corrected at PT 5 by command 60 or 260 (logic 0.5.1-2, A18), which senses a second gate conditioned by the 1 side of the respective carry-storage flip-flop. The pulse provided by this gate performs the shift-left functions previously performed by commands 2, 4, 81, and 89, or by commands 202, 204, 281, and 289.

If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at PT 5. If either memory was selected, the contents of the selected register are transferred at PT 6. At the same time, command 31 clears the TMAR, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and the step counter, command 101 (logic 0.3.1, B13) clears the index interval and operations registers, and commands 66 and 266 (logic 0.5.1-2, B22) record overflow conditions if they are present.

Dual 16-bit operation is performed when bit L12 of the instruction word contains a 0. If 17-bit operation is required, bit L12 of the instruction word is made to contain a 1. Under the latter condition, the carry lines from the bit RS adder are disconnected from the right carry storage and right overflow control circuits and connected to the bit LS adder. At the same time, the carry lines from the bit L1 adder are disconnected from the bit LS adder. Either 16- or 17-bit operation may be specified for any of these instructions; however, 17-bit operation is meaningful only for the *ADD* and *SUB* instructions.

3.2.2 *Clear and Add (CAD), Clear and Subtract (CSU), Clear and Add Magnitudes (CAM), Clear and Add Clock (CAC)*

The *CAD, CSU,* and *CAM* instructions involve the transfer of data from memory, and the *CAC* instruction involves the transfer of data from the clock register. The *CAD* and *CAC* instructions transfer the subject information, without modification, to the accumulator register. The *CSU* instruction transfers the complement of the specified memory register content to the accumulator register. The *CAM* instruction transfers the positive value of the specified memory register content to the accumulator register.

Preliminary operations determine whether the *CAC* or one of the other three instructions has been programmed. If the operations register specifies *CAC* and a test memory address is selected, then the contents of the clock register will be transferred to the right MBR and the *CAC* instruction will be executed. If the operations register specifies one of the other instructions, then the address register will specify a memory location, and the contents of that location will be transferred to the MBR's for processing.

The commands generated during the PT cycle of any of these four instructions are the same as those described in 3.2. The commands generated during OT 0 through OT 9 of the OT cycle differ according to the instruction. At OT 0 of the *CAC* instruction, command 31 (logic 0.4.1, B5) clears MAR 2. At OT 0 delayed, command 71 (logic 0.4.1, C11) sets the select clock register flip-flop and transfers the contents of the address register to MAR 1, MAR 2, and TMAR (although the specified test memory transfer is not performed). At OT 0 delayed 0.5 μsec, commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's. At OT 1, commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6) clear the A registers. At OT 3, command A6 (logic 0.2.6, A5) transfers the contents of the clock register to the right MBR.

Although test memory was selected, the contents of test memory are not transferred to the MBR's at OT 5. At OT 6, the accumulator registers are cleared by com-

mands 10 (logic 0.5.1-2, B21) and 210 (logic 0.5.2-2, B21), and the TMAR is cleared by command 31. At OT 7, commands 43 (logic 0.1.1, D9) and 51 (logic 0.1.2, D8) transfer the data in the MBR's to the A registers in preparation for the addition process that follows.

For the *CAD, CSU,* and *CAM* instructions, the same commands are executed during OT 0 through OT 7 of the OT cycle. At OT 0, command 31 clears MAR 2. At OT 0 delayed, command 71 transfers the contents of the address register to MAR 1, MAR 2, and TMAR and starts the selected memory device, and commands 41 and 53 clear the MBR's. At OT 1, commands 21 and 230 clear the A registers. If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at OT 5. At OT 6, the accumulator registers are cleared by commands 10 and 210, the TMAR is cleared by command 31, and, if core memory was selected, the contents of the selected core memory register are transferred to the MBR's. At OT 7, commands 43 and 51 transfer the data in the MBR's to the A registers in preparation for the addition process that follows.

If the *CAM* instruction is being executed, commands 22 and 229 (logic 0.5.1, D5) are issued at OT 8 to make the A registers positive. If the *CSU* instruction is being executed, commands 26 and 226 (logic 0.5.1, B5) are generated at OT 9 to complement the A registers.

The remaining commands (beginning at OT 10) for the *CAD, CSU, CAM,* and *CAC* instructions are the same as those described for the *ADD, TAD, SUB,* and *TSU* instructions. However, although commands 63, 263, 60, 260, 66, and 266 are generated, they do not execute any action because end carry and overflow cannot occur in the *CAD, CSU, CAM,* or *CAC* instruction.

3.2.3 *Difference Magnitudes (DIM)*

The *DIM* instruction determines the difference between the absolute values of two quantities. One of the quantities involved is contained in the accumulator register before the *DIM* instruction is executed. The other quantity is transferred from the selected memory register to the A registers at the beginning of the instruction. Subsequent operations transfer the initial contents of the accumulator register to the B register, and then record the difference quantity in the vacated accumulator register.

The commands generated during the PT cycle perform the same function as that described in 3.2. At OT 0 of the OT cycle, command 31 (logic 0.1.4, B5) clears MAR 2. At OT 0 delayed, command 71 (logic 0.4.1, C11) transfers the contents of the address register to MAR 1, MAR 2, and TMAR and starts the selected memory device, and commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's. At OT 1, commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6) clear the A registers. If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at OT 5. At OT 6, the B registers are cleared by commands 84 (logic 0.5.1-3, A7) and 284 (logic 0.5.2-3, A7), the TMAR is cleared by command 31, and, if core memory was selected, the contents of the specified core memory register are transferred to the MBR's. At OT 7, commands 43 (logic 0.1.1, D9) and 51 (logic 0.1.2, D8) transfer the data in the MBR's to the A registers.

At OT 8, the A registers are made positive by commands 22 and 229 (logic 0.5.1, D5), and the contents of the accumulator registers are transferred into the corresponding B registers by commands 1 and 201 (logic 0.5.1-2, E19). At OT 9, the accumulator registers are made positive by commands 13 (logic 0.5.1-2, B21) and 213 (logic 0.5.2-2, A22) and the A registers are complemented by commands 26 and 226 (logic 0.5.1, B5).

The difference quantity is now obtained by the addition process, which is initiated at OT 10 by commands 64 (logic 0.5.1-2, B22) and 264 (logic 0.5.2-2, A1). The execution of the remaining commands is the same as that described for the corresponding portion of the instructions in 3.2.1. However, commands 66 and 266 (logic 0.5.1-2, B22) are not effective because overflow cannot occur.

3.2.4 *Add B Registers to Accumulator Registers (ADB)*

The *ADB* instruction differs from the other add instructions in the method used for placing an operand in the A register. In the *ADB* instruction, one of the operands is transferred directly from the B register to the A register; in the other instructions, the operand is transferred from the memory element to the A registers through the MBR. Except for this difference in preparatory operations, the *ADB* instruction is identical with the *ADD* instruction.

The execution of commands during the PT cycle is the same as that described in 3.2. At OT 0 time of the OT cycle, command 31 (logic 0.1.4, B5) clears MAR 2. At OT 0 delayed, command 71 (logic 0.4.1, C11) transfers the contents of the address register to MAR 1, MAR 2, and TMAR and starts the specified memory device, and commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's. At OT 1, commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6) clear the A registers. If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at OT 5. At OT 6, the TMAR is cleared by command 31, and, if core memory was selected, the contents of the specified core memory register are transferred to the MBR's. At OT 7, commands 43 (logic 0.1.1, D9) and 51 (logic 0.1.2, D8) transfer the data in

the MBR's to the A registers. The data transferred by these commands is meaningless, but the transfer is not inhibited.

At OT 8, the A registers are again cleared by commands 21 and 230. At OT 9, the A registers are loaded from the corresponding B registers by commands 88 (logic 0.5.1-3, D6) and 288 (logic 0.5.2-3, C7).

The remaining commands in the *ADB* instruction, beginning with commands 64 (logic 0.5.1-2, A1) and 264 (logic 0.5.2-2, A1) at OT 10, are the same as those described for the *ADD* instruction in 3.2.1. The 17-bit option described for the *ADD* instruction is also applicable for the *ADB* instruction.

3.3 MULTIPLY CLASS

Each instruction in this class is indexable and requires a PT cycle, an OT cycle, and an arithmetic pause. The instruction cycle begins at PT 7, at which time command 42 (logic 0.1.1, D9) transfers the contents of the left MBR to the operation and index interval registers, command 52 (logic 0.1.2, D8) transfers the contents of the right MBR to the address register and the step counter, and command 92 (logic 0.4.1, B1) steps the program counter. At PT 8, the contents of the selected index register (if one is selected) are transferred to the address register. At PT 11, command 131 (logic 0.3.1, B3) sets the PT-OT flip-flop to OT. The operations that occur during the subsequent OT cycle, arithmetic pause, and the remainder of the PT cycle vary according to the instruction being accomplished and are discussed in the following paragraphs.

3.3.1 *Multiply (MUL), Twin and Multiply (TMU)*

The only difference between these two instructions is the method of providing multiplicands to the A registers. In the *MUL* instruction, the multiplicands for the left and right A registers are supplied by the respective left and right MBR's. In the *TMU* instruction, the left MBR supplies the same multiplicand to both A registers. The multipliers are contained in the accumulator registers before either instruction is executed.

At OT 0, command 31 (logic 0.1.4, B5) clears MAR 2. At OT 0 delayed, command 71 (logic 0.4.1, C11) transfers the contents of the address register to MAR 1, MAR 2, and TMAR and starts the selected memory device, and commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's. At OT 1, commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6) clear the A registers. (If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at OT 5.) At OT 6, the TMAR is cleared by command 31, and, if core memory was selected, the contents of the specified core memory register are transferred to the MBR's. At OT 7, command 75 (logic 0.5.3, B3) sets the step counter to 15, commands 43 (logic 0.1.1, D9) and 51 (logic 0.1.2, D8) transfer the data in the MBR's to the A registers for the *MUL* instruction (or commands 43 and 44 (logic 0.1.1, D8) transfer the data in the left MBR to both A registers for the *TMU* instruction), commands 84 (logic 0.5.1-3, A7) and 284 (logic 0.5.2-3, A7) clear the B registers, and commands 13 (logic 0.5.1-2, B21) and 213 (logic 0.5.2-2, A22) make the accumulator registers positive. Commands 13 and 213 also complement the associated sign control flip-flops if the sign bit of the corresponding accumulator register contains a 1.

The remaining commands are identical for both instructions. At OT 8, the 2-mc sync flip-flop is set by command 138 (logic 0.5.3, D7), the contents of the accumulator registers are transferred to the B registers by commands 1 and 201 (logic 0.5.1-2, E19), the accumulator registers are cleared by commands 10 (logic 0.5.1-2, B21) and 210 (logic 0.5.2-2, B21), and the A registers are made positive by commands 22 and 229 (logic 0.5.1, D5). Commands 22 and 229 also complement the associated sign control flip-flops if the sign bit of the corresponding accumulator register contains a 1.

The 2-mc operation flip-flop is set at OT 9, and the 2-mc sync flip-flop is cleared at OT 10. Also at OT 10, command 134 (logic 0.2.2, A8) sets the pause flip-flop, command 73 (logic 0.5.3, D2) steps the step counter from 15 to 14, and commands 83 (logic 0.5.1-3, C8) and 283 (logic 0.5.2-3, C7) accomplish the multiplication for the first partial product.

Each execution of command 83 or 283 examines the multiplier bit that is currently held in bit 15 of the associated B register. The multiplicand (in the A register) is added to the contents of the accumulator register if this bit is a 1, and the resulting partial product appears in the accumulator register shifted one bit position to the right. No end carry is involved because the numbers being added are positive. If the multiplier bit is a 0, the subject command only shifts the current partial product one bit position to the right. In either case, command 83 or 283 shifts the associated B register one bit position to the right, which brings the next multiplier bit up for examination by the next partial product command. This results in the loss of one multiplier bit, but the effect of that bit on the final product already has been taken into account.

Command 73 reduces the contents of the step counter by 1 each time it is issued, and the step counter (initially set to $15)_b$ determines the number of times that commands 83 and 283 will be issued. Each occurrence of commands 73, 83, and 283 is followed by the generation of a partial product. The first partial product is generated at OT 10 of the OT cycle, which immediately precedes the arithmetic pause.

The arithmetic pause begins after OT 11, at which time command 161 (logic 0.3.1, B4) clears the PT-OT flip-flop, and the step counter goes from 14 to 13. The

second through the 11th partial products are generated during the arithmetic pause. The clear pause delay sync flip-flop (logic 0.5.3, D6) is set when the step counter goes from 7 to 6, the clear pause delay flip-flop (logic 0.5.3, D6) is set when the step counter goes from 6 to 5, and the pause (logic 0.2.2, B8), the clear pause delay, and the clear pause delay sync flip-flops are cleared when the step counter goes from 5 to 4.

The PT cycle is resumed when the step counter goes from 4 to 3. At PT 0 (when the step counter content is 4), command 31 clears MAR 2. At PT 0 delayed, command 91 (logic 0.4.1, A4) transfers the contents of the program counter to MAR 1, MAR 2, and TMAR and starts the selected memory device, and commands 41 and 53 clear the MBR's. At PT 1, the step counter goes from 3 to 2, and the clear pause delay sync flip-flop is set. At PT 2, the step counter goes from 2 to 1, the clear pause delay flip-flop is set, and action is initiated to clear the 2-mc operation flip-flop. (The clearing operation requires 1 μsec and is not completed until PT 4 time.) At PT 3, the pause, the clear pause delay, and the clear pause delay sync flip-flops are cleared, the step counter goes from 1 to 0, and the last partial product is generated.

If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at PT 5. If a core memory address was selected, the contents of the specified register are transferred to the MBR's at PT 6. At the same time, command 31 clears the TMAR, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and the step counter, and command 101 (logic 0.3.1, B13) clears the index interval and operation registers. If the associated sign control flip-flops are set, commands 9 (logic 0.5.1-2, C21) and 209 (logic 0.5.2-2, C22) complement the respective accumulator–B register combinations and clear the sign control flip-flop.

Each half-word product appears as a 30-bit word plus a sign in the combined accumulator register and B register. The sign of the product is in the sign bit of the accumulator register, the most significant bit of the product is in bit 1 in the accumulator register, and the least significant bit of the product is in bit 14 in the B register.

3.3.2 *Divide (DVD), Twin and Divide (TDV)*

The difference between these instructions is the same as that described for the *MUL* and *TMU* instructions in 3.3.1. In addition, the same commands are generated during the first PT cycle as those discussed for the *MUL* and *TMU* instructions in 3.3.1. Therefore, the following analysis will discuss only the operations performed from OT 0 on.

At OT 0, command 31 (logic 0.1.4, B5) clears MAR 2. At OT 0 delayed, command 71 (logic 0.4.1, C11) transfers the contents of the address register to MAR 1, MAR 2, and TMAR and starts the selected memory device, and commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's. At OT 1, commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6) clear the A registers, and commands 67 and 267 (logic 0.5.1-2, A19) complement the divide-connect flip-flop and clear the carry-storage flip-flops. If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at OT 5. At OT 6, the TMAR is cleared by command 31, and, if core memory was selected, the contents of the selected register are transferred to the MBR's. At OT 7, command 74 (logic 0.5.3, B2) sets the step counter to 17, commands 43 (logic 0.1.1, D9) and 51 (logic 0.1.2, D8) transfer the data in the MBR's to the A registers for the *DVD* instruction (or commands 43 and 44 (logic 0.1.1, D8)) transfer the data in the left MBR to both A registers for the *TDV* instruction), and commands 16 (logic 0.5.1-2, D21) and 20 (logic 0.5.2-2, C22) make the accumulator and B registers positive. Commands 16 and 20 also complement the associated sign control flip-flops if the sign bit of the corresponding accumulator register contains a 1.

The remaining commands are identical for both instructions. At OT 8, the 2-mc sync flip-flop is set by command 138 (logic 0.5.3, D7) and the A registers are made positive by commands 22 and 229 (logic 0.5.1, D5). Commands 22 and 229 also complement the associated sign control flip-flops if the sign bit of the corresponding accumulator register contains a 1.

The 2-mc operation flip-flop is set at OT 9, and the 2-mc operate pulses begin at OT 10. At OT 10, command 134 (logic 0.2.2, A8) sets the pause flip-flop to stop the generation of instruction pulses at the end of the OT cycle. The arithmetic pause begins after OT 11, at which time command 161 (logic 0.3.1, B4) clears the PT-OT flip-flop.

The 2-mc operate pulses that begin at OT 10 are converted into command 80 (logic 0.5.3, C2) pulses, which activate the divide time pulse distributor. The divide time pulse distributor transmits a continuous repetition of five regularly timed pulses. These are the divide time pulses (DVTP's) and are identified as follows: DVTP 0, 1, 2, 3, and 4. The DVTP 0, 1, and 4 pulses control the divide operations in the arithmetic element; the DVTP 2 pulse sets the clear pause delay flip-flop when the step counter content is 2; and the DVTP 3 pulse is used to decrease the contents of the step counter by 1. This series of five pulses, which results in the generation of one partial quotient, is issued 16 times during the divide operation.

The DVTP 0 shifts the combined accumulator and B registers one bit position to the left, to double the dividend, and it makes the signs of the divisor and the

dividend unlike. If the signs of the operands are alike before the shift, this pulse is gated to complement the A register.

The DVTP 1 pulse examines the A register sign bit. If the sign bit is 0 (positive), the carry 0 line to the bit 15 adder circuit is pulsed and the positive form of the divisor (in the A register) is added to the 16 most significant bits of the negative dividend (in the accumulator register). If the sign bit is 1 (negative), the carry 1 line to the bit 15 adder circuit is pulsed and the negative form (complement) of the divisor plus the 1 that is carried (which makes this the 2's complement) is added to the 16 most significant bits of the positive dividend. The add process, which requires more time than is provided between DVTP pulses, is completed prior to the DVTP 4 pulse. The inherent shift right that occurs during the add process is corrected by DVTP 4.

During the 16th cycle of the divide process, the DVTP 2 pulse sets the clear pause delay sync flip-flop. During the same cycle, the DVTP 3 pulse steps the step counter from 2 to 1 and develops an output which is used to initiate the operation to clear the 2-mc operation flip-flop. When this DVTP 3 pulse is generated, the clear pause delay flip-flop is set by a 2-mc TPD on pulse. In step with the following DVTP 4 pulse, the next 2-mc TPD on pulse clears the pause, the clear pause delay, and the clear pause delay sync flip-flops. When the pause flip-flop is cleared, the time pulse distributor is again started to resume the generation of instruction pulses for the execution of the program.

The PT cycle is resumed at PT 0 with command 31, which clears MAR 2. At PT 0 delayed, command 91 (logic 0.4.1, A4) transfers the contents of the program counter to MAR 1, MAR 2, and TMAR and starts the selected memory device, and commands 41 and 53 clear the MBR's. At PT 1, commands 67 and 267 complement the divide-connect flip-flops to disconnect the carry lines of the respective sign-bit adder circuits from bit 15 of the associated B registers and to clear the carry storage flip-flops in preparation for correction of the remainder, and commands 61 and 261 (logic 0.5.1-2, A17) complement the respective A registers if the signs of the corresponding divisors and remainders are the same. At PT 2, commands 11 and 211 (logic 0.5.1, C5) sense bit S of the left and right A register, respectively. If this bit contains a 0, the associated bit 15 adder circuit is pulsed with a carry 0 signal (command 64 or 264) to cancel the effect of the last trial subtraction.

At PT 5, commands 60 and 260 (logic 0.5.1-2, A18) correct the inherent shift right if commands 11 and 211 resulted in an add operation. At the same time, the contents of the selected test memory register are transferred to the MBR's if a test memory address was selected. If a core memory address was selected, the contents of the selected core memory register are transferred to the MBR's at PT 6. At the same time, command 31 clears the TMAR, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address and step counter, and command 101 (logic 0.3.1, B13) clears the index interval and operation registers. If the associated sign control flip-flops are set, commands 9 (logic 0.5.1-2, C21) and 209 (logic 0.5.2-2, C22) complement the respective accumulator-B register combination and then clear the sign control flip-flops.

At the end of the *DVD* or *TDV* instruction, the quotient appears in B register bits S through 15, the remainder appears in accumulator bits 1 through 15, and the sign appears in the accumulator sign bit position.

3.4 SHIFT CLASS

The eight instructions of the shift class utilize the accumulator and B registers in either a shifting or a cycling ring. Shifting is accomplished between adjacent bits in either the left or the right direction. The left and right shifts in the B registers and the left shift in the accumulator registers are broadside shifts; the right shifts in the accumulator registers are ripple-shift processes. The instructions in this class are grouped into three categories: cycling *(DCL* and *FCL)*, accumulator-B register shifts *(DSL, DSR, LSR,* and *RSL)*, and accumulator shifts *(ASL* and *ASR)*. The shift commands and the register configurations for each of these instructions are shown in figure 3–11. This class of instructions is not indexable.

The sequence of commands common to these instructions depends on the number of shifts specified. If one shift is specified, the operation is accomplished in one PT cycle as follows: at PT 7 of the PT cycle, command 42 (logic 0.1.1, D9) transfers the contents of the left MBR to the operation and index interval registers, command 52 (logic 0.1.2, D8) transfers the contents of the right MBR to the address register and the step counter, and command 92 (logic 0.4.1, B1) steps the program counter.

At PT 9, the 2-mc sync flip-flop is set by command 138 (logic 0.5.3, D7); at PT 10, the 2-mc operation flip-flop is complemented to the set state, and the clear pause delay sync flip-flop (logic 0.5.3, D6) is set. At PT 11, command 73 steps the step counter from 1 to 0, the clear pause delay flip-flop is set by a 2-mc TPD on pulse, the 2-mc sync flip-flop is cleared by a 2-mc operate pulse, the 2-mc operate flip-flop is complemented to the cleared state by a 2-mc TPD on pulse, and the shift commands for the instruction being executed are generated by the 2-mc operate pulse. The applicable shift commands, which occur simultaneously at PT 11, are covered in the discussions of the specific instructions.

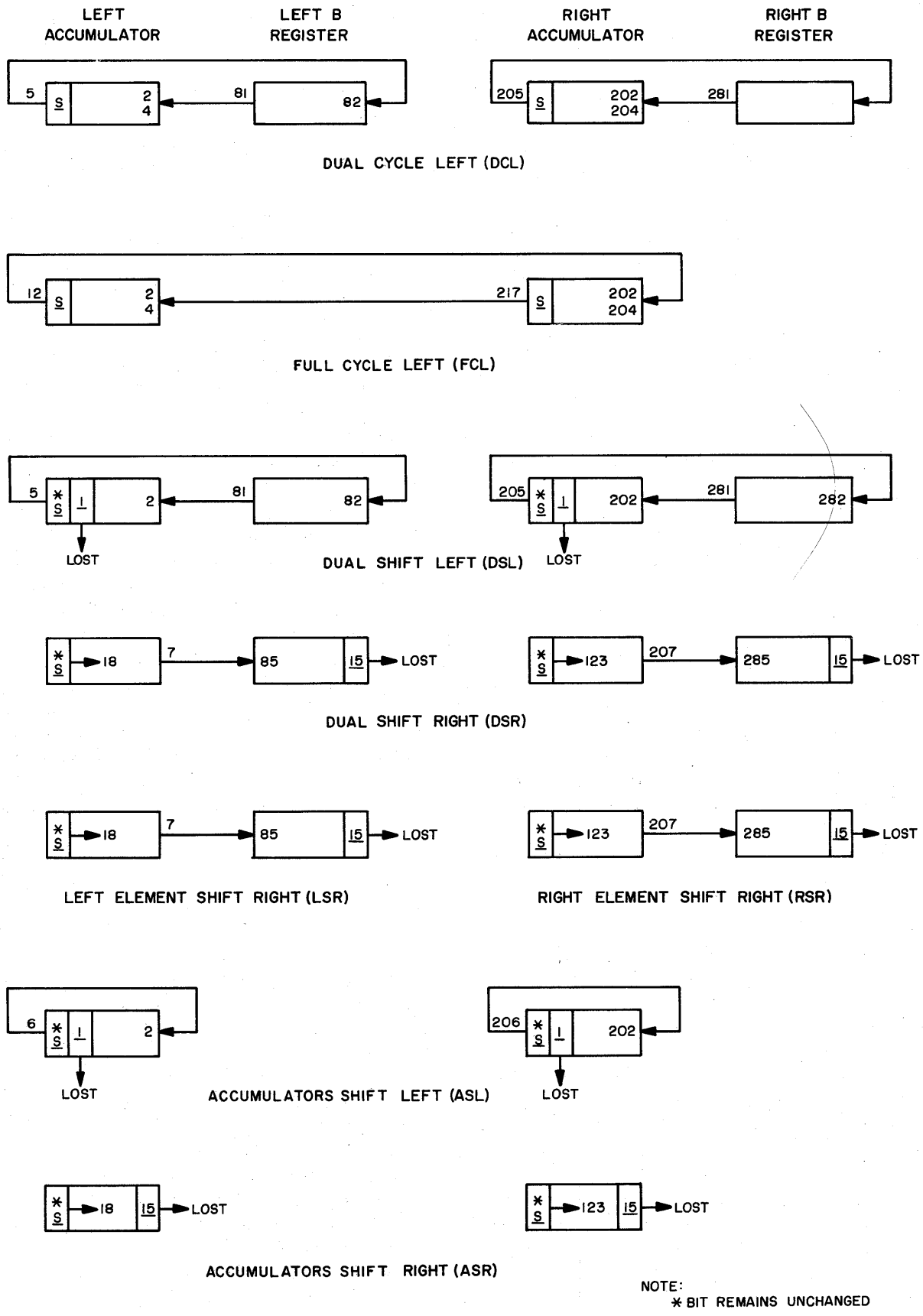


Figure 3-11. Execution of Shift Commands

At PT 0 time, MAR 2 is cleared by command 31 (logic 0.1.4, B5) and the pause, the clear pause delay, and the clear pause delay sync flip-flops are cleared by a 2-mc TPD on pulse. At PT 0 delayed, commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's, and command 91 (logic 0.4.1, A4) transfers the contents of the program counter to MAR 1, MAR 2, and TMAR and starts the selected memory device. If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at PT 5. If a core memory address was selected, the contents of the selected register are transferred to the MBR's at PT 6. At the same time command 31 (logic 0.1.3, B13) clears the TMAR, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and step counter, and command 101 (logic 0.3.1, B13) clears the index interval and operation registers.

If two through six shifts are specified, the shift operation is accomplished during the PT cycle as follows. At PT 7, command 42 transfers the contents of the left MBR to the operation and index interval registers, command 52 transfers the contents of the right MBR to the address register and the step counter, and command 92 steps the program counter.

The 2-mc sync flip-flop is set by command 138 at PT 9, and the 2-mc operation flip-flop is set at PT 10. At PT 11, the 2-mc sync flip-flop is cleared by a 2-mc operate pulse, command 73 reduces the contents of the step counter by 1, and the shift commands for the instruction in progress are generated by the 2-mc operate pulse to accomplish the first shift. Commands 73 and those required to accomplish the shift are repeated in step with the PT 0 through PT 4 pulses, depending upon how many shifts are to be performed. These commands are generated by 2-mc operate pulses. When the step counter content is reduced from 2 to 1, action is initiated to clear the 2-mc operation flip-flop. Because of inherent delays, this flip-flop is not cleared until the step counter is stepped to zero. In addition to controlling the generation of commands, the step-counter pulses initiate action to clear the pause flip-flop. However, since this flip-flop is not set under these conditions, this action does not produce any positive results. The applicable shift commands, which occur simultaneously each time the step counter is stepped, are covered in the discussions of the specific instructions.

At PT 0 of the next PT cycle, command 31 clears MAR 2. At PT 0 delayed, commands 41 and 53 clear the MBR's, command 91 transfers the contents of the program counter to the three memory address registers, and the selected memory is started. If test memory was selected, the contents of test memory are transferred to the MBR's at PT 5. At PT 6, the contents of core memory are transferred to the MBR's (if a core memory address was selected), command 31 clears the TMAR, command 77 clears the address register and the step counter, and command 101 clears the index interval and operation registers.

If 7 through 63 shifts are specified, the shift operation requires a PT cycle and an arithmetic pause as follows. At PT 7, command 42 transfers the contents of the left MBR to the operation and index interval registers, command 52 transfers the contents of the right MBR to the address register and the step counter, and command 92 steps the program counter. At PT 9, the 2-mc sync flip-flop is set by command 138. At PT 10, the 2-mc operation flip-flop is set by a 2-mc TPD on pulse, and the pause flip-flop is set by command 134 (logic 0.2.2, A8).

The arithmetic pause begins after the PT 11 pulse, at which time the 2-mc sync flip-flop is cleared. At PT 11, command 73 reduces the contents of the step counter by 1, and the shift commands for the instruction in progress accomplish the first shift. The applicable shift commands, which occur simultaneously each time the step counter is stepped, are covered in the discussions of the specific instructions.

The shift commands and command 73 are issued at 0.5-μsec intervals during the arithmetic pause until the step counter content is reduced to 7. When the step counter is reduced from 7 to 6, the clear pause delay sync flip-flop is set by an odd-to-even-step-counter pulse. (If only seven shifts are specified, this flip-flop is set when command 73 is first issued.) When the step counter is reduced from 6 to 5, the clear pause delay flip-flop is set by a 2-mc TPD on pulse. When the step counter is reduced from 5 to 4, the pause, the clear pause delay, and the clear pause delay sync flip-flops are cleared by a 2-mc TPD on pulse.

The PT cycle is resumed when the step counter content is reduced from 4 to 3. At the same time (PT 0), command 31 clears MAR 2. At PT 0 delayed, commands 41 and 53 clear the MBR's, command 91 transfers the contents of the program counter to the three memory address registers, and the selected memory device is started. At PT 1, the clear pause delay sync flip-flop is set, and command 73 steps the step counter from 3 to 2. At PT 2, the clear pause delay flip-flop is set, and command 73 steps the step counter from 2 to 1. At PT 3, the pause, the clear pause delay, the clear pause delay sync, and the 2-mc operation flip-flops are cleared, and command 73 steps the step counter from 1 to 0. If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at PT 5. At PT 6, the contents of the selected core memory register are transferred to the MBR (if a core memory address was selected), command 31 clears the TMAR, command 77 clears the address register and step counter, and command 101 clears the index interval and operation registers.

3.4.1 *Dual Cycle Left (DCL), Cycle Accumulators Left (FCL)*

The *DCL* and *FCL* instructions connect the applicable registers to form 32-bit cycling rings so that the information in all bits can be shifted without losing any data. The 32-bit ring is formed by the accumulator and B register combination for the *DCL* instruction and by the left and right accumulator registers for the *FCL* instruction.

The cycle operation of the *DCL* instruction is initiated by commands 2, 4, 5, 81, and 82 for the left arithmetic registers and by commands 202, 204, 205, 281, and 282 for the right arithmetic registers. These commands occur simultaneously each time command 73 steps the step counter. Commands 2 and 4 (logic 0.5.1-2, E21) and 202 and 204 (logic 0.5.2-2, D21) shift accumulator register bits 1 through 15 one bit position to the left, commands 5 and 205 (logic 0.5.1-2, D19) shift the accumulator register sign bits into bit 15 of the B registers, commands 82 (logic 0.5.1-3, D7) and 282 (logic 0.5.2-3, D7) shift B register bits 1 through 15 one bit position to the left, and commands 81 (logic 0.5.1-3, E6) and 281 (logic 0.5.2-3, E6) shift the B register sign bit into bit 15 of the associated accumulator register.

The shift operation of the *FCL* instruction is initiated by commands 2, 4, 12, 202, 204, and 217. As in the *DCL* instruction, these commands occur simultaneously each time command 73 is issued. Commands 2 and 4 shift the left accumulator register bits 1 through 15 one bit position to the left; command 12 (logic 0.5.1-2, 18D) shifts the left accumulator register sign bit into bit 15 of the right accumulator register; commands 202 and 204 (logic 0.5.2-2, 21D) shift the right accumulator register bits 1 through 15 one bit position to the left; and command 217 (logic 0.5.2-2, 18D) shifts the right accumulator register sign bit into bit 15 of the left accumulator register.

The shift commands for either instruction are repeated at a 2-mc rate until the desired number of shifts has been accomplished.

3.4.2 *Dual Shift Left (DSL), Dual Shift Right (DSR), Left Element Shift Right (LSR), Right Element Shift Right (RSR)*

Each of the instructions in this category (accumulator B register shifts) connects the applicable registers to form a 31-bit shifting register. The execution of one of these instructions causes the information in the 31-bit register to be shifted one bit position in the specified direction, and it causes one bit of information to be lost. In the *DSL* instruction, the information in the most significant bit position (accumulator bit 1) is lost; in the *DSR, LSR,* or *RSR* instruction, the information in the least significant bit position (B register bit 15) is lost. The sign bit (accumulator bits) of the 31-bit register is not changed by the execution of any instruction in this category.

The shift left specified by the *DSL* installation is initiated by commands 2, 81, 82, and 5 for the left arithmetic registers and by commands 407, 281, 282, and 205 for the right arithmetic registers. These commands are issued simultaneously each time command 73 steps the step counter, and they are repeated until the desired number of shifts has been accomplished. Commands 2 (logic 0.5.1-2, E21) and 202 (logic 0.5.2-2, D21) shift accumulator register bits 2 through 15 one bit position to the left, commands 81 (logic 0.5.1-3, E6) and 281 (logic 0.5.2-3, E6) shift the B register sign bit into bit 15 of the associated accumulator register, commands 82 (logic 0.5.1-3, D7) and 282 (logic 0.5.2-3, D7) shift B register bits 2 through 15 one bit position to the left, and commands 5 and 205 (logic 0.5.1-2, D19) shift the accumulator register sign bit into bit 15 of the associated B register. Since the information in the accumulator register sign bits remains the same throughout the execution of the *DSL* instruction, the number of identical, least significant bits in each B register agrees with the number of shifts accomplished.

The *DSR* instruction differs from the *DSL* instruction in that the shift is to the right and is initiated by commands 7, 18, and 85 for the left arithmetic registers and by commands 123, 207, and 285 for the right arithmetic registers. These commands are issued simultaneously each time command 73 steps the step counter and are repeated until the desired number of shifts has been accomplished. Commands 85 (logic 0.5.1-3, D6) and 285 (logic 0.5.2-3, D7) shift B register bits 5 through 14 one bit position to the right, commands 7 (logic 0.5.1-2, D1) and 207 (logic 0.5.2-2, D1) shift accumulator register bits 15 into the sign bits of the B registers, and commands 18 (logic 0.5.1-2, D1) and 123 (logic 0.5.2-2, D1) ripple-shift accumulator bits 5 through 14 one bit position to the right. Each shift causes the information in the sign bits to appear in the next most significant bit position in the associated accumulator register; at the end of the *DSR* instruction, the number of identical, most significant bits in each accumulator agrees with the number of shifts accomplished.

The shifts in the *LSR* and *RSR* instructions are identical with the shifts in the corresponding registers formed for the *DSR* instruction (fig. 3–10). As the designations imply, the *LSR* instruction affects only the left arithmetic registers and the *RSR* instruction affects only the right arithmetic registers.

3.4.3 *Shift Accumulators Left (ASL), Shift Accumulators Right (ASR)*

The *ASL* and *ASR* instructions utilize the accumulator registers separately as 16-bit shifting registers. The execution of either instruction causes the information to be shifted one bit position and causes one bit of in-

formation to be lost. In the *ASL* instruction, the information in the most significant bit position (accumulator bit 1) is lost; in the *ASR* instruction, the information in the least significant bit position (accumulator bit 15) is lost. The accumulator sign is not changed by the execution of either instruction.

The shift left in the *ASL* instruction is initiated by commands 2 (logic 0.5.1-2, E21) and 202 (logic 0.5.2-2, D21), which shift accumulator register bits 2 through 15 one bit position to the left, and commands 6 and 206 (logic 0.5.1-2, D19), which shift the respective sign bit into bit 15. These commands are issued simultaneously each time command 73 steps the step counter and are repeated until the desired number of shifts has been accomplished. Since the information in the sign bit remains the same, the number of identical, least significant bits in each accumulator register agrees with the number of shifts accomplished.

The shift right in the *ASR* instruction is initiated by commands 18 (logic 0.5.1-2, D1) and 123 (logic 0.5.2-2, D1), which are applied to bit 14 of the respective accumulator registers to start the ripple-shift. Each shift causes the information in bits 1 through 14 to be shifted one bit position to the right, and it causes the information in the sign bit to be shifted into bit 1. Commands 18 and 123 are issued at a 2-mc rate until the desired number of shifts has been accomplished.

Since the sign bit information does not change, the number of identical, most significant bits in each accumulator register agrees with the number of shifts accomplished.

3.5 STORE CLASS

Each of the seven store-class instructions is indexable, and the *RST, STA,* and *AOR* instructions can be controlled to provide either 16- or 17-bit operation. Except for the *FST* instruction, each instruction effects the transfer of old data from the memory element and the transfer of new data to the memory element. The *AOR* and *DEP* instructions involve arithmetic or logical modification of the old data before it is stored back into memory. The *FST* instruction requires a PT cycle and an OTB cycle, and the remaining instructions in this class require a PT cycle, an OTA cycle, and an OTB cycle. The first part of the instruction cycle, which is the same for all store-class instructions, begins at PT 7. At this time, command 42 (logic 0.1.1, D9) transfers the contents of the left MBR to the operation and index interval registers, command 52 (logic 0.1.2, D8) transfers the contents of the right MBR to the address register and the step counter, and command 92 (logic 0.4.1, B1) steps the program counter. At PT 8, the contents of the selected index register (if one is selected) are added to the contents of the address register. At PT 11, the PT-OT is set by command 131 (logic 0.3.1, B3) to terminate the first PT cycle and to start the OT cycle. The operations that occur during the subsequent OT and PT cycles vary according to the instruction being accomplished and are discussed in the following paragraphs.

3.5.1 *Full Store (FST), Left Store (LST), Right Store (RST), Store Address (STA)*

The *FST* instruction transfers data from the accumulator registers to the MBR's. Since this instruction does not require an OTA cycle, the A-B flip-flop is set by command 132 (logic 0.3.1, B3) at PT 11. At OTB 0, command 31 (logic 0.1.4, B5) clears MAR 2. At OTB 0 delayed, command 71 (logic 0.4.1, C11) transfers the contents of the address register to the three memory address registers and starts the selected memory device, and commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR. At OTB 2, commands 17 (logic 0.5.1-2, E19) and 121 (logic 0.5.2-2, D19) transfer the contents of the left and right accumulator registers to the corresponding MBR's.

The selected core memory, started at OTB 0 delayed, is inhibited at OTB 3 so that the contents of the selected memory register will not be transferred into the MBR's. During the latter portion of this memory cycle, the new information is stored in memory. At OTB 6, the TMAR is cleared by command 31.

At OTB 11, the A-B flip-flop is cleared by command 167 (logic 0.3.1, B4) and the PT-OT flip-flop is cleared by command 161 (logic 0.3.1, B4). These two commands terminate the OTB cycle and start the next PT cycle. At PT 0, command 31 clears MAR 2. At PT 0 delayed, command 91 (logic 0.4.1, A4) transfers the contents of the program counter to the three memory address registers and starts the selected memory device, and commands 41 and 53 clear the MBR's. If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at PT 5. If a core memory address was selected, the contents of the specified register are transferred to the MBR's at PT 6. At the same time, command 31 clears the test MAR, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and the step counter, and command 101 (logic 0.3.1, B13) clears the index interval and operation registers.

The *LST* instruction is used to replace the data contained in the left half-word of the selected memory word by the data contained in the left accumulator register. At OT 0 of the OTA cycle, command 31 clears MAR 2. At OT 1, commands 41 and 53 clear the MBR's, and command 71 transfers the contents of the address register to the three memory address registers and starts the selected memory device. If test memory was selected, the contents of test memory are transferred to the MBR's at OT 5. At OT 6, the TMAR is cleared by command 31, the A registers are cleared by commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6), and, if core mem-

ory was selected, the contents of the selected register are transferred to the MBR's. At OT 7, command 43 (logic 0.1.1, D9) transfers the contents of the left MBR to the left A register (this data is not used), and command 51 (logic 0.1.2, D8) transfers the contents of the right MBR to the right A register for temporary storage. At OT 11, command 132 sets the A-B flip-flop to terminate the OTA cycle and start the OTB cycle.

At OTB 0, command 31 clears MAR 2. At OTB 0 delayed, command 71 transfers the contents of the address register to the three memory address registers and starts the selected memory device, and commands 41 and 53 clear the MBR. At OTB 2, command 17 (logic 0.5.1-2, E19) transfers the contents of the left accumulator register to the left MBR, and command 228 (logic 0.5.2, D7) returns the data in the right A register to the right MBR. Bit LS of the right A register is inhibited during this instruction.

The remaining commands of the *LST* instruction are executed in the same manner as those described for the *FST* instruction.

The *RST* instruction stores the right half-word in the same manner that the *LST* instruction stored the left half-word; but, in this case, either 16 or 17 bits may be transferred. At OT 0 of the OTA cycle, command 31 clears MAR 2. At OT 0 delayed, commands 41 and 53 clear the MBR's, and command 71 transfers the contents of the address register to the three memory address registers and starts the selected memory device. If test memory was selected, the contents of test memory are transferred to the MBR's at OT 5. At OT 6, the TMAR is cleared by command 31, the A registers are cleared by commands 21 and 230, and, if core memory was selected, the contents of the specified core memory register are transferred to the MBR's. At OT 7, command 43 transfers the contents of the left MBR to the left A register for temporary storage. At OT 11, command 132 sets the A-B flip-flop to end the OTA cycle and start the OTB cycle.

At OTB 0, command 31 clears MAR 2. At OTB 0 delayed, command 71 transfers the contents of the address register to the three memory address registers and starts the selected memory device, and commands 41 and 53 clear the MBR's.

Commands 23 (logic 0.5.1, D5) and 121 (logic 0.5.2-2, D19) are issued at OT 2 of the OTB cycle. If 16-bit operation is specified (bit L12 of the instruction word is a 0), then command 23 returns the contents of the left A register to the left MBR unaltered, and command 121 transfers the contents of the right accumulator register to the right MBR. If 17-bit operation is specified (bit L12 of the instruction word is a 1), then command 23 transfers bits 1 through 15 of the left A register to the corresponding bit positions in the left MBR, and command 121 transfers the contents of the right accumulator register to the right MBR and the contents of bit LS of the accumulator register to bit LS of the MBR. The sign bit of the left A register is inhibited during 17-bit operation.

The remaining commands of the *RST* instruction are executed in the same manner as those described for the *FST* instruction.

The execution of the *STA* instruction is similar to that of the *RST* instruction except that, in this case, address data that was previously transferred to the right A register (by a branch instruction) is stored in the right half-portion of the specified memory word. At OT 0 of the OTA cycle, command 31 clears MAR 2. At OT 0 delayed, commands 41 and 53 clear the MBR's, and command 71 transfers the contents of the address register to the three memory address registers and starts the selected memory device. If test memory was selected, the contents of the selected test memory register are transferred to the MBRs at OT 5. At OT 6, the test MAR is cleared by command 31, the left A register is cleared by command 21, and the contents of the selected core memory register (if core memory was selected) are transferred to the MBR's. At OT 7, command 43 transfers the contents of the left MBR to the left A register for temporary storage. At OT 11, command 132 sets the A-B flip-flop to end the OTA cycle and start the OTB cycle.

At OTB 0, command 31 clears MAR 2. At OTB 0 delayed, command 71 transfers the contents of the address register to the three memory address registers and starts the selected memory device, and commands 41 and 53 clear the MBR's.

Commands 23 (logic 0.5.1, D5) and 228 (logic 0.5.2, D7) are issued at OT 2 of the OTB cycle. If 16-bit operation is specified (bit L12 = 0), then command 23 returns the content of the left A register to the left *MBR*'s unaltered, and command 228 transfers the contents of bits RS through R15 of the right A register to the right MBR. If 17-bit operation is specified (bit L12 = 1), then command 23 returns bits L1 through L15 of the left A register to the corresponding bit positions in the left MBR, and command 228 transfers bits RS through R15 of the right A register to the right MBR and bit LS of the right A register to the sign bit position of the left MBR. The sign bit of the left A register, inhibited as it was in the RST instruction, is lost.

The remaining commands of the *STA* instruction are executed in the same manner as those described for the *FST* instruction.

3.5.2 *Exchange (ECH)*

The *ECH* instruction exchanges the data in the accumulator registers with the data in the selected core memory register. This is accomplished by storing the data from the selected memory register in the A regis-

ters during the OTA cycle, and transferring the data in the accumulator registers to the specified memory location during the OTB cycle. During the latter cycle, the contents of the A registers are added to the cleared accumulator registers.

The OTA cycle begins at OT 0, at which time command 31 (logic 0.1.4, B5) clears MAR 2. At OT 0 delayed, commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's, and command 71 (logic 0.4.1, C11) transfers the contents of the address register to the three memory address registers and starts the selected memory device. If test memory was selected, the contents of test memory are transferred to the MBR's at OT 5. At OT 6, the TMAR is cleared by command 31, the A registers are cleared by commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6), and, if core memory was selected, the contents of the selected register are transferred to the MBR's. Commands 43 (logic 0.1.1, D9) and 51 (logic 0.1.2, D8) transfer the contents of the MBR's to the A registers at OT 7. At OT 11, command 132 (logic 0.3.1, B3) sets the A-B flip-flop to end the OTA cycle and start the OTB cycle.

Command 31 clears MAR 2 at OT 0 of the OTB cycle. At OTB 0 delayed, command 71 transfers the contents of the address register to the three memory address registers and starts the selected memory device, and commands 41 and 53 clear the MAR. At OT 2, commands 17 (logic 0.5.1-2, E19) and 121 (logic 0.5.2-2, D19) transfer the contents of the accumulator registers to the MBR's.

The selected core memory, started at OTB 0 delayed, is inhibited at OTB 3 so that the contents of the selected memory register will not be transferred into the MBR's. During the latter part of this memory cycle, the new information is stored in memory. At OTB 6, the TMAR is cleared by command 31, and the accumulator registers are cleared by commands 10 (logic 0.5.1-2, B21) and 210 (logic 0.5.2-2, B21).

At OTB 10, the contents of the A registers are transferred to the accumulator registers by commands 64 (logic 0.5.1-2, A1) and 264 (logic 0.5.2-2, A1), which pulse the carry 0 lines of the associated bit 15 adders. The addition process continues (for 1.5 μsec) through PT 0 of the second PT cycle.

At OTB 11 of the OTB cycle, the A-B flip-flop is cleared by command 167 (logic 0.3.1, B4) and the PT-OT flip-flop is cleared by command 161 (logic 0.3.1, B4). These two commands end the OTB cycle and start the next PT cycle. At PT 0, command 31 clears MAR 2. At PT 0 delayed, command 91 (logic 0.4.1, A4) transfers the contents of the program counter to the three memory address registers and starts the selected memory device, commands 41 and 53 clear the MBR's, and commands 2, 4, 81, 89, 202, 204, 281, and 289 correct the inherent shift right that occurred as a result of the add operation. Commands 2 (logic 0.5.1-2, E21) and 202 (logic 0.5.2-2, D21) initiate broadside shifts which transfer the information in bits 2 through 15 of each accumulator register one bit position to the left, commands 4 (logic 0.5.1-2, E21) and 204 (logic 0.5.2-2, D21) shift the contents of bit L1 and R1 into the corresponding accumulator register sign bit, commands 81 (logic 0.5.1-3, E6) and 281 (logic 0.5.2-3, E6) transfer the contents of each B register sign bit into bit 15 of the corresponding accumulator register, and commands 89 (logic 0.5.1-3, B7) and 289 (logic 0.5.2-3, B7) return the stored sign bit information to the B register sign bit positions.

If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at PT 5. If a core memory address was selected, the contents of the selected register are transferred to the MBR's at PT 6. At the same time, command 31 clears the TMAR, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and step counter, and command 101 (logic 0.3.1, B13) clears the index interval and operation registers.

3.5.3 *Add One Right (AOR)*

The *AOR* instruction adds a 1 to the right half-portion of the specified memory word. This is accomplished by transferring the word from the specified memory register to the A registers, pulsing the carry 1 line of the bit R15 adder circuit so that the new sum appears in the accumulator register, and then transferring the contents of the accumulator register back into the specified memory register. The left half-portion of the memory is not affected by this instruction but is temporarily stored in the left A register, and later transferred back into memory in the same manner described for the *RST* instruction.

At OT 0 of the OTA cycle, command 31 (logic 0.1.4, B5) clears MAR 2. At OT 0 delayed, commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's, and command 71 (logic 0.4.1, C11) transfers the contents of the address register to the three memory address registers and starts the selected memory device. If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at OT 5. At OT 6, the contents of the selected core memory register (if core memory was selected) are transferred to the MBR's, the TMAR is cleared by command 31 (logic 0.1.3, A16), the A registers are cleared by commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6), and the right accumulator register is cleared by command 210 (logic 0.5.2-2, B21). If 17-bit operation is specified by the instruction word (bit L12 = 1), command 10 (logic 0.5.1-2, B21) is also generated to clear the left accumulator register.

Commands 43 (logic 0.1.1, D9) and 51 (logic 0.1.2, D8) transfer the contents of the MBR's to the A register at OT 7. At OT 8, command 69 (logic 0.5.2-2, A1) pulses the carry 1 line of the bit R15 adder circuit. At OT 11, the shift right that resulted from the add operation is corrected by commands 202 and 204 (logic 0.5.2-2, D21) and commands 281 and 289 (logic 0.5.2-3, E6 and B7). If 17-bit operation is specified, command 204 also shifts accumulator register bit L1 to bit LS.

At the same time (OTA 11), the A-B flip-flop is set by command 132 (logic 0.3.1, B3) to end the OTA cycle and to start the OTB cycle. At OTB 0, command 31 clears MAR 2, and command 14 (logic 0.3.1, E3) sets accumulator bit R15 to a 1 if the previous addition resulted in an end carry. At OTB 0 delayed, command 71 transfers the contents of the address register to the three memory address registers and starts the selected memory device, and commands 41 and 53 clear the MBR's.

Commands 23 (logic 0.5.1, D5) and 121 (logic 0.5.2-2, D19) are issued at IP 2 of the OTB cycle. If 16-bit operation is specified by the instruction word (bit L12 = 0), command 23 transfers the contents of the left A register to the left MBR unaltered, and command 121 transfers the contents of the right accumulator register to the right MBR. If 17-bit operation is specified (bit L12 of the instruction word is a 1), command 23 transfers bits 1 through 15 of the left A register to the corresponding bit positions in the left MBR, and command 121 transfers the contents of the right accumulator register to the right MBR and transfers bit LS of the left accumulator register to bit LS of the left MBR. The sign bit of the left A register is inhibited during 17-bit operation.

The selected core memory, started at OTB 0 delayed, is inhibited at OTB 3 so that the contents of the selected memory register will not be transferred into the MBR's. During the latter part of this memory cycle, the new information is stored in memory. At OTB 6, TMAR is cleared by command 31.

At OTB 11, the A-B flip-flop is cleared by command 167 (logic 0.3.1, B4) and the PT-OT flip-flop is cleared by command 161 (logic 0.3.1, B4) to terminate the OTB cycle and to start the next PT cycle. At PT 0, command 31 clears MAR 2. At PT 0 delayed, command 91 (logic 0.4.1, A4) transfers the contents of the program counter to the three memory address registers and starts the selected memory device, and commands 41 and 53 clear the MBR's. If test memory was selected, the contents of the specified test memory register are transferred to the MBR's at PT 5. If a core memory address was selected, the contents of the specified register are transferred to the MBR's at PT 6. At the same time, command 31 clears the TMAR, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and step counter, and command 101 (logic 0.3.1, B13) clears the index interval and operation registers. If execution of this instruction results in an arithmetic overflow, the overflow condition is recorded at PT 6 by command 66 (logic 0.5.1-2, B22) if 17-bit operation was specified, or by command 266 (logic 0.5.1-2, B22) if 16-bit operation was specified.

3.5.4 *Deposit (DEP)*

The *DEP* instruction is used to replace specific bits of a memory word with the corresponding bits of the accumulator register. This is accomplished by reading the content of the selected memory register and modifying the specified bits during the OTA cycle, and then storing the modified memory word back into the same memory register during the OTB cycle.

Prior to the initiation of this instruction, each B register is loaded (by the *LDB* instruction) with a mask that contains 1's in the bit positions that are to be modified, and 0's in the remaining bit positions. At OT 0 of the OTA cycle, command 31 (logic 0.1.4, B5) clears MAR 2. At OT 0 delayed, commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's, and command 71 (logic 0.4.1, C11) transfers the contents of the address register to the three memory address registers and starts the selected memory device. At OT 1, commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6) clear the A registers. At OT 2, commands 19 (logic 0.5.1-2, B21) and 219 (logic 0.5.2-2, A22) complement the contents of the accumulator registers. At OT 4, commands 88 (logic 0.5.1-3, D6) and 288 (logic 0.5.2-3, C7) transfer the masks from the B registers to the A registers. If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at OT 5.

Commands 25 (logic 0.5.1, D4) and 225 (logic 0.5.2, D7) are also issued at OT 5 to logically multiply (AND function) the masks in the A registers and the complemented operands in the accumulator registers. The logical multiplication is accomplished by gating the 0 side outputs of the A register flip-flops to the 0 side inputs of the corresponding accumulator flip-flops. The A register 0 bits (unmasked bits), which identify the memory word bits that are not to be modified, change the corresponding bits in the accumulator registers to 0's. The 1's then appearing in the accumulator register indicate which of the masked bits are to be replaced by 0's. The masked accumulator bits that contain 0's indicate which of the masked bits are to contain 1's.

This first step in the *Deposit* instruction is shown numerically as follows:

1	110011	Accumulator register
2	001100	Complement accumulator register (*1*)

3	010100	A register, mask transferred from B register. Indicates that bits 2 and 4 of *1* are to be deposited in the memory word.
4	000100	Logical multiplication of *2* and *3*, appears in accumulator register. Indicates that bit 2 of memory word is to contain a 1 and that bit 4 is to contain a 0. All other bits of memory word retain their original content.

At OT 6, the TMAR is cleared by command 31, and, if core memory was selected, the contents of the specified core memory register are transferred to the MBR's. At OT 7, the accumulator registers are complemented by commands 19 and 219 so that the 0's of the masked bits appear as 0's and all other bits appear as 1's. At the same time, the selected memory word in the MBR's is logically added (OR function) to the contents of the A registers by commands 43 (logic 0.1.1, D9) and 51 (logic 0.1.2, D8). The 0's now in the A registers correspond to the 0's of the unmasked bits of the memory word that were not selected for replacement. All other bits in the A registers are 1's.

This second step of the *Deposit* instruction is shown numerically as follows:

5	111011	Complement accumulator register (*4*)
6	101101	Memory word in memory buffer register
7	111101	Logically add *6* to *3*; sum (in A register) indicates that bit 5 of original memory word (*6*) is a 0, all other unmasked bits are 1's.

At OT 9, commands 25 and 225 again logically multiply (AND function) the accumulator and A registers. The operands in these registers correspond to steps *5* and *7* of the previous numerical illustration, and the logical multiplication of these two operands produces 111001. Bits 1, 3, 5, and 6 of this number are the same as those of the original memory word (*6*), and bits 2 and 4 are the same as the masked bits (*1*) originally contained in the accumulator register.

At OT 11, command 132 (logic 0.3.1, B3) sets the A-B flip-flop to end the OTA cycle and to start the OTB cycle. At OTB 0, command 31 clears MAR 2. At OTB 0 delayed, command 71 transfers the contents of the address register to the three memory address registers and starts the selected memory device, and commands 41 and 53 clear the MBR's. Commands 17 (logic 0.5.1-2, E19) and 121 (logic 0.5.2-2, D19) transfer the contents of the accumulator registers to the MBR's at OTB 2.

The remaining commands of the *DEP* instruction are executed in the same manner as those described for the *FST* instruction.

3.6 BRANCH CLASS

Generally, branch class instructions are used to determine whether specific conditions exist that would require a change in the sequential execution of the operating program. If the branch condition is met during the execution of any branch class instruction, the contents of the program counter are transferred to the right A register for address storage, and the program continues sequentially from the branch address specified by the address portion of the branch instruction. If the branch condition is not met, the program continues sequentially from the next instruction. Of the six instructions contained in this class, two (the *Branch and Index [BPX]* and the *Branch and Sense [BSN]*) sense for conditions that are external to the arithmetic element. The four branch instructions discussed in this chapter sense the accumulator registers for negative sign bits, or for zero content (either positive or negative).

3.6.1 *Branch on Full Zero (BFZ)*

The *BFZ* instruction causes the program to branch to the address specified by the instruction if the contents of the two accumulator registers are zero, either positive or negative, in any combination. The content of each accumulator register with respect to the *BFZ* branch condition is determined by making the register negative, adding a 1, complementing the number, and then checking the sign bit for a 1.

This instruction requires a PT cycle and an OT cycle. At PT 7 of the instruction cycle, command 42 (logic 0.1.1, D9) transfers the contents of the left MBR to the operation and index interval registers, command 52 (logic 0.1.2, D8) transfers the contents of the right MBR to the address register and the step counter, and command 92 (logic 0.4.1, B1) steps the program counter. At PT 9, commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6) clear the A registers. At PT 11, command 131 (logic 0.3.1, B3) sets the PT-OT flip-flop to terminate the PT cycle and to start the OT cycle.

The OT cycle begins with command 31 (logic 0.1.4, B5), which clears MAR 2 at OT 0. At OT 0 delayed, commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's, and command 71 (logic 0.4.1, C11) transfers the contents of the address register to the three memory address registers and starts the selected memory device. At OT 1, commands 13 (logic 0.5.1-2, B21) and 213 (logic 0.5.2-2, A22) make the accumulator registers positive. Commands 13 and 213 also complement the associated sign control flip-flops if the sign bit of the corresponding accumulator register contains a 1. At OT 2, commands 19 (logic 0.5.1-2, B21) and 219 (logic 0.5.2-2, A22) complement the accumulator reg-

isters to make them negative. At OT 3, commands 62 (logic 0.5.1-2, A1) and 69 (logic 0.5.2-2, A1) pulse the carry 1 lines of the bit L15 and R15 adder circuits. The add operation initiated by commands 62 and 69 continues for 1.5 μsec.

If test memory was selected, the contents of the selected register are transferred to the MBR's at OT 5. At OT 6, the TMAR is cleared by command 31, and, if core memory was selected, the contents of the selected register are transferred to the MBR's. At the same time, the inherent shift right that occurred during addition is corrected by commands 2, 4, 81, 89, 202, 204, 281, and 289. Commands 2 (logic 0.5.1-2, E21) and 202 (logic 0.5.2-2, D21) initiate broadside shifts which transfer the information in the left and right accumulator bits 2 through 15 one bit position to the left, commands 4 (logic 0.5.1-2, E21) and 204 (logic 0.5.2-2, D21) shift the contents of accumulator bit 1 into the corresponding accumulator sign bits, commands 81 (logic 0.5.1-3, E6) and 281 (logic 0.5.2-3, E6) transfer the contents of the respective B register sign bit to the corresponding accumulator register bit 15, and commands 89 (logic 0.5.1-3, B7) and 289 (logic 0.5.2-3, B7) transfer the B register sign storage information to the B register sign bit positions. If the negative number in the accumulator register was negative 0 before addition, then the addition process changed the sign bit of that number from 1 to 0; but if the negative number contained one or more 0 bits, then the sign bit remained a 1.

At OT 7, the contents of the accumulator registers are again complemented by commands 19 and 219. At OT 8, the accumulator register sign bits are sensed for 1's by command 162 (logic 0.5.2-2, D19). The sign bits of the two accumulator registers are sensed in series so that, if both are 1's, the branch flip-flop will be set. At the same time, the contents of the accumulator registers are restored to their original value by commands 62 and 69, which add 1's to the accumulator registers to offset the previous addition of 1's to the complemented numbers. At OT 11, commands 2, 4, 81, 89, 202, 204, 281, and 289 correct the inherent shift right that resulted from the second addition process, and command 161 (logic 0.3.1, B4) clears the PT-OT flip-flop to end the OT cycle. If the branch condition was met at OT 8 (i.e., branch flip-flop was set), then command 93 (logic 0.4.1, C5) will be executed at OT 11 to transfer the program counter content to the right A register, and to clear the program counter. If the branch condition is not met, command 93 is not executed.

The PT cycle is resumed at the next instruction pulse (PT 0) with command 31, which clears MAR 2, and command 154 (logic 0.4.1, D13), which transfers the contents of the address register to the program counter if the branch flip-flop is set. At PT 0 delayed, the MBR's are cleared by commands 41 and 53, and the contents of either the address register (if the branch flip-flop is set) or the program counter (if the branch flip-flop is cleared) are transferred to the three memory address registers by command 100 (logic 0.4.1, D12) or command 91 (logic 0.4.1, A4). This transfer also starts the selected memory device. If test memory was selected, the contents of the selected register are transferred to the MBR's at PT 5. If a core memory address was selected, the contents of the selected register are transferred to the MBR's at PT 6. At the same time, command 31 clears the TMAR, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and the step counter, command 101 (logic 0.3.1, B13) clears the operation and index interval registers, and command 163 (logic 0.3.1, B2) clears the branch flip-flop. If the associated sign control flip-flops are set, commands 214 and 216 (logic 0.5.1-2, A22) complement the respective accumulator registers and clear the sign control flip-flops.

3.6.2 *Branch on Full Minus (BFM), Branch on Left Minus (BLM), Branch on Right Minus (BRM)*

The *BFM, BLM,* and *BRM* instructions sense the left and/or right accumulator register sign bits for negative (1) sign. Each instruction requires one PT cycle, and, except for the *sense-for-branch* command, each instruction is accomplished by the same command sequence.

At PT 7, command 42 (logic 0.1.1, D9) transfers the contents of the left MBR to the operation and index interval registers, command 52 (logic 0.1.2, D8) transfers the contents of the right MBR to the step counter, and command 92 (logic 0.4.1, A4) steps the program counter. At PT 9, commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6) clear the A registers, and the branch flip-flop is set if the branch condition specified by the instruction is met. If the *BFM* instruction is being executed and the accumulator register sign bits are 1's, the branch flip-flop is set by command 162 (logic 0.5.2-2, D19). If the *BLM* instruction is being executed and the left accumulator register sign bit is a 1, the branch flip-flop is set by command 165 (logic 0.5.1-2, D21). If the *BRM* instruction is being executed and the right accumulator register sign bit is a 1, the branch flip-flop is set by command 166 (logic 0.5.1-2, D22).

If the branch flip-flop was set at TP 9, then command 93 (logic 0.4.1, C5) transfers the contents of the program counter to the right A register and clears the program counter at TP 11. At TP 0, command 31 (logic 0.1.4, B5) clears MAR 2, and command 154 (logic 0.4.1, D13) transfers the contents of the address register to the program counter. At PT 0 delayed, commands 41 (logic 0.1.1, D9) and 53 (logic 0.1.2, B8) clear the MBR's and command 100 (logic 0.4.1, D12) transfers the con-

tents of the address register to the three memory address registers. If the branch flip-flop was not set, then command 100 is not executed as before, but the contents of the program counter are transferred to the three memory address registers by command 91 (logic 0.4.1, A4). In either case, the transfer command also starts the selected memory device. If test memory was selected, the contents of the selected register are transferred to the MBR's at PT 5. If core memory was selected, the contents of the selected register are transferred to the MBR's at PT 6. At the same time, command 31 (logic 0.1.4, B5) clears the TMAR, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and the step counter, command 101 (logic 0.3.1, B13) clears the operation and index interval registers, and command 163 (logic 0.3.1, B2) clears the branch flip-flop.

3.7 MISCELLANEOUS CLASS

Twelve of the 16 instructions in the miscellaneous class utilize registers in the arithmetic element for their execution. Four of the instructions (*SLR, CSW, ETR,* and *LDB)* perform a unique control or loading function, and eight *(CMM, CDM, CMF, AML, CMR, CDF, CDL,* and *CDR)* perform a logical comparison between the specified word in the memory element and the contents of the accumulator registers. The function of each of these instructions is defined in the following discussions.

3.7.1 *Shift Left and Round (SLR)*

The *SLR* instruction is used to position a 32 number in the accumulator and B registers, and then rounds off the 16 least significant bits so that the number can be processed in the 16-bit accumulator registers. The required number of shifts (to the left) is specified by the address part of this instruction, which usually follows a *Multiply* or a *Divide* instruction. If executed after a *Multiply* instruction, the *SLR* instruction usually specifies 0 shifts because the product is correctly positioned in the accumulator register. However, if executed after a *Divide* instruction, the *SLR* instruction usually specifies 15 shifts because the quotient is contained in the B register (2.2.2.3).

The round-off operation takes into consideration the value of the 17th bit (in the B register sign bit position) and the sign bit of the accumulator register. If both sign bits are 0's, the 16-bit number in the accumulator register remains as it is. If both sign bits are 1's, the accumulator register is complemented twice and no change is effected. If the B register sign bit is a 1 and the accumulator register sign bit is a 0, a 1 is added to the 16-bit number in the accumulator register. If the accumulator register sign bit is a 1 and the B register sign bit is a 0, then the accumulator register is complemented, a 1 is added to the 16-bit number, and the accumulator register is complemented again to restore the number to its original form.

The execution of the *SLR* instruction is similar to that described for the *DSL* instruction (3.4.2) in that it is not indexable, the accumulator register sign bit remains unchanged, and, with each shift, bit 1 of the accumulator register is lost and the sign bit advances into bit 15 of the B register. If zero shifts are specified, this instruction is accomplished during the PT cycle. If one or more shifts are specified, a PT cycle and an arithmetic pause are required.

The initial commands of the instruction cycle which occur at PT 7 are 42, 52, and 92. Command 42 (logic 0.1.1, D9) transfers the contents of the left MBR to the operation and index interval registers, command 52 (logic 0.1.2, D8) transfers the contents of the right MBR to the address register and the step counter, and command 92 (logic 0.4.1, B1) steps the program counter.

If zero shifts are specified by the instruction, the next command occurs at PT 0 of the next PT cycle. At this time, command 31 (logic 0.1.4, B5) clears MAR 2. At PT 0 delayed, commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's and command 91 (logic 0.4.1, A4) transfers the contents of the program counter to the three memory address registers and starts the selected memory device. At PT 1, commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6) clear the A registers, and, if the associated accumulator register sign bits are 1's, commands 16 (logic 0.5.1-2, D21) and 20 (logic 0.5.2-2, C22) complement the respective accumulator B register combinations and sign control flip-flops. If the associated B register sign bits are 1's, commands 87 (logic 0.5.1-3, E7) and 287 (logic 0.5.2-3, E6) pulse the carry 1 lines of the respective adders and set the associated carry storage flip-flops at PT 2.

At PT 5, the contents of the selected test memory register are transferred to the MBR's if test memory was selected, and commands 60 and 260 (logic 0.5.1-2, A18) perform correctional shifts left if commands 87 and 287 resulted in an addition. At PT 6, the contents of the selected core memory register (if a core memory address was selected) are transferred to the MBR's, command 31 clears the test MAR, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and step counter, command 101 (logic 0.3.1, B13) clears the index interval and operation registers, commands 84 (logic 0.5.1-3, A7) and 284 (logic 0.5.2-3, A7) clear the B registers, commands 214 and 216 (logic 0.5.1-2, A22) complement the accumulator registers and clear the associated sign control flip-flops if the sign control flip-flops are set, and commands 66 and 266 (logic 0.5.1-2, B22) record the overflow if overflow occurred.

If one shift is specified by the instruction, the following sequence of commands is applicable after PT 8 of the first PT cycle. At PT 9, the 2-mc sync flip-flop is set by command 138 (logic 0.5.3, D7). At PT 10, the 2-mc operation flip-flop is complemented to the set state (to control the generation of 2-mc operate pulses), the clear pause delay sync flip-flop is set, and the pause flip-flop is set by command 134 (logic 0.2.2, D7) to start the arithmetic pause, which begins after the PT 11 pulse is generated.

At PT 11, the first 2-mc operate pulse is converted into command 73 (logic 0.5.3, D2), which steps the step counter from 1 to 0, sets the clear pause delay flip-flop, clears the 2-mc sync flip-flop, and complements the 2-mc operate flip-flop to the cleared state. The 2-mc operate pulse is also converted into the required *SLR* shift commands. These commands are 2, 5, 81, and 82 for the left arithmetic element and 202, 205, 281, and 282 for the right arithmetic element. Commands 2 (logic 0.5.1-2, E21) and 202 (logic 0.5.2-2, D21) shift accumulator register bits 2 through 15 one bit position to the left, commands 5 and 205 (logic 0.5.1-2, D19) shift the accumulator register sign bits into bit 15 of the B registers, commands 81 (logic 0.5.1-3, E6) and 281 (logic 0.5.2-3, E6) shift the B register sign bits into bit 15 of the accumulator registers, and commands 82 (logic 0.5.1-3, D7) and 282 (logic 0.5.2-3, D7) shift B register bits 1 through 15 one bit position to the left.

The first 2-mc TPD on pulse that occurs during the pause (these pulses being continually generated when the computer is operating) clears the pause, the clear pause delay, and the clear pause delay sync flip-flops to terminate the arithmetic pause. Thus, although the termination of the arithmetic pause is initiated by a 2-mc operate pulse, the actual termination is performed by 2-mc TPD on pulses. The remaining commands, beginning with PT 0 of the second PT cycle, are the same as those described for operation with zero shifts.

If two shifts are specified by the *SLR* instruction, the following sequence of commands is applicable after PT 8 of the first PT cycle. At PT 9, the 2-mc sync flip-flop is set by command 138. At PT 10, the 2-mc operation flip-flop is set, and the pause flip-flop is set to start the arithmetic pause, which begins after the PT 11 pulse.

At PT 11, the first 2-mc operate pulse clears the 2-mc sync flip-flop, sets the clear pause delay sync flip-flop, initiates action to clear the 2-mc operation flip-flop (1.0 μsec required because of inherent delays), and is converted into command 73, which steps the step counter from 2 to 1, and shift commands 2, 5, 81, 82, 202, 205, 281, and 282, which shift the accumulator and B register one bit position to the left. When the second 2-mc operate pulse occurs, the 2-mc operation flip-flop is cleared, command 73 steps the step counter from 1 to 0, and the *SLR* shift commands are executed again.

The first 2-mc TPD on pulse following PT 11 sets the clear pause delay flip-flop, and the second 2-mc TPD on pulse following PT 11 clears the pause, the clear pause delay, and the clear pause delay sync flip-flops to terminate the arithmetic pause. The remaining commands, beginning with PT 0 of the second PT cycle, are the same as those described for operation with zero shifts.

If three shifts are specified by the *SLR* instruction, the following sequence of commands is applicable after PT 8 of the first PT cycle. At PT 9, the 2-mc sync flip-flop is set by command 138. At PT 10, the 2-mc operation flip-flop is set, and the pause flip-flop is set to start the arithmetic pause, which begins after the PT 11 pulse.

At PT 11, the first 2-mc operate pulse clears the 2-mc sync flip-flop, sets the clear pause delay sync flip-flop, and is converted into command 73, which steps the step counter from 3 to 2, and shift commands 2, 5, 81, 82, 207, 205, 281, and 282, which perform one shift left. At the next 2-mc operate pulse, command 73 steps the step counter from 2 to 1, the *SLR* shift commands are executed a second time, and action is initiated to clear the 2-mc operation flip-flop. At the same time, the 2-mc TPD on pulse sets the clear pause delay flip-flop. When the third 2-mc operate pulse is generated, command 73 steps the step counter from 1 to 0, the *SLR* shift commands are executed a third time, and the 2-mc TPD on pulse clears the pause, the clear pause delay, the clear pause delay sync, and the 2-mc operation flip-flops. The remaining commands, beginning with PT 0 of the second PT cycle, are the same as those described for operation with zero shifts.

If four or more shifts are specified by the *SLR* instruction, the following sequence of commands is applicable after PT 8 of the first PT cycle. At PT 9, the 2-mc sync flip-flop is set by command 138. At PT 10, the 2-mc operation flip-flop is set, and the pause flip-flop is set by command 134 to stop the time pulse distributor after PT 11, thus causing an arithmetic pause.

The arithmetic pause begins after PT 11, at which time the 2-mc sync flip-flop is cleared, command 73 reduces the contents of the step counter by 1, and *SLR* shift commands 2, 5, 81, 82, 202, 205, 281, and 282 accomplish the first shift. Command 73 and the shift commands occur simultaneously at a 2-mc rate during the arithmetic pause until the required number of shifts have been executed.

When all but three shifts have been executed, the step counter content is 3. At the next 2-mc operate pulse, command 73 reduces the step counter contents from 3 to 2 and sets the clear pause delay sync flip-flop (to initiate termination of the pause). The *SLR* shift commands are also repeated. When command 73 reduces

the contents of the step counter from 2 to 1, the clear pause delay flip-flop is set (by a 2-mc TPD on pulse), the *SLR* shift commands are executed, and the clear pulse for the 2-mc operation flip-flop is issued. When command 73 steps the step counter from 1 to 0, the last shift for the *SLR* instruction is accomplished and the 2-mc TPD on pulse clears the pause, the clear pause delay, the clear pause delay sync, and the 2-mc operation flip-flops. The remaining commands, beginning with PT 0 of the second PT cycle, are the same as those described for operation with zero shifts.

3.7.2 *Clear and Subtract Word Counter (CSW)*

The *CSW* instruction transfers the contents of the IO word counter to the right accumulator register. The number transferred is one less than the number of words remaining to be processed, is in complement form, and may contain 16 or 17 bits. The *CSW* instruction is not indexable.

This instruction is accomplished during a PT cycle as follows. At PT 7, command 42 (logic 0.1.1, D9) transfers the contents of the left MBR to the operation and index interval registers, command 52 (logic 0.1.2, D8) transfers the contents of the right MBR to the step counter and the address register, and command 92 (logic 0.4.1, B1) steps the program counter. At PT 9, command 157 (logic 0.7.3, A8) clears the CSW gate flip-flop and sets the CSW control flip-flop, and command 210 (logic 0.5.2-2, B21) clears the right accumulator register if 16-bit operation is specified. If 17-bit operation is specified, command 210 also clears the left accumulator sign bit.

At PT 0, the CSW control flip-flop is cleared, and command 31 (logic 0.1.4, B5) clears MAR 2. At PT 0 delayed, commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's, and command 91 (logic 0.4.1, A4) transfers the content of the program counter to the three memory address registers and starts the selected memory device. At PT 1, command 153 (logic 0.7.3, B5) transfers the contents of the IO word counter to the right accumulator register if the IO word counter is not being stepped. If the IO word counter is being stepped at PT 1, then command 153 is executed at PT 5. If 17-bit operation is specified, this command also transfers bit LS of the IO word counter to bit LS of the left accumulator register.

If test memory was selected, the contents of the selected register are transferred to the MBR's at PT 5. If core memory was selected, the contents of the selected core register are transferred to the MBR's at PT 6. At PT 6, command 31 clears the TMAR, command 101 (logic 0.3.1, B13) clears the operation and index interval registers, and command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and the step counter.

3.7.3 *Extract (ETR)*

The *ETR* instruction is used to transfer selected bits of the specified memory word into the accumulator registers. Before this instruction is executed, the accumulator registers are loaded with a mask which specifies which bits of the memory word are to be transferred to the accumulator register. The unmasked accumulator bits contain 0's before and after the execution of the instruction. After the instruction is executed, the masked accumulator bits duplicate the associated bits of the memory word. This instruction can also be used in the opposite sense; that is, the specified memory word can contain the mask, and the accumulator registers can contain the data to be processed. The final accumulator register content will be the same regardless of how the instruction is used.

This instruction is indexable and requires a PT cycle and an OT cycle for execution. The instruction cycle begins at PT 7 with command 42 (logic 0.1.1, D9), which transfers the contents of the left MBR to the operation and index interval registers, command 52 (logic 0.1.2, D8), which transfers the contents of the right MBR to the step counter and the address register, and command 92 (logic 0.4.1, B1), which steps the program counter. At PT 8, the contents of the selected index register are added into the address register if an index register is selected. At PT 11, command 131 (logic 0.3.1, B3) sets the PT-OT flip-flop to terminate the PT cycle and to start the OT cycle.

At OT 0, command 31 (logic 0.1.4, B5) clears MAR 2. At OT 0 delayed, commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's, and command 71 (logic 0.4.1, C11) transfers the contents of the address register to the three memory address registers and starts the selected memory device. At OT 1, commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6) clear the A registers. If test memory was selected, the contents of the selected test memory register are transferred to the MBR's at OT 5. At OT 6, command 31 clears the TMAR, and, if a core memory address was selected, the contents of the selected core memory register are transferred to the MBR's.

The operation within the arithmetic element begins at OT 7. At this time, commands 43 and 51 transfer the contents of the MBR's to the A registers. At OT 9, commands 25 (logic 0.5.1, D4) and 225 (logic 0.5.2, D7) logically multiply the content of the accumulator registers by the content of the A registers. The logical multiplication is effected by a 0 side transfer of the A registers to the accumulator registers, which clears all the associated accumulator register bits.

The PT-OT flip-flop is cleared by command 161 (logic 0.3.1, B4) at OT 11 to start the next PT cycle. At PT 0, command 31 clears MAR 2. At PT 0 delayed,

commands 41 and 53 clear the MBR's, and command 91 (logic 0.4.1, A4) transfers the contents of the program counter to the three memory address registers and starts the selected memory device. If a test memory address was selected, the contents of the selected test memory register are transferred to the MBR's at PT 5. If a core memory address was selected, the contents of the selected core memory register are transferred to the MBR's at PT 6. At the same time, command 31 clears the TMAR's, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and the step counter, and command 101 (logic 0.3.1, B13) clears the operation and index interval registers.

3.7.4 *Load B Registers (LDB)*

The *LDB* instruction transfers information from the specified memory register to the B registers. This instruction is indexable and requires a PT cycle and an OT cycle for execution.

The instruction cycle begins at PT 7 with command 42 (logic 0.1.1, A9), which transfers the contents of the left MBR to the operation and index interval registers, command 52 (logic 0.1.2, B8), which transfers the contents of the right MBR to the step counter and the address register, and command 92 (logic 0.4.1, B1), which steps the program counter. At PT 8, the contents of the selected index register are added into the address register if an index register is selected. At PT 11, command 131 (logic 0.3.1, B3) sets the PT-OT flip-flop to terminate the PT cycle and to start the OT cycle.

At OT 0, command 31 (logic 0.1.4, B5) clears MAR 2. At OT 0 delayed, commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's and command 71 (logic 0.4.1, C11) transfers the contents of the address register to the three memory address registers and starts the selected memory device. If a test memory address was selected, the contents of the selected test memory register are transferred to the MBR's at OT 5. At OT 6, commands 84 (logic 0.5.1-3, A7) and 284 (logic 0.5.2-3, A7) clear the B registers, command 31 clears the TMAR, and, if a core memory address was selected, the contents of the selected core memory register are transferred to the MBR's.

At OT 7, commands 39 and 40 (logic 0.1.1, D9) transfer the content of the MBR's to the B registers. At OT 11, command 161 (logic 0.3.1, B4) clears the PT-OT flip-flop to terminate the OT cycle and to start the next PT cycle.

At PT 0, command 31 clears MAR 2. At PT 0 delayed, commands 41 and 53 clear the MBR's and command 91 transfers the contents of the program counter to the three memory address registers and starts the selected memory device. If a test memory address was selected, the contents of the selected test memory register are transferred to the MBR's at PT 5. If a core memory address was selected, the contents of the selected core memory register are transferred to the MBR's at PT 6. At the same time, command 31 clears the TMAR, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and the step counter, and command 101 (logic 0.4.1, D12) clears the operation and index interval registers.

3.7.5 *Compare Full Words (CMF), Compare Left Half-Words (CML), Compare Right Half-Words (CMR)*

The *CMF, CML,* and *CMR* instructions are used to perform a logical comparison of the specified memory word with the contents of the left and/or right accumulator registers. This comparison is accomplished by transferring the selected memory word into the A registers, complementing the accumulator registers, and then checking the content of the left and/or right registers to determine whether a compare condition exists (fig. 3–7). If the words (in their original form) do not compare, a no-compare signal is developed which steps the program counter by a factor of 1 (in addition to the normal stepping that occurs at PT 7). The extra step causes the program counter to skip the next sequential instruction, thus indicating that the words did not compare. If the words in their original form do compare, then the accumulator register will contain the complement of the A register during the test period, and the check-for-compare pulse will not be converted into a no-compare pulse.

These instructions are indexable, and each requires a PT cycle and an OT cycle for execution. The *CMF* instruction cycle begins at PT 7 with command 42 (logic 0.1.1, D9), which transfers the contents of the left MBR to the operation and index interval register, command 52 (logic 0.1.2, D8), which transfers the contents of the right MBR to the step counter and the address register, and command 92 (logic 0.4.1, B1), which steps the program counter. At PT 8, the contents of the selected index register are added into the address register if an index register is selected. At PT 9, commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6) clear the A registers, and commands 19 (logic 0.5.1-2, B21) and 219 (logic 0.5.2-2, A22) complement the accumulator registers. At PT 11, command 131 (logic 0.3.1, B3) sets the PT-OT flip-flop to terminate the PT cycle and to start the OT cycle.

Command 31 (logic 0.1.4, B5) clears MAR 2 at OT 0. At OT 0 delayed, commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's, and command 71 (logic 0.4.1, C11) transfers the contents of the address register to the three memory address registers and starts the selected memory device. If a test memory address was selected, the contents of the selected test memory register are transferred to the

MBR's at OT 5. If a core memory address was selected, the contents of the selected core memory register are transferred to the MBR's at OT 6. At the same time, command 31 clears the TMAR. At OT 7, commands 43 (logic 0.1.1, D9) and 51 (logic 0.1.2, D8) transfer the contents of the MBR's to the A registers.

At OT 9, command 99 (logic 0.6.2, E4) is issued to check for comparison. If any pair of corresponding bits in the left and right accumulator and A registers are alike, the output from the associated AND circuit will condition the no-compare gate tube. If this condition exists, command 99 becomes a no-compare pulse which is used to step the program counter. At OT 11, command 161 (logic 0.3.1, B4) clears the PT-OT flip-flop to terminate the OT cycle and to start the next PT cycle.

At PT 0, command 31 clears MAR 2. At PT 0 delayed, commands 41 and 53 clear the MBR's, and command 91 (logic 0.4.1, A4) transfers the contents of the program counter to the three memory address registers and starts the selected memory device. At PT 1, commands 21 and 230 clear the A registers. If a test memory address was selected, the contents of the selected test memory register are transferred to the MBR's at PT 5. If a core memory address was selected, the contents of the selected core memory register are transferred to the MBR's at PT 6. At the same time, command 101 (logic 0.3.1, B13) clears the operation and index interval registers, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and the step counter, command 31 clears the TMAR, and commands 19 and 219 complement the contents of the accumulator registers to return them to their original form.

The application of the *CMF* instruction is shown by the following abbreviated program (all numbers are in octal notation):

Memory Location	*Instruction*	*Address*
0.00001	*CAD*	0.01000
0.00002	*CMF*	0.01002
0.00003	*BPX*	To subroutine A
0.00004	*BPX*	To subroutine B
0.01000	0.07035	0.70451
0.01002	0.07025	0.70441

The *CAD* instruction places octal numbers 0.07035 and 0.70451 in the left and right accumulator registers, respectively, and the *CMF* command sequence produces the following:

Since the memory word did not compare with the accumulator register contents, program step 0.0003 will be skipped and the next instruction to be executed will be the *Unconditional Branch (BPX)* to subroutine B.

The command pulse sequence generated during the execution of the *CML* and *CMR* instructions is exactly the same as the command pulse sequence generated during the execution of the *CMF* instruction. The only difference between these three instructions is that command 99 is gated to check the content of the left and/or right arithmetic registers as designated by the instruction. As the designations imply, the *CML* instruction affects only the left half-word and the *CMR* instruction affects only the right half-word.

3.7.6 Compare Difference Full Words (CDF), Compare Difference Left Half-Words (CDL), Compare Difference Right Half-Words (CDR)

The *CDF*, *CDL*, and *CDR* instructions are used to perform a logical comparison of the specified memory word with the contents of the left and/or right accumu-

Time	*Command*	*Left Half-Word*	*Right Half-Word*	*Remarks*
		0.07035	0.70451	Accumulator register contents after *CAD* instruction
PT_19	Complement accumulator registers	1.70742	1.07326	In accumulator register
	Clear A registers	0.00000	0.00000	In A register (cleared)
OT7	Transfer MBR to A registers	0.07025	0.70441	In A register (received from memory location 0.01002)
OT9	Check accumulator and A registers for comparison	No-compare	No-compare	Program counter stepped by 1
PT_21	Clear A registers	0.00000	0.00000	In A register
PT_26	Complement accumulator registers	0.07035	0.70451	In accumulator register

lator register, and to provide the arithmetic differences between the two half-words. The compare operation performed during the execution of these instructions is identical with, and produced by, the same pulse sequence that is generated during the execution of the *CMF, CML,* and *CMR* instructions. After the check for compare is performed, the operands in the accumulator registers are subtracted from the operands in the associated A registers to provide the arithmetic differences for further analysis by subsequent programmmed instructions.

The three compare difference instructions are indexable, and each requires a PT cycle and an OT cycle for execution. Since the command sequence generated between PT 7 and OT 8 time of these instruction cycles is identical with that described for the corresponding portion of the *CMF* instruction, this analysis continues with the commands that occur after OT 8. At OT 9, command 99 (logic 0.6.2, E4) checks for a comparison of the full words or the left or right half-words, depending on which of the three compare-difference instructions is being executed. If any pair of corresponding bits in the left and/or right accumulator and A registers are alike, command 99 will be gated as a no-compare pulse to step the program counter by a factor of 1. At OT 10, commands 64 (logic 0.5.1-2, A1) and 264 (logic 0.5.2-2 A1) initiate the add operation to add the contents of the A registers and the complemented accumulator registers. Since the accumulator registers were complemented at PT 9, this add operation effectively subtracts the original content of the accumulator registers from the A register which contains the data obtained from the memory element. At OT 11, command 161 (logic 0.3.1, B4) clears the PT-OT flip-flop to terminate the OT cycle and to start the next PT cycle.

At PT 1, command 31 (logic 0.1.4, B5) clears MAR 2. At PT 0 delayed, commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's, and command 91 transfers the contents of the program counter to the three memory address registers and starts the selected memory device. At PT 1, the inherent shift right that resulted from the add operation is corrected by commands 2 and 4 (logic 0.5.1-2, E21), 81 (logic 0.5.1-3, E6), and 89 (logic 0.5.1-3, B7) for the left accumulator, and by commands 202 and 204 (logic 0.5.2-2, D21), 281 (logic 0.5.2-3, E6), and 298 (logic 0.5.2-3, B7) for the right accumulator. At the same time, commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6) clear the A registers.

If the add operation resulted in an end carry, the sum is corrected by commands 63 and 263 (logic 0.5.1-2, A17) at PT 2. At PT 5, commands 60 and 260 (logic 0.5.1-2, A18) perform a correctional shift left if an end carry resulted from the preceding add operation and the contents of the specified test memory register are transferred to the MBR's if a test memory address was selected. If a core memory address was selected, the contents of the selected core memory register are transferred to the MBR's at PT 6. At the same time, command 31 clears the TMAR, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and the step counter, command 101 (logic 0.3.1, B13) clears the operation and index interval registers, and commands 66 and 266 (logic 0.5.1-2, B22) record the arithmetic overflow if the condition exists.

3.7.7 Compare Masked Bits (CMM), Compare Difference Masked Bits (CDM)

The *CMM* and *CDM* instructions differ from the *CMF* and the *DCF* instructions in that a mask is used to limit the compare and difference operations to selected bits. The desired mask must be loaded into the B registers before either of these instructions is executed. The instructions involve all bits of each half-word, but before the compare and difference operations are accomplished, the unmasked bits are made to compare. In the difference operation which follows the compare operation in the *CDM* instruction, the unmasked bits become 0's.

The unmasked bits in the accumulator and A registers are made to compare so they will not affect the comparison check of the masked bits. This is accomplished by clearing the unmasked bits in the complemented accumulator register to 0's and setting the unmasked bits in the A register to 1's. The operation consists of transferring the mask to the A registers, complementing the accumulator registers, logically multiplying the accumulator registers by the A register mask (to clear the unmasked accumulator bits), complementing the A register (to set the unmasked bits and clear the masked bits), and then logically adding the specified memory word to the A registers. At the time of the comparison check and the difference operation, the unmasked bits in the accumulator register are 0's and the unmasked bits in the A register are 1's.

These instructions are indexable, and each requires a PT cycle and an OT cycle for execution. At PT 7 of the instruction cycle, command 42 (logic 0.1.1, D9) transfers the contents of the left MBR to the operation and index interval registers, command 52 (logic 0.1.2, B8) transfers the contents of the right MBR to the step counter and the address register, and command 92 (logic 0.4.1, B1) steps the program counter. At PT 8, the contents of the selected index register are added into the address register if an index register is selected. At PT 9, commands 21 (logic 0.5.1, B4) and 230 (logic 0.5.2, B6) clear the A registers, and commands

19 (logic 0.5.1-2, B21) and 219 (logic 0.5.2-2, A22) complement the operands in the accumulator registers. At PT 11, command 131 (logic 0.3.1, B3) sets the PT-OT flip-flop to terminate the PT cycle and to start the OT cycle.

At OT 0, command 31 (logic 0.1.4, B5) clears MAR 2. At OT 0 delayed, commands 41 (logic 0.1.1, A9) and 53 (logic 0.1.2, B8) clear the MBR's, and command 71 (logic 0.4.1, C11) transfers the contents of the address register to the three memory address registers and starts the selected memory device.

At OT 4, the masks are transferred from the B registers to the A registers by commands 88 (logic 0.5.1-3, D6) and 288 (logic 0.5.2-3, C7). At OT 5, the contents of the selected test memory register are transferred to the MBR's if a test memory address was selected, and commands 25 (logic 0.5.1, D4) and 225 (logic 0.5.2, D7) logically multiply the associated operands in the accumulator by the mask contained in the A registers (0 side transfer). The masked bits of the accumulator registers remain as they were, but the unmasked bits are changed to 0's. At OT 6, the masks (in the A registers) are complemented by commands 26 and 226 (logic 0.5.1, B5) so that the unmasked bits in the A registers will now contain 1's, and the masked bits will contain 0's. At the same time, the contents of core memory are transferred to the MBR's if a core memory address was selected. (If a test memory address was selected, the contents of the selected test memory register are transferred to the MBR's at OT 5.) At OT 7, the selected memory word (in the MBR's) is logically added (OR function) to the complemented masks (in the A registers) by commands 43 (logic 0.1.1, A9) and 51 (logic 0.1.2, B8). The masked bits of the selected memory words will appear in the masked bits of the A registers, but the unmasked A register bits will remain set to 1.

Command 99 (logic 0.6.2, E4) is issued at OT 9 to check the contents of the accumulator and A registers. If the operands compare (that is, if each bit in the accumulator registers is the complement of the corresponding bit in the A registers), then a no-compare pulse will not be generated. However, if one or more masked bits do not compare, then a no-compare pulse is generated to step the program counter by a factor of 1.

The command pulse sequence generated during the execution of the *CMM* and *CDM* instructions is identical up to this point. The following paragraphs provide separate discussions of the commands generated during the remainder of these instruction cycles.

If the *CMM* instruction is in process, the sequence of commands after the execution of command 99 is as follows. At OT 11, command 161 (logic 0.3.1, B4) clears the PT-OT flip-flop to terminate the OT cycle and to start the next PT cycle. At PT 0, command 31 clears MAR 2. At PT 0 delayed, commands 41 and 53 clear the MBR's and command 91 (logic 0.4.1, A4) transfers the contents of the program counter to the three memory address registers and starts the selected memory device. At PT 1, commands 21 and 230 clear the A registers. If a test memory address was selected, the contents of the selected test memory register are transferred to the MBR's at PT 5. If a core memory address was selected, the contents of the selected core memory register are transferred to the MBR's at PT 6. At the same time, command 101 (logic 0.3.1, B13) clears the operation and index interval registers, command 77 (logic 0.4.1, B13 and 0.5.3, B2) clears the address register and the step counter, command 31 clears the TMAR, and commands 19 and 219 complement the accumulator registers to return the masked bits to their original form. The unmasked accumulator bits will be set to the 1 state.

If the *CDM* instruction is in progress, the sequence of commands after the execution of command 99 is as follows. At OT 10, commands 64 (logic 0.5.1-2, A1) and 264 (logic 0.5.2-2, A1) initiate the add operation to add the contents of the A registers to the accumulator registers. At OT 11, command 161 clears the PT-OT flip-flop to terminate the OT cycle and to start the next PT cycle.

At PT 0, command 31 clears MAR 2. At PT 0 delayed, commands 41 and 53 clear the MBR's, and command 91 transfers the contents of the program counter to the three memory address registers and starts the selected memory device. At PT 1, the inherent shift right that resulted from the previous addition is corrected by commands 2 and 4 (logic 0.5.1-2, E21), 81 (logic 0.5.1-3, E6), and 89 (logic 0.5.1-3, B7) for the left accumulator, and by commands 202 and 204 (logic 0.5.2-2, D21), 281 (logic 0.5.2-3, E6), and 289 (logic 0.5.2-3, B7) for the right accumulator. At the same time, commands 21 and 230 clear the A registers.

If the addition process resulted in an end carry, the sum is corrected at PT 2 by commands 63 and 263 (logic 0.5.1-2, A17). At PT 5, commands 60 and 260 (logic 0.5.1-2, A18) perform a correctional shift left if an end carry addition was performed, and the contents of the selected test memory register are transferred to the MBR's if a test memory address was selected. At PT 6, the contents of the selected core memory register are transferred to the MBR's if a core memory address was selected. At the same time, command 31 clears the TMAR, command 77 clears the address register and the step counter, command 101 clears the operation and index interval registers, and, if the add operation produced an overflow, commands 66 and 266 record the overflow.

Missing From Original Document

CHAPTER 3
INDEXING

3.1 BASIC ANALYSIS

3.1.1 General

Indexing is the process whereby an instruction address may be modified during the execution of iterative (cyclic) program loops. The modification is accomplished by adding the contents of one of four index registers (Nos. 1, 2, 4, or 5) to the address portion of an instruction word just before the address is transferred out of the address register. The information in the memory register that corresponds to the original, unmodified address, however, is not altered. The right accumulator register can also be used as an index register for certain types of address modification such as table lookup procedures. When so used, the right accumulator is designated index register No. 3.

3.1.2 Operational Characteristics

The circuitry required to perform the indexing function includes, in addition to one of the index registers noted above, the address register and its integrated index adder.

In order to utilize the indexing principle, provision must be made to initially load the index register with a number that is one less than the desired number of iterations. This is accomplished by programming the *XIN* (or *XAC*) instruction whereby the address portion of this instruction word (or the right accumulator register) contains the desired number and operation bits L1-L3 determine which of the four index registers is to be loaded with this number by way of the address register. (The right accumulator, i.e., index register No. 3, may be loaded or modified by so many instructions that no special instruction is required to set it for indexing purposes.) With the index register thus loaded, the programmer may then specify iterative program loop operation utilizing both indexable and nonindexable instructions. The last instruction in such a program loop is the *BPX* instruction.

The indexable instruction in an iterative loop always specifies the index register whose contents are to be added to the address portion of the indexable instruction word. The addition is performed in the index adder which is an integral part of the address register. In this manner, the address register contents are modified by the addition process. The modified address is then used in the subsequent operate cycle of the instruction to obtain the operand from this modified memory address. At the completion of each iterative loop, the *BPX* instruction senses the index register sign bit for a 0 to determine whether the loop is to be repeated. If the condition is met, program control is branched back to the address contained in the address portion of the *BPX* instruction word, which in this case is the memory location of the first instruction in the iterative loop. The *BPX* instruction will also specify, when needed, an index interval in bits L10-L15. These bits are transferred to address register bits R10-R15 in complement form. To obtain a true complement, address register bits LS, RS-R9 are also complemented at the same time. The content of the selected index register is then added to the complemented number in the address register, effectively subtracting or stepping down the index register contents by the stipulated index interval constant. The resultant modified number (if still positive) is then transferred back to the index register awaiting the next pass in the cyclic loop. Since the program branches to the first instruction in the cyclic loop, its address will again be modified; however, the modified address will now reflect the stepped-down index register value. This process will be repeated in accordance with the programmed data until the last iteration is reached. The last iteration will be determined when, during the *BPX* instruction, the index register is sensed and it is found that the sign bit contains a 1. This signifies the no-branch condition and the end of the cyclic loop. Under these conditions, the address portion of the *BPX* instruction is meaningless and the program counter will be stepped by 1 to select the next sequential memory address; the program will then continue in a conventional manner.

The information paths to the indexing circuitry are controlled by the commands associated with the instruction being executed. In this manner, the same circuits are used repeatedly to produce the desired end results. No overlap or duplication of information arises because the computer processes instructions sequentially and in proper time relationship. Consequently, the circuits capable of handling any indexing operation may be represented in a single, simplified illustration as shown in figure 4–3, foldout. This figure contains index register No. 1, the address register and its associated index adder, and the pertinent transfer circuitry between the applicable registers. The commands and other timing pulses that are used in conjunction with the indexing circuitry are

also identified. This includes the reset class instructions, *XIN*, *XAC*, and *ADX*, and the branch class instruction, *BPX*. The illustration should be used in connection with the following text.

3.1.3 Index Register

Index registers Nos. 1, 2, 4, and 5 each contain 17 bits. Since 17-bit arithmetic operation is possible, the right accumulator register can also be considered as a 17 register when used as index register No. 3. The bits of the four physical index registers are designated sign bit, bit 0, and bit 1 through bit 15. (See fig. 4–3.) Actually, only the 16 bits, 0-15, can be used to express a numerical indexing quantity. Thus, the maximum quantity that can be contained is $200{,}000_{(8)}$. The sign bit cannot be considered a part of the numerical quantity since it is not transferred to the address register although a transfer is made to it from the corresponding bit LS of the address register during the index register loading operation. That is, the address register transfers 17 bits of information to the index register although only the last 16 bits of the index are transferred to the address register during the indexing operation. Figure 4–3 reveals that this transfer is not automatic upon the issuance of the proper command because the transfer is contingent on the content of bit LS of the address register. For example, if address register bit LS is a 0, then GT's 14, 15, 16, and 17 are conditioned. If index register No. 1 is selected, command 79 is generated at PT 6 and examines GT 14. The resultant output from GT 14 sets the index register sign bit to the 0 side and also strobes the 16 gates (GT 14) connected to the 1 side of the remaining address register bits RS-R15. If any of these bits is set, the corresponding flip-flops in index register No. 1 will also be set, thus completing the transfer of 17 bits. However, if address register bit LS contained a 1, no transfer to the index register would be possible and the index register would remain in its cleared state. The cleared state of any index register is 2.00000. This is due to the fact that the clear pulse sets the sign bit to the 1 side and the remaining bits to the 0 side. Hence, the natural cleared state, expressed octally, is 2.00000 where the number 2 represents the combination of binary 1 and 0 in the sign bit and bit 0 positions, respectively. The reason for the foregoing bit LS control is to prevent the index register from being loaded with the address register contents of negative 0 (all 17 bits contain 1's) on the last pass of a cyclic loop. Thus, the index register can never be set to a value lower than its cleared state (2.00000).

3.1.4 Index Adder

The numerical contents of the selected index register are used to modify the address portion of the indexable instruction in an iterative loop. The modification process consists of adding the contents of both the index register and the address register in the index adder. Further, the index adder is also used to add the index interval complement to the index register contents, which effectively subtracts the index interval from the number in the index register. The index adder accomplishes both functions by employing parallel additions; that is, the 16 bit positions, 0-15, of the selected index register are added simultaneously and independently to the corresponding 16 bits, RS-R15, of the address register, with provision made to take the resulting individual bit carry pulses into account some time after the initial addition. The index adder is faster than the arithmetic adder and contains a minimum of equipment to develop the speed required to synchronize the indexing operation with the other operations of the computer.

Briefly, FF's RS-R15 of the address register receive information from the corresponding flip-flops, 0-15, of the selected index register (index register No. 1 in fig. 4–3) by means of a complement line. For any of the flip-flops, this complement line is the output of a gate conditioned by the 1 side of the index register flip-flop and pulsed by the appropriate indexing command: command 114 for index register No. 1; command 115 for index register No. 2; command 117 for index register No. 4; command 124 for index register No. 5; and command 212 for the right accumulator register when that register is functioning as index register No. 3. Consequently, a 1 in any index register flip-flop causes the corresponding address register flip-flop to reverse its status when an indexing operation is initiated. A 0 in any index register flip-flop does not affect the status of its associated address register flip-flop. Thus, column by column, the index register contents modify the address register contents in accordance with the rules of binary addition.

If the address register and index register flip-flops for any given bit position each contain a 1, their sum is a 0, and a carry pulse results when the two bits are added. This carry pulse must be taken into account when adding the next-highest-order bit positions. In the parallel add process employed here, addition is performed simultaneously in all 16 bit positions; hence, any resulting carry pulse must be delayed to provide sufficient settling time for those address register flip-flops which were complemented during the initial phase of the addition process. This is accomplished in the following manner. Consider that bit 14 of the index register and corresponding bit R14 of the address register each contain a 1. Under these conditions, GT 3 is conditioned and, when pulsed by command 114, will pass the index register pulse to OR 4. The output from OR 4 follows two paths; it passes through OR 5 to complement the R14 flip-flop and also examines GT 9.

Examination of GT 9 takes place, however, before the R14 FF status is changed to the 0 side of the complement pulse. Since bit R14 originally contains a 1, GT 9 is conditioned and feeds a carry pulse to the 0.5-μsec fixed delay line. Before considering the action of the resultant delay carry pulse on the left adjacent bit position, note that in the add process discussed up to this point the binary addition of 1 and 1 (bits 14 and R14) is satisfied since a 0 is placed in bit R14 with a carry to the left adjacent bit. The foregoing process, although limited to a single bit position in the discussion, is simultaneously executed in all 16 bit positions.

The 0.5-μsec delay allotted the carry pulse is required to enable any of the address register flip-flops, which were complemented in the initial add process, to settle down. This is necessary to prepare a flip-flop for a subsequent complement operation. In the previous example, the resultant delay carry pulse is then passed through OR 3 to the complement line of the left adjacent bit as well as through OR 9 to the fast carry gate designated GT 8. To understand the propagation of both types of carry pulses (delay and fast carry) through the address register, assume that bit RS is the left adjacent bit (in practice this will be R13) and that it contains a 1. The delay carry pulse originating from GT 9 of bit R14 complements the bit RS flip-flop and simultaneously inspects the fast carry gate tube (GT 8). Because of the inherent delay in changing the status of the RS flip-flop from a 1 to a 0, the gate tube (GT 8) is conditioned and transmits a fast carry pulse to examine GT 6 and complement bit LS. The pulse from GT 8 is termed fast carry because it must be made available to complement bit LS before bit RS changes its status to a 0 to decondition GT 8.

The foregoing was intended to provide the basic mechanics of operation, particularly with respect to the inherent delay of flip-flops that dictate the use of delay and fast carry lines. To better understand the overall operation, an example is presented involving the addition of two numbers. For this example, assume that index register No. 1 and the address register in figure 4–3 constitute complete 4-bit registers. Assume also that index register No. 1 contains denary number 3(0011) and that the address register contains denary number 7(0111). The contents in each of the subject registers for the corresponding bits, therefore, are:

	Sign Bit	*Bit 0*	*Bit 14*	*Bit 15*
Index register No. 1	0	0	1	1
	Bit LS	*Bit RS*	*Bit 14*	*Bit 15*
Address register	0	1	1	1

The contents are added in the following sequential order:

a. At PT 8 +0.1 μsec, command 114 senses each index register gate. Gate tubes 3 and 4 associated with bits 14 and 15 will be conditioned due to the presence of a 1 in those bits.

b. Each of the conditioned gates feeds a pulse to corresponding circuits OR 4 and OR 6 in the address register. The OR 4 output is simultaneously fed to OR 5 and GT 9. Similarly, the OR 6 output is applied to OR 7 and GT 11.

c. The gates, GT 9 and GT 11, for address register flip-flops R14 and R15 are conditioned due to a 1 in these flip-flops, thereby supplying carry pulses to their respective 0.5-μsec fixed delays.

d. The output pulse from OR 5 and OR 7 complement flip-flops R14 and R15, changing their status from a 1 to a 0. The flip-flops settle to their new condition in approximately 0.5 μsec. Reviewing, it will be seen that up to this point the process reflects the addition of 1 and 1 in two of the four bit positions with a resultant 0 placed in address register bits R14 and R15 and a carry started in each of their respective delay lines. When the flip-flops settle to their 0 states, the carry gates, GT 9 and GT 11, are deconditioned; however, the timing of the carry pulses from these gates already in the carry line is such that the pulses are available to perform the following function.

e. The delay carry pulses from the 0.5-μsec delay circuits of bits R14 and R15 are fed simultaneously to their respective left adjacent bit circuits. For bit R15, the delay carry pulse is fed to OR 5 and OR 10; for bit R14, the delay carry pulse is fed to OR 3 and OR 9. Therefore, the delay carry pulse from bit R15 passes through OR 10 and examines GT 10. Since the R14 flip-flop now contains a 0, GT 10 is deconditioned and no pulse is passed. Coincident with this action, the R15 delay carry pulse also passes through OR 5 to complement bit R14 again (the first time was noted in step d), setting it back to the 1 state in approximately 0.5 μsec.

Simultaneous with the above action, the delay carry pulse from bit R14 passes through OR 9 and examines GT 8 associated with the 1 side of bit RS. Since this flip-flop was originally loaded with a 1 and nothing has happened to change its status, the gate is conditioned and passes a fast carry pulse to OR 1 and OR 8 of the left adjacent bit LS. Coincidentally, the R14 delay carry pulse also passes through OR 3 to complement bit RS, changing its status from a 1 to a 0 in approximately 0.5 μsec. Although this action deconditions GT 7 and GT 8, the fast carry pulse

from GT 8 is already in the line and performs the following function.

f. The fast carry pulse from GT 8 of bit RS passes through OR 8 and examines GT 6 associated with the 1 side of bit LS. Since this flip-flop was initially loaded with a 0 and nothing has happened to change its status, GT 6 is deconditioned and cannot pass a pulse. However, the fast carry pulse from GT 8 of bit RS passes through OR 1 to complement bit LS, changing its status from a 0 to 1 in approximately 0.5 μsec.

g. At the completion of the last step there are no pulses left in any of the carry lines; hence, the address register flip-flops should contain the sum of the original contents of the index and address registers; that is,

0011 (denary 3) index register

0111 (denary 7) address register

Sum 1010 (denary 10) in address register

This may be verified by referring to the final states of the flip-flops in the previous steps as follows:

step d — R15 contains a 0.

step e — R14 contains a 1 and RS contains a 0.

step f — LS contains a 1.

3.2 APPLICATIONS OF INDEXING

3.2.1 Cyclic Loops

A cyclic loop is developed when the same sequence of instructions must be used many times in the processing of certain types of data located in sequential memory locations. Only indexable instructions can be used for this purpose.

3.2.1.1 Programmed Cyclic Loop

Any one of the index registers may be used for a data cyclic loop except index register No. 3 (right accumulator register). The right accumulator register, when it is used for indexing as index register No. 3, cannot be used for executing *The Branch and Index (BPX)* instruction and, therefore, cannot be used directly for cyclic data processing. However, it can be used to reset one of the index registers, which is then used in a cyclic loop.

To illustrate the advantages of a cyclic loop produced by indexing, two programs are described below which produce the same end result. The instruction listing for these programs is tabulated in table 4–2. Both programs provide for the addition of 10 pieces of data which are stored in consecutive memory locations 200 to 209. In each case, the sum of these numbers is then stored in memory location 300. For the purpose of discussion, denary numbers are used, but it should be remembered that the computer deals only in binary notation.

TABLE 4–2. COMPARISON OF A CYCLIC AND A NONCYCLIC PROGRAM

PROGRAM STEP	WITHOUT INDEXING INSTRUCTION WORD			WITH INDEXING INSTRUCTION WORD				
	MEMORY LOCATION	OPERATION	ADDRESS	MEMORY LOCATION	INDEX REGISTER	OPERATION	INDEX INTERVAL	ADDRESS
1	100	*CAD*	209	99	1	*XIN*		8
2	101	*ADD*	208	100		*CAD*		209
3	102	*ADD*	207	101	1	*ADD*		200
4	103	*ADD*	206	102	1	*BPX*	1	101
5	104	*ADD*	205	(Repeat 101 and 102 nine times)				
6	105	*ADD*	204	103		*FST*		300
7	106	*ADD*	203					
8	107	*ADD*	202					
9	108	*ADD*	201					
10	109	*ADD*	200					
11	110	*FST*	300					

The program without the indexing feature starts with the instruction in memory address 100. (Refer to table 4–2.) This is the *Clear and Add (CAD)* instruction and calls for a transfer operation to be performed on the contents of memory address 209. The operation consists of transferring the data in address 209 to the right accumulator register through the memory buffer register, the right A register, and the right adder circuits.

The next program step, which is in memory address 101, calls for an arithmetic addition to be performed on the contents of memory address 208. The contents of address 208 are accordingly transferred to the right accumulator register and are added to the data transferred from address 209. The program steps continue in this manner until the sum of the contents of memory addresses 209 through 200 appears in the right accumulator register. The sum is then stored in memory address 300. In this performance, the add process is repeated nine times, and 22 memory addresses are used, 11 for storing the instructions and 11 for storing the data utilized in the arithmetic operations.

The program which incorporates the indexing feature uses the *Reset Index Register (XIN)* and the *Branch and Index (BPX)* instructions to provide a cyclic loop (table 4–2). The instruction commands and their time sequences are shown separately in tables 4–3 and 4–4. The applicable tables should be referred to in the detailed analysis that follows.

TABLE 4–3. *RESET INDEX REGISTER (XIN)* INSTRUCTION ANALYSIS

PULSE TIME	COMMAND NUMBER	COMMAND
PT 7	42	*Left memory buffer register bits L1-L15 to operation register; bit LS to address register*
	47	*Memory parity count*
	52	*Right memory buffer register to address register and step counter*
	92	*Add 1 to program counter*
PT 8+	—	*Clear memory No. 1 controls (cleared internally)*
PT 10+	55	*Parity check* — No OT cycle
PT 0	31	*Clear memory No. 2 controls*
PT 0+	91	*Program counter to memory address registers and test memory address register. Start selected memory.*
PT 1	41	*Clear left memory buffer register*
	53	*Clear right memory buffer register*
PT 4	113	*Clear index register No. 1* — Conditional: Only the selected register is cleared.
	116	*Clear index register No. 2*
	118	*Clear index register No. 4*
	125	*Clear index register No. 5*
PT 6	79	*Address register to index register No. 1* — Conditional: Transfer to selected register occurs only if bit LS of address register contains a 0.
	78	*Address register to index register No. 2*
	95	*Address register to index register No. 4*
	96	*Address register to index register No. 5*
	77	*Clear address register*
	101	*Clear operation register*
		Clear index interval register
	31	*Clear test memory address register*
		Core memory to left memory buffer register
		Core memory to right memory buffer register

TABLE 4–4. *BRANCH AND INDEX (BPX)* INSTRUCTION ANALYSIS

PULSE TIME	COMMAND NUMBER	COMMAND
PT 7	42	*Left memory buffer register bits L1-L15 to operation register; bit LS to address register*
	47	*Memory parity count*
	52	*Right memory buffer register to address register and step counter*
	92	*Add 1 to program counter*
PT 8+	—	*Clear memory No. 1 controls (cleared internally)*
PT 9	21	*Clear left A register*
	230	*Clear right A register*
	170	*Set branch flip-flop*
	164	*Test index register No. 1 S bit for 0* } Conditional: Test performed on selected register only
	174	*Test index register No. 2 S bit for 0*
	119	*Test index register No. 4 S bit for 0*
	126	*Test index register No. 5 S bit for 0*
PT 10+	55	*Parity check*
PT 11	93	*Program counter to right A register and clear program counter. Conditional: branch flip-flop is set*
		No OT cycle
PT 0	31	*Clear memory No. 2 controls*
PT 0+	100	*Address register to memory address registers* } Conditional on branch flip-flop being set
	154	*Address register to program counter* (if branch FF is cleared, execute command 91 to transfer program counter to MAR's)
PT 1	41	*Clear left memory buffer register*
	53	*Clear right memory buffer register*
PT 2	103	*Index interval complement to address register*
PT 3	114	*Index register No. 1 to address register* } Conditional: Transfer from selected register only
	115	*Index register No. 2 to address register*
	117	*Index register No. 4 to address register*
	124	*Index register No. 5 to address register*
	212	*Right accumulator register to address register. Carry in address register.*
PT 4	113	*Clear index register No. 1* } Conditional: Only the selected register is cleared
	116	*Clear index register No. 2*
	118	*Clear index register No. 4*
	125	*Clear index register No. 5*
PT 6	77	*Clear address register*
	101	*Clear operation register*
		Clear index interval register

TABLE 4–4. *BRANCH AND INDEX (BPX)* INSTRUCTION ANALYSIS (cont'd)

PULSE TIME	COMMAND NUMBER	COMMAND	
	78	*Address register to index register No. 2*	Conditional: Transfer to selected register occurs only if bit LS of address register contains a 0.
	79	*Address register to index register No. 1*	
	95	*Address register to index register No. 4*	
	96	*Address register to index register No. 5*	
	163	*Clear branch flip-flop*	
	31	*Clear test memory address register*	
		Core memory to left memory buffer register	
		Core memory to right memory buffer register	

The *XIN* instruction word specifies, by bits L1 to L3, which one of the four physical index registers is to be used in a cyclic loop. If bits L1 to L3 are 0, no index register is specified. In this example, index register No. 1 is specified; therefore, bits L1 and L2 contain 0's and L3 contains a 1.

At PT 7, the *XIN* instruction word, bits L1-L15, are transferred from the left memory buffer register to the operation register which was cleared at PT 6 of the previous instruction cycle. (The PT 6 pulse clears all of the registers which are to receive specific bits of the instruction word.) At the same time, bit LS of the left memory buffer and bits RS-R15 of the right memory buffer register are transferred to the address register, and a 1 is added to the program counter to specify the address of the next instruction.

At PT 6, the address portion of the *XIN* instruction word is transferred from the address register to the designated index register, and the address and operation registers are cleared for the next instruction. The address part of the *XIN* instruction specifies a number which is one less than the number of repetitions that are to take place. If 10 numbers are to be added, as in this example, the process of addition is repeated 10 times, once by the *CAD* instruction which constitutes the second step of the outlined program (refer to table 4–2) and nine times by the repetitive cyclic process. In this example, the index register is loaded with the number 8 so that 9 repetitive loops will be performed.

After the *XIN* instruction in memory address 99 is executed, the *Clear and Add (CAD)* instruction in memory address 100 is brought from memory. (Refer to table 4–2.) Since no index register is specified for the *CAD* instruction, the contents of memory address 209 are then transferred to the memory buffer register, then to the A registers, and finally through the adders to the accumulator registers. The next program instruction (*ADD* in memory address 101) specifies index register No. 1, the contents of which are added to memory address 200. Index register No. 1 contains the number 8, which, when added to address 200 in the address register, produces a new address, 208. The *ADD* instruction then transfers the operand contained in the modified address 208 to the A registers, adding this operand to the contents of address 209 which were previously transferred to the accumulator registers. The next program step brings the instruction in memory address 102 into the memory buffer registers. This is the *Branch and Index (BPX)* instruction, and specifies index register No. 1 as well as an index interval of 1.

The command sequence table for the *Branch and Index (BPX)* instruction (table 4–4) shows that the *BPX* instruction is transferred from the memory buffer registers to the operation register and the address register at PT 7, and a 1 is added to the program counter. At PT 9, the specified index register sign bit is tested for 0. If the index register sign bit were 1, no branch would take place; but since the specified index register (No. 1) contains a value of 8, the sign bit is 0; therefore, the branch condition is met and the branch flip-flop is set. At the same time, the left and right A registers are cleared before the the program counter contents are transferred to the right A register at PT 11. A branch operation takes place to the memory address specified by the *BPX* instruction (in this case, memory address 101). The program counter is cleared at PT 11, and the *BPX* address is transferred from the address register to the program counter and to the memory address register at PT 0+. At PT 1, the memory buffer registers are cleared in anticipation of the transfer of the instruction from memory address 101. At PT 2, the complement of the index interval is placed in the address register. The contents of the index register are transferred to the address register at PT 3 and added

to the complement of the index interval. In terms of the denary notation used in this discussion, this entails algebraically subtracting the 1 in the index interval from the 8 in the index register, thereby reducing the number 8 to a 7. At PT 4, the index register is cleared, and at PT 6, the stepped-down contents of the address register are transferred to the index register. The operation and address registers, as well as the branch flip-flop, are cleared for the new instruction from memory address 101, which has been transferred to the memory buffer register at PT 6. Since one add process has been executed, the index register contains a value of 7. The eight steps that follow are repetitions of the above procedure; that is, the program performs an add process, goes on to the next memory address, and then branches back to the add process at memory address 101. Since the contents of the index register are reduced by 1 and then added to the contents of the address register at each branch operation, address 200 is first added to 8, then to 7, then to 6, etc., until 0 is reached in the index register. Thus, the contents of address 209 (transferred by the *CAD* instruction) and addresses 208 through 200 are added consecutively in the accumulator registers.

As long as the index register sign bit contains a 0, the repetitive branch process continues. When the index register is reduced to the cleared state (a 1 in the sign bit), no further branch of program control can occur. Since the program counter is increased by 1 by command 91 for each instruction, the next instruction is obtained from memory address 103. This is the *FULL STORE (FST)* instruction which stores the sum of all the data in memory address 300.

The performance of this cyclic program requires 17 memory addresses, 5 for storing the instructions and 12 for storing the data.

Although the cyclic program takes more time than the noncyclic program (because two instructions, *BPX* and *ADD*, are necessary for each addition process), it uses only 17 memory addresses, as opposed to 22 required by the noncyclic program. Since core memory storage capacity is the prime factor in the execution of the operational program, the additional time needed by a cyclic program is not considered a waste.

3.2.1.2 Indeterminate Cyclic Loop

A cyclic loop may comprise indeterminate repetitive operations, the actual number of which is unknown to the programmer. This occurs when the number of repetitions to be performed depends upon the amount of data that is transferred from an IO device. The number of data words actually transferred from an IO device must be calculated by the computer. Table 4–5 tabulates a short program to illustrate how an index register may be reset from the right accumulator register after data has been transferred from a status drum to the computer. In this program, the right accumulator register resets the index register with a number that represents the number of words that have been transferred and upon which a common operation such as *ADD*, *MUL*, etc., must be performed.

Since the number of words to be read into memory is unknown, the drum is programmed by an *RDS* instruction to read 8,192 words. The IO interlock is set when the *RDS* instruction is given. This instruction places the complement of 8,192 into the IO word counter. This number is immediately stepped by 1 and then increased by 1 for each word transferred into memory from the Drum System. Therefore, the IO word counter will normally contain a number that is one more than the actual number of words transferred from the Drum System. The drum transfer operation is in process as long as the

TABLE 4–5. PROGRAM: RESET INDEX REGISTER FROM RIGHT ACCUMULATOR REGISTER

PROGRAM STEP	MEMORY ADDRESS	INDEX REGISTER	INSTRUCTION WORD OPERATION	ADDRESS
1	019		*RDS*	(8192)
2	020		*BSN* (IO Interlock)	020
3	021		*CAD*	030
4	022		*CSW*	
5	023		*ADD* 10	(8191)
6	024		*BFZ*	
7	025		*SUB* 10	031
8	026	1	*XAC* 10	
9	027	Program upon which repetitions are to be performed		
	030	0.00000 0.00000		
	031	0.00000 0.00001		

IO interlock is set. This condition is sensed for by the *BSN* instruction in memory address 020 which branches on itself as long as the condition is satisfied. This branching operation continues until the drum makes one revolution, at which time a disconnect signal is generated by the drum, clearing the IO interlock. At this time, the program falls through to the *CAD* instruction which clears both the right and left accumulators. The *CSW* instruction, in memory address 022, then transfers the contents of the IO word counter to the cleared right accumulator register. The program continues with the *ADD* instruction in memory address 023, which transfers the number 8191 from the address register to the right A register and adder circuits where it is added to the right accumulator register contents previously transferred from the IO word counter. Since the right accumulator contents are in complement form, they are subtracted from 8191. The resultant difference in the right accumulator represents the actual number of words that have been transferred. The *BFZ* instruction in memory location 024 is then executed to determine whether any word(s) transfer has been made. Assuming that a word transfer has been made, the program falls through to the *SUB* instruction in memory location 025. This instruction effectively subtracts a 1 from the number contained in the right accumulator. Hence, the final contents in the right accumulator register represent a number that is one less than the number of words transferred. The *XAC* instruction, in memory location 026, then transfers the right accumulator contents to index register No. 1 which is to take part in the iterative loop operation starting program step 9. Thus, the requirement of loading an index register with a number that is one less than the desired number of repetitions is satisfied. The timing sequence and commands involved in the *XAC* instruction are shown in table 4–6. At PT 7, the *XAC* instruction is brought from the left and right memory buffer registers to the operation and address registers. The instruction is decoded in the instruction matrix, and index register No. 1 is specified. Since the address portion of this instruction is meaningless, the address register is cleared at PT 9. At PT 10, the contents of the right accumulator register are transferred to the address register and from there to index register No. 1 at PT 6, thereby resetting the index register from the right accumulator register. The remainder of the commands in this instruction are ignored since they are not relevant to this discussion.

3.2.2 Table Lookup Procedure

The indexing process has many applications in the preparation of a program. The procedure given here describes one of these, called table lookup procedure, which affords a means for quickly obtaining tabular material from core memory storage.

Any one of the four index registers or the right

TABLE 4–6. *RESET INDEX REGISTER FROM RIGHT ACCUMULATOR REGISTER (XAC)* INSTRUCTION ANALYSIS

PULSE TIME	COMMAND NUMBER	COMMAND
PT 7	42	*Left memory buffer register bits L1-L15 to operation register; bit LS to address register*
	47	*Memory parity count*
	52	*Right memory buffer register to address register and step counter*
	92	*Add 1 to program counter*
PT 8+	—	*Clear memory No. 1 controls (cleared internally)*
PT 9	77	*Clear address register*
PT 10+	55	*Parity check*
	212	*Right accumulator register to address register. Left accumulator register bit LS to address register conditional on a 1 in bit L12.*
		No OT cycle
PT 0	31	*Clear memory No. 2 controls*
PT 0+	91	*Program counter to memory address registers and test memory address register. Start selected core memory.*
PT 1	41	*Clear left memory buffer register*
	53	*Clear right memory buffer register*

TABLE 4–6. *RESET INDEX REGISTER FROM RIGHT ACCUMULATOR (XAC)* INSTRUCTION ANALYSIS (Cont'd)

PULSE TIME	COMMAND NUMBER	COMMAND	
PT4	113	*Clear index register No. 1*	Conditional: Only the selected register is cleared
	116	*Clear index register No. 2*	
	118	*Clear index register No. 4*	
	125	*Clear index register No. 5*	
PT 6	79	*Address register to index register No. 1*	Conditional: Transfer to selected register occurs only if bit LS of address register contains a 0.
	78	*Address register to index register No. 2*	
	95	*Address register to index register No. 4*	
	96	*Address register to index register No. 5*	
	77	*Clear address register*	
	101	*Clear operation register*	
		Clear index interval register	
	31	*Clear test memory address register*	
		Core memory to left memory buffer register	
		Gore memory to right memory buffer register	

accumulator register can be used for indexing when tabular material stored in the core memory is to be used. The table to be referred to may be a table of trigonometric functions, a logarithmic table, or the like. These tables are stored in the core memory in such a manner that the memory address for a specific variant of a function differs from the value of the variant it contains by a constant, arbitrary address prefix. This arrangement in the case of trigonometric sines is exemplified in table 4–7. In this case, the sine values of an angle are located in memory addresses which always differ from the angle values themselves by the arbitrary address prefix 100.

As an illustration of table lookup procedure, suppose that the range of an object is expressed by c sin b, where c is a distance and b is an azimuth angle. Assume that c is 50 miles and b is 4 degrees. The table lookup procedure is used to find the numerical value of sin b so that the expression may be evaluated by a simple multiplication process. If the value of the angle is the result of a previous computation, it will appear in the right accumulator register; the right accumulator register is therefore used as index register No. 3. A *Clear and Add (CAD)* instruction which specifies 100 as its operand address, and No. 3 as the index register, is programmed for indexing purposes. This instruction adds the value of the angle (4 degrees) to the operand address (100); the result (104) is sent to the memory address register. The result is that the numerical value of the sine of 4 degrees is contained in the accumulator register at the completion of the *CAD* instruction. It remains only to program a *Multiply (MUL)* instruction specifying as the operand the address in which c (50 miles) is contained. At the completion of the multiply operation, the evaluation of c sin b will appear in the right accumulator register.

TABLE 4–7. MEMORY ADDRESSES FOR STORAGE OF TABLES OF FUNCTIONS

MEMORY ADDRESS	CONTENTS
100	sin 0°
101	sin 1°
102	sin 2°
103	sin 3°
104	sin 4°
105	sin 5°
.	.
.	.
.	.
190	sin 90°

PART 5
SELECTION ELEMENT

CHAPTER 1
INTRODUCTION

1.1 FUNCTION OF SELECTION ELEMENT

The selection element of the Central Computer supplements the instruction control element in the execution of six specific instructions *(BSN, PER, TOB, TTB, SEL,* and *SDR)*. The instruction control element completely decodes the instruction and generates all the commands necessary for instruction execution. However, the *BSN, PER, TOB, TTB, SEL,* and *SDR* instructions can perform a similar operation on one of many associated circuits. The selection element was designed to perform the function of selecting the variation called for by the instruction code. The four general functions of the selection element, from which a variation may be selected, are:

a. Sense for one of many conditions *(BSN)*
b. Operate control circuits and electromechanical devices *(PER)*
c. Test a specific bit (or bits) of a memory word *(TOB* and *TTB)*
d. Select an IO device *(SEL* and *SDR)*

(See 4.5, Part 1, Ch. 4, for general theory of operation of selection element.)

1.2 BLOCK DIAGRAM ANALYSIS

1.2.1 General

The selection element and its relationship to the other Central Computer elements was described in Part 1, Chapter 4. This part presents a brief discussion of the selection element logic circuit groups and their relationship (Ch 1), and a detailed analysis of the operate codes (Ch 2), sense codes (Ch 3), *TOB* and *TTB* instructions (Ch 4), and the *Select* and *Select Drum* codes (Ch 5).

It may be seen from figure 5–1 that the selection element consists of the index interval register, the *PERSELBSN* matrix, test circuits, *PER* circuits, *BSN* circuits, and *SEL* gates. During execution of *BSN, PER, TOB, TTB, SDR,* and *SEL* instructions, the index interval portion (L10-L15) of the instruction word is transferred from the left memory buffer register to the index interval register. The d-c levels generated by the index interval register are used to condition the *PERSELBSN* matrix. The *PERSELBSN* matrix decodes these levels and produces levels to condition the output line which reflects the contents of the index interval register. This

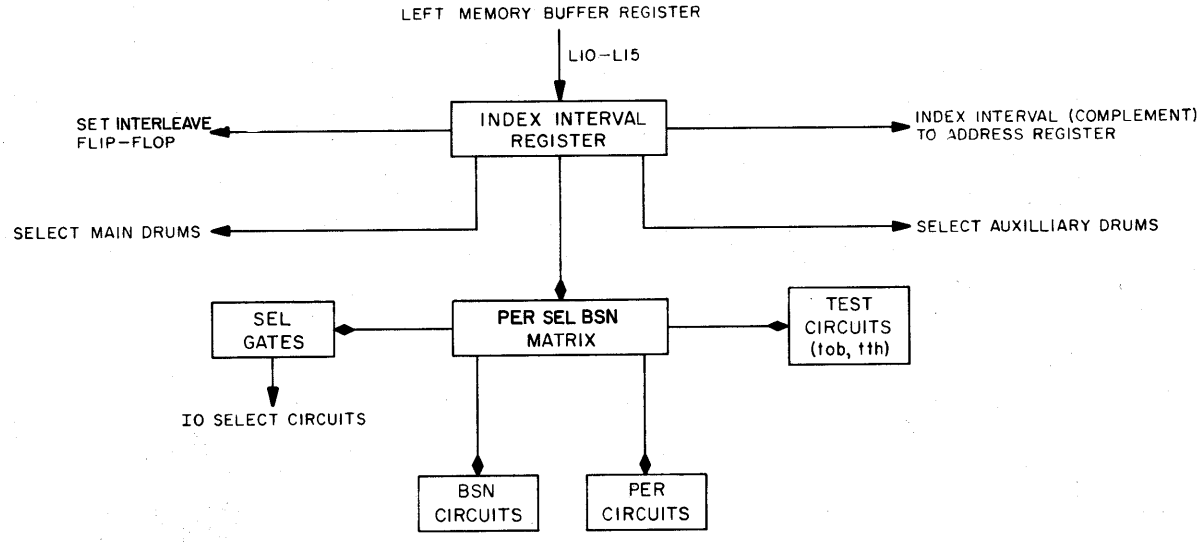


Figure 5–1. Selection Element, Simplified Block Diagram

selected output is used to condition the circuit associated with the desired sense unit, device to be operated, word bit(s) to be tested, or IO device to be selected.

1.2.2 Index Interval Register

The index interval register is composed of six flip-flops and their associated output gates. The index interval portion of the instruction word (L10-L15) is loaded into this register from the memory element by way of the left memory buffer register. The significance of the index interval bits depends upon the instruction word. The d-c levels generated by the flip-flops are used to drive the *PERSELBSN* matrix and also to control the action of the associated gates.

The outputs of the index interval register gates perform the following functions:

a. Select a main drum field (see Ch 5)
b. Select an auxiliary drum field (see Ch 5)
c. Transfer the index interval (complement) to the address register during the execution of the *BPX* instruction (see Part 4)
d. Set interleave flip-flop (see Part 6)

1.2.3 *PERSELBSN* Matrix

The d-c output levels of the index interval register flip-flops drive the *PERSELBSN* matrix. The matrix produces only one positive output level at a time. This single output is determined by the instruction code. To obtain a positive output level for a specific index interval, the corresponding levels from the index interval register are mixed in an AND circuit. Each possible combination of the index interval bits has an AND circuit assigned to it. Because some of the index interval codes are utilized by several instructions, each output line of the *PERSELBSN* matrix conditions several gates. The selection of which gate is to be used for a given instruction is made by the command pulses originating in the instruction control element. During execution of an instruction, only the conditioned gate which pertains to that instruction is sensed. The circuits controlled by the output levels of the *PERSELBSN* matrix are discussed in Chapters 2, 3, 4, and 5.

CHAPTER 2
OPERATE CODES

2.1 INTRODUCTION

The AN/FSQ-7 Combat Direction Central contains a number of electrical and electromechanical devices which are operated under the control of the Central Computer. These devices (called operate units) are activated by control signals originating in the instruction control element during the execution of the *Operate (PER)* instruction. The index interval bits (L10-15) of the *PER* instruction word determine which of the devices is to be operated. The selection element selects the device by decoding the index interval bits in the *PERSELBSN* matrix. At OT 9 of the *PER* instruction, command 104 is issued by the instruction control element. The command pulse is routed to the selection element, where it senses all the gates associated with the various electronic and electromechanical devices. The d-c output of the *PERSELBSN* matrix conditions the gate associated with the designated device; all other gate tubes associated with the other devices are deconditioned. The command pulse is passed by the conditioned gate to perform the specified operation.

2.2 OPERATE CODES, GROUP 1

The operate codes in this group are used to set flip-flops in the Central Computer. The circuitry involved is shown in figure 5–2.

2.2.1 Condition Lights, *PER* $(01)_8$ through $(04)_8$

Four condition lights are provided. They are located on the duplex maintenance console and may be used to indicate the course of the program.

The d-c output level generated by the *PERSELBSN* matrix conditions one of the gates associated with the four condition lights. Command 104 of the *PER* instruction senses GT's 1 through 4 at OT 9 (fig. 5–2.) Since only one of the four gates is conditioned, only one gates the command pulse. The output pulse of the selected gate sets the condition-light flip-flop specified by the index interval portion of the instruction word. The resultant d-c level from the 1 side of the flip-flop turns on the selected condition light.

2.2.2 Intercommunication, *PER* $(10)_8$ through $(13)_8$

The function of operate codes $(10)_8$ through $(13)_8$ is to notify the standby computer associated with the Combat Direction Central that a specific condition exists in the active computer. Conversely, the standby computer may execute any of the above instructions, thus notifying the active computer that a specific condition exists in the standby computer.

Assume that the active computer of the Combat Direction Central is executing one of these *PER* instructions. Accordingly, the *PERSELBSN* matrix generates a d-c level which conditions the gate associated with the operate unit specified by the index interval portion of the instruction word. All four gates (7 through 10) shown in figure 5–2 are sensed at OT 9 of the *PER* instruction, but only the conditioned gate passes the sensing pulse. The generated pulse from the active computer sets a flip-flop in the standby computer. The relays associated with these lights are energized when the standby computer is marginal checking these lights. These relays prevent the intercommunication between the active and standby computers.

2.2.3 Clear IO Interlock, *PER* $(27)_8$

The circuits used to execute the *PER* $(27)_8$ instruction are shown in figure 5–2. This *PER* instruction provides a means of clearing the IO interlock flip-flops.

The d-c level which results from the decoding performed by the *PERSELBSN* matrix conditions GT 11. This gate is sampled at OT 9 of the *PER* $(27)_8$ instruction by command 104. Its output pulse clears the IO interlock flip-flops.

2.2.4 Lock Address Counter, *PER* $(75)_8$

If a *PER* $(75)_8$ instruction is executed before an *RDS* or a *WRT* instruction, the IO address counter is prevented from being stepped during the subsequent IO operation. As soon as the IO operation is completed, the lock-address-counter circuit is returned to its original status, thus allowing the IO address counter to be stepped during succeeding IO operations. The effect of executing a *PER* $(75)_8$ instruction before an *RDS* instruction is to cause all the words which are read from the selected input device to be placed in one register of the memory element. The address of this memory register is the address contained in the IO address counter before the *PER* $(75)_8$ instruction was executed. Similarly, the effect of executing a *PER* $(75)_8$ instruction before a *WRT* instruction is to cause the contents of one register of the memory element (the register specified by the contents of the IO address counter) to be written many times in succession by the output device.

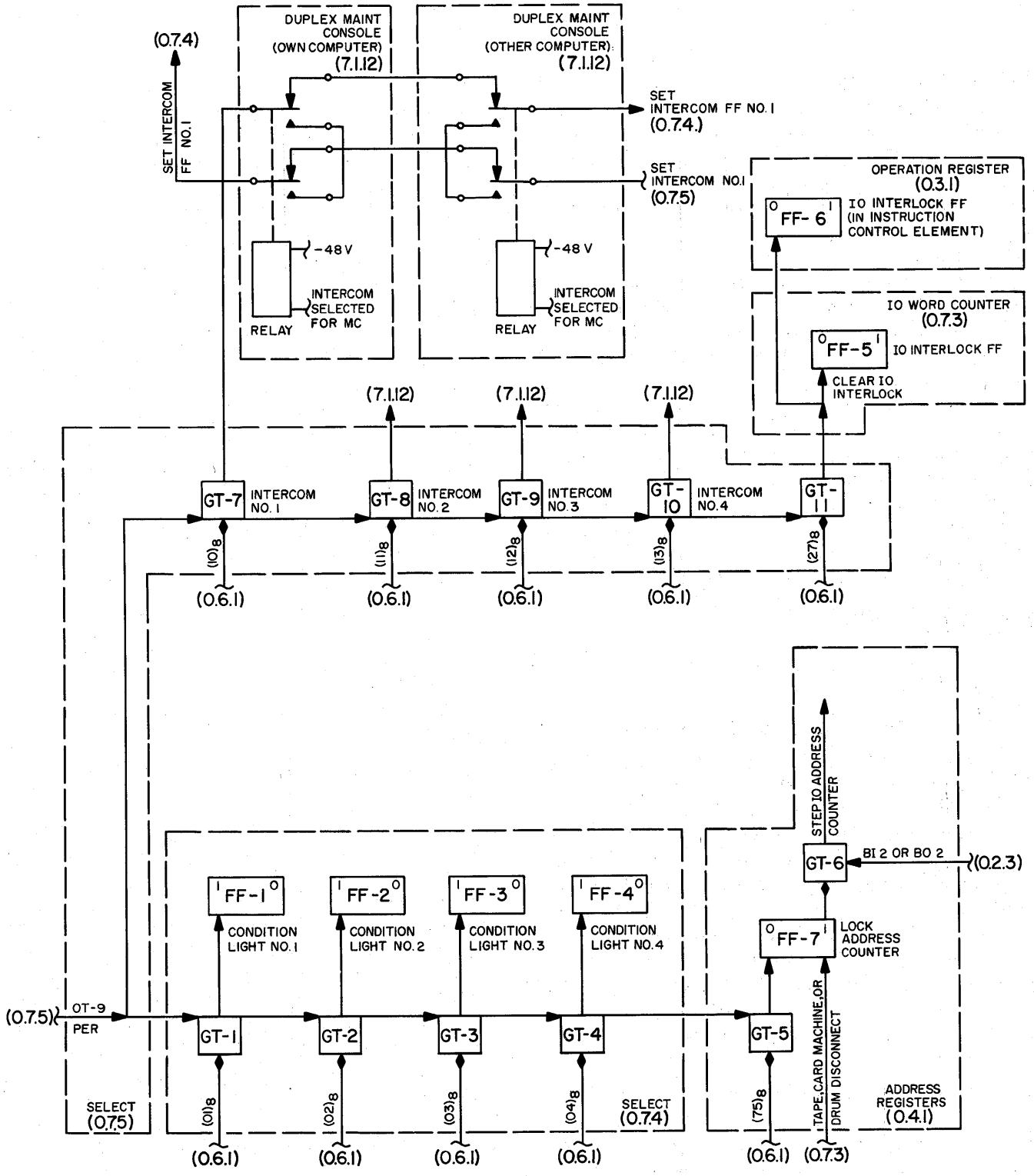


Figure 5–2. Operate Circuits, Group 1, Simplified Logic Diagram

The method used to accomplish the desired results is illustrated in figure 5–2. At OT 9 of the *PER* $(75)_8$ instruction, GT 5, conditioned by *PERSELBSN* matrix output line $(75)_8$, passes a pulse to clear the lock-IO-address-counter flip-flop. Consequently, GT 6 is deconditioned and will not gate a pulse when sensed at BI 2 or BO 2, and the contents of the IO address counter remain unchanged. As soon as the IO process is terminated (by an IO disconnect pulse), the lock-IO-address counter flip-flop is set. The d-c level generated by the 1 side of the lock-address-counter flip-flop conditions GT 6. Therefore, at BO 2 or BI 2 of the succeeding IO operations, the gate tube will pass a pulse to step the IO address counter.

2.3 OPERATE CODES, GROUP 2

Four operate codes are provided to control operations peculiar to the magnetic tape units:

a. *PER* $(70)_8$: backspace (tapes)
b. *PER* $(71)_8$: rewind (tapes)
c. *PER* $(72)_8$: write EOF (tapes)
d. *PER* $(67)_8$: set prepared (tapes)

Since only one tape adapter is provided for the control of four tape drives, the computer program must specify which tape drive is to be affected before any of the above *PER* instructions can be executed. For a detailed analysis of the tape system, refer to Part 9. The circuits of the selection element which are used to execute the *PER* $(70)_8$, *PER* $(71)_8$, *PER* $(72)_8$, and *PER* $(67)_8$ instructions are shown in figure 5–3.

As noted in the figure, the output pulse generated by GT 1, GT 2, or GT 3 performs three functions:

a. Sets IO interlock FF 1, located in the IO element.
b. Sets the IO interlock flip-flop which is located in the instruction control element.
c. Transmits a control pulse to the tape adapter unit.

The control pulse that is supplied to the tape adapter is routed to the selected tape drive to perform the specified operation. When the specified operation has been completed, the selected tape drive transmits a pulse (through the tape adapter unit) to set the disconnect-IO interlock-sync flip-flop (FF 2) in the IO element to notify the computer that the operation has been performed. This flip-flop gates the next 2-mc oscillator pulse to clear the IO interlock (in the IO element) (fig. 5–3) and to clear the disconnect-IO-interlock-sync flip-flop. The d-c level generated by the 0 side of the IO interlock flip-flop is used to condition GT 5. Gate 5 is repeatedly sensed by TP 11 or 2-mc pulses. The output pulse gated by GT 5 clears the IO interlock flip-flop located in the instruction control element.

The operations described above are common for the execution of either the *PER* $(70)_8$, *PER* $(71)_8$, or *PER* $(72)_8$ instruction. The other characteristics of each of these instructions, as well as those of the *PER* $(67)_8$ instruction, are discussed in the following paragraphs.

2.3.1 Backspace Tapes, *PER* $(70)_8$

The backspace operation moves the tape backward for one record (a block of information). As shown in figure 5–3, the index interval portion of the *PER* $(70)_8$ instruction, after being decoded by the *PERSELBSN* matrix, conditions GT 3. At OT 9, the command 104 pulse of the *PER* instruction senses GT's 1, 2, 3, and 4. Gate 3 gates the pulse, but the other gates do not because they are deconditioned.

The output pulse from GT 3 goes to the tape adapter unit, where it activates circuits which cause the tape to be moved back one record. The tape is stopped at the beginning of the record. While this operation is in progress, the IO interlock is on.

2.3.2 Rewind Tapes, *PER* $(71)_8$

The rewind operation moves the tape on the selected tape drive back to its starting or load point. The circuits of the selection element which are used to initiate a rewind operation are shown in figure 5–3. The d-c level generated by the *PERSELBSN* matrix conditions GT 2. The output pulse of GT 2 is used to set the IO interlock flip-flops and to initiate the rewind operation in the selected tape drive. When the rewind operation is actually started, the tape adapter unit transmits a disconnect signal to the IO element to clear the IO interlock.

2.3.3 Write End-of-File, *PER* $(72)_8$

The *PER* $(72)_8$ instruction is used to write a 1-word control record on the selected tape to indicate the end of a file area (used to indicate the end of a group of records).

The circuits of the selection element which are used to initiate a write-end-of-file operation are shown in figure 5–3. The d-c level generated by the *PERSELBSN* matrix conditions GT 1. The output pulse of GT 1 is used to set the IO interlock flip-flops and initiate the write-end-of-file operation in the tape adapter unit. At the completion of the write-end-of-file operation, the tape adapter unit transmits a disconnect signal to the IO element to clear the IO interlock.

2.3.4 Set Prepared (Tapes), *PER* $(67)_8$

The *PER* $(67)_8$ instruction, which clears the not-in-file-area flip-flop, is used to place the tape drive unit in a prepared condition. Once cleared, the flip-flop will remain cleared until an end-of-file control word is either read or written.

The circuits of the selection control element which are used to execute the *PER* $(67)_8$ instruction are shown in figure 5–3. The d-c level generated by *PERSELBSN*

matrix output line $(67)_8$ conditions GT 4. Gate 4 is sensed at OT 9 of the *PER* instruction by command 104. The resulting output pulse is transferred to the selected tape adapter unit to clear the not-in-file-area flip-flop.

2.4 OPERATE CODES, GROUP 3

The operate codes discussed under this heading all use thyratron relay drivers to perform the specified operation. These codes perform operations in the line printer and the card punch and start and stop marginal checking excursions.

The operate circuits for all the codes in this group are identical. A typical circuit is shown in figure 5–4. During the execution of one of these instructions, the *PERSELBSN* matrix conditions the appropriate gate tube. At OT 9 of the *PER* instruction, these gates are sampled by command 104. The gate that is conditioned passes a pulse to trigger a single-shot multivibrator, the output of which fires a thyratron relay driver. The relay driver output is applied to its associated control circuit in the line printer, the card punch, or the Power System, where it causes the designated operation to be performed.

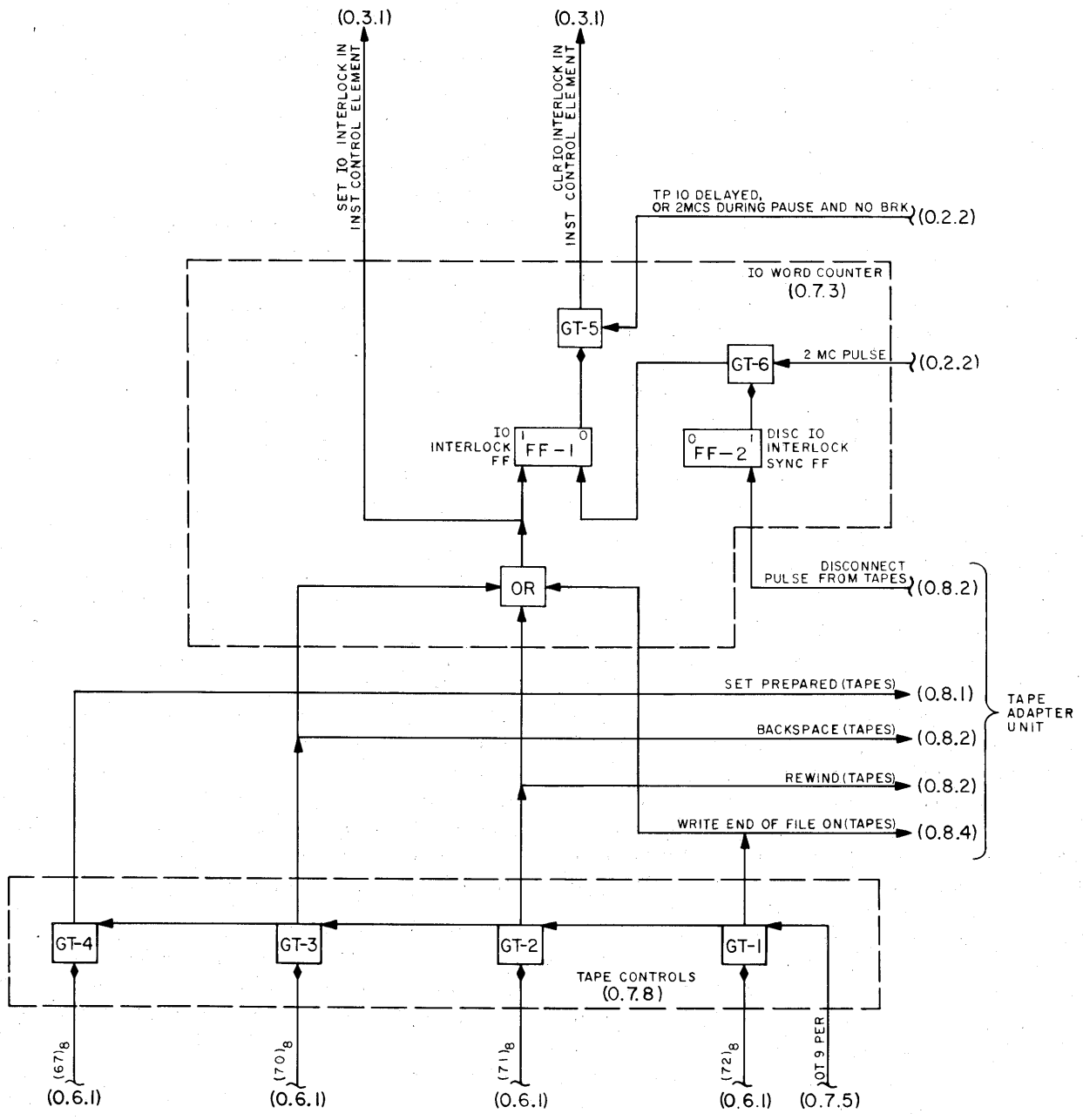


Figure 5–3. Operate Circuits, Group 2, Simplified Logic Diagram

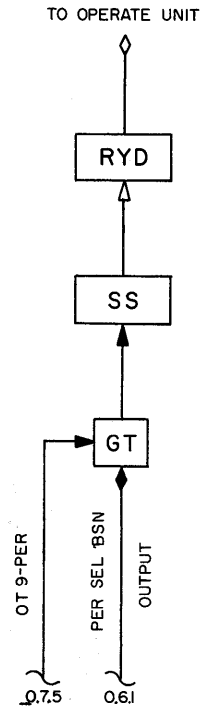


Figure 5–4. Typical Operate Circuit, Group 3, Simplified Logic Diagram

2.4.1 Line Printer Control Hubs 1-10, *PER* $(51)_8$ *through* $(62)_8$

The line printer has 120 print wheels which may be used for printing operations. However, only 64 of these are at any one time to print out computer data. The format or configuration of the printed data is determined by the wiring of a control panel located in the printer. The control panel (plugboard) is wired so that when one of the 10 control-panel exit hubs is specified by a *PER* $(51)_8$ through $(62)_8$ instruction (before a *WRT*), the printer will print out information which is arranged in the predetermined configuration. This control feature can cause the printer to group and/or rearrange bits of data in a number of prescribed ways; examples are listed below:

a. Print 64 bits in 64 consecutive columns.
b. Print groups of bits (e.g., 5 bits to a group).
c. Transpose right and left words.
d. Transpose bits.

The 10 control-panel exit hubs which may be specified by this group of instructions are provided in addition to the hubs normally used during a line printer operation. The additional hubs may be used whenever a special configuration (or variation from normal format) is desired. One *PER* $(51)_8$ through $(62)_8$ instruction (preceding a *WRT*) will allow one line of data to be printed in the specified configuration. If the same configuration is desired for more than one line of information, it is necessary to repeat (program) a *PER* $(51)_8$ through $(62)_8$ instruction for each line to be printed.

2.4.2 Card Punch *PER* $(73)_8$ and $(74)_8$

During normal card-punch operation, two binary data words are simultaneously punched into columns 17-80 of each of the 12 rows of a standard IBM card. However, since it is often desirable to identify the cards of a program deck, identification information can also be punched into columns 1-16 of each card. This control is accomplished by use of the *PER* $(73)_8$ and *PER* $(74)_8$ instructions. The *PER* $(73)_8$ instruction sets up control in the card punch so that the information normally punched into columns 17-32 of the next card will be punched into columns 1-16 (identification field). The identification information punched in columns 1 through 16 of the first card may be punched into columns 1-16 of subsequent cards (together with data in columns 17 through 80) by executing a *PER* $(74)_8$ instruction before processing each subsequent card. Execution of a *PER* $(74)_8$ instruction sets up control in the card punch so that the identification information punched on the preceding card will be read, fed back, and punched into the next card as data is being punched into columns 17 through 80.

2.4.3 Marginal Checking, *PER* $(21)_8$, $(22)_8$, and $(23)_8$

The *PER* $(21)_8$, *PER* $(22)_8$, and *PER* $(23)_8$ instructions are used to control the automatic marginal checking facilities of the AN/FSQ equipment. Automatic marginal checking is performed by a program called the marginal checking (MC) executive program acting in conjunction with a test routine.

For each control line to be checked, the MC executive program will contain a control word (stored in the test memory register) which specifies:

a. Voltage
b. Equipment
c. Circuit group
d. Line
e. Amount of excursion
f. Duration of test
g. Type of automatic restart to be performed.

A *PER* $(21)_8$ instruction is used to initiate the marginal checking (MC) test by selecting a particular control line (or lines) and applying a voltage excursion to the associated circuits. Selection of the desired controls is accomplished by the use of an MC control word which is preloaded into the test register by the MC executive program. The MC control word controls thyratron relay drivers which are energized by the *PER* $(21)_8$ instruction. The contents of the test register are decoded by a relay matrix which controls the marginal checking controls. After the MC voltage has been ap-

plied to the selected line, a test routine is run to determine the reliability of the circuits under test. After the test is completed, the MC executive program will generate a *PER* $(22)_8$ or *PER* $(23)_8$ instruction to terminate the marginal checking excursion in the duplex or simplex equipment, respectively. The MC executive program then selects the next line to be tested and repeats the routine described above. The process is repeated until all lines that can be selected by the MC executive program are tested.

2.5 OPERATE CODES, GROUP 4

Group 4 of the operate codes is used to provide conditioning levels to 22 identical operate gates (fig. 5–5) which are located in the selection element. The output pulses from these gates actuate one of 22 control circuits (depending on the code used) that are contained in other systems of the AN/FSQ-7 equipment. A brief summary of the function of each of these control circuits is presented in the following paragraphs.

2.5.1 Area Discriminator, *PER* $(17)_8$ and $(20)_8$

Under normal operating conditions, target track assignment is initiated manually by using a light gun. The light gun, which is sensitive to blue light of a certain minimum intensity, is directed at one particular point by the display console operator. When it is desired to initiate a track for a particular target, pertinent information is manually inserted into an appropriate group of switches and the light gun is pointed at the radar data display on a display console. When the particular display is intensified, pertinent identification information which is to be associated with manually assigned track number is transferred to the Drum System for processing.

The above process can be programmed and done automatically by using the area discriminator. Where the light gun only covers a particular spot, the area discriminator takes in the entire display scope. However, it can only detect the target (radar) data which has been increased in intensity. The *PER* $(20)_8$ instruction sets up control circuits in the Display System so that radar identification information detected by the area discriminator during a 2.54-second display cycle will be transferred to the manual inputs drum so that the computer can assign track numbers to it. The *PER* $(17)_8$ instruction is actually a spare code which is provided for future expansion.

2.5.2 Situation Display Cameras, *PER* $(31)_8$ through $(34)_8$

There are two situation display cameras provided with the AN/FSQ-7: one for computer A, and one for computer B. The camera associated with the active computer is used to photograph processed situation display information as it is displayed. This may be done manually or under program control using the *PER* $(31)_8$ and *PER* $(32)_8$ instructions. (The *PER* $(33)_8$ and $(34)_8$ codes are provided for future expansion.)

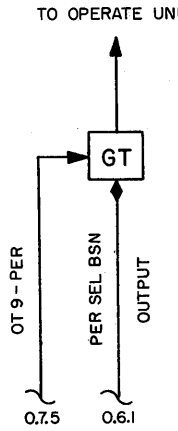


Figure 5–5. Typical Operate Circuit, Group 4, Simplified Logic Diagram

The camera may be operated in two modes: I and II. What type of information will fall into mode I or mode II is predetermined by the program. If mode I operation is selected, only mode I information is displayed and photographed. The same holds true for mode II. Under program control, *PER* $(31)_8$ selects mode I and *PER* $(32)_8$ mode II operation.

2.5.3 Start Digital Display, *PER* $(35)_8$ and $(36)_8$

The digital displays present new information upon request only; therefore, it is necessary for the program to initiate a digital display cycle whenever it is desired to update this information. The old information, which is displayed on 5-inch digital display tubes, is first erased and then replaced by updated information.

The digital data messages are prepared in the Central Computer System and stored in the digital display drum field. The *PER* $(35)_8$ instruction causes this stored information to be transferred to the digital display element (in sequence) for display. The *PER* $(36)_8$ instruction is provided for expansion purposes.

2.5.4 Test Pattern Generator Operate Codes

a. *PER* $(24)_8$, Select Program Pattern Generator *LRI*

b. *PER* $(25)_8$, Select Program Pattern Generator *XTL*

c. *PER* $(64)_8$, Start Program Pattern Generator *GFI*

d. *PER* $(63)_8$, Start *GFI* Continuous Pattern Generator and Start *LRI* and *XTL* Computer Pattern Generator

e. *PER* $(65)_8$, *LRI* sync, *XTL* sync, *GFI* az

f. *PER* $(66)_8$, *LRI* data, *XTL* data, *GFI* target

The purpose of this group of *PER* instructions is to provide data and controls which are utilized during a computer maintenance program which is specifically designed to test selected *LRI, XTL,* and *GFI* channels.

2.5.4.1 LRI Channel Test

To perform this test, the TEST switch on the test pattern generator (*TPG*) must be set to the MODE II position. Mode II operation is differentiated into two types (type 1 and type 2), either of which may be selected by command pulses generated by the Central Computer. Type 1 operation is initiated at OT 9 by execution of a *PER* $(24)_8$ instruction. The *PER* OT 9 pulse prepares the *TPG* for message generation; the message itself is then composed by *PER* $(65)_8$ OT 9 and *PER* $(66)_8$ OT 9 command pulses from the Central Computer. Thus, type 1 operation is characterized by complete computer control over the *TPG* message.

Type 2 operation is accomplished by the execution of a *PER* $(63)_8$ instruction. A *PER 63* OT 9 pulse initiates a message, the composition of which is determined by the settings of the BIT SELECTION switches on the *TPG*. Each *PER* $(63)_8$ OT 9 pulse initiates a single message. The Central Computer can cause resumption of type 1 operation at any time by executing a *PER* $(24)_8$ instruction.

2.5.4.2 XTL Channel Test

Execution and control of this test (both types of operation) is identical with that for *LRI* with one exception. A *PER* $(25)_8$ instruction is used instead of the *PER* $(24)_8$.

2.5.4.3 GFI Channel Test

The GFI channel test involves three types of operation. The first two (types 1 and 2) are identical with those for LRI and XTL except that a *PER* $(64)_8$ instruction is used in place of the *PER* $(24)_8$ or *PER* $(25)_8$. Type 2 operation may be changed to type 3 by depressing the START pushbutton on the TPG.

In type 3 operation, message generation is completely under manual control. Type 3 operation may be changed to type 1 by executing a *PER* $(64)_8$ instruction.

2.5.5 Scan Counter, *PER* (76 and 77)$_8$ (0.7.7)

Input radar data is processed by the Central Computer, and the resultant radar data messages are stored on nine radar data drum fields. The nine fields contain data showing successive target positions. The data on eight of these nine fields is displayed during a given pass of the operational program. During this time, the ninth field is receiving new data from the computer. The contents of the scan counter in the Drum System determine which eight of the nine fields are read during any operational program cycle. To ensure proper synchronization between the Drum and Central Computer Systems, the contents of the scan counter are controlled by the computer. The *PER* $(77)_8$ instruction steps the scan counter. The *PER* $(76)_8$ instruction is provided as a maintenance tool. It is used to reset the scan counter at any point during the stepping action initiated by the *PER* $(77)_8$ as desired by the operator.

2.5.6 Set Inactivity, *PER* (05)$_8$

The *PER* $(05)_8$ instruction is employed during the execution of the air defense operational program to ensure that the sequential execution of the program is progressing normally. The *PER* $(05)_8$ is actually a test for inactivity in the active computer. It will sound an alarm whenever time pulse distributor action is interrupted for at least 1/32 of a second or in the event of Central Computer hangup. The instruction is programmed so that it is executed during the time between consecutive 8-second clock pulses. This is accomplished by programming an interative loop which executes a *PER (05)*$_8$ every eight seconds. The following discussion (2.5.6.1 and 2.5.6.2) of both types of inactivity test refers to figure 5–6.

2.5.6.1 Inactivity Test for Computer Hangup

Any previous inactivity test will have been terminated by clearing inactivity counters A and B. Therefore, at the beginning of an inactivity test, flip-flops A and B will both be cleared (A = 0 and B = 0). The first OT 9 *(PER 05)* pulse passes through GT 4 to set A and clear B (A = 1, B = 0). The first 8-second clock pulse (real-time clock) passes through GT 2 to set B and complement A to clear (A = 0, B = 1). The second OT 9 *(PER 05)* pulse passes through GT 4 to set A and clear B (A = 1, B = 0). The second 8-second clock pulse passes through GT 2 to set B and complement A to clear (A = 0, B = 1). This sequence of events will continue as long as the computer is operating normally. However, if an OT 9 *(PER 05)* pulse does not occur at the proper time (because of computer hangup in an idle loop or an indefinite IO pause), the next 8-second clock pulse will pass through GT 5, OR 1, and GT 6 to clear A and complement B to the clear state (this terminates the test). The same 8-second clock pulse will also pass through OR 3 to sound the computer audible alarm, set the inactivity flip-flop, and sense GT 12. If the INACTIVITY switch is on, the pulse passes through GT 12 and senses GT's 13 and 14. If the auto branch control flip-flop is set, GT 13 passes a pulse to GT 17, and GT 14 passes a pulse to perform an alarm disconnect and sense GT's 15 and 16. Gate 17 receives a conditioning level through the ACTIVE-INACTIVE switch in the active computer, so when it is sensed by the output of GT 13 it passes a pulse to set alarm 1 in the other (inactive) computer. If the STOP-BRANCH flip-flop switch is set to STOP when GT's 15 and 16 are sensed by the output of GT 14, an alarm stop will result; if the flip-flop is set to BRANCH, an alarm branch will

be performed. (Discussion of the action taken when an inactivity alarm is generated is purposely limited because the alarm circuits involved are covered in detail in Part 8.) The action described above is summarized in table 5–1.

2.5.6.2 Test for TPD Inactivity

During execution of the programmed iterative loop, 32-pps clock pulses are continually sensing GT's 3 and 7. Since one of these gates is always conditioned during an inactivity test (see table 5–1), a pulse will be passed through OR 2 to sense GT 9. If the INACTIVITY switch is on, the pulse passes through GT 9 and examines GT 10. If the STOP-BRANCH switch is on BRANCH, the sense-TPD-inactivity pulse is generated and senses GT 11. If the inactive TPD flip-flop is set, the pulse passes through GT 11, performs an alarm branch and an alarm branch delayed, and passes through OR 3. (The action of pulses passing through OR 3 is described in 2.5.6.1.) If the inactive TPD flip-flop is not set, GT 11 is not conditioned; therefore, no 32-pps pulse will be passed and no alarm action will be taken.

The sense-TPD-inactivity pulse also sets (to 1) the inactive TPD flip-flop; this conditions GT 11. If the time pulse distributor is active, the flip-flop is reset (to 0) by the next TP 11, thus removing the conditioning level from GT 11 before it is sensed by another 32-pps pulse and thereby preventing alarm action.

2.5.7 Reset Inactivity, *PER* $(06)_8$

The *PER* $(06)_8$ instruction is used to terminate an inactivity test. It does this by generating a pulse which

TABLE 5–1. SUMMARY OF COMPUTER HANGUP INACTIVITY TEST

PULSE SEQUENCE	EFFECT ON STATUS OF INACTIVITY COUNTER A	EFFECT ON STATUS OF INACTIVITY COUNTER B	RESULTING ACTION
(Start of Test)	0	0	
OT 9 (*PER* 5)	1	0	
8-sec clock pulse	0	1	
OT 9 (*PER* 5)	1	0	
8-sec clock pulse	0	1	
8-sec clock pulse	0	0	Sound audible alarm and terminate test

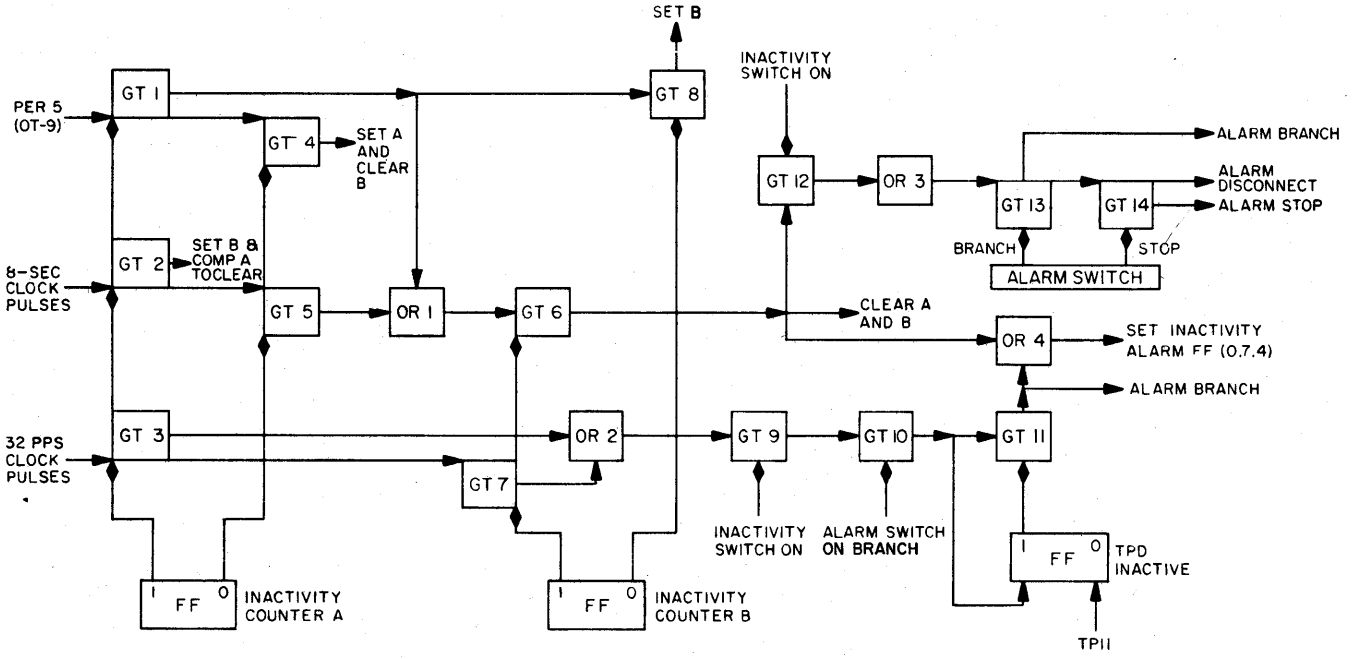


Figure 5–6. Inactivity Test, PER $(05)_8$ Instruction, Simplified Logic Diagram

Missing From Original Document

CHAPTER 3
BRANCH AND SENSE CODES

3.1 INTRODUCTION

The *Branch and Sense (BSN)* instruction enables the Central Computer System to determine the presence or absence of numerous conditions which may exist in AN/FSQ-7 Combat Direction Central. The condition to be sensed for is specified by the index interval portion of the instruction word. If the particular condition exists, the computer branches control of the program to the instruction word whose core memory location is specified by the address portion of the *BSN* instruction. If the particular condition does not exist, executing the *BSN* instruction does not affect the sequential execution of the computer program.

The *BSN* instruction utilizes a PT and an OTA cycle. The OTA cycle is used to provide time for the *PERSELBSN* matrix to stabilize. After the operations which are common to every instruction have been executed by the computer, the index interval portion of the instruction word is transferred from the left memory buffer register to the index interval register. The d-c levels generated by the index interval register are used to condition the *PERSELBSN* matrix. The *PERSELBSN* matrix decodes the index interval portion of the instruction word and conditions the output line which reflects the contents of the index interval register. This selected output is used to condition the circuit associated with the desired sense unit. (A sense unit is an electronic or electromechanical device which may be sensed by the computer, and the desired sense unit refers to that electronic or electromechanical device specified by the index interval portion of the instruction word.) The desired sense unit participates in the execution of the *BSN* instruction.

Branch control for the *BSN* instruction is shown in figure 5–8. The d-c level of each of the 46 input lines represents the presence or absence of a particular condition. In actual practice, only one input line may be activated at any particular time. The activated input line is associated with the sense unit specified by a particular *BSN* instruction. The +10V level of the activated line is transferred through OR circuits to condition GT 1. Therefore, when GT 1 is sensed at TP 8, it generates an output pulse which sets the sense-sync flip-flops. The d-c output level from the 1 side of the sense-sync flip-flop conditions GT 2.

At OT 9 of the *BSN* instruction, the instruction control element issues command 140. This pulse is routed to the selection element. Because the particular sense condition is present, the command 140 pulse is returned to the instruction control element where it sets the branch flip-flop. (See fig. 5–8.) The d-c level generated by the 1 side of the branch flip-flop conditions GT's 3 and 4. The sense-sync flip-flop is cleared at TP 10. Gate 3 gates a pulse at OT 11, transferring the contents of the program counter to the right A register (command 93). Gate 4 gates an output pulse at TP 0, transferring the contents of the address register to the program counter (command 154). The address register contains the address portion of the *BSN* instruction, specifying the core memory address of the next instruction.

Thus, when the sense condition is present, the computer branches control of the program, and the core memory address of the next instruction is specified by the *BSN* instruction. If the sense condition is not present, the branch flip-flop will not be set, GT 3 will not pass the command 140 pulse, and the computer continues its program in the normal manner.

The sequence of events resulting in the branch operation discussed above is analyzed in the following paragraphs. For this discussion, the sense codes have been divided into three groups on the basis of similarity of operation and circuitry.

3.2 SENSE CODES, GROUP 1

The sense codes discussed in this section sense for the existence of various conditions in the equipment as a means of reporting equipment status. No action is taken other than a branch of program control if the condition sensed for is present. The circuits involved are shown in figure 5–9, foldout.

3.2.1 Sense Switches, *BSN* $(21)_8$ through $(24)_8$

The four SENSE/ACTIVE switches are 2-position switches located on the duplex maintenance console. When the Central Computer System executes a *BSN* instruction which senses a SENSE/ACTIVE switch, for a particular setting, the circuits shown in figure 5–9 (foldout) are utilized. Four *BSN* instruction codes are provided, one for each SENSE/ACTIVE switch. If a switch is in the ACTIVE position when examined by the computer, a branch of program control occurs; if the switch is in the INACTIVE position, no branch occurs and the computer continues its program sequentially.

During the execution of this instruction, the *PERSELBSN* matrix generates a d-c level which partially conditions the AND circuit associated with the SENSE/ACTIVE switch specified by the index interval portion

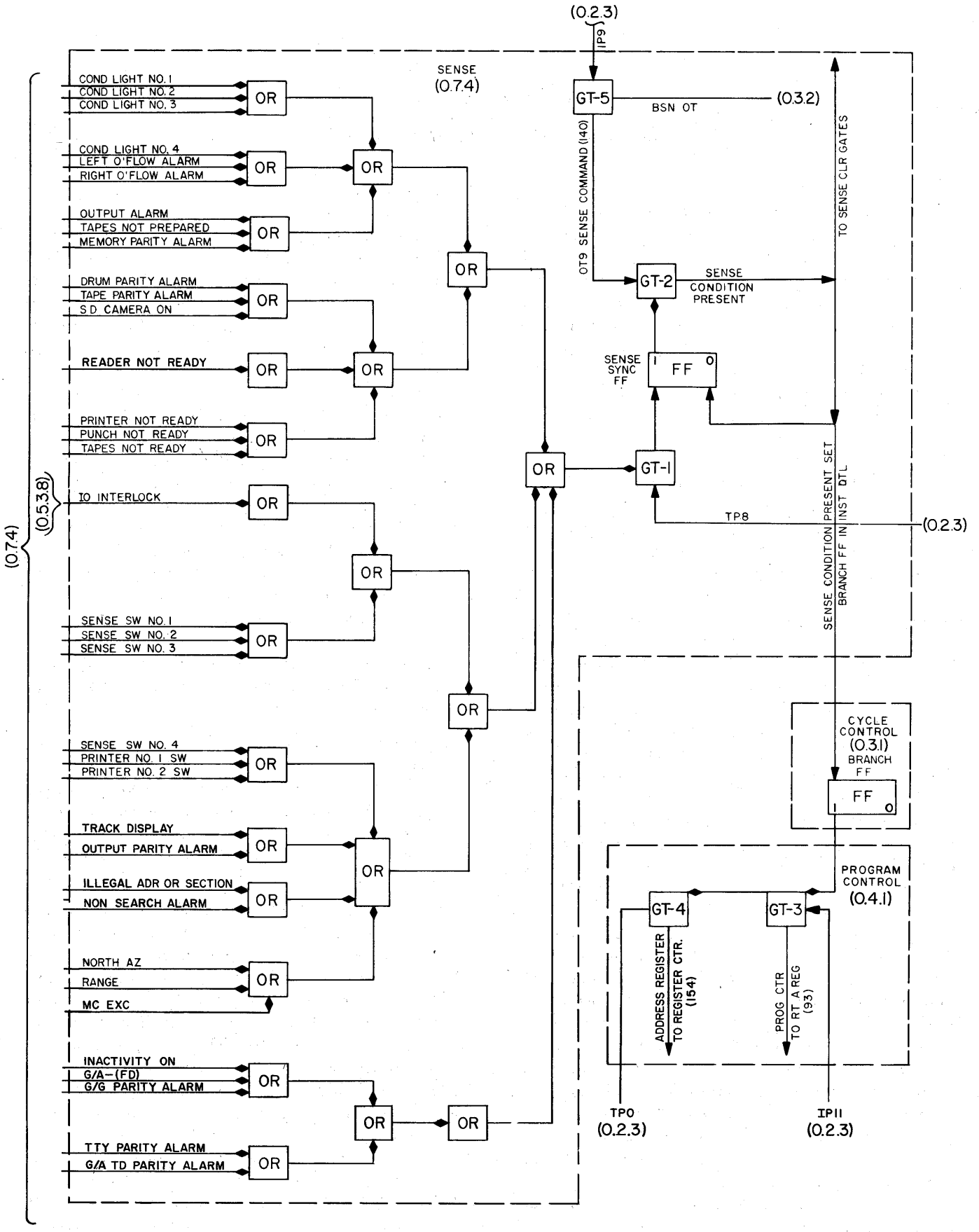


Figure 5–8. Branch Control, BSN Instruction, Simplified Logic Diagram

of the *BSN* instruction (fig. 5–9). If the specified switch is in the ACTIVE position, a d-c level generated by the conditioned AND circuit indicates to the computer that the sense condition is present. The computer then executes a branch of program control as described in 3.1. If the switch is in the SENSE (inactive) position, the AND circuit is deconditioned, indicating the absence of the sense condition to the computer.

The SENSE/ACTIVE switches are used to provide a means of manual control of maintenance programs, provided that the programs are written to utilize them.

3.2.2 Marginal Checking Excursions, *BSN* $(20)_8$ and $(27)_8$

The marginal-checking-excursion sense circuits enable the computer to determine the presence or absence of a marginal checking excursion. When a duplex marginal checking excursion is on, the Power System supplies an activating level to PG 1, which sets FF 1 (fig. 5–9). The output from the flip-flop partially conditions AND 5. If a *BSN* $(20)_8$ instruction is programmed, the output of the *PERSELBSN* matrix fully conditions AND 5. The resultant d-c level from the AND circuit indicates to the computer that a duplex marginal checking excursion is in progress and the computer branches control of the program as described in 3.1.

If a simplex marginal checking excursion is on when the *BSN* $(27)_8$ instruction is executed, a similar operation will result using PG 3, FF 2, and AND 6.

3.2.3 IO Interlock On, *BSN* $(14)_8$

The *BSN* $(14)_8$ instruction is used to determine whether the IO interlock flip-flop is set whenever an IO operation is in progress.

When the IO interlock is on, the IO interlock flip-flop is set and the output from its 1 side partially conditions AND 7. When the *BSN* $(14)_8$ instruction is executed, the *PERSELBSN* matrix supplies another input to AND 7 to fully condition it. The resultant output from AND 7 indicates that the IO interlock is on, and the computer branches control of the program as described in 3.1.

3.2.4 Duplex Switch Alarm, *BSN* $(30)_8$

The DUPLEX SWITCH ALARM switch is located on the duplex maintenance console. When this switch is in its active position, AND 8 (fig. 5–9) is partially conditioned. The execution of a *BSN* $(30)_8$ instruction, while the switch is in its active position, fully conditions AND 8. The resulting output indicates to the computer that the sense condition is present; that is, the duplex switch is active. If the switch is in its inactive position when the computer executes a *BSN* $(30)_8$ instruction, AND 8 is not conditioned and the resulting output level indicates to the computer that the duplex switch is inactive.

3.2.5 Situation Display Camera, *BSN* $(35)_8$

The operation of a situation display camera may be initiated under control of the Central Computer System. When any of the four situation display cameras located in the Display System are operating, the Display System transmits a pulse which sets the situation display camera flip-flop in the selection element. The d-c level generated by the 1 side of this flip-flop partially conditions AND 9. (See fig. 5–9.) If a *BSN* $(35)_8$ instruction is executed at a time when one of the situation display cameras is operating, the d-c level of the *PERSELBSN* matrix fully conditions AND 9. The output level of AND 9 indicates to the computer that the sense condition is present, and the computer branches control of the program as described in 3.1. At the completion of the situation display camera cycle, the Display System generates a pulse which clears the situation display camera flip-flop.

If the situation display camera flip-flop is cleared at the time of the execution of a *BSN* $(35)_8$ instruction, AND 9 remains deconditioned. Under these conditions, the d-c level generated by AND 9 indicates to the computer that the sense condition is not present, and no branch of program control occurs.

3.2.6 Track Display, *BSN* $(37)_8$

When the Display System is reading the track display field of the Drum System, it transmits a pulse which sets the track display flip-flop in the selection element. The d-c level generated by the 1 side of this flip-flop partially conditions AND 10 (fig. 5–9). If a *BSN* $(37)_8$ instruction is executed while the track display flip-flop is set, the d-c level of the *PERSELBSN* matrix fully conditions AND 10. The output level of AND 10 indicates to the computer that the sense condition is present, and the computer branches control of the program as described in 3.1. The track display flip-flop remains set until the Display System executes a read operation on the radar data drum field, at which time the Display System transmits a pulse which clears the track display flip-flop.

If the track display flip-flop is cleared at the time of the execution of a *BSN* $(37)_8$ instruction, AND 10 remains deconditioned. Under these conditions, the d-c level generated by AND 10 indicates to the computer that the sense condition is not present, and no branch of program control occurs.

3.2.7 IO Unit Not Ready, *BSN* $(11)_8$

The IO units not-ready sense circuits enable the Central Computer System to check the status of the card reader, the card punch, the line printer, and the tape units. These circuits are sensed after one of the above units is selected by an *SEL* instruction and before the associated *RDS* or *WRT* instruction is executed. If the selected IO unit is not mechanically ready to perform

its functions (e.g., if the line printer is selected but is out of paper, or if a tape unit is selected but has a door open), a d-c level is applied to the branch-control-on-sense circuits. A *BSN* instruction sensing the IO not-ready circuits executes a branch of control if the selected IO unit is not ready. If the IO unit is ready, the program is executed sequentially.

3.2.7.1 Tapes Not Ready

The tapes-not-ready circuits are shown in figure 5–9. The tape-operation and tapes-not-ready flip-flops are set when one of the tape units is selected. The tape-operation flip-flop, when set, conditions GT 4, GT 7, and the tape control circuits. The tapes-not-ready flip-flop partially conditions AND 14. When the *BSN* $(11)_8$ instruction is decoded, the output of the *PERSELBSN* matrix fully conditions AND 14. Circuit AND 14 now has an output, indicating to the computer that the selected tape unit is not ready. At OT 9 of the *BSN* $(11)_8$ instruction, the computer will branch program control as described in 3.1.

However, if the tape unit is ready, the tapes-not-ready flip-flop must be cleared before OT 9 to remove the conditioning level from AND 14. This is accomplished by PT 11 of the *BSN* instruction. Pulse PT 11 is gated through GT 7 and examines GT 6. If the tape unit is under computer control, GT 6 passes the pulse to GT 5. If the tape unit is in either the read, write, or rewind status, it is ready and GT 5 is conditioned. Gate 5 passes PT 11, and the tapes-not-ready flip-flop is cleared. This removes one of the +10V inputs from AND 14 with the result that the output level is down. The absence of a conditioning output from AND 14 when the OT 9 pulse arrives indicates the selected tape unit is ready. The OT 9 pulse is passed by GT 4 and OR 1 to set the tapes-not-ready flip-flop in preparation for the next *BSN* $(11)_8$ instruction.

3.2.7.2 Card Reader Not Ready

The card-reader-not-ready circuits are shown in figure 5–9. After being selected, the card reader is always sensed for readiness before the read instruction is issued.

When the card reader is selected, the select pulse passes GT 1 and sets the card-reader-not-ready flip-flop, FF 5; FF 5 supplies a conditioning level to AND 11. If the card reader is not ready, a not-ready level is applied to level originator LO 1, and the resulting output is applied to AND 11 as a second conditioning level. When the *BSN* $(11)_8$ instruction is decoded, the *PERSELBSN* matrix supplies a third level to AND 11. Circuit AND 11 is now fully conditioned, and its output level indicates to the computer that the card reader is not ready. If the card reader is ready, LO 1 has an output level of −30V. Under this condition, AND 11 has a deconditioning output, indicating that the card reader is ready.

3.2.7.3 Card Punch Not Ready

The card-punch-not-ready circuitry is identical with that used for the card reader. (Refer to 3.2.7.2 and fig. 5–9.)

3.2.7.4 Line Printer Not Ready

The line-printer-not-ready circuitry is identical with that used for the card reader. (Refer to 3.2.7.2 and fig. 5–9.)

3.2.8 Line Printer, *BSN* $(31)_8$ and $(32)_8$

Two sense codes are provided for sensing conditions in the line printer. The printer is wired to determine which conditions are sensed for. Usually, one code senses for the presence of the plugboard and the other senses for overflow. The two sense circuits are identical. Only one is discussed here.

When a particular sense condition exists, LO 4 is conditioned by the printer and its output partially conditions AND 16. If a *BSN* $(31)_8$ instruction is executed at this time, the *PERSELBSN* matrix supplies another input to AND 16. Circuit AND 16 is now fully conditioned. The output from AND 16 indicates to the computer that the selected sense condition exists, and the computer branches program control as described in 3.1.

3.2.9 Tapes Not Prepared, *BSN* $(10)_8$

The process of sensing for tapes not prepared is very similar to sensing for tapes not ready (see 3.2.7.1) and many of the same circuits are involved. (See fig. 5–9.) The tapes-not-prepared flip-flop is set when a tape drive is selected and must be cleared before OT 9 if the tape is prepared. This is accomplished by PT 11 of the *BSN* instruction. This pulse passes GT 8 which is conditioned by the output of the inverter when the selected tape drive is prepared. The tape drive is prepared when neither the rewind status nor the not-in-file-area condition exists.

3.3 SENSE CODES, GROUP 2

The sense codes in this group are similar to those in group 1 in that they sense for the existence of various conditions in the equipment. However, if the sense condition exists, the computer clears the sense unit in addition to branching program control.

All 20 of the sense circuits in this group are identical; one of them is shown in figure 5–10. In some cases, the flip-flop is a part of the circuit whose condition is to be sensed. In others, it is an indicator which indicates the condition of a remotely located circuit. Whenever one of the sense conditions comes into existence, the appropriate flip-flop is set and the AND circuit is partially conditioned. When one of the *BSN* instructions in this group is programmed, the *PERSELBSN* matrix output supplies a conditioning level to the appropriate AND circuit. If the condition being sensed for is present, the AND circuit is now fully conditioned and its output in-

dicates to the computer that the selected sense condition is present and a branch of program control occurs. If the sense condition is not present, the flip-flop is not set and the AND circuit output indicates the absence of the sense condition, in which case the program does not branch.

The output of the *PERSELBSN* matrix also conditions a gate tube associated with the flip-flop to be sensed. At OT 9 of the *BSN* instruction, this gate tube passes a pulse to clear the sensed flip-flop to prepare it to receive another sense condition signal.

The conditions sensed for by this group of codes are described in the paragraphs which follow.

3.3.1 Condition Lights, *BSN* $(01)_8$ through $(04)_8$

Four condition lights are provided on the duplex maintenance console to provide visual indication of the status of maintenance programs. By means of the *BSN* instruction, the computer can also determine the status of the program. The condition lights are lit by the appropriate *PER* instruction.

3.3.2 Intercommunication Flip-Flops, *BSN* $(43)_8$ through $(46)_8$

The computer has four intercommunication flip-flops, each of which is set when some specific condition exists in the other computer. The *BSN* instruction enables the computer to determine the existence of conditions in the other computer.

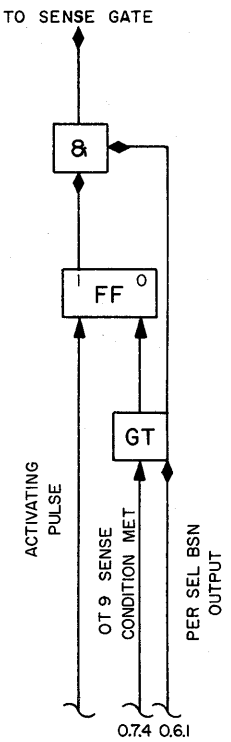


Figure 5–10. Typical Sense Circuit, Group 2, Simplified Logic Diagram

3.3.3 Range Flip-Flop, *BSN* $(34)_8$

This sense code is used to determine whether the test pattern generator in the Input System is generating a range pattern.

3.3.4 North-Azimuth On Flip-Flop, *BSN* $(47)_8$

This sense code is used to determine whether the test pattern generator is generating a north-azimuth signal.

3.3.5 Nonsearch Alarm Flip-Flop On, *BSN* $(50)_8$

In the Output System, a section searches the OB drum fields for new data and stores this data in the output storage circuits. After the search for data is completed, the computer goes into nonsearch time, during which previous data is transmitted. Should an error occur during nonsearch operation, a nonsearch alarm is generated in the Output System.

The computer senses for the presence of a nonsearch alarm by executing a *BSN* $(50)_8$ instruction.

3.3.6 Output Buffer Drum Parity, *BSN* $(51)_8$

Output data is transferred from the Drum System to the output buffer register in the Output System. At this point, a parity check is made. When a parity error is detected, the OB parity flip-flop is set. The *BSN* $(51)_8$ instruction senses the condition of this flip-flop to indicate possible OB parity errors to the computer.

3.3.7 Illegal Address or Section, *BSN* $(52)_8$

This sense code is used to determine the presence or absence of an illegal address or section alarm in the Output System.

3.3.8 Output Parity Alarms, *BSN* $(53\text{-}60)_8$

There are six output parity alarms:

a. G/A FD $(53)_8$
b. G/G $(54)_8$
c. TTY $(55)_8$
d. G/A TD $(56)_8$
e. BO No. 1 $(57)_8$
f. BO No. 2 $(60)_8$

These codes sense for the presence of an output parity alarm. An output-parity-alarm pulse is generated when a parity error is detected in the transfer of an output word to the Output System.

3.4 SENSE CODES, GROUP 3

The codes in this group sense for conditions which activate or are otherwise associated with computer alarms. If the condition sensed for exists, a branch of program control is executed and the sense unit is cleared. The alarm, however, is not cleared. The circuits involved are shown in figure 5–11, foldout.

There are nine sense codes in this group. When desired by the operator, six of these sense conditions may

stop the computer or initiate an automatic branch operation. The six conditions are: memory parity error, tape parity error, status drum parity error, addressable drum parity error, right overflow, and left overflow.

The memory-parity-error pulse which sets the memory parity error flip-flop also senses GT 8 (fig. 5–11). When the MEMORY PARITY ACTIVE-INACTIVE switch on the duplex maintenance console is in the ACTIVE position, GT 8 is conditioned and passes the pulse to OR 3. Drum-and-tape-parity-error pulses and overflow-alarm pulses may be passed through similar circuits and applied to OR 3 also. The output pulse from OR 3 clears the continue and TPD control flip-flops, thus stopping the computer.

If a branch operation is desired, the STOP-BRANCH switch on the duplex maintenance console is placed in the BRANCH position, thus conditioning GT 17. With the circuits in this condition, a pulse from OR 3 not only stops the computer, but is passed through GT 17 and a series of delay lines to various computer circuits, where it performs numerous functions necessary to execute an alarm branch operation to restart the computer. (The computer must be stopped when setting up the branch operation because a branch instruction is not programmed. During the time that the computer is stopped, controls are set up so that when the computer is restarted the branch operation will begin.)

All the sense conditions in this group except alarm 1 and alarm 2 set alarm-indicator flip-flops. The output of each of these flip-flops is applied to OR 2 to sound the computer alarm. The alarm indicators can only be cleared manually from the duplex maintenance console.

The sensing operations controlled by the codes in this group are performed in the same manner as those of group 2. (Refer to 3.3 and fig. 5–11.) The conditions sensed for are described in the paragraphs which follow.

3.4.1 Output Alarm On, *BSN* $(33)_8$

The output-alarm circuits are provided in the output control element to indicate different types of errors in the Output System. The Output System generates the following six types of alarms:

a. OB word parity
b. Illegal section
c. Illegal address
d. Nonsearch comparison
e. Telephone line parity
f. Busy bit

The output-alarm flip-flop is set when any of these conditions occurs.

3.4.2 Left Overflow On and Right Overflow On, *BSN* $(12)_8$ and $(13)_8$

The computer cannot handle numbers with an absolute value greater than 1. When such a number appears in either accumulator register, an overflow condition exists and the proper overflow flip-flop is set.

3.4.3 Memory Parity Error, *BSN* $(15)_8$

The memory-parity-error flip-flop is set whenever a word with incorrect parity is stored into or read out of core memory.

3.4.4 Tape Parity Error, *BSN* $(17)_8$

The tape parity error flip-flop is set whenever a word with incorrect parity is read from the tapes and stored in core memory.

3.4.5 Addressable Drum Parity Error, *BSN* $(16)_8$

The addressable drum parity error flip-flop is set whenever a word with incorrect parity is read from the addressable drums and stored in core memory.

3.4.6 Status Drum Parity Error, *BSN* $(25)_8$

The status drum parity error flip-flop is set whenever a word with incorrect parity is read from the status drum and stored in core memory.

3.4.7 Alarm 1 On and Alarm 2 On, *BSN* $(41)_8$ and $(42)_8$

These two codes allow one computer to sense for the presence of alarm conditions in the other computer. The alarm 1 flip-flop is set when parity errors or overflow conditions occur in the other computer. The alarm 2 flip-flop is set by tape- or drum-parity-error signals from the other computer.

3.4.8 Inactivity On, *BSN* $(05)_8$

This code senses for TPD inactivity (e.g., to determine whether the Central Computer is hung up in an IO pause). The inactivity flip-flop is set whenever TPD inactivity occurs.

CHAPTER 4

TEST ONE BIT AND *TEST TWO BITS* INSTRUCTION ANALYSIS

4.1 INTRODUCTION

The *Test One Bit* and *Test Two Bits* instructions provide a means of testing a specific bit or bits of a memory word without affecting the memory word composition. The *TOB* instruction is used to test one particular bit, and the *TTB* instruction is used to test two particular bits of the operand specified by the right-half of the instruction word.

During the execution of either of these instructions, the specified bit or bits are tested to determine whether the program counter will be stepped an additional number of times to ship any of the next sequential instructions. During the execution of a *TOB* instruction, the specified bit is tested for content. If the bit contains a 1, the program counter is stepped one additional time, causing it to skip the next sequential instruction following the *TOB* instruction. If the tested bit contains a 0, the program counter will only be stepped normally to the next instruction. The bit to be tested is determined by the contents of L11-L15 of the instruction word.

During the execution of the *TTB* instruction, two particular bits are tested and the program counter is stepped accordingly. The possible combinations of the two bits and the action taken as a result of the combinations are:

a. 00 - The program counter is stepped in the normal manner to the next instruction.

b. 01 - The program counter is stepped one additional time, thereby causing the next instruction to be skipped.

c. 10 - The program counter is stepped two additional times, thereby causing the next two instructions to be skipped.

d. 11 - The program counter is stepped three additional times, thereby causing the next three instructions to be skipped.

The word bits to be tested are determined by the contents of L11-L15 of the instruction word. These bits specify only one word bit (same as *TOB*), but the instruction coding automatically causes the bit adjacent to, and to the left of, the specified bit to be also tested.

4.2 DETAILED ANALYSIS

The information flow during the execution of the *TOB* and *TTB* instructions is shown in simplified form in figure 5–12. As shown in the figure, at PT 7, the contents of the left memory buffer register are transferred to the operations register (only bit L11 is shown for simplicity) and the index interval register (L10 through L15). The contents of bit L10 of the instruction word are also transferred to the L10 storage flip-flop. This flip-flop is used to specify whether a *TOB* or a *TTB* instruction is to be executed. If this flip-flop contains a 0, the *TOB* instruction is specified; if the flip-flop contains a 1, the *TTB* instruction is specified. The right memory buffer register contents (not shown), specifying the address of the operand which contains the bit or bits to be tested, are transferred to the address register at PT 7. At OT 1, command 102 sets bits L10 and L11 of the index interval register to the 1 side. Combining these bits with the original contents of index interval bits L12 through L15 in the *PERSELBSN* matrix will result in an output level from the matrix which designates the bit position to be tested according to the following octal code identification. Since bits L10 and L11 are 1's, the lowest value possible with all 0's in the remaining index interval register bits (L12 through L15) is octal 60 (this specifies sign bit selection). The highest value possible with all 1's in bits L12 through L15 is octal 77 (this specifies bit 15 selection). Therefore, octal code 61 specifies bit 1, etc.

As noted in table 5–2, the state of the operations register bit L11 flip-flop determines whether the bit (or bits) to be tested is located in the left or right half-word of the operand. If bit L11 contains a 0, the test bit (or bits) is located in the left half-word. If bit L11 contains a 1, the right half-word is specified. The two gates associated with this flip-flop are sensed during the *TOB* and *TTB* instructions to determine which portion of the memory word is to be used. The output pulse of the selected gate is used to transfer the memory buffer content to sense the individual *TOB-TTB* gates. If a sense pulse is applied to the conditioned *TOB-TTB* gate, an output pulse is generated (specifying that the test condition exists). This pulse senses the gates associated with the L10 storage flip-flop. The output of these latter gates is used to step the program counter.

During the execution of the *TOB* instruction, the contents of the specified half-word are transferred to

the *TOB-TTB* gates only once, at OT 7. However, during the execution of the *TTB* instruction, this transfer is executed twice, first at OT 7 and again at OT 9. To illustrate the operation of the transfer circuits, consider the following examples. If *PERSELBSN* matrix output line 61_8 (which designates that bit 1 is to be tested) is selected during the execution of a *TOB* instruction (L10 storage flip-flop contains a 0), the *TOB-TTB* gate associated with bit 1 and the step-by-1 gate will both be conditioned. If bit 1 of the selected half-word contains a 1, the conditioned gate (bit 1) will generate a pulse (at OT 9) which will sense the bit L10 storage flip-flop gates. Since the latter flip-flop contains a 0, the program counter will be stepped once by the step-by-1 pulse.

During the execution of a *TTB* instruction (L10 storage flip-flop contains a 1), with bit 1 again selected, the AND circuit output, which combines the *PERSELBSN* matrix output with the bit L10 storage flip-flop output, will first condition the *TOB-TTB* gate associated with the sign bit of the selected half-word. It must be noted that on a *TTB* instruction the bit adjacent to, and to the left of, the bit selected (in this case bit 1) is always tested first. At OT 7, the selected memory buffer contents are transferred to the *TOB-TTB* gates, and, if the sign bit contains a 1, the program counter will be stepped by the step-by-2 pulse. At OT 8, the L10 storage flip-flop is cleared, with the result that the *TOB-TTB* gate specified by the instruction and the step-by-1 gate will both be conditioned to set up the same control used in the *TOB* instruction. At OT 9 of the *TTB* instruction, the operations register bit L11 gates are sensed a second time to execute the *TOB* operation described above. If the sign bit and the bit 1

TABLE 5–2. *TOB-TTB* TEST BIT SELECTION

OCTAL NOTATION OF BITS L11 THROUGH L15 OF INSTRUCTION WORD	TEST BIT SELECTED	
	TOB INSTRUCTION (0.0050—)*	*TTB* INSTRUCTION (0.0054—)*
00	LS	LS**
01	L1	LS, L1
02	L2	L1, L2
03	L3	L2, L3
04	L4	L3, L4
05	L5	L4, L5
06	L6	L5, L6
07	L7	L6, L7
10	L8	L7, L8
11	L9	L8, L9
12	L10	L9, L10
13	L11	L10, L11
14	L12	L11, L12
15	L13	L12, L13
16	L14	L13, L14
17	L15	L14, L15
20	RS	RS**
21	R1	RS, R1
22	R2	R1, R2
23	R3	R2, R3
24	R4	R3, R4
25	R5	R4, R5
26	R6	R5, R6
27	R7	R6, R7
30	R8	R7, R8
31	R9	R8, R9
32	R10	R9, R10
33	R11	R10, R11
34	R12	R11, R12
35	R13	R12, R13
36	R14	R13, R14
37	R15	R14, R15

TABLE 5–2. *TOB-TTB* TEST BIT SELECTION (cont'd)

**The fourth bit in the octal code specifies bit L10 of the instruction (0 for TOB and 4 for TTB). For either instruction, the octal equivalent must include the addition of the L10 code. Thus, TTB 34 is 0.00574; TOB 32 is 0.00532, etc.*

***The TTB 00 (0.00540) and TTB 20 (0.00560) instruction codes are not selectable as two test bit combinations. If specified, they will be executed as a TOB instruction which can only result in a 1-bit addition to bit 15 of the program counter.*

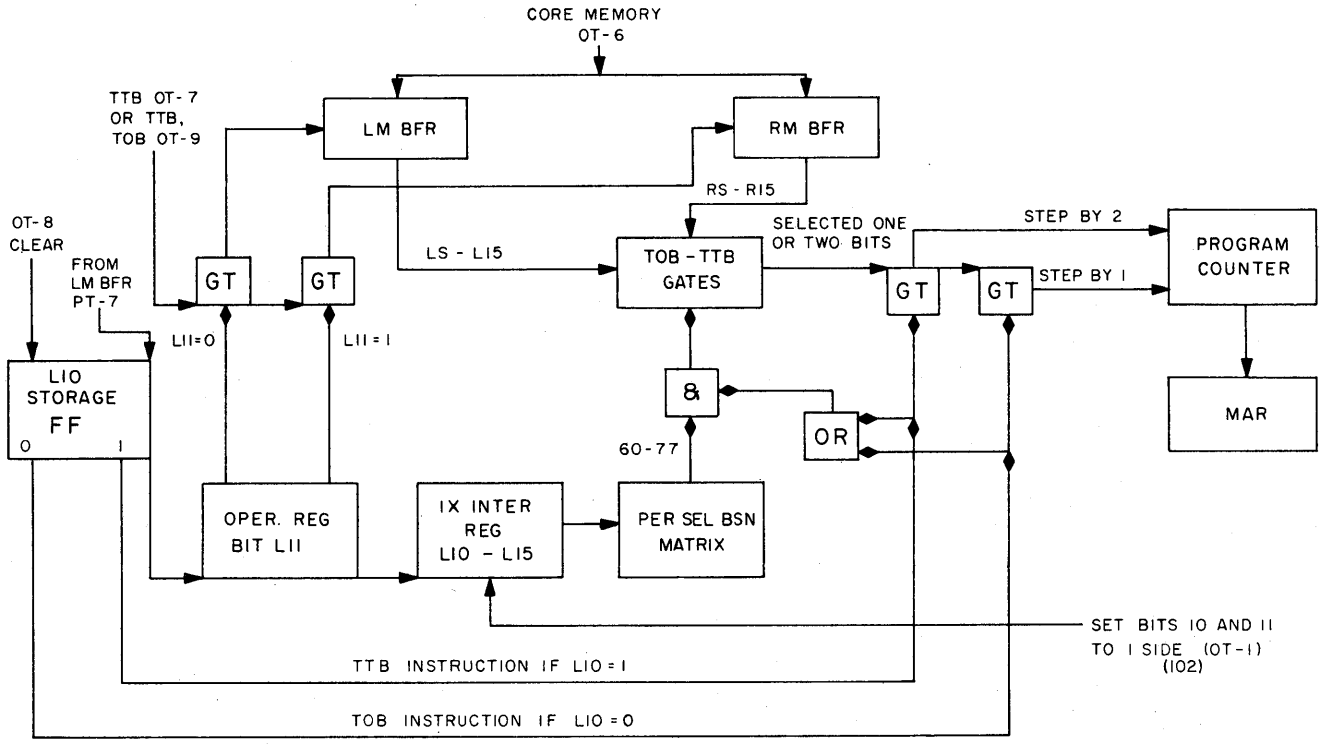


Figure 5–12. TOB, TTB Instruction Analysis, Logic Diagram

flip-flops of the selected memory buffer both contain 1's, the program counter is stepped a total of three times in addition to the normal program stepping.

The control circuits required to perform the operations specified by the *TOB* and *TTB* instructions are shown in simplified form in figure 5–13. A detailed analysis of these control circuits using examples of each of the two instructions follows.

Analysis of the *TOB* instruction is based on the following sample program:

0.00001	*TOB* 01 (0.00501)	0.20000
	0.00501	
0.00002	*BPX*	Subroutine A
0.00003	*BPX*	Subroutine B

As noted above, the *TOB* instruction is used to determine the content of bit L1 (table 5–2) of the operand contained in memory location 0.20000, and to set up program control to take appropriate action. Referring to figure 5–13, it will be noted that at PT 7 the program counter is stepped by 1, and the index interval portion of the instruction word is transferred to the index interval register, the operations register bit L11 flip-flop, and the L10 storage flip-flop. Since the index interval code contains $01_{(8)}$, the operations register bit L11 flip-flop and the L10 storage flip-flop will both be set to the 0 state. At OT 1, command 102 sets index interval register flip-flops L10 and L11 to the 1 state; thus, the index interval register now contains $110001_{(2)}$ or $61_{(8)}$. Under these conditions, output line 61 of the *PERSELBSN* matrix will be at a +10V level to partially condition AND circuits 1 and 2. Since the L10 storage flip-flop contains a 0, AND 1 is fully selected and its output level conditions GT 1. At OT 6, the operand is transferred from memory location 0.20000 to the memory buffer register. At OT 9, the gates (GT's 3 and 4) associated with the operations register bit L11 flip-flop are sensed to determine which memory buffer register is to be involved in the transfer operation. Since this flip-flop contains a 0, GT 3 is conditioned to generate a pulse which transfers the contents of the left memory buffer register to the *TOB-TTB* gates. Since GT 1 is the only *TOB-TTB* gate conditioned, an output pulse from OR 1 indicates that bit L1 of the memory buffer register contains a 1. If bit L1 did contain a 1, the output pulse from OR 1 will sense GT's 5 and 6. Since the L10 storage flip-flop contains a 0, GT 5 will gate the command 98 pulse to add a 1 to the program counter. Thus, if bit LS contains a 0, the next instruction executed will cause the program to branch to subroutine A. However, if bit LS contains a 1, the next instruction executed will cause the program to branch to subroutine B.

Analysis of the *TTB* instruction is based on the following sample program:

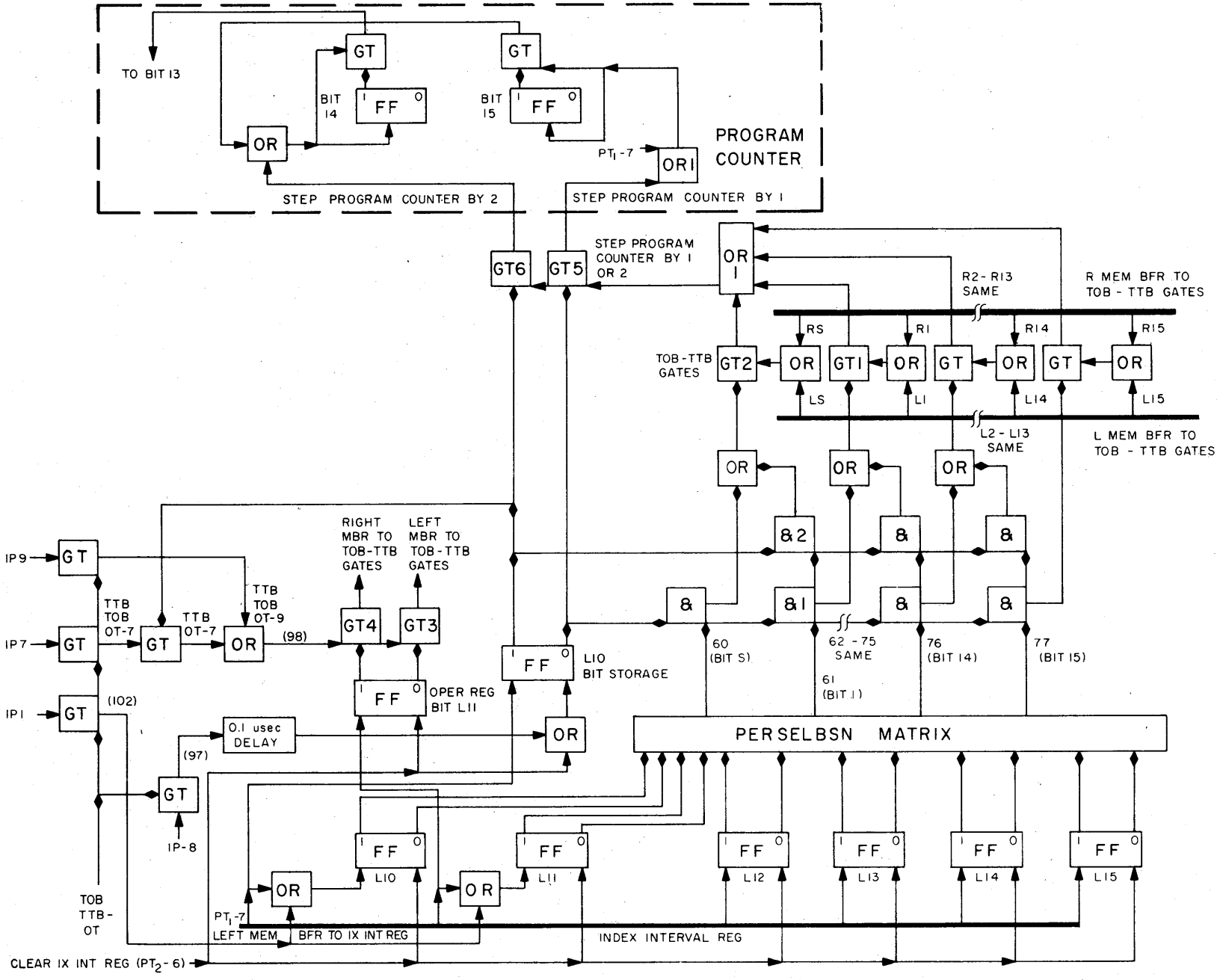


Figure 5–13. TOB, TTB Control Circuits, Logic Diagram

0.00001	*TTB* 01 (0.00541)	0.20000
0.00002	*BPX*	Subroutine A
0.00003	*BPX*	Subroutine B
0.00004	*BPX*	Subroutine C
0.00005	*BPX*	Subroutine D

As noted above, the *TTB* instruction is used to determine the contents of bits LS and L1 (table 5–1) of the operand contained in memory location 0.20000, and to set up program control to take appropriate action. Referring to figure 5–13, it will be noted that circuit operation during the execution of the *TTB* instruction is similar to that noted during the analysis of the *TOB* instruction. At PT 7, the program counter is stepped by 1, and the index interval portion of the instruction is transferred to the index interval register, the operations register bit 11 flip-flop, and the L10 storage flip-flop. Since the index interval code contains $41_{(8)}$, the operations register bit L11 flip-flop is set to 0, and the L10 storage flip-flop is set to 1 (indicating that the *TTB* instruction is selected). At OT 1, command 102 sets index interval register flip-flops L10 and L11 to the 1 state. As a result, the index interval register will now contain $110001_{(2)}$ or $61_{(8)}$ and output line 61 of the *PERSELBSN* matrix will partially condition AND circuits 1 and 2. Since the L10 storage flip-flop contains a 1, AND 2 is fully selected to condition GT 2. Thus, although bit 1 is specified by the index interval content, the *TOB-TTB* sign-bit gate (adjacent to, and to the left of, the selected bit) is conditioned and tested first. At OT 6, the operand is transferred to the memory buffer register from memory location 0.20000. At OT 7, GT's 3 and 4 are sensed and, since GT 3 is conditioned, the contents of the left memory buffer register are transferred to the *TOB-TTB* gates. Since GT 2 is the only *TOB-TTB* gate conditioned, an output pulse from OR 1 indicates that bit LS of the memory buffer contains a 1. If bit LS contains a 1, the output pulse from OR 1 will sense GT's 5 and 6. Since the L10 storage flip-flop contains a 1, GT 6 will gate the command 98 pulse to step the program counter by 2. At OT 8, the L10 storage flip-flop is cleared to the 0 side by command 97, thereby setting up control to execute a *TOB* instruction. Since bit L1 was specified as the test bit in both examples (*TOB* and *TTB*), the OT 9 pulse will perform exactly the same action noted in the *TOB* instruction analysis.

As a result of executing a *TTB* instruction, the program counter may be stepped to skip a maximum of three instructions, depending on the content of the selected test bits. Since the two test bits represent four possible combinations of content, four separate *BPX* instructions are required so that the program may be made to branch to a specific subroutine representing the test bit content.

CHAPTER 5
SELECT CODES

5.1 INTRODUCTION

The institution of an IO operation to transfer data between an IO unit and the memory element invariably requires the programming of at least three instructions. Two of these instructions are preparatory, and the third is the instruction of execution. The two preparatory instructions are *Select (SEL)* or *Select Drums (SDR)* and *Load IO Address Counter (LDC)*. The *LDC* instruction loads the IO address counter with the core memory location specified as the starting address for the IO process. Succeeding IO transfer operations proceed from the starting address in consecutive order. The *SEL* instruction is used to select one of the IO devices (other than drums), and the *SDR* instruction is used to select one of 75 drum fields. In addition to specifying which IO device is to be involved in the subsequent IO operation, these instructions also set up the proper control circuits in the IO element. Selection of an IO unit is specified by the index interval portion (bits L10 through L15) of the *SEL* instruction; selection of a drum field is specified by the index interval portion and bit R1 of the *SDR* instruction. This chapter provides an analysis of the select and select drums codes. A simplified logic diagram of the circuit operation involved in the selection of a drum field or other IO unit is shown in figure 5–14, foldout.

5.2 DRUM FIELD SELECTION

Selection of the drum field (1 of 75) to be used in an IO operation is identified by bits R1 and L10 through L15 of the *SDR* instruction word. If the selected drum field is address-controlled, the drum-starting address is specified by the address portion of the instruction word. If the selected drum field is to be read by a status identity mode, the address portion of the instruction word supplies the identifying code to be used. Since some of the drum fields can be controlled in several ways, individual *Select Drum* instruction codes specify which drum field is selected as well as the type of control that is to be exercised. The *SDR* instruction sets up control circuits in both the selected drum group and the IO element to prepare these circuits for the subsequent transfer of data. These circuits will remain selected until a new *SDR* or an *SEL* instruction is initiated. If the IO interlock is on (indicating that a previously programmed IO operation is still in process), execution of the instruction is delayed until the interlock is returned to the off condition.

The drum fields and their select codes are listed in table 5–3. Certain drum fields can only be read by the computer whereas others can only be written on. In the column labeled Operations, the words "read" and "write" indicate the instruction normally used to initiate the transfer of data. In certain cases, however, special provisions are made for testing the drums and their associated circuits; these require special codes for field selection. The column in the table labeled "Mode" identifies the method of data transfer used for each drum field.

Drum field selection involves a decoding process performed by the index interval register, the *PERSELBSN* matrix, and bit R1 of the address register. Briefly stated, during the execution of the *SDR* instruction, the index interval register (bits L10 through L15 of the instruction word) specifies which drum field of the main or auxiliary group is to be selected, and bit R1 of the address register is used to differentiate between the selection of a main drum field and that of an auxiliary drum field. The manner in which this is accomplished is described below; the circuits involved appear in figure 5–14.

At OT 5 of both the *SEL* and the *SDR* instructions, the address-register-bit-R1 flip-flop in the selection element is cleared in anticipation of a transfer of information from bit R1 of the address register. If bit R1 of the address register is a 1, the flip-flop in the selection element is set at PT 3 and the d-c level generated by its 1 side conditions GT 13. If bit R1 is a 0, the flip-flop remains cleared to condition GT 14. At PT 5 of the *SDR* instruction, GT's 13 and 14 are sensed to determine which drum group is selected. If GT 13 is conditioned, it will develop an output pulse to sense GT's 7 through 12, and thus transfer the content of the index interval register to the auxiliary drum selection register of the Drum System. That is, if bit R1 of the address register is a 1, the *SDR* instruction selects an auxiliary drum field. Conversely, if bit R1 of the address register is a 0, GT 14 is conditioned so that the content of the index interval register (GT's 1 through 6) will be transferred to the main drum selection register of the Drum System at PT 5 of the *SDR* instruction.

During the execution of an *RDS* or a *WRT* instruction, the last three bits of the index interval register are used to specify the interleaving constant, if one is to be used. Interleaving is not performed if

TABLE 5–3. DRUM FIELD AND MODE SELECT CODES

OCTONARY CODE R1	L10-15	DRUM FIELD	MODE	OPERATION
0	02	Auxiliary memory 1	Address	Read, write
0	03	Auxiliary memory 2	Address	Read, write
0	04	Auxiliary memory 3	Address	Read, write
0	05	Auxiliary memory 4	Address	Read, write
0	06	Auxiliary memory 5	Address	Read, write
0	07	Auxiliary memory 6	Address	Read, write
0	10	Auxiliary memory 7	Address	Read, write
0	11	Auxiliary memory 8	Address	Read, write
0	12	Auxiliary memory 9	Address	Read, write
0	13	Auxiliary memory 10	Address	Read, write
0	14	Auxiliary memory 11	Address	Read, write
0	15	Auxiliary memory 12	Address	Read, write
1	41	Auxiliary memory 13	Address	Read, write*
1	42	Auxiliary memory 14	Address	Read, write*
1	43	Auxiliary memory 15	Address	Read, write*
1	44	Auxiliary memory 16	Address	Read, write*
1	45	Auxiliary memory 17	Address	Read, write*
1	46	Auxiliary memory 18	Address	Read, write*
1	51	Auxiliary memory 19	Address	Read, write*
1	52	Auxiliary memory 20	Address	Read, write*
1	53	Auxiliary memory 21	Address	Read, write*
1	54	Auxiliary memory 22	Address	Read, write*
1	55	Auxiliary memory 23	Address	Read, write*
1	56	Auxiliary memory 24	Address	Read, write*
1	61	Auxiliary memory 25	Address	Read, write*
1	62	Auxiliary memory 26	Address	Read, write*
1	63	Auxiliary memory 27	Address	Read, write*
1	64	Auxiliary memory 28	Address	Read, write*
1	65	Auxiliary memory 29	Address	Read, write*
1	66	Auxiliary memory 30	Address	Read, write*
1	71	Auxiliary memory 31	Address	Read, write*
1	72	Auxiliary memory 32	Address	Read, write*
1	73	Auxiliary memory 33	Address	Read, write*
1	74	Auxiliary memory 34	Address	Read, write*
1	75	Auxiliary memory 35	Address	Read, write*

TABLE 5–3. DRUM FIELD AND MODE SELECT CODES (cont'd)

OCTONARY CODE R1 L10-15		DRUM FIELD	MODE	OPERATION
1	76	Auxiliary memory 36	Address	Read, write*
1	02	Auxiliary memory 37	Address	Read, write*
1	03	Auxiliary memory 38	Address	Read, write*
1	04	Auxiliary memory 39	Address	Read, write*
1	05	Auxiliary memory 40	Address	Read, write*
1	06	Auxiliary memory 41	Address	Read, write*
1	07	Auxiliary memory 42	Address	Read, write*
1	10	Auxiliary memory 43	Address	Read, write*
1	11	Auxiliary memory 44	Address	Read, write*
1	12	Auxiliary memory 45	Address	Read, write*
1	13	Auxiliary memory 46	Address	Read, write*
1	14	Auxiliary memory 47	Address	Read, write*
1	15	Auxiliary memory 48	Address	Read, write*
0	24	Crosstell input	Status	Read, test write
0	25	Crosstell input	Identity (R11-R15)	Read
0	40	Crosstell marker (single channel)	Address	Write (contents of LS only)
0	27	Digital display	Address	Read, write
0	17	Digital display	Identity (R14-R15)	Test read
0	32	Gap-filler input	Status	Read, test write
0	33	Gap-filler input	Identity (R11-R15)	Read
0	16	Intercommunication (other)	Address	Read
0	26	Intercommunication (own)	Address	Write
0	76	Intercommunication	Address	Test read
0	34	Long-range radar input 1	Status	Read, test write
0	35	Long-range radar input 1	Identity (R12-R15)	Read
0	50	Long-range radar input 1	Identity (R7-R15)	Read
0	36	Long-range radar input 2	Status	Read, test write
0	37	Long-range radar input 2	Identity (R12-R15)	Read
0	51	Long-range radar input 2	Identity (R7-R15)	Read
0	22	Manual input	Status	Read, test write
0	23	Manual input	Identity	Read
0	30	Output buffer odd	Status Identity (R14-R15)	Write

TABLE 5–3. DRUM FIELD AND MODE SELECT CODES (cont'd)

OCTONARY CODE R1 L10-15		DRUM FIELD	MODE	OPERATION
0	31	Output buffer even	Status	Write, test read
0	60	Radar data 1	Address	Read, write
0	61	Radar data 2	Address	Read, write
0	62	Radar data 3	Address	Read, write
0	63	Radar data 4	Address	Read, write
0	64	Radar data 5	Address	Read, write
0	65	Radar data 6	Address	Read, write
0	66	Radar data 7	Address	Read, write
0	67	Radar data 8	Address	Read, write
0	70	Radar data 9	Address	Read, write
0	47	Situation display	Identity (R5-R10)	Test read
0	20	Spare 1 (XTL)	Address	Read, write
0	21	Spare 2 (AM)	Address	Read, write
0	41	Track display 1	Address	Read, write
0	42	Track display 2	Address	Read, write
0	43	Track display 3	Address	Read, write
0	44	Track display 4	Address	Read, write
0	45	Track display 5	Address	Read, write
0	46	Track display 6	Address	Read, write

**Write permitted only if interlock switch is unlocked.*

none of these bits contains a 1. The interleaving information is transferred to the appropriate circuits of the IO element at PT 6 of either the *RDS* or *WRT* instruction.

In order to properly set up the drum transfer control circuits in the IO element, the *PERSELBSN* matrix decodes the contents of the index interval register to condition the single output line which reflects the contents of the register. The applicable output conditions a set of gates which, upon being sensed during the *SDR* instruction, alter the status of certain drum control flip-flops in the IO element so that the characteristics of the selected drum field are reflected by the status of these control flip-flops. In the following step-by-step analysis of main drum selection, only those circuits pertinent to an IO transfer operation between the Drum System and the Central Computer System are included in the discussion.

When a drum field is designated as the IO unit to be linked with the Central Computer System, the *SDR* instruction is executed as the first instruction in the basic IO program pattern. The index interval of the *SDR* instruction assigns one of the available drum fields in accordance with the code given in table 5–3. If the selected field utilizes the address mode or identity mode of operation, the starting address or the identity code is contained in the address part of the instruction. The starting core memory address to be used in the subsequent transfer of data is loaded into the IO address counter as a result of the *LDC* instruction.

Execution of the *SDR* instruction is delayed if the IO interlock is on, indicating that a previously scheduled IO process is still in progress. As soon as the IO interlock is cleared, the *SDR* instruction proceeds. Command 155 (deselect IO device) is generated at OT 5 by the instruction control element and is sent to the selection element, the IO element, and the Drum System. In the Drum System, it prepares the drum circuits for the selection of a new drum field by resetting all the circuits affected by a previous information transfer.

Command 155 deselects the IO unit last used and deactivates the drum selection controls. The drum selection controls, consisting of eight flip-flops, are shown in figure 5–12. The drum-operation flip-flop indicates that a Drum System/Central Computer System transfer has been requested by the program; the address register bit R1 flip-flop indicates whether a main or auxiliary drum field is selected; and the six transfer mode flip-flops reflect the inherent characteristics of the selected drum field. When the identity 14-15, identity 11-15, identity 5-10, identity 12-15, identity 7-15, and address mode flip-flops are in the 0 state, it is assumed that the drum field to be selected is processed by the address mode and that it has a parity bit associated with each of its registers. The assumptions may be altered in the course of the *SDR* instruction to conform with the chracteristics of the drum field actually selected.

At PT 2 of the instruction, the drum control register in the IO element is cleared by command 146 (clear drum control register) to prepare it for the receipt of the address part of the *SDR* instruction. This transfer occurs at PT 3. Command 325 (set drum operate flip-flop) is issued next by the instruction control element. This command alters, if necessary, the original assumptions made by the drum selection controls concerning the mode of operation and the existence of parity bits. Command 325 also causes the transfer of the index interval register contents to the drum selection register.

The index interval register conditions only one of the outputs of the *PERSELBSN* matrix; namely, the one associated with the selected drum field. To illustrate the action of the drum selection controls, assume that output line 23 of the *PERSELBSN* matrix is conditioned in accordance with the contents of the index interval register. Table 5–1 gives the characteristics of this drum field, which is read by a status identity mode of operation in which bits R14 and R15 specify the identity code to be used. In addition, the data contained on this drum field does not have a parity bit associated with it. Accordingly, GT's 44 and 45 are conditioned by the *PERSELBSN* matrix output (fig. 5–14). When command 325 is given, all gate tubes associated with *SEL* or *SDR* instructions are sensed. Since GT's 44 and 45 are conditioned, the interrogation results in the setting of the identity 14-15 flip-flop and the clearing of the address-mode and parity-check flip-flops. The circuits conditioned by these flip-flops control the subsequent transfer of data so that only those words with the desired identity code will be accepted. In addition, a parity bit will be assigned to each accepted word as it is stored in the memory element.

5.3 IO DEVICE SELECTION

The IO device (with the exception of the magnetic drums) that is to function during an IO operation is identified by the last six bits (L10 through L15) of the operation part of the *Select (SEL)* instruction. Provision is made for selecting one of eight IO units. The address part of this instruction is meaningless; the use of indexing, therefore, does not affect its execution. The *SEL* instruction sets up controls in the IO element to prepare it for the subsequent transfer of data. These control circuits remain selected until another *SEL* instruction or an *SDR* instruction is instituted.

5.3.1 Card Reader, *SEL* $(01)_8$

The card reader can only be used for a read operation. Figure 5–14 shows the circuits immediately affected when the card reader is selected. The index interval contents are decoded by the *PERSELBSN* matrix, and output line 1 of this matrix conditions GT's 15 and 16. At OT 5 of the *SEL* instruction, command 155 clears all IO selection flip-flops except the read/write-0 flip-flop, which it sets. At PT 5, command 156 senses GT's 15 and 16. Since both of these gates are conditioned, the result of the interrogation is that the card-reader, card-machine, and card-reader-not-ready flip-flops are set, and the read/write-0 flip-flop is cleared.

The card-reader flip-flop is used to gate the start-read pulse (occurring during the subsequent *RDS* instruction) to the card reader. The card-machine flip-flop is used to gate BI pulses, which carry on the actual IO transfer following the *RDS* instruction. The read/write-0 flip-flop is used to inform the computer whether a read-0 or write-0 condition is logical for the IO unit selected. In the case of the card machines it is logical, and the flip-flop is cleared. The card-reader-not-ready flip-flop is used during the execution of the *BSN 11* (sense IO units not ready) instruction to specify that the status of the card reader is being interrogated. If the *BSN 11* instruction is executed after the *SEL 01* instruction, and the card reader is not ready, a branch of program control will occur to indicate that manual intervention is necessary, before this device can be used.

5.3.2 Card Punch, *SEL* $(02)_8$

The card punch can only be used for a write operation. Figure 5–14 shows the circuits immediately affected when the card punch is selected. The index interval contents are decoded by the *PERSELBSN* matrix, and output line 2 conditions GT's 16 and 17. At OT 5 of the *SEL* instruction, command 155 clears all IO selection flip-flops except the read/write-0 flip-flop, which it sets. At PT 5, command 156 senses GT's 16 and 17. Since both of these gate tubes are conditioned, the result of the interrogation is that the card-punch,

card-machine, and card-punch-not-ready flip-flops are set, and the read/write-0 flip-flop is cleared.

The card-punch flip-flop is used to gate the start-write pulse (occurring during the subsequent *WRT* instruction) to the card punch. The card-machine flip-flop is used to gate BO pulses, which carry on the actual IO transfer following the *WRT* instruction. The read/write-0 flip-flop is used to inform the computer whether a read-0 or write-0 condition is logical for the IO unit selected. In the case of the card machines it is logical, and the flip-flop is cleared. The card-punch-not-ready flip-flop is used during the execution of the *BSN 11* (sense IO units not ready) instruction to specify that the status of the card punch is being interrogated. If the *BSN 11* instruction is executed after the *SEL 02* instruction, and the card punch is not ready, a branch of program control will occur to indicate that manual intervention is necessary before this device can be used.

5.3.3 Line Printer, *SEL* $(03)_8$

The line printer can only be used for a write operation. Figure 5–14 shows the circuits immediately affected when the line printer is selected. The index interval contents are decoded by the *PERSELBSN* matrix, and output line 3 conditions GT's 16 and 18. At OT 5 of the *SEL* instruction, command 155 clears all IO selection flip-flops except the read/write-0 flip-flop, which it sets. At PT 5, command 156 senses GT's 16 and 18. Since both of these gate tubes are conditioned, the result of the interrogation is that the line-printer, card-machine, and line-printer-not-ready flip-flops are set, and the read/write-0 flip-flop is cleared.

The line-printer flip-flop is used to gate the start-write pulse (occurring during the subsequent *WRT* instruction) to the line printer. The card-machine flip-flop is used to gate BO pulses, which carry on the actual IO transfer following the *WRT* instruction. The read/write-0 flip-flop is used to inform the computer whether a read-0 or write-0 condition is logical for the IO unit selected. In the case of the card machines it is logical, and the flip-flop is cleared. The line-printer-not-ready flip-flop is used during the execution of the *BSN 11* (sense IO units not ready) instruction to specify that the status of the line printer is being interrogated. If the *BSN 11* instruction is executed after the *SEL 03* instruction, and the line printer is not ready, a branch of program control will occur to indicate that manual intervention is necessary before this device can be used.

5.3.4 IO Register, *SEL* $(04)_8$

Whenever it is necessary to clear core memory, the IO register is selected as an input device and is used as a source of words containing only 0's to be written into core memory. The IO register can also be used to load a particular pattern into core memory. This is accomplished by selecting the IO register, writing the desired test pattern word into the register, and then reading the IO register to load the core memory under test with the desired pattern.

Selection of the IO register as an IO device is accomplished by executing the *SEL 04* instruction. During the execution of this instruction, the *PERSELBSN* matrix conditons a select gate which is used to set the IO register selected flip-flop in the IO element. The output levels of this flip-flop are used to control the transfer of data to or from the IO register during the subsequent IO operation.

5.3.5 Manual Input Matrix, *SEL* $(06)_8$

The manual input matrix can only be used as an input device. Figure 5–14 shows the circuits immediately affected when the manual input matrix is selected. During the execution of the *SEL 06* instruction, *PERSELBSN* matrix output line 6 develops a d-c level to condition GT 46. At OT 5 of the *SEL* instruction, command 155 clears all the IO selection flip-flops. At PT 5, command 156 senses all of the select gates. Since GT 46 is conditioned, the result of the interrogation is the setting of the manual-input-matrix flip-flop. The output level of the manual-input-matrix flip-flop conditions the transfer control circuits in the IO element which are used during the subsequent read operation.

5.3.6 Warning Lights, *SEL* $(10)_8$

The warning lights are selected as an IO device for writing only. Figure 5–14 shows the circuits immediately affected when the warning lights are to be selected. During the execution of the *SEL 10* instruction, *PERSELBSN* matrix output line 10 conditions GT 19. At OT 5 of the *SEL* instruction, command 155 clears all the IO selection flip-flops. At PT 5, command 156 senses all the select gates. Since GT 19 is conditioned, the interrogation results in the setting of the warning-light flip-flop.

The warning-light flip-flop is used to condition the transfer control circuits in the IO element which affect the transfer of data to the Warning Lights System during the subsequent write operation.

5.3.7 Magnetic Tapes, *SEL* $(11)_8$ through $(16)_8$

The six magnetic tape drives contained in the tape system may be used for reading and writing. Figure 5–14 shows the circuits immediately affected when a magnetic tape unit is selected. During the execution of the appropriate *SEL* instruction, *PERSELBSN* matrix output lines 11 through 16 are applied through OR 3 to condition GT 28. At OT 5 of the *SEL* instruction, command 155 clears all the IO selection flip-flops except the read/write-0 flip-flop, which it sets. At PT 1, command 156 senses GT 28. Because this gate is conditioned, the result of the interrogation is that the tape-operation, tapes-not-ready, and tapes-not-prepared flip-flops are set, and the read/write-0 flip-flop is cleared.

Command 156 also senses GT's 31 through 36. One of these gates is conditioned by an output from the *PERSELBSN* matrix. The conditioned gate passes the pulse to one of six gates, GT's 38 through 43. If the tape system is under computer control, these gates are conditioned and the pulse is passed to set the flip-flop associated with the selected tape unit. The output level from the selected flip-flop prepares the selected tape unit for the subsequent transfer of data.

5.3.8 Burst-Time Counters, *SEL* $(21)_8$

The burst-time counters are selected for a reading operation only. Figure 5–14 shows the circuits immediately associated with the burst-time-counter flip-flop. During the execution of the *SEL 21* instruction, *PERSELBSN* matrix output line 21 conditions GT 21. At OT 5 of the *SEL* instruction, command 155 clears all the IO selection flip-flops. At PT 5, command 156 senses all the select gates. Since GT 21 is conditioned, it passes the pulse to set the burst-time-counter flip-flop. The burst-time-counter flip-flop conditions the transfer control circuits in the IO element which affect the transfer of the contents of the burst-time counters during the subsequent read operation.

PART 6
IO ELEMENT

CHAPTER 1
INTRODUCTION

1.1 GENERAL

The purpose of Part 6 is to explain the circuits of the IO element in the Central Computer and their interrelationship with the circuits of the IO units. These units are: the card reader, card punch, line printer, tape drive units, warning lights, the manual input matrix and burst time counters in the Input and Output System, respectively, the IO register, and each of the 75 drum fields in the Drum System.

Previous parts in this manual have discussed the instruction control, arithmetic, program, and selection elements of the Central Computer System. The primary function of these elements is the control and computation of data which has been loaded into the Central Computer under program control. It is significant to realize, however, that the results of computed data must be periodically transferred out to some external device and that new data must be periodically brought in from some other external device. Data transfers into or out of the Central Computer, or IO operations, occur at specific times as determined by program instructions of the IO class.

Refer to Part 1, paragraph 3.7 for a general discussion of the IO element and a simplified diagram of the IO element in the Central Computer System.

1.1.1 Basic IO Program

There are five program instructions in the IO class: *SDR (Select Drums)*, *SEL (Select)*, *LDC (Load IO Address Counter)*, *RDS (Read)*, *and WRT (Write)*. A combination using either *Select* instruction, the *LDC* instruction, and either the *RDS* or *WRT* is required to initiate any IO operation. This combination is called the basic IO program. For example, the *SDR* and *LDC* instructions followed by the *RDS* instructions will prepare the IO element and Drum System circuits to perform an input transfer from a specific drum field to the memory element. Similarly, the *SEL* and *LDC* instructions followed by the *WRT* instruction prepare circuits in the IO element and a specific IO unit to perform an output transfer from the memory element to the IO unit specified by the *SEL* instruction.

The status of card machines or tape drives may be checked to ascertain whether these IO units are ready to engage in the IO operation. To perform this check function, *Branch and Sense* instructions *BSN (10)* and *BSN (11)* are used in conjunction with the basic sequence previously mentioned. The computer program can determine whether any type of IO operation is in progress by executing the *BSN (14)* (*Sense IO Interlock*) instruction. The IO interlock function is explained in 1.1.3.

The detailed function of the *SEL* and *SDR* instructions is explained in Chapter 5 of Part 5.

1.1.2 Break Cycles

The transfer of a single word into or out of the memory element requires one memory cycle (6 μsec). The rate at which successive word transfers can be made depends on the speed of the selected IO device. The transfer rates vary between one word every 10 μsec (for a drum field) and one word every 25 ms (for a card punch). The difference in speed between the IO devices and the computer machine cycle permits the execution of internal operations between IO word transfers. When an IO word transfer is to be made the IO device issues a break request. (The term break refers to interrupting internal operations.) The computer when executing a program normally senses every machine cycle for the break request at TP 11 and recognizes the break request by initiating a break cycle. (During an IO pause, explained in 1.1.3, the computer checks for break request at a 2-mc rate.) It is during the break cycle (designated BO for breakout and BI for break-in) that the IO transfer is executed. (Pulses used to effect the transfer are gated time pulses and are designated BI (or BO) 0, 2, 4, etc.) If during the execution of a break cycle another break request is made, the computer recognizes (at TP 11) the second break request by immediately instituting another break cycle. If no break request has been made the computer

will resume execution of internal operations from the point at which it was interrupted.

1.1.3 IO Interlock

The AN/FSQ-7 is designed so that only one IO device can be operated on at any one time. Once an IO operation has been initiated, that is, a *WRT* or *RDS* instruction has been executed, an attempt to execute any instruction in the IO class will cause the computer to go into an IO pause (i.e., cease execution of program until IO operation complete). The control feature which prevents the execution of an IO instruction while an IO operation is in process is called the IO interlock. To prevent placing the computer in an IO pause, a sense code (*BSN 14*) is provided so that the status of the IO interlock can be determined by the program.

The IO interlock control feature also prevents the computer from being halted (either with a *HLT* instruction or by manual pushbutton) while an IO transfer operation is in progress. The IO operation is completed and then automatically halted if the *HLT* instruction is issued during the IO operation.

1.1.4 Termination of IO Transfer

Normally the IO transfer is terminated and the involved IO unit disconnected when the number of words specified in the *RDS* or *WRT* instruction are transferred into or out of the memory element. This is accomplished through the use of the IO word counter and an IO word counter equals 0 pulse.

Recall that the IO word counter is loaded from the address register during the execution of the *RDS* or *WRT* instruction. Actually, the complement of the number of words to be read or written is placed in the word counter. This permits adding 1 to the word counter each time a word is transferred to effectively reduce the contents of the counter. When the last word is transferred, the word counter is stepped from negative 0 to positive 0. (It appears that the stepping of the IO word counter from negative 0 to positive 0 would provide an additional word to the number of words specified in the *RDS* or *WRT* instruction. However, during the execution of the *RDS* or *WRT* instruction the IO word counter is stepped one count before any words are transferred. This compensates for the final stepping of the IO word counter, and the number of words transferred is exactly the number specified in the *RDS* or *WRT* instruction.) With this final stepping an IO word counter equals 0 pulse is generated (end-carry pulse) which sets up controls to clear the IO interlock, and to send a disconnect pulse to the IO unit to prevent further word transfers.

Under certain conditions the involved IO unit will itself terminate the IO transfer by generating a disconnect pulse which clears the IO interlock even though the IO word counter has not been stepped to 0.

An example of this is when a tape drive unit is being read by the computer. Data is written on the tapes by records, each of which may contain any number of words. When the computer reads back the data in a particular record with the *RDS* instruction specifying more words than are in the record, the tape adapter will generate a disconnect pulse when the end-of-record is sensed which clears the IO interlock.

In the case of status drum fields, a disconnect pulse is generated by the Drum System after one complete revolution of the drum ($2{,}048_{10}$ registers). If the involved *RDS* or *WRT* instruction specifies more than $2{,}048_{10}$ words, a maximum of $2{,}048_{10}$ words are transferred regardless of the setting of the IO word counter. There are other cases of the IO unit terminating IO transfers, but these are discussed in the chapters explaining the IO transfers involving these units.

1.2 LOGIC ANALYSIS OF IO ELEMENT

The diagram of the Central Computer System (fig. 1–25) shows the general circuit contents of the IO element. (See fig. 1–27 for a more detailed diagram of the IO element.) In the following explanations in this chapter and the remaining chapters in the part, reference is often made to engineering logic as well as to simplified diagrams. Table 6–1 provides the reader with a list of the actual engineering logic involved.

Other blocks on figure 1–6 (selection circuits and transfer circuits) not included in the table are spread over several logic sheets in the 0.7 logic group. Direct reference will be made to these circuits as they are applicable to the explanation.

The logic analysis is broken down in this manner: the remainder of Chapter 1 covers those circuits which are common to all types of IO transfers; Chapters 2, 3, 4 and 5 will cover the logic analysis particular to each type of IO unit.

TABLE 6–1. IO ELEMENT ENGINEERING LOGIC

REGISTER OR LOGIC CIRCUIT	LOGIC NUMBER	GRID LOCATION
IO Address Counter	0.4.1	E 1-5
Break Cycle Generation	0.2.3	A-E 1-7
Left IO Buffer and Left IO Register	0.7.1	All
Right IO Buffer and Right IO Register (also Drum Control Register)	0.7.2	All
IO Word Counter	0.7.3	A-B 2-6
IO Termination Controls	0.7.3	C-E 5-11

1.2.1 Break Cycle Generation

The initial conditioning of the IO control circuitry for an IO operation is accomplished through the execution of the *SEL* and *LDC* instructions. With the execution of the *RDS* or *WRT* instruction further setting up is performed and the transfer operation is initiated with a start-read or a start-write pulse. However, every transfer of a word requires the initiation of a break cycle.

Three flip-flops (0.2.3, B-C 2-3) are involved in this portion of the operation: break-request-sync, break-request, and break. The break-request-sync flip-flop is set by any one of six pulses, provided that at the time the pulse is generated the word counter is not equal to 0 (note gate). The source of these pulses will be covered later in connection with the transfers to or from the various IO units. The first 2-mc pulse generated after the setting of break request sync will set the break-request flip-flop. The gate conditioned by the 1 side of break-request flip-flop is strobed by a TP 11 pulse or a 2-mc pulse (if pause no break condition). A gated TP 11 then clears the break flip-flops. (A second break flip-flop exists in unit 4. It is set and cleared with the one on 0.2.3, but its output levels perform different functions.) The break request flip-flop is also set directly by a 2-mc pulse generated under an IO register empty-IO buffer full condition. This condition occurs only during a read operation of drums, BTC (burst time counters), or MI matrix.

The level output from the break flip-flop is applied to two 2-way AND circuits (0.2.3 D4). The second input to each of the AND circuits is from the write and read flip-flops. The output from the AND circuit with the read level input is applied to gates which pass TP's to generate the BI pulses. The output from the AND with the write level input is applied to gates which pass TP's to generate BO pulses. The BI pulses or the BO pulses are distributed to control the transfer of the word for which the break cycle was initiated. Tables 6–2 and 6–3 give the function of the BI and BO pulses, respectively.

1.2.2 IO Pause Circuits

Whenever a *RDS, WRT* or certain *PER* instructions (that affect selected IO devices) are executed, the IO interlock flip-flop is set to indicate that an IO operation is in process and accordingly prevent the execution of an IO class instruction or the *HLT* instruction. During the execution of these interlocked instructions, the IO interlock is effectively sensed and, if found set, the pause flip-flop (0.2.2, B8) is set. Actually, an IP 10 strobes a gate conditioned by a level which is up when the IO interlock is set during an IO or *HLT* PT cycle. With the pause flip-flop set, the break or no pause level will be down except when the break flip-flop is set. This level

TABLE 6–2. FUNCTIONS OF BI PULSES

PULSE	FUNCTIONS
BI 0	1. Clear break request FF 2. Transfer IO address counter to MAR 3. Start memory
BI 1	Not used
BI 2	1. Step IO address counter (0.4.1) 2. Clear IO register status flip-flop (0.7.7) 3. Step IO word counter (if drums not selected) 4. Transfer IO registers to memory buffers
BI 3	1. Start parity count (0.1.2) 2. Inhibit sample pulse (0.1.4) 3. Clear parity write flip-flop if IO word has parity bit (0.1.1)
BI 4	Transfer memory buffers to test registers (if test memory selected) (0.1.1)
BI 5	Not used
BI 6	Generate IO register break request pulse (if IO register selected) (0.7.5)
BI 7	1. Transfer IO buffer to IO register (if card machine operate flip-flop set; 0.7.6) 2. Sense parity check control flip-flop (0.1.1, B11) 3. Parity count (if no parity assigned)
BI 8	Not used
BI 9	Generate card mach 2nd break request (if 2nd break request flip-flop set; 0.7.6)
BI 10	Not used
BI 11	1. Generate clear IO interlock pulse (if word counter equals 0 and if IO buffer and IO register status is empty or BTC or MI matrix selected) 2. Generate word counter equal 0 pulse for tape adapter (if word counter equals 0 and tapes selected, 0.7.8) 3. Generate clear IO interlock pulse if IO register selected

is applied to an AND in the output of the TPD 0 flip-flop (0.2.3, B11) and when the break or no pause level is down the TPD cannot be stepped and no TP or IP pulses can be generated. The effect of inhibiting the generation of TP's and IP's renders the computer incapable of further execution of the instruction currently held in the operation register. This condition is called an IO pause.

When the break flip-flop is set to indicate that the next machine cycle will be a break cycle, the break or no pause level comes up and the TPD starts running. To prevent the generation of IP's during the break cycle to maintain the IO pause condition, a gate (0.2.2, C7) connected to 0 side of the break flip-flop is deconditioned. This gate normally passes 2-mc pulses to the IP driver line of the TPD.

TABLE 6–3. FUNCTIONS OF BO PULSES

PULSE	FUNCTIONS
BO 0	1. Clear break request FF 2. Transfer IO address counter to MAR 3. Start memory
BO 1	Not used
BO 2	1. Step IO address counter (0.4.1) 2. Step IO word counter (if MI matrix burst time counter, or drums not selected, 0.7.3)
BO 3	Not used
BO 4	Not used
BO 5	Transfer test memory to memory buffer (if test memory selected, 0.1.3)
BO 6	Generate IO register break request pulse (if IO register selected, 0.7.5)
BO 7	1. Transfer memory buffers to IO register 2. Start parity count and set parity check flip-flop 3. Set IO register status flip-flop
BO 8	1. Transfer IO register to tape write register (if tapes selected) 2. Transfer IO register to warning light register (if warning lights selected)
BO 9	Not used
BO 10	Not used
BO 11	1. Generate clear IO interlock pulse (if word counter equals 0 and drums selected) 2. Generate word counter equals 0 pulse for tape adapter (if word counter equals 0 and tapes selected) 3. Clear write flip-flop (if warning lights selected)

CHAPTER 2
MAIN AND AUXILIARY DRUM TRANSFERS

2.1 GENERAL

The Drum System of the Combat Direction Central has two functions: It serves as temporary storage and transfer point for most of the data entering and leaving the Central Computer and it also provides auxiliary memory storage facilities for the computer. Most of this storage space is used for the storage of program data.

The transfer of data between the Central Computer System and the Drum System is designated as a CD (Central Computer System to and from Drum System) operation. Data transfers between the Drum System and external systems are designated OD (other than Central Computer System to and from the Drum System) operations. The following paragraphs discuss the logical circuits of the computer involved in CD operations.

The Drum System contains 39 separate fields located on six physical main drums and 36 separate fields located on six physical auxiliary drums. With the exception of the nine radar data (RD) fields and the three output buffer (OB) fields, each drum field has a maximum storage capacity of 2,048 registers. The radar data fields have a maximum storage capacity of 2,060 registers, but only 2,048 registers can be used for data storage; the output buffer fields have a maximum storage capacity of 2,048 registers, but only 2,036 registers can be used for data storage. Except for the OB fields each drum field is treated as a distinct IO unit by the computer. The fields of the Drum System differ from one another with respect to mode of reading and writing data, the presence or absence of parity bits in the stored word contents, and in the type of information stored. Furthermore, some drum fields only accept information from the computer (write), whereas others only supply information to the computer (read). Because of the inherent characteristics of the various fields, the method of instituting and executing IO processes with the computer varies from field to field.

The drum fields are read or written under different methods of control. The input fields (gap-filler input, long-range-radar input, and manual input) are read by the status-identification mode of operation. The cross-telling fields are read by a modification of status identification called the marker status mode. The output buffer fields are written by still another modification of the status mode. All other fields of the Drum System are written or read by the address mode of operation or by a modification of the address mode known as the interleave mode.

2.2 INITIATION OF DRUM TRANSFERS

Execution of the *SDR* instruction indicates which drum group and the particular field within that group which is selected. The drum group (main or auxiliary) is indicated by the contents of bit R1 of the *SDR* instruction. If the drum field selected is addressable, the right half-word of the *SDR* instruction indicates the starting location on the drum field for the transfer. If a drum field is read by status identification, the identity code to be used is contained in the right half-word of the *SDR* instruction. The starting location in memory for the subsequent operation is specified by the *LDC* instruction. Finally, the direction of transfer is specified by a *RDS* or *WRT* instruction. The number of words involved in the operation is indicated by the right half-word of the *RDS* or *WRT* instruction.

Once these three instructions have been executed, the IO element has all the necessary information to control the IO operation, and a start-read or start-write pulse will be developed in the IO element and sent to the Drum System.

2.3 READING OPERATION

2.3.1 Address Mode

Reading of the auxiliary-memory, track display, radar-data digital-display and intercommunication drum fields is accomplished by using a method called the address mode.

In the address mode the address of the *SDR* instruction specifies the starting register of the selected field with which the transfer is involved. The drum control register is loaded with the address of this first drum register. In the Drum System the angular position counter (APC) contains, at any time, the address of the register which is about to come under the drum read-write heads. By transferring the contents of the APC to the IO buffer register a comparison check can be made with the specified address in the drum control register. By repeatedly (at a 10-μsec rate) transferring the APC and making this comparison check, the circuits can determine when to begin the transfer of data words themselves.

The data words read from the drum at a 10-μsec rate are transferred to the IO buffer register to be read

into memory after passing through the IO registers and the memory buffer registers. An IO buffer loading pulse is associated with each word transferred to the IO buffer register. This pulse (a CD-1 delayed) is synced with the 2-mc pulses to initiate the control function which generated the break request and transfers the IO buffers to the IO registers. It is during the break-in cycle that the word is transferred from the IO registers to the memory buffers.

The operation is terminated when the word counter becomes equal to zero. At this time the IO interlock is cleared and a drum-disconnect pulse is subsequently generated and sent to the Drum System to disable the drum read circuits.

Figure 6–1 (foldout) is a simplified logic diagram of the circuits of the Central Computer System and the Drum System utilized during an address-mode read operation for which a main drum field is selected. The control portion of an address-mode read operation is initiated in the Drum System by the deselect pulse. The deselect pulse is transmitted to the Drum System at OT 5 of the *SDR* instruction where it clears the CD field-selection register and the CD selection decoder, triggers SS 2, clears flip-flops 2 and 3, and sets FF 1.

The output of SS-2, when fired, is a negative level for a 120-μsec duration. Its purpose is to decondition GT 1 through AND 5 and AND 6 and thereby prevent gating of CD pulses until the drum field selection levels have settled.

When the computer executes the *RDS* instruction, the start-read pulse arrives at the Drum System to clear FF 4, set FF 2, set FF 1 and strobe GT 3. (The setting of FF 1 and the strobing of GT 3 have significance when the *RDS* instruction is executed immediately following a write operation on the same drum field without another *SDR* instruction being issued. In this case the pulse strobing GT 3 fires SS 1 which serves a function similar to that of SS 2, that is, to provide a 120-μsec delay until the drum head circuits have settled from the write to the read condition.) The output of FF 2 brings up one input to AND 4 and one input to AND 1. The output of FF 4 brings up one input to AND 3. The entire control circuit is now partially conditioned to begin the comparison transfer process.

After the 120-μsec delay period effected by SS 2 has expired, AND 5 is fully conditioned and it in turn brings up the second input to AND 4 which conditions GT 1. The next CD 4 pulse generated by the CD time pulse distributor will pass GT 1 to set FF 3. The one output of FF 3 brings up the second input to AND 1, the output of which brings up the second input to AND 3 and one input to AND 2. The output of AND 3 is applied to the CD time pulse distributor as a start-compare level conditioning that circuit to transfer the APC with a CD 1 pulse to the IO buffer register and to pass a CD 1 as a start-compare pulse. The start-compare pulse is sent to the IO element to make the comparison of the words in the IO buffer and the drum control register, and also to set up control circuits preparatory to accepting the IO word from drums in the event a successful comparison is made (explained in detail below).

In the Drum System the start-compare pulse sets FF 4 to bring up the second input to AND 2 which in turn conditions GT 2. This action is preparatory to gating the next CD-1 pulse to read the drum coming under the read heads in the event that a successful comparison is made.

Note that both the IO element and the Drum System are set up to effect the transfer of a word from drum to core pending the results of the comparison going on in comparison circuits. If the comparison is successful the word is transferred. If unsuccessful, a no-compare pulse is generated and distributed to disable the circuits in both the IO element and the Drum System. In the Drum System the no-compare pulse clears FF 4. This results in deconditioning GT 2, blocking the CD-1 pulse which reads out the drum word to the CD read bus. The zero output of FF 4 also brings up the second input to AND 3 again, and the start-compare level is reapplied to the CD time pulse distributor to cause the repeat of the entire comparison process just described, but for the new contents of the APC.

The start-compare pulse which is transmitted to the computer sets the accept flip-flop and, after being delayed for 1 μsec, senses the five identity-mode flip-flops and the address-mode flip-flop. The 1-μsec delay is necessary to allow the drum address in the right IO buffer register to stabilize before the comparison is performed. Since this discussion assumes that the read operation involves an addressable drum field, the address-mode flip-flop is set. Consequently, the start-compare pulse is gated by GT 9 and routed to the comparison circuits.

At the completion of the comparison process, the right IO buffer register is cleared. If the current drum address held by the right IO buffer register does not agree with the desired starting address held by the drum control register, a no-compare pulse is generated by the comparison circuits. The no-compare pulse is routed through OR 6 and OR 5. The output pulse of OR 5 clears the accept flip-flop and that of OR 6 senses the address-register-bit-R1 flip-flop. The d-c level generated by the 0 side of the accept flip-flop indicates to the circuits of the computer that the comparison process was unsuccessful. As previously stated, a 1 in the bit-R1 position of the right half-word of the *SDR* instruction specifies the selection of an auxiliary drum field, and a 0 in the bit-R1 position specifies the selection of a main drum field. Since this discussion assumes that a main drum field has been selected, the address-register-bit-R1 flip-flop is clear, and, as a result, GT 9 is deconditioned, and GT 20 is condi-

tioned. Accordingly, the no-compare pulse which was routed through OR 6 is gated by GT 20. The resultant output pulse of GT 20 is transmitted to the CD read-write control circuits of the main drums as a no-compare pulse.

As shown in figure 6–1, the no-compare pulse is routed through OR 9 to clear FF 4. The d-c level generated by the 0 side of FF 4 conditions AND 3 and as a result AND 3 again applies a start-compare level to the CD time-pulse distributor. The CD time-pulse distributor causes the new address contained in the angular-position counter to be transmitted to the right IO buffer register of the Central Computer System with directions to repeat the comparison process. Accordingly, the start-compare pulse, which is generated by the CD time-pulse distributor, sets FF 4 and is routed to the comparison circuits of the computer.

The comparison operation is repeated until a successful comparison is achieved, at which time the no-compare pulse is not generated, and FF 4 remains in its 1 state; as a result, the d-c level generated by AND 2 conditions GT 2. Gate tube 2 is sensed by a CD 1 address-select pulse generated by the CD time-pulse distributor. The output pulse of GT 2 is delayed for 1.1 μsec to form a read-sample pulse and is routed to the CD read circuit, causing reading of the word which is under the drum read heads at that time. The read-sample pulse is also routed to the Central Computer System as an IO-buffer-load pulse. As a result of the generation of the read sample pulse, the word under the drum read heads is transferred from the CD read circuit to the IO buffer register by way of the CD read bus. From this point until the termination of the IO process, no more CD 1 start-compare pulses are generated. An IO-buffer-load pulse is transmitted to the computer each time the contents of a drum field register are transferred to the IO buffer register.

The IO-buffer-load pulse (CD 1) which is routed to the computer sets the IO-buffer-load sync flip-flop after being delayed for 1 μsec. This pulse is coincident in time with the transfer of the word from the drum register to the IO buffer registers. The 1-μsec delay is introduced so that the IO buffer registers can stabilize before attempting to transfer a word out of them. After the IO-buffer-load sync flip-flop is set, the next 2-mc pulse is gated through GT 9 to set the IO-buffer-load flip-flop. The d-c level generated by the 1 side of the IO buffer load flip-flop conditions GT 10. Gate tube 10 is also sensed at a 2-mc rate, and the resultant output pulse simultaneosuly clears the IO buffer load and the IO buffer load sync flip-flops and strobes GT 11. Gate tube 11 is conditioned by the d-c level generated at the 1 side of the drum operate flip-flop (set at PT 5 of the *SDR* instruction). The resultant output pulse of GT 12 senses GT 13 to determine the status of the accept flip-flop.

As shown in figure 6–1, the accept flip-flop is cleared at PT 6 of an *RDS* or a *WRT* instruction or by the no-compare pulse and is set by the CD 1 start-compare pulse. The sequence in which the above occurs is as follows: At PT 6 of an *RDS* or a *WRT* instruction, the accept flip-flop is cleared. When the drums transmit a CD 1 start-compare pulse, the accept flip-flop is set. If the comparison is unsuccessful the no-compare pulse clears the accept flip-flop. At this point in the discussion it is assumed that a successful comparison has been achieved; consequently, the no-compare pulse is not generated, and the accept flip-flop is in its 1 status when interrogated by the pulse which is gated through GT 11. The output pulse of GT 11 is gated through GT 12 to set the IO buffer-status flip-flop and to step the IO word counter.

The IO-register-status flip-flop was initially cleared at PT 1 of the *RDS* instruction by command 294. Consequently, the d-c level generated by the 0 side of the IO register status flip-flop and the d-c level generated by the 1 side of the IO buffer status flip-flop condition AND 7. The resultant d-c level generated by AND 7 conditions GT 16. Gate tube 16 is sensed at a 2-mc rate and the resultant output pulse, which is the drum-read break-request pulse, performs the following operations:

a. Transfers the contents of the IO buffer register to the IO register
b. Sets the break request sync flip-flop
c. Clears the IO-buffer-status flip-flop
d. Sets the IO-register-status flip-flop
e. Senses the status of the interleave flip-flop.

Clearing the IO-buffer status flip-flop and setting the IO register status flip-flop ensures that only one break-request pulse is generated for each word transferred to the IO buffer registers.

As soon as the break request is honored, a break-in cycle is initiated to complete the transfer of the data word currently in the IO registers to its destination in the memory element. At BI 0 (delayed), the contents of the IO address counter are transferred to the memory address register. At BI 2, the IO address counter is stepped. The BI 2 pulse also transfers the data word contained in the IO registers to the memory buffer registers, clears the IO registers, and clears the IO-register-status flip-flop. At BI 3, a parity count is performed. The BI 7 pulse senses the parity check control flip-flop and the BI 11 pulse determines the status of the word transfer by simultaneously sensing the IO-word counter status flip-flop and the break request flip-flop. At the end of the break-in cycle, the first word has been placed in its specified memory register.

The Drum System asynchronously transmits another CD-1 buffer-load pulse to the Central Computer System and the entire process is repeated. When the

next-to-last word is in the IO buffer registers, the IO word counter, after being stepped, contains negative 0 and the next-to-last break-in-cycle is requested. When the last word is in the IO buffer register, the IO word counter is stepped to positive 0 and an end-carry pulse is generated. The 2-mc pulse, which steps the IO word counter, also requests the last break-in cycle. The end-carry pulse clears the IO word counter status flip-flop and no new break request is honored. As soon as the IO word counter contents are equal to positive 0, AND 10 generates a d-c level which is applied to AND 11. (See fig. 6–1.) The d-c level generated by AND 11 conditions GT 21 and, as a result, the next 2-mc pulse is gated by GT 21. After being gated by GT 21, the 2-mc pulse branches. In one branch, the output pulse is delayed 0.5 μsec and then routed through OR 8 to clear the disconnect drum-control flip-flop; in the other branch, the output pulse of GT 21 is routed to the main drums as a disconnect-main-drum pulse.

Following the drum-disconnect pulse, the last word is loaded into its memory location by the break-in pulses which are issued during the current break-in cycle. When the IO buffer register and the IO register are both empty, the d-c levels generated by the 0 side of their respective status flip-flops partially condition AND 6. The third d-c level required to fully condition AND 6 is generated when the IO word counter contents are equal to 0, and the resultant d-c level conditions GT 14. Gate tube 14 is pulsed at BI 11 of the break-in cycle, and the resultant output pulse is gated through GT 15 (which is conditioned by the 1 side of the drum operate flip-flop) to clear the IO-interlock flip-flop.

As it may not always be desirable to read consecutive drum field registers, provisions are made to read every 8th, 16th or 64th drum register by the process of interleaving. Interleaving by 8, 16 or 64 is specified by one of the index interval bits, L13, L14 or L15 respectively, of the *RDS* instruction. At PT 6 of *RDS*, gates conditioned by the levels for these bits are strobed. Whichever of the gates is conditioned, an output pulse is sent to set the interleave flip-flop and to set the corresponding interleave mode flip-flop, 8, 16 or 64.

Going back in the read operation where a successful compare has been accomplished and GT 16 is conditioned by the output of AND 7, the 2-mc pulse passed by GT 16 sets the IO register status flip-flop, transfers the IO buffer to the IO register, and senses GT 18. This gate is now conditioned from the one side of the interleave flip-flop and the output pulse, called a false-no-compare, is sent to the Drum System and to the drum control register as a stepping pulse. A false no-compare pulse performs the same function as a true no-compare pulse, that is, to clear FF 4 and bring up the start compare level. The drum control register stepping pulse strobes the gates conditioned by the interleave mode flip-flops. The output pulse from the conditioned gate is sent to the appropriate flip-flop in the drum control register to effectively add 8, 16, or 64 to the contents of the drum control register.

From this point on, the reading process is identical to that described for simple address controlled reading, with one exception. After each successful address comparison, the output pulse from GT 16 passes through GT 18, which is conditioned because the interleave flip-flop is set. This pulse is passed by GT 5, GT 6, or GT 7 and steps the drum control register by 8, 16, or 64, depending on the request expressed by bit L13, L14, or L15 of the *RDS* instruction. The output of GT 18 also passes through OR 6, to be gated by GT 20 and sent to the main drums as a false no-compare pulse, which clears FF 4, causing the Drum System to execute a new address search.

In this way, only every 8th, 16th or 64th drum register is read from the Drum System into the memory element of the computer. When the IO word-counter contents are equal to positive 0, the IO interlock is cleared and the computer transmits a disconnect pulse to the Drum System.

2.3.2 Status Mode

In reading by the status method, the Drum System examines each register of the selected field and reads each word found. For control purposes, each of the fields which may be read by the status method has two status channels whose contents indicate the status of each register in the field. A 1 bit in the CD status channel indicates to the CD read circuits the presence of a word in the corresponding register; a 0 bit means that the register is empty. The OD status channel provides the same information to the OD writing circuits.

When the CD status control circuits of the Drum System detect a 1 bit in the CD status channel, the word in the corresponding register is transferred into the IO buffer register of the computer and the status control circuits write a 0 bit in the OD status channel to indicate to the OD circuits that the register is empty and available for the writing of a new data word from the Input System.

Simultaneously with the transfer of the word to the IO buffer register, a compare pulse and an IO-buffer-load pulse are sent to the IO element in the computer. The only function of the compare pulse is to set the accept flip-flop. In the status mode, the address mode flip-flop and the five identity mode flip-flops are all cleared, and the compare pulse is not gated to the comparison circuits. A no-compare pulse is never developed. The IO-buffer-load pulse performs the same function as in address mode reading, and the word is stored in the memory element.

In this mode of operation, the status disconnect counter in the Drum System counts the drum registers

that are examined for the presence of words. After one complete revolution of the drum (2,048 registers) the end-carry pulse from the counter disconnects the drums, terminating the reading operation.

2.3.3 Status Identification Mode

Reading in this mode is similar to that in the status mode except that an identification process is used. One of the identity flip-flops is set at PT 5 of the *SDR* instruction. The flip-flop to be set depends upon the field selected. The identity flip-flop conditions a gate tube that passes the compare pulse to the comparison circuits where it compares the identity bits of the word from the drum with the desired identity bits which were placed in the drum control register by the *SDR* instruction. When the comparison is successful, a no-compare pulse is not generated, and the word is stored in memory as in the status mode. If, however, the comparison is not successful, a no-compare pulse is generated to clear the accept flip-flop, thus preventing the transfer of the word to the memory element. The no-compare pulse also goes to the Drum System, where it causes the status control circuits to write a 1 instead of a 0 in the OD status channel, indicating to the OD writing circuits that the register still contains information and cannot be used for the writing of a new word.

The various modifications of the status identification mode used for the transfer of multiword messages are essentially the same as status identification. Where the IO element is concerned, the only difference is that a compare pulse is received only with the first word of the message. If the comparison is successful, all words in the message are accepted because the accept flip-flop remains set as a result of the compare pulse transmitted with the first word. In the case of an unsuccessful comparison, all the words are rejected since the accept flip-flop had been cleared by the no-compare pulse resulting from the first word.

2.4 WRITING OPERATION

When the computer is writing on a drum field, the original information exists in a memory location and its ultimate destination is a drum register on the selected drum field. The word passes through three registers: the memory buffer register, the IO register, and the drum write register. (See fig. 6–2, foldout.) The drum write register, physically located in the Drum System, serves the same purpose during a breakout cycle that the IO buffer register serves during a break-in cycle. During the time interval between BO 2 and BO 6 of the breakout cycle, the first data word to be transferred is placed in the MBR's. At BO 7 of the breakout cycle, the data word is transferred from the MBR's to the IO registers. The data word is then transferred from the IO registers to the drum write register at approximately BO 8. These transfers occur as the result of a break request made by the IO element during the execution of a *WRT* instruction. During the first breakout cycle, the circuits of the IO element request a second breakout cycle. As a result, the second data word is placed in the IO registers at BO 7 of the second breakout cycle. Both processes, the one placing a data word in the drum write register and the other placing a second data word in the IO registers, are completed before the first data word is written on the drum surface. Once the first data word is written the Drum System generates a word-demand (break-request) pulse to initiate successive breakout cycles.

The setting up of the controls by the *WRT* instruction is the same for both the address and the status modes.

The final command of the *WRT* instruction, *start write* (182), is generated at PT 6 and gated by GT 30. As shown in figure 6–3, the resultant pulse performs the following operations:

a. Senses index interval bits L13, L14, and L15 for interleaving
b. Clears the sense-IO-word-counter flip-flop
c. Sets the write flip-flop (if the IO word counter contents are not equal to 0)
d. Gated by GT 27 or 28 clears write register status flip-flop and becomes:
    1. Start write pulse
    2. First break request pulse, and sets:
        a. Not read drums flip-flop
        b. 2nd break request flip-flop
        c. Write drums flip-flops.

Bit position L13, L14, or L15 of the index interval register specifies a mode of interleaving when the write operation specifies writing on an addressable field of the Drum System. If a 1 appears in any one of these three bit-positions, the associated gate tube (GT 9, GT 7, or GT 8, respectively) gates the PT 6 pulse, which sets the appropriate interleave-mode flip-flop. Note that the gated PT 6 pulse is also routed through OR 5 to set the interleave flip-flop. Command 182 clears the sense-IO-word counter flip-flop (0.7.3, D8) in anticipation of the sense-IO-word-counter operation at PT 3 of

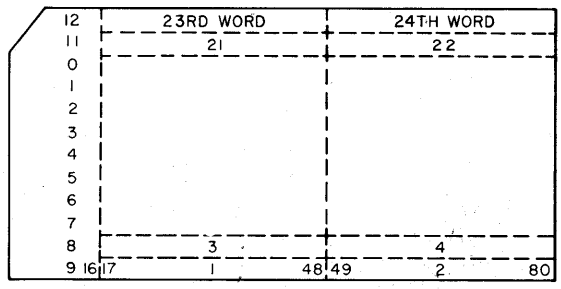


Figure 6–3. IBM Card, Binary Word Layout

the next *RDS* or *WRT* instruction. The PT 6 pulse is also gated through GT 10 to set the write flip-flop. Gate tube 10 is associated with the 1 side of the IO-word-counter-status flip-flop. Consequently, the write flip-flop is not set unless the IO word counter is not equal to 0. The accept flip-flop, which normally reflects the result of a comparison process, is also cleared by command 182 in anticipation of the comparison process which is initiated during the subsequent data transfer. If a main drum field is selected, the PT 6 pulse is gated through GT 28 (fig. 6–3) and routed to the main drums as a start-write pulse. If an auxiliary drum field is selected, the PT 6 pulse is routed to the auxiliary drums as a start-write pulse by way of GT 27.

The output pulses from GT 28 and GT 27 are also applied to OR 7, and the resultant output pulse from OR 7 is the first break-request pulse. In addition to requesting the first break cycle, the output pulse of OR 7 sets the not-read-drums flip-flop, sets the second-break-request flip-flop, and the write-drums flip-flop, and passes through OR 8 to clear the write-register-status flip-flop. Since the write flip-flop is set and a break request has been issued, the computer initiates the generation of breakout pulses.

The breakout cycle commences as follows (not illustrated on fig. 6–3): The first breakout pulse (BO 1) clears the break-request flip-flop and transfers the contents of the IO address counter to the MAR. The memory address register now contains the memory address of the first word which is to be written on the selected drum field. At BO 2, the status of the lock-address-counter flip-flop is examined; if this flip-flop is set, the BO 2 pulse steps the IO address counter so that this counter now specifies the memory address of the second word to be written on the selected drum field. At BO 7, a parity count is initiated and the first data word is transferred from the memory buffer registers to the IO registers. The BO 7 pulse also sets the IO-register-status flip-flop. As shown in figure 6–2, the d-c level which is generated by the 1 side of the IO-register-status flip-flop is applied to AND 6. Since the PT 6 pulse (command 182) set the write-drums flip-flop and cleared the write-register-status flip-flop, AND 6 is fully conditioned and the resultant output level partially conditions AND 7 and AND 8. Since it is assumed that a main drum field is selected, AND 7 is fully conditioned and the resultant output level from this circuit conditions GT 21. Upon being sensed by a 2-mc pulse (delayed 0.8 μsec), GT 21 gates the pulse, which transfers the first data word from the IO registers to the drum write register. The output pulse of GT 21 also passes through OR 6, with the following results:

a. The write-register-status flip-flop is set
b. The write-register-status-sync flip-flop is cleared
c. The IO-register-status flip-flop is cleared (through OR 4)
d. The second break-request flip-flop is examined by sensing GT 29.

The 1 in the write-register-status flip-flop and the 0 in the IO-register-status flip-flop reflect a correct representation of current circuit conditions; that is, the drum write register now contains the first data word and the IO registers are empty. The sensing of the second break-request flip-flop (by pulsing GT 29) results in the generation of a second break-request pulse, which requests a second breakout cycle and clears the second break-request flip-flop.

As a result of the generation of a second break-request pulse while the write flip-flop is set, the next memory cycle is assigned as a second breakout cycle. As in the first breakout cycle, a data word is transferred from the memory element to the IO registers by way of the MBR's. Therefore, at BO 7 of the second breakout cycle, the second data word is in the IO registers and the first data word is in the drum write register. The transfer of the second data word from the IO registers to the drum write register is inhibited at this stage of the IO operation because the 3-way AND circuit which controls this transfer is deconditioned by the d-c level generated by the 0 side of the write-register-status flip-flop. At BO 7, the IO-register-status flip-flop is again set, indicating that the IO registers contain the second data word of the transfer. At the end of the second breakout cycle, therefore, the first data word of the transfer is in the drum write register and the second word is in the IO registers. The computer now awaits a word-demand pulse from the Drum System in order to complete the IO process. Note that two breakdown cycles are executed as the result of a *WRT* instruction involving the Drum System.

2.4.1 Address Mode

In writing on the drums by the address mode, the first word transferred is written on a specific register of the selected field. The address search process is the same as that used in reading in the address mode. The control circuits in the Drum System are essentially the same as those used in reading. The primary difference is that when comparison is successful, a write pulse is generated in the Drum System rather than a read-sample pulse.

The write pulse activates the drum writing circuits, thereby causing the word in the drum write register to be written on the drum surface. The write pulse is also delayed 2.5 μsec. The delayed pulse clears the write register (not shown) and is transferred to the computer as a word-demand pulse indicates that the drum write register is empty and ready to receive a new word.

Figure 6–2 is a simplified logic diagram of the Drum System operation during an address mode write

operation. The word-demand pulse from the Drum System is fed to OR 13. The output pulse from OR 13 generates a break request, sets the word-demand-sync flip-flop, and examines the interleave flip-flop by sensing GT 15. The breakout cycle assigned as a result of the break request arising from the first successful comparison is the third breakout cycle in the drum write operation.

The d-c level generated by the 1 side of the word-demand-sync flip-flop conditions GT 18, which, as a result, gates the next 2-mc pulse. The gated 2-mc pulse sets the word demand flip-flop and simultaneously examines it by sensing GT 19. Since the word-demand flip-flop is still in its 0 state when examined, the gated 2-mc pulse is suppressed by GT 19. (See fig. 6–2.) However, since the word-demand flip-flop and the word-demand-sync flip-flop are now set, the next 2-mc pulse is gated through both GT 18 and GT 19. The resultant output pulse of GT 19 clears the word-demand and word-demand-sync flip-flops and steps the IO word counter. Therefore, after the first word-demand pulse is generated, the contents of the IO word counter specify N-1, N being the total number of data words to be written. The word-demand pulse then sets the write-register-status-sync flip-flop after passing through a 2-μsec delay line. As a result of the d-c level generated by the 1 side of the write-register-status-sync flip-flop, the next 2-mc pulse is gated through GT 20 and, consequently, the write-register-status flip-flop is cleared after a delay of 0.08 μsec. The output of the clear side of the write register status flip-flop brings up the third output to AND 6. The output of AND 6 conditions either GT 21 or GT 22 through AND 7 and 8. These gates pass the next 2-mc pulse which transfers the IO register to the write registers, clears the IO register, clears the IO register status flip-flop, clears the write register status sync flip-flop and sets the write register status flip-flops. The word now in the write register is read out of the memory register during the second break cycle. The IO register is now clear, ready to receive the third word from memory.

As previously stated, the generation of a word-demand pulse results in the setting of the break request flip-flop, ensuring the eventual assignment of the third breakout cycle. The first breakout pulse (BO 1) clears the break request flip-flop and transfers the contents of the IO address register to the memory address register.

The BO 7 pulse of the third breakout cycle transfers the third data word from the memory buffer registers to the IO register, executes a parity count, and sets the IO-register-status flip-flop. At the completion of the third breakout cycle, the second data word is in the drum write register and the third data word is in the IO registers.

Each CD-3 pulse writes another data word on the drum surface. The succeeding CD-4 pulse demands another breakout cycle, steps the IO word counter, and causes the transfer of the next word from the IO register to the drum write registers.

After the (N–1)th word is written on the drum surface, the Nth word is transferred to the write register, the IO word counter is stepped to negative zero, and another break request is generated. This break request and the following one are made only as a by-product of transferring the Nth word from the IO register to the write register and then writing the Nth word on the drum. The (N+1)th and (N+2)nd word will end up in the word register and the IO register respectively, from which they will be cleared.

After the Nth word has been written, the IO word counter is stepped from negative zero to positive zero with the resultant generation of the end-carry pulse. The IO word counter end-carry pulse clears the IO word counter status flip-flop and the write drums flip-flop. Since the write drums flip-flop is clear, AND 6 is deconditioned; consequently the transfer of the data word currently held in the IO registers does not occur. Since the IO word counter contents are equal to 0, AND 11 is conditioned and the resultant d-c level is applied to AND 13. Because of the d-c level generated by the 1 side of the disconnect-drum-control flip-flop (set at PT 3 of the *WRT* instruction) and the d-c level applied by AND 11, AND 13 generates a d-c level which conditions GT 25. This gate tube is sensed at a 2-mc rate, and the resultant output pulse is routed to the main drums as a drum-disconnect pulse. The Drum System receives the disconnect pulse about 3.75 μsec after the last CD 4 word-demand pulse. As shown in figure 6–3, the disconnect pulse passes through OR 3 to clear FF 4 and FF 3 and clears FF 2. Since flip-flops 4 and 3 are cleared, AND 3 is deconditioned; therefore, the generation of another write pulse is inhibited. The IO interlock is not cleared until BO 11 of the last breakout cycle, at which time GT 23 (conditioned by the output level of AND 10, through OR 9) gates the BO 11 pulse. The gated BO 11 pulse is again gated by GT 24 and, as a result, the not-read-drums flip-flop and the IO interlock are both cleared, thus terminating the address mode write drums operation.

Inasmuch as it may not always be desirable to write on consecutive drum field registers, provisions are made to write on every 8th, 16th, or 64th drum register. The interleave circuits start a new address search after each word transfer in the same manner as when a reading operation is being performed.

2.4.2 Status Mode

Computer-to-drums (CD) write operation under status control is identical to operation in the address mode except for the following variations.

At PT 5 of the *SDR* instructions, the address-mode

flip-flop is cleared. After the *WRT* instruction is executed and the drum write register and IO registers are loaded with the first and second data words, respectively, the CD status channel of the selected drum field is read at CD 1. A 0 status bit in this channel indicates that the drum register is empty and available for the storage of new information. Accordingly, the subsequent CD 3 pulse writes the contents of the drum write register (i.e., the first data word) onto the drum surface, and a 1 is written in the OD status channel to indicate a full drum register. The subsequent CD 4 pulse is sent to the computer as a word-demand pulse. This pulse steps the IO word counter, requests a break, and initiates the transfer of the next data word from the IO registers to the drum write register, as explained in the discussion of operation in the address mode.

CHAPTER 3
CARD MACHINE TRANSFERS

3.1 GENERAL

Three card machines are associated with the Central Computer: the card reader, the line printer, and the card punch. The card reader is used when instruction and data words are to be transferred from punched cards to the memory element (read); the card punch and the line printer are used when information (instruction and/or data) that is transferred from the memory element is to be punched on an IBM card or printed on a paper form (write). As a group, the card machines have the ability to transfer information into and out of the memory element, although each individual card machine can perform only one of these two functions.

The cards used with the card reader and the card punch are standard IBM cards. There are 80 columns on each card and 12 punch positions or rows in each column. The columns are divided into three fields: columns 1 through 16, 17 through 48, and 59 through 80. Punch positions in the rows across the second and third fields are used to store 32-bit binary data words; that is, one binary word is punched in each row of the second and third fields. Since there are 12 rows, the storage capacity of a card is 24 words. The distribution of words on a card is shown in figure 6–3. The first field is used for storing identification information, either in the form of 16-bit binary half-words or by Hollerith code.

When an IO process involves the card reader or the card punch, one row of information is transferred at a time; that is, two 32-bit words are processed simultaneously. The transfer sequence is words 1 and 2, then 3 and 4, and so forth, ending with the 23rd and 24th words. (See fig. 6–3.) An index point is associated with each row on the card. The card is at an index point when the row is under the brushes of the card reader or under the punches of the card punch. Thirteen index points constitute a card machine cycle: 12 for the 12 rows of binary words and the 13th denoting the point at which the rear edge of the card passes from under the brushes or punches. Thus, during a card-machine cycle, one full card is either read or punched by the Central Computer System.

In the subsequent discussion, attention is directed to the circuits of the IO element which, by responding to the IO program, effect the data-word transfers between the memory element and a selected card machine. A complete simplified logic diagram of all card-machine control circuits, except the sense and operate circuits, is given in figure 6–4, foldout. The following paragraphs discuss these circuits in detail, and the diagram will aid in an understanding of the overall operations and the interconnections among the several circuits.

3.2 INITIATION OF CARD MACHINE TRANSFERS

As with any IO device, three basic IO instructions are required to initiate an IO operation with any one of the card machines. The *SEL* instruction specifies which card machine is to be involved in the IO operation, and the *LDC* instruction specifies the starting memory location to be used. Since the card machines can become not ready during normal operation (run out of cards or paper), a *BSN 11* instruction is usually included with the IO program. If the card machine is not ready when the *BSN 11* instruction is executed, a branch of program control is performed which informs the operator that manual intervention is necessary. If the card machine is ready the program continues sequentially so that the applicable *RDS* or *WRT* instruction is executed. As a result, a start-read or start-write pulse is generated to initiate the operation of the selected card machine.

3.3 CARD READER TRANSFERS

The word transfer from the card reader is initiated by command 180, which occurs at PT 6 of the *RDS* instruction. As shown in figure 6–4, the IP 6 pulse is gated through GT 22, and then examines the status of the IO word-counter-status flip-flop (0.7.3, C10) and the card-reader flip-flop (0.7.6, B10) by pulsing GT 14 and GT 2, respectively. The PT 6 pulse also passes through OR 4 to clear the sense-IO-word-counter flip-flop (0.7.3, D8). The sense-IO-word-counter flip-flop is cleared at this time because it must be readied for a subsequent *RDS* or *WRT* instruction. If an end-carry pulse was generated when the IO word counter was stepped at PT 3, the IO-word-counter-status flip-flop is in its 0 state when examined at PT 6. However, if an end-carry pulse was not generated (i.e., a read-0-operation was not programmed), the IO-word-counter-status flip-flop is in its 1 status when examined. In the latter case, GT 14 gates the PT 6 pulse and as a result the read flip-flop (0.7.6, E9) is set. The resultant d-c level generated by the 1 side of the read flip-flop partially conditions the break-in command generators (0.2.3). Note that if the IO-word-counter-status flip-flop was cleared by the end-carry pulse (read-0 operation speci-

fied), the PT 6 pulse does not set the read flip-flop. Since the card reader flip-flop was set during the *SEL* instruction, GT 2 is conditioned and the PT 6 pulse is gated to trigger SS 1. The resultant output level of SS 1 conditions relay driver RYD 1. The output level of this relay driver is applied to the card reader, causing the card feed mechanism to advance.

The computer continues to carry out the internal program while awaiting a break request from the card reader. The card-reader generates a break request in a manner different from either the line printer or the card punch. As explained in paragraph 3.1, an index point affecting the card machine cycle is associated with each of the 12 rows of card information. At each index point, two words (taken across columns 17 through 48 and 49 through 80) are simultaneously transferred to the computer. One word is placed in the IO register and the other in the IO buffer register. Coincident with the transfer of the two words, the card reader applies a d-c level to pulse generator PG 3 which generates a single pulse. (See fig. 6–4.) This pulse sets the break-request flip-flop (0.2.3, B3) and the second-break-request flip-flop (0.7.6, B9).

As a result of the first break request, a break-in cycle is eventually assigned. At BI 0 of the first break-in cycle, the MBR's and the break request flip-flop are cleared. The memory buffer registers are cleared to receive the first word, and the break request flip-flop is cleared to record the next break request. At BI 0 (delayed) the contents of the IO address counter are transferred to the MAR. The BI 2 pulse steps the IO address counter, indicating the next memory address to be used. The BI 2 pulse also senses GT 11 to determine the status of the card-machine flip-flop (0.7.6, B10); since a card machine is selected, the pulse is gated through GT 11 to step the IO word counter, thus changing its contents to $-N+2$. The BI 2 pulse also transfers the first word from the IO register to the MBR. At BI 3, a parity count is performed on the word while it is in the MBR's and a parity bit is assigned. The BI 7 pulse examines the status of the card-machine flip-flop by pulsing GT 9. (See fig. 6–4.) Since the card-machine flip-flop is set, the BI 7 pulse is gated and the second word is transferred from the IO buffer register to the IO register and the IO buffer register is cleared. The BI 9 pulse examines the status of the card machine flip-flop and the second-break-request flip-flop by sensing GT 8. Since both flip-flops are set, the BI 9 pulse is gated by GT 8, and the output pulse clears the second-break-request flip-flop after passing through OR 5. The gated BI 9 pulse also becomes the second break-request pulse. If the IO word counter is not equal to 0 (and it is assumed that it is not), the second break-request pulse sets the break-request flip-flop, and the computer assigns the next memory cycle as the second break-in cycle.

At the start of the second break-in cycle, the first word of the transfer is already in the memory and the second is in the IO register. The IO address counter contains the memory address of the second word. The break flip-flop, the read flip-flop, and the break-request flip-flop are all set. At TP 0, the memory address register is cleared so that it may receive the second memory address from the IO address counter at BI 0 (delayed). Until BI 9, the procedure of the second break-in cycle is identical to that of the first break-in cycle. By the time BI 9 occurs, the second word has been stored in memory. When the BI 9 pulse senses GT 8 to determine the status of the second-break-request flip-flop, it is suppressed because the second-break-request flip-flop is now in the 0 status. Consequently, no break-request pulse is developed, and the break-request flip-flop, cleared at BI 0, is not reset; the break flip-flop is cleared when the break request flip-flop is sensed at TP 11. The computer now resumes internal operations until the next card-reader index pulse is received about 20 ms later. At that time, the entire process is repeated and another word pair is stored in the memory element. Because the IO address counter is stepped in increments of 1, the successive words of the IO transfer are placed in consecutive memory locations. In the interval between the first and second card reader index pulses, the IO address counter contains the desired address for the third word to be transferred and the IO word counter contains $-N+3$.

The IO process continues in the manner described above until the last word is being transferred. At BI 2, a pulse is gated by GT 11 to step the IO word counter. As a result of stepping the IO word counter, an end-carry pulse is developed which clears the IO word counter flip-flop (0.7.3, C10) to indicate to the computer that the IO word counter contents are equal to $+0$ and that no more break-in cycles are to be assigned. The cleared IO word-counter-status-flip-flop disables the break request circuit so that no more break requests (in the form of card reader index pulses) are honored. However, the IO-interlock flip-flop (0.7.3, D6) remains in its 1 state until the end of the card machine cycle, thereby preventing a new IO class instruction from being executed.

At 12.5 time of every card machine cycle (the end of the card) the card reader supplies a d-c level to PG 1 (fig. 6–4) to determine whether the feed mechanism should stop, or whether another card should be processed. As a result of the conditioning level applied to its input, PG 1 generates an output pulse which clears the IO register and the second-break-request flip-flop. The output of PG 1 also examines the status of the IO word counter (GT 23) to determine whether the card reader is to be stopped. If this flip-flop was cleared by the end-carry pulse, GT 23 is conditioned and gates the examining pulse. The output pulse of GT 23 simultane-

ously clears the IO interlock and triggers SS 4. The clearing of the IO interlock terminates the IO operation, and the triggering SS 4 generates a control pulse to disconnect the card reader. Since the IO interlock is clear, the computer can now execute another IO-class instruction.

3.4 CARD PUNCH AND LINE PRINTER TRANSFERS

The execution of the *WRT* instruction in connection with the card punch or the line printer is similar to the execution of the *RDS* instruction in connection with the card reader. The only difference between these two instructions is that command 182 replaces command 180 at PT 6. During the execution of the *RDS* instruction, command 180 sets the read flip-flop and examines the card-reader flip-flop; during the execution of the *WRT* instruction, command 182 sets the write flip-flop and examines both the card-punch flip-flop (0.7.6, A12) and the line-printer flip-flop (0.7.6, A12). Upon being examined, one of these flip-flops will result in the generation of the start-write pulse. The start-write pulse is sent to the selected card machine, and the computer continues internal operations while waiting for the first break-request pulse.

The line printer and the card punch are similar to the card reader in that they transfer data words in pairs. However, the computer time required to transfer a word pair out of memory to the printer or the punch is greater than that required to transfer a word pair from the card reader to the memory element. Whereas two successive break-in cycles are required to transfer a word pair from the card reader to memory element, 12 memory cycles (2 breakout and 10 dummy cycles) are necessary to transfer a word pair from memory to the line printer or the card punch. The additional time is required because of the slow action of the thyratron buffer registers used in card machine write operations. A series of delaying flip-flops is an integral part of the break-request generation circuits for the line printer or the card punch. These delaying flip-flops cause the computer to assign dummy breakout cycles; since the mode of data transfer is identical for both the line printer and the card punch, the same IO element transfer control circuits are used for both of these devices. When the card punch is selected the thyratron buffer registers are associated with the card punch latching magnets; when the line printer is selected the thyratron buffer registers are associated with the line printer latch and unlatch magnets. Although data words are transferred from the memory element in exactly the same manner for both of these devices, the card punch and the line printer process word pairs in completely different ways. During line printer operation, each word pair is individually punched into one row of a standard IBM card. Since a standard card contains 12 rows, a total of 24 binary words (12 word pairs) can be punched into one card. Since the line printer operation is based on the Hollerith code, a total of 12 word pairs (card image) must be transferred to the printer in order to print one line (2 words) of information. During line printer operation, the 12 word pairs are transferred at specific times so that the correct character on each type wheel (64) will be properly positioned when the information is actually printed.

Since the same IO element transfer control circuits are used for controlling the transfer of data to the card punch and line printer, the following discussion, which assumes that the card punch is selected, is also applicable to the line printer.

A short time after receiving the start-write pulse from the computer, the card punch conditions PG 2 (fig. 6–4), which generates an index pulse. This pulse clears the thyratron-buffer-control flip-flop (0.7.6, D10) and the write flip-flop (0.7.3, E10) and sets the conditioning delay (0.7.6, D12), card-word-transfer (0.7.6, D12) and break request flip-flop (0.2.3, B3). The d-c level generated by the 0 side of the thyratron-control flip-flop conditions the grids of 32-bit thyratron buffer register A.

The break-request flip-flop is examined at TP 11 of the current memory cycle (or by a 2-mc pulse if the computer is in an arithmetic or IO pause). As a result the break flip-flop is set and, unless some preventative provision were made, a break cycle would be immediately assigned. To prevent the generation of breakout pulses at this time, the d-c level developed by the 0 side of the write flip-flop deconditions the breakout command generators, preventing the pulses from being issued even though the break flip-flop is set. The TP 7 pulse of the first wasted machine cycle examines the status of the card-word-transfer flip-flop by sensing GT 27. Since this flip-flop is set, GT 27 is conditioned and, upon being examined, gates TP 7. As a result, the interword-delay flip-flop (0.7.6, D13) is simultaneously set and interrogated through GT 24; however, the shift in status of the flip-flop is not fast enough to permit TP 7 to proceed any further.

The TP 7 pulse of the second wasted machine cycle appears 6 μsec later and, because the card-word-transfer and interword delay flip-flops are both set, the pulse is gated by GT 27 and GT 24. The output pulse of GT 24 clears the interword delay flip-flop and simultaneously examines the delay (0.7.6, D13), firing-delay (0.7.6, C12), and conditioning-delay flip-flops. Note that, in this circuit arrangement, every alternate TP 7 is gated by the interword-delay gate tube (GT 24). Thus the status of the delay, firing-delay, and conditioning-delay flip-flops is examined once every two machine cycles. Since the delay and firing delay flip-flops are both clear, the gate tubes associated with these flip-flops suppress the examin-

ing pulse. The conditioning-delay flip-flop was set by the index pulse; therefore, its associated gate tube (GT 26) is conditioned. The output pulse of GT 26 clears the conditioning-delay flip-flop and, simultaneously, clears the card-word transfer flip-flop and sets the write flip-flop. Setting the write flip-flop conditions the breakout command generators, causing the computer to assign the next memory cycle as a breakout cycle. Two memory cycles have elapsed since the generation of the index pulse so that the 32-bit thyratron buffer register A has had sufficient time to raise its shield-grid potential to the proper level.

At BO 0 of the first breakout cycle, the break-request flip-flop is cleared and the contents of the IO address counter are transferred to the memory address register. At BO 2, both the IO address counter and the IO word counter are stepped. In the time interval between BO 2 and BO 6, the word specified by the contents of the MAR is transferred from core memory to the MBR's. The BO 7 pulse transfers this word from the MBR's to the IO registers. The same pulse also examines the status of the card-machine flip-flop by pulsing GT 10. Since the card-machine flip-flop remains set for the duration of an IO operation involving any card machine, the examining BO 7 pulse is gated; as a result, the write flip-flop is cleared and the firing-delay, break-request (if the IO word counter contents are not equal to 0), and card-word-transfer flip-flops are set. The d-c levels generated by the 1 side of the IO register flip-flops are applied to the No. 1 grids of both of the 32-bit thyratron buffer registers. Since the shield grids of 32-bit thyratron buffer register A were conditioned by the 0 side of the thyratron-control flip-flop before the occurrence of the breakout cycle, thyratron buffer register A fires, energizing the associated magnets in the card punch and completing the transfer of the first word.

Since the write flip-flop was cleared by the BO 7 pulse, the next machine cycle is not assigned as a breakout cycle, even though the break flip-flop is set. In fact, six wasted machine cycles are developed by the action of the break-request generation circuits before the second breakout cycle is assigned. The sequence is as follows: At TP 7 of the first memory cycle, the card-word-transfer flip-flop is examined by sensing GT 27. Since the flip-flop was set by the BO 7 pulse, GT 27 gates TP 7 and, as a result, the interword-delay flip-flop is set.

As the card-word-transfer and the interword delay flip-flops are both set, TP 7 of the next machine cycle is gated by GT 27 and GT 24. The output pulse of GT 24 clears the interword-delay flip-flop and examines the firing delay, conditioning-delay, and delay flip-flops by sensing GT 25, GT 26, and GT 28, respectively. The conditioning-delay and delay flip-flops are both clear; consequently, the associated gate tubes suppress the examining pulse. The firing delay flip-flop was set at BO 7 of the first breakout cycle, and GT 25 is conditioned and gates the examining pulse. The output pulse of GT 25 clears the firing-delay flip-flop and the IO registers and also complements the delay flip-flop. The IO registers are cleared for transfer of the second word.

During the next machine cycle, the card-word-transfer flip-flop (0.7.6, D12) is again examined by TP 7. Because the flip-flop is set, GT 27 is conditioned and passes the examining pulse, with the result that the interword delay flip-flop (0.7.6, D13) is set.

The next TP 7 (fourth machine cycle) is gated by GT 27 and GT 24. The output pulse of GT 24 clears the interword-delay flip-flop and examines the delay (0.7.6, D13), firing-delay (0.7.6, C12), and conditioning-delay flip-flops (0.7.6, D12). The firing-delay and conditioning-delay flip-flops are clear and the gate tubes associated with these flip-flops suppress the examining pulse. The delay flip-flop is set; GT 28 is conditioned and consequently gates the examining pulse. The output pulse of GT 28 sets the conditioning-delay and thyratron control flip-flop (0.7.6, D10). The d-c level generated by the 1 side of the conditioning-delay flip-flop conditions GT 26 so that when this gate tube is sensed, the output pulse will set the write flip-flop. Reversing the status of the thyratron-control flip-flop conditions the shield grids of 32-bit thyratron buffer register B.

At TP 7 of the fifth machine cycle, a pulse is gated by GT 27, associated with the card-word-transfer flip-flop. As a result, the interword-delay flip-flop is simultaneously set and examined. The time required for this flip-flop to shift status from 0 to 1 is sufficient to suppress the pulse. At TP 7 of the next machine cycle, however, both the card-word-transfer and interword delay flip-flops are set and TP 7 is gated by both GT 27 and GT 24. The output pulse of GT 24 simultaneously examines the firing-delay, conditioning-delay, and delay flip-flops by sensing GT 25, GT 26, and GT 28, respectively. The firing-delay flip-flop is clear, and GT 25 suppresses the examining pulse. However, both the conditioning-delay and delay flip-flops are set and, consequently, GT 26 and GT 28 gate the examining pulse. The output of GT 26, in conjunction with the pulse issued by GT 28, clears the conditioning-delay flip-flop. The output of GT 28 also pulses the thyratron-control flip-flop but, since the thyratron-control flip-flop was set at TP 7 of the fourth machine cycle (after the first breakout cycle), the pulse issued by GT 28 does not reverse the status of this flip-flop. The output pulse of GT 26 also clears the card-word-transfer flip-flop and sets the write flip-flop (0.7.3, F10). At the end of the current machine cycle, TP 11 senses the break request flip-flop (0.2.3, B3). Since the break-request flip-flop was set at BO 7 of the first breakout cycle, TP 11 is gated and, as a result, the break flip-flop is set. Because the break and write flip-flops are set, the breakout command generators

are conditioned. At this point, eight memory cycles and one breakout cycle have elapsed since the generation of the first break request pulse. The next memory cycle is assigned as the second breakout cycle.

At BO 0 of the second breakout cycle, the break-request flip-flop is cleared and the contents of the IO address counter are transferred to the MAR. At BO 2, both the IO word counter and the IO address counter are stepped. In the time interval between BO 2 and BO 6, the second word is transferred from core memory to the MBR's. The BO 7 pulse transfers the word from the MBR's to the IO registers. As shown in figure 6–4, the BO 7 pulse is also gated by GT 10 and the firing-delay and card-word-transfer flip-flops are set and the write flip-flop is cleared. This pulse also requests a break if the IO word counter is not equal to 0. A break request at this point assures the next machine cycle will be a dummy breakout cycle and not an instruction cycle, which would terminate the IO process prematurely.

The d-c levels generated by the IO-register flip-flops are applied to the No. 1 grids of both 32-bit thyratron buffer registers. However, only thyratron buffer register B is conditioned since the thyratron-control flip-flop is in its 1 state. Since time must be allowed for the thyratrons to fire, the next TP 7 is gated by GT 27 and the interword-delay flip-flop is set. During the following machine cycle, TP 7 is gated by GT 27 and GT 24. The resultant output pulse of GT 24 clears the interword-delay flip-flop and examines the status of the firing-delay, conditioning-delay and delay flip-flops by sensing GT 25, GT 26, and GT 28, respectively. The conditioning delay flip-flop is clear and GT 26 suppresses the examining pulse. The firing delay flip-flop is set; therefore, GT 25 gates the pulse and performs the following operations:

a. Clears the firing-delay flip-flop
b. Clears the IO registers
c. Complements the delay flip-flop
d. Examines GT 29

Gate tubes 29 and 28 are both associated with the 1 side of the delay flip-flop. Since GT 28 is sensed by the output of GT 24, and GT 29 is sensed by the output of GT 25, both gate tubes of the delay flip-flop are examined and the flip-flop remains in its 1 state long enough to allow the gate tubes to pass these examining pulses. As a result, the following operations are performed:

a. GT 28:
   Sets the conditioning delay flip-flop
b. GT 29:
   Clears the break request flip-flop
   Clears the card-word transfer flip-flop
   Sets the write flip-flop

If (as assumed here) no arithmetic pause exists, the break-request flip-flop (0.2.3, B3) is examined at TP 11 of the current memory cycle. Since this flip-flop was cleared by the output pulse of GT 29, the TP 11 pulse clears the break flip-flop. (If the break flip-flop were not cleared, the next memory cycle would be assigned as a breakout cycle because the write flip-flop is set.)

At this point, the first memory word is in 32-bit thyratron buffer register A, and the second memory word is in 32-bit thyratron buffer register B. The last two dummy breakout cycles were utilized to reset the break request circuits to their original state. The two words are now punched in one row (columns 17 through 80) of a card under the control of the card punch. The next index pulse does not appear for at least 20 ms and in this interval the computer continues its internal operations.

The word pair transfers continue in the above manner until the IO word counter is reduced to +0. As shown in figure 6–4, the IO word counter end-carry pulse clears the IO word counter status flip-flop (0.7.3, C10). Due to the d-c level generated by the 0 side of this flip-flop, no further break request will be honored and GT 23 is conditioned. However, the IO interlock (0.7.3, D6) is not cleared until the end of the card punch cycle at which time the card punch generates a request disconnect pulse. The card punch request disconnect pulse, which is generated at 12.5 time of the card punch cycle, is applied to PG 1. If the IO word counter status flip-flop is cleared, the output pulse of PG 1 is gated by GT 23. The output pulse of GT 23 triggers SS 4 and clears the following flip-flops:

a. IO-interlock
b. Interword delay
c. Firing-delay
d. Card-word-transfer
e. Delay

The resultant d-c level output of SS 1 conditions relay driver RYD 4, and, as a result, the card punch is disconnected. Since the IO interlock is cleared the computer may initiate another IO process.

The write-0 operation is a logical operation when the card punch or line printer are selected as output units. This operation permits the programmer to cause the card punch to pass one blank card without punching any information in it or to skip one line on the line printer. The operation is usually used to designate the start or end of a particular program routine.

The write-0 operation is similar to the read-0 operation involving the card reader. When the card punch or line printer is selected, a start-write pulse is transmitted to physically start the selected unit. Break requests are generated as noted above; however, since the IO word counter is equal to 0, the break-request pulses are not honored by the computer and no information is transferred from the memory element. At the end of the particular card machine cycle, the card punch or line printer generates a request disconnect pulse which is honored to terminate the operation.

CHAPTER 4
MAGNETIC TAPE TRANSFERS

4.1 GENERAL

The magnetic tape element of the AN/FSQ-7 provides a convenient means of storing large blocks of information to which random access is not required. (See Part 9 for a detailed explanation of the tape element.)

The magnetic tape element consists of the tape adapter unit, six tape drive units, and the tape power supply unit. The tape adapter unit contains all of the electronic circuits necessary for selecting and controlling the six tape drive units, each of which controls a reel of magnetic tape, while the tape power supply unit generates the nonstandard voltages required by the tape drive units. The magnetic tape used in the tape drive unit is ½ inch wide and can store seven tracks of information in the traverse direction. Since a computer word contains 33 bits, the individual bits of a data word must be combined into groups for recording on the tape. These groups are called characters – each character consists of six information bits and one synchronizing bit. Six characters are required to store one data word on the tape. Each of the first five characters contains six consecutive bits of the data word, while the sixth character contains the last three bits of the data word.

During an IO transfer operation, individual data words are transferred between the IO register of the Central Computer and the word register of the tape element. The tape element control circuits transfer consecutive data word characters between the word register and the magnetic tape. These internal tape operations, although started by the computer, are controlled by the tape-timing circuits which operate asynchronously to the computer-timing circuits. Since the tape operations are independent of the computer, the tape element generates break-request and disconnect pulses as required, and the necessary synchronization is accomplished in the IO element.

Data is recorded on magnetic tape in blocks of words at a rate of one character approximately every 50 μsec, wherein the interword spacing is the same as the intercharacter spacing. These blocks of data, which do not contain any fixed number of words, are called records. In storing blocks or records of data on a magnetic tape, a relatively large gap is provided between the last character of one record and the first character of the next record. This interrecord gap, which does not contain synchronizing pulses, is used during tape read operations to provide a means for controlling the transfer of single blocks or records of data. During an IO write process, the IO transfer operation is always terminated by the computer transfer control circuits (0.8.2) which also generate a disconnect signal to stop the tape operation. An IO read process which involved tapes usually specifies that a complete block or record of data be transferred to the computer. Under this condition, the IO transfer operation is terminated and the selected tape is disconnected by the tape transfer control circuits which respond to the absence of synchronizing pulses in the interrecord gap. However, if the computer requests fewer data words than are contained in the record being read, then the IO transfer operation is terminated as a result of the word counter going to 0 although tape motion is not stopped (tape is not disconnected) until the interrecord gap is reached.

Another division of data on tape is called the file. It is made up of one or more records and its end is indicated by a special one-word record, called the end-of-file record. (The end-of-file record is written by executing the *PER 72* instruction.)

The tape drive selected for operation by the computer must be in a state that it can be operated. Two conditions, referred to as ready and prepared, are two states of operability that can be sensed by the computer. These conditions are explained in detail in Part 9.

4.2 INITIATION OF TAPE TRANSFERS

The tape units are selected for a reading or writing operation by execution of the *SEL* instruction. This instruction will select one of six possible tape drive units and deselect all other tapes units and IO devices. The initial address in core memory which is to be used in a tape operation is determined by execution of the *LDC* instruction. Finally, a *RDS* or *WRT* instruction is executed, depending on the desired direction of transfer, and either a start-read or start-write pulse will be generated and sent to the selected tape drive unit.

4.3 READING OPERATION

Figure 6–5 (foldout) is a simplified logic diagram of the tape drive units control circuits. As shown in the diagram, command 180, which occurs at PT 6 of the *RDS* instruction, examines GT 4. This gate tube, which is conditioned by the tape-operation flip-flops, generates a start-read-tape pulse to the selected tape drive unit to start tape motion. When the tape comes up to speed and the first character is read, the tape time pulse distributor

is started (by the sync pulse developed in the sync track read head) to control the transfer of six tape characters to compose a 33-bit word in the tape word register. As soon as the word is composed, the tape time pulse distributor effects the transfer from the IO word register to the IO register and generates a break-request pulse. This break-request pulse passes through GT 5 (fig. 6–5, foldout) and examines a gate tube controlled by the word counter not equal to 0 status. If the IO word counter content is not equal to 0, the break-request pulse sets the break request sync, which in turn sets the break-request flip-flop. As a result, the succeeding memory cycle is assigned as a break-in cycle. At TP 0, the memory address register is cleared. At BI 1, the MAR receives the contents of the IO address counter. The BI 2 pulse effects the transfer of the word in the IO registers to the memory buffer registers and then, as shown in figure 6–5, steps the IO word counter. At BI 3, the IO address counter is stepped. The normal action of the memory circuits transfers the word now in the MBR's to the memory location specified by the MAR. The first data word transfer is now complete.

The data transfer operation just discussed is repeated until the last word in the record has been transferred, or until the IO word counter is stepped to positive 0. If the latter condition occurs, an end-carry pulse is generated by the IO word counter, which clears the IO-word-counter-status flip-flop. As a result, all future break-request pulses generated by the tape element are suppressed, although the selected tape drive continues its motion. A short time after the last character of the last word in the record is read from the tape, the tape element circuits respond to the absence of synchronizing pulses from the tape to generate a disconnect pulse. This pulse is used to set the disconnect IO-interlock-sync flip-flop (fig. 6–5) which conditions GT 2. This gate tube, which is examined by 2-mc pulses, clears the IO interlock to terminate the IO transfer operation. If the tape unit was programmed to read a full record of words, the disconnect pulse disconnects the tape unit from the computer controls and terminates the IO transfer operation by the single action of clearing the IO interlock.

The read-0 operation is a logical operation when used with the tape element in that this operation causes the selected tape reel to advance by one record without transferring the data contained in that record. If such an operation is specified, the read flip-flop remains in the cleared state although the IO process is not terminated, since the IO interlock remains on. At PT 6 of the *RDS* instruction, command 180 initiates the tape transfer. In reading the data words of the record, the tape unit develops break-request signals; however, because the IO word counter is equal to 0, these pulses do not set the break-request-sync flip-flop and, consequently, break cycles are not generated.

The selected tape unit continues to transmit data words to the IO registers until it detects the end-of-record gap, at which time it transmits a disconnect-tape pulse which clears the IO interlock. As a result, the read-0 operation is terminated. The fact that an unintelligible jumble of data is loaded into the IO registers is immaterial, since no transfer is effected beyond that point; for all practical purposes, the computer simply skips a tape record. At the conclusion of the read-0 operation another IO process may be programmed to transfer data from the second record by the process described above.

The operation of reading a word from the tape requires approximately 4.5 ms if the tape is not at the load point; if it is at the load point, an additional 20 ms are required.

4.4 WRITE OPERATION

The operation of writing data on a tape is very similar to the read operation described above. During the execution of the *WRT* instruction, the selected tape drive is started, and when the tape has come up to operating speed, the tape time pulse distributor is started to initiate the IO transfer operation. At 8 time of the first tape time pulse distributor cycle, a break request is generated to transfer the first word from core memory to the word register via the IO register. The remainder of the first cycle and the next 5 cycles are used to write the six characters of the first word on the tape. This process of obtaining a word and storing it on tape continues until the IO word counter contents are reduced to positive 0 at which time an end-carry pulse is generated. This end-carry pulse clears the IO interlock and stops the tape-time pulse distributor to terminate the IO transfer operation. The selected tape drive motion stops a short time after the tape-time pulse distributor is stopped.

CHAPTER 5
MISCELLANEOUS IO UNITS

5.1 GENERAL

The four miscellaneous IO units used by the computer are: the manual input matrix, the burst-time counters, the IO register, and the Warning Light System. Only a read operation can be performed when the computer is linked to any of the first two of these miscellaneous units; only a write operation can be performed when the computer is linked to the Warning Light System. The IO register may be used in both reading and writing operations.

5.2 INITIATION OF TRANSFERS

An IO program for an operation involving the manual input matrix or the burst time counters consists of the *SEL, LDC,* and *RDS* instructions. Upon execution of the *RDS* instruction, a start-read pulse is generated in the IO element, transferred to the selected device, and initiates the first word transfer.

If the Warning Light System is the selected IO device, the basic IO program consists of the *SEL, LDC,* and *WRT* instructions. Execution of the *WRT* instruction will cause a start-write pulse to be transferred to the Warning Light System.

The basic IO program used for the IO register consists of the *SEL, LDC,* and either a *RDS* or *WRT* instruction, depending on the direction of transfer. As with all basic IO programs, execution of these instructions will generate either a start-read or start-write pulse which will initiate the transfer of data from or to the IO register.

5.3 MANUAL INPUT MATRIX

5.3.1 Description

Certain types of data (described below) are fed into the Central Computer through a buffer storage device called the manual input (MI) matrix and which is a part of the manual input element of the Input System. The matrix is made up of tape cores arranged in 128 rows of 33 (one not used) cores each. Information is read from the matrix under computer command at a rate of one row (one computer word) every 20 μsec. The MI matrix can be read entirely or in part during the execution of one transfer operation, but the reading is always sequential starting from the first row.

The status of the cores in rows 7-128 is determined by the setting of the keyboard switches on keyboard panels associated with various display consoles to which the cores are connected. In addition, certain cores in these rows are connected to simplex equipment (e.g. Input System channels and Display System consoles) so that the active or standby status of this equipment can be determined by the computer program. The reading of the cores in rows 7-128 does not affect the status of these cores (nondestructive readout).

The cores in the first six rows are connected to the ACTION pushbuttons on the several keyboard panels and to the light guns associated with certain situation display consoles. A core is set to the 1 state when the ACTION button to which it is connected is depressed (or if connected to a light gun, when the light gun is fired). When the cores in rows one through six are read, they are switched to the 0 state in the process (destructive readout). A core in the first six rows serves to indicate to the computer program whether the keyboard data for a particular console is to be recognized. The program determines which cores in rows 7 through 128 are associated with a particular core in rows 1 through 6.

5.3.2 Reading Operation

Reading of the MI matrix is initiated at PT 6 of the *RDS* instruction, which gates a start-read pulse to the core matrix control circuits.

This pulse is used to initiate the transfer of data from the MI matrix to the computer. When this operation is initiated, the control functions of reading and transfer are performed by the core-matrix control circuits, the core shift register, the thyratron core drivers, and the sense amplifier. (See fig. 6–6, foldout.) The core-matrix control circuits initiate, control, and terminate the readout operation. The core shift register and thyratron core drivers cause a sequential (word-by-word) readout. The sense amplifiers amplify, filter, and shape the pulses which transfer the data word from the core matrix to the Central Computer System.

As shown in figure 6–6, the MI matrix derives its timing pulses from the MIXD drum-timing circuits which are located on the OD side of the Drum System. The timing pulses are designated OD 1, OD 2, OD 3, and OD 4, and they occur at intervals of 2.5 μsec. Since OD 1 pulses, which are generated every 10 μsec, are frequency divided readout of the core matrix occurs every 20 μsec. When the readout process is initiated, a shift level is produced to transfer a word from the matrix, and a break-request pulse is generated and

routed to the Central Computer System to inform it that a data word has been transferred from the core matrix. The computer keeps a cumulative count (in the IO word counter) of the transfers and, when the specified number of transfers has been received, it transmits a disconnect MI matrix pulse to the core-matrix control circuits. As a result of receiving the disconnect pulse, the core-matrix-control circuits stop the generation of shift levels and terminate the readout of the core matrix. A detailed logic circuit analysis of the core-matrix-control circuits is presented in the following paragraphs. As shown in figure 6–6, the read-manual-input-matrix pulse triggers SS 1. The resultant output pulse of the single-shot is designated the load-shift-register pulse. The load-shift-register pulse simultaneously writes a 1 in the first core (CS 1) of the core shift register and sets FF 1. The resultant d-c level which is generated at the 1 side of FF 1 conditions GT 1.

The next OD 1 pulse simultaneously examines FF 1 and FF 2, by pulsing GT 1 and GT 2, respectively. Since FF 1 is set and FF 2 is clear, the OD 1 pulse is gated by GT 1 and suppressed by GT 2. The gated OD 1 pulse sets FF 2, and the resultant d-c level generated at the 1 side of FF 2 conditions GT 2. Because FF 2 is now in its 1 state, the next OD 1 pulse is gated through GT 2; as a result, FF 3 and FF 4 are both set. The d-c level generated by the 1 side of FF 3 is applied (through OR 6) to the 16 core shift drivers (shift/reset). At OD 2, a pulse passes through OR 4 to clear FF 3, thus establishing the length of the shift pulse at 2.5 μsec. The 2.5-μsec shift pulse is applied to the 16 core shift drivers which in turn apply a read-current pulse to the 128 core-shift circuits. Since core shift bit CS 1 contains a 1 (all others contain 0's) an output pulse is developed which sets core shift bit CS 2 and which applies a pulse to thyratron core driver TCD 1. The output of the thyratron core driver is used to drive all of the cores of the associated word to the 0 state. The cores that contained 1's develop output pulses on the associated sense lines which, when amplified and shaped into 0.1-μsec standard pulses by the sense amplifier blocking oscillator, are used to set the associated bit of the IO buffer register.

At OD 3, the status of FF 4 is examined by pulsing GT 3. Since FF 4 is set, GT 3 is conditioned and the OD 3 pulse is gated. The output pulse of GT 3 clears FF 4 and FF 2 and is routed to the Central Computer System as a break-request pulse. At the same time, the first data word is being transferred into the IO buffer register.

Since the speed of the MI matrix is comparable to that of the Drum System, break-request circuits are also used for the MI matrix. As shown in figure 6–5, the break-request pulse is gated through GT 11 and, as a result, the IO-buffer-load-sync flip-flop is set.

The break-in cycle is subsequently assigned, during which the IO word counter and the IO address counter are stepped by BI 1 and BI 3, respectively. The word in the IO buffer register is transferred successively to the IO register and to the memory buffer register, and the break-request flip-flop is cleared. Each new word is treated in the same way, the rate of their arrival being constant and dictated by the design of the MI matrix. As soon as the IO word counter contents are equal to 0, the end-carry pulse resulting from the final stepping of this counter clears the IO-word-counter-status flip-flop. The 0 side of this flip-flop and the 1 side of the MI-matrix flip-flop fully condition AND 2 whose output is gated by GT 13 to become a clear-IO-interlock pulse and a disconnect-manual-input-matrix pulse at BI 11 (fig. 6–5).

As shown in figure 6–6 (foldout), the disconnect MI-matrix pulse clears FF 1 and FF 2 after passing through OR 1 and OR 2, respectively. The pulse also triggers SS 2 after passing through OR 3. The resultant output level of SS 2 is applied through OR 6 to the core shift register as a reset level which clears the core shift register in anticipation of the subsequent readout operation.

The disconnect operation can also be accomplished manually through the duplex maintenance console. In this case, the manual-reset pulse, which originates at the duplex maintenance console, is applied to OR circuits 1, 2, 3, 4, and 5 simultaneously, thus clearing flip-flops 1, 2, 3, and 4. The clearing of these flip-flops permits the operator to immediately terminate the readout operation. The manual-reset pulse also triggers SS 2, which applies the reset level to the core shift register through OR 6. The reset level clears the core shift register in anticipation of the subsequent readout operation.

5.4 BURST TIME COUNTERS

5.4.1 Description and Purpose

The types of output data processed and transmitted by the Output System are ground-to-ground (G/G), teletype (TTY), ground-to-air, frequency division (G/A-FD), and ground-to-air, time division (G/A-TD). The Output System reads output data from the OD side of the output buffer drum fields and examines each word to make up messages according to type of data and word sequence. The messages are made up by temporarily storing the individual words in particular locations of a ferrite core array, before reading out the completed messages to the transmission equipment. The left half-portion of the output words contains the information for making up the messages.

Besides separating data words by type and word sequence it is necessary to identify data words as being associated with a particular message in a given type. To accomplish this the Output System has four message counters (called burst counters), one for each type of

output. Each output data word has a burst time count in the left half-portion of the word. The Output System checks for comparison of the burst count with the setting of the appropriate burst counter. Those words with burst counts less than the burst counter contents are rejected and lost. Those that compare are stored for subsequent transmission. Those words with burst counts greater than the burst counter contents are rejected but left on the OB drum for future reading. A burst time counter is stepped by the Output System as each message is completed.

For the operating program to assign burst numbers to output data words, it must be able to obtain the contents of the various burst counters at any time. This facility is provided in the equipment so that the burst time counters can be read using the basic IO program.

In addition to the four burst time counters in the Output System there is an elapsed time counter which is used for the G/A-FD section only. The elapsed time counter is stepped by the 32-pps computer clock pulses. As each G/A-FD message is completed, it is cleared at the same time the G/A-FD burst time counter is stepped. The elapsed time counter serves to give the computer an indication of the elapsed time since the last G/A-FD message was completed and is read with the G/A-FD burst time counter.

5.4.2 Reading Operation

Execution of the *SEL (21) (Select Burst Time Counters)* instruction results in setting the burst time counters select flip-flop (0.7.7, A-B). The output from the 1 side of this flip-flop gates the PT 6 pulse of the *RDS* instruction. The gated PT 6 pulse is sent to the output computer section of the Output System to set up the circuitry for reading out the contents of the burst time counters.

The burst time counters are read one at a time under control of the circuits in the output computer section. (See fig. 6–7, foldout.) The PT 6 (select and read) pulse sets FF 1. The following OD 3 pulse will be gated by the 1 side to set FF 2. The 1 side of FF 2 conditions GT 4 to pass successive OD 1 pulses which are used to set a 3-flip-flop counter. The output levels of the counter are decoded with five 3-way AND circuits to produce burst counter selection levels. As the counter is stepped by OD 1 pulses the proper selection level is applied to a number of 2-way AND circuits. The second input to these AND circuits is from the 1 side of the flip-flops in one of the burst time counters. The outputs of these AND circuits are applied to the readout gates. The readout gates are strobed by the OD 1 pulses for transfer to the IO buffer register.

This circuit arrangement provides transfers at a 10-μsec rate (the same as for addressable drum transfers). To read all four burst counters the *RDS* instruction must specify at least 5 words. Note that 0's will be transferred when the selection counter equals 011_2.

The read operation is terminated when the word counter goes to 0 or when the selection counter is stepped from 111_2 to 000_2 depending on the number of words specified in the *RDS* instruction.

5.5 IO REGISTER

Although the IO register is generally used as the intermediate storage register between core memory and a selected IO device, it may itself be selected as an IO device using the *SEL (04)* instruction. The IO register, so selected, serves as a source of words containing all 0's when it is desired to clear memory locations, or a word of any configuration when it is desired to write a test pattern into memory.

When any *SEL* instruction is executed, the IO register is cleared (0.7.1, B-7, ctrl clr/deselect pulse). Normally, during the execution of a subsequent *RDS* or *WRT* instruction the IO register is cleared again, (0.7.7, C-7, command 144). Prior to the transfer of each word the IO register is cleared. However, the *SEL (04)* instruction causes a gate (0.7.1, C-7, IO reg not sel level) to be conditioned which prevents the clearing of the IO register between transfers and when the *RDS* or *WRT* instruction is executed.

The basic IO program (see table 6–4) is used to clear memory. To clear all memory locations including the test register, the number specified in the *RDS* instruction is 3.77760_8.

To write a particular pattern into core memory, the program in table 6–5 is used. The location specified in the *LDC* instruction contains the test word. Note that the IO register is not selected a second time before issuing the *RDS* instruction. To do so would cause clearing of the IO register and the loss of the test word.

5.6 WARNING LIGHT SYSTEM

Warning lights are used to alert operators at different consoles of certain Central Computer System operations. Provision is made for 256 lights, divided into eight groups of 32 lights each. Each group is considered a 32-bit register. To make use of the Warning Light System, *LDC, SEL, WRT* is the basic IO program which

TABLE 6–4. PROGRAM TO CLEAR MEMORY LOCATIONS

OPERATION	ADDRESS	COMMENTS
SEL (04)	—	Select IO register
LDC	0.00000	To clear starting at location zero.
RDS	0.00000 - 3.77760	Number of locations to be cleared.

must be executed. The circuits in the IO element required to implement the use of the Warning Light System are shown in figure 6–5.

The principle of operation depends upon the fact that complementing a flip-flop an even number of times does not alter its original status. To write new information into the warning light registers, the IO registers are loaded with a new word every second memory cycle. For each word transferred, the first of the two cycles is a breakout cycle during which the word in the IO registers is used to complement the eight warning light registers and the selected warning light register is cleared; the second cycle is a dummy cycle during which the word in the IO register is again used to complement the eight warning light registers. Under this condition the original content of the seven non-selected warning light registers will not be destroyed. Each of the eight warning light registers is modified in a similar manner so that, at the end of the IO process, each register will duplicate the status of the core memory word assigned to it.

The address of the first core memory word to be used is placed in the IO address counter by the *LDC* instruction. During the *SEL* instruction, the warning-light flip-flop is set and the auxiliary-warning-light flip-flop is cleared. (See fig. 6–5.) After the *WRT* instruction, the IO interlock and the IO-word-counter-status flip-flops are set, and the IO word counter contains the complement of the right half-word of the *WRT* instruction. The magnitude of the number in the IO word counter cannot exceed 8 because there are only eight warning light registers, but it can have any intermediate value between 1 and 8.

The final pulse of the *WRT* instruction, PT 6, requests a break, sets the warning-light-counter No. 1 flip-flop, and clears the other two warning-light-counter flip-flops. The register matrix converts the reading of the warning light counter into a positive d-c level on matrix output 001. AND circuit 9 is therefore energized conditioning GT 25.

As a result of the break request, the next memory cycle is assigned as a breakout cycle. The IO word counter and the IO address counter are stepped at BO 2 and BO 3, respectively. After the stepping process is completed, the IO word counter contains N —1 and the IO address counter contains the address of the core memory word which is to affect the second warning light register. The word for the first warning light register has already been placed in the MBR's. This first word is transferred from the MBR's to the IO registers at BO 7.

As shown in figure 6–5, the BO 8 pulse is gated by GT 28 associated with the warning-light flip-flop. Since this flip-flop was set at PT 5 of the *SEL* instruction, the BO 8 pulse is passed to cause the content of the IO registers to complement the eight warning light registers. The warning light register flip-flops are slow-speed flip-flops; therefore about 2 μsec must elapse before they can be operated upon again. Since there is not enough time between BO 8 and BO 11 to provide the 2 μsec, a dummy cycle must be assigned. Accordingly, the BO 8 pulse requests a break. To prevent the generation of breakout pulses during the dummy cycle, the write flip-flop is cleared at BO 11. The BO 11 pulse also sets the auxiliary-warning-light flip-flop, conditioning the circuits which are to be inspected during the dummy cycle.

TABLE 6–5. PROGRAM TO LOAD MEMORY 1 WITH TEST PATTERN

OPERATION	ADDRESS	COMMENTS
SEL (04)	—	Select IO register
LDC	3.77777	Test word in test register
WRT	0.00001	Place test word in IO register
LDC	0.00000	Start loading test pattern at location zero.
RDS	2.00000	Number of words in memory 1.

Since the auxiliary-warning-light flip-flop is set, the gate tubes associated with its 1 side are conditioned. At TP 4, the IO word counter is stepped and the eight gate tubes associated with the register matrix are simultaneously sensed. At this stage in the process, GT 25 is conditioned; therefore, warning light register No. 1 is cleared. The status of the other warning light registers remains unchanged. The TP 4 pulse also steps the warning light register counter by complementing the warning-light-counter No. 1 and warning-light-counter No. 2 flip-flops. The resultant output from the register matrix is therefore 010, which conditions GT 26 and deconditions GT 25. However, the clearing pulse generated by GT 25 for warning light register No. 1 is developed before the gate tube is disabled.

The IO register is not affected during the dummy cycle because of the absence of breakout pulses. At TP 11 (of the dummy cycle), the flip-flops of the warning light registers are again complemented in accordance with the contents of the IO register. The TP 11 pulse also clears the auxiliary-warning-light flip-flop, resets the write flip-flop, and clears the IO register.

With the break flip-flop and the write flip-flop set,

the following memory cycle is another breakout cycle. During this cycle and its attendant dummy cycle, the contents of the second warning light register are changed to conform to the dictates of the word in the IO register. After TP 11 of the dummy cycle, the first two warning light registers are correctly conditioned, GT 24 is conditioned by the register matrix, and the remaining registers still retain their initial contents. Each TP 11 pulse that is generated examines the status of the IO word counter. As soon as the contents of this counter are equal to 0, a gate tube conditioned by the IO-word-counter-status flip-flop gates the TP 11 pulse. As a result, the IO interlock and the counter-control flip-flop are cleared, terminating the IO process.

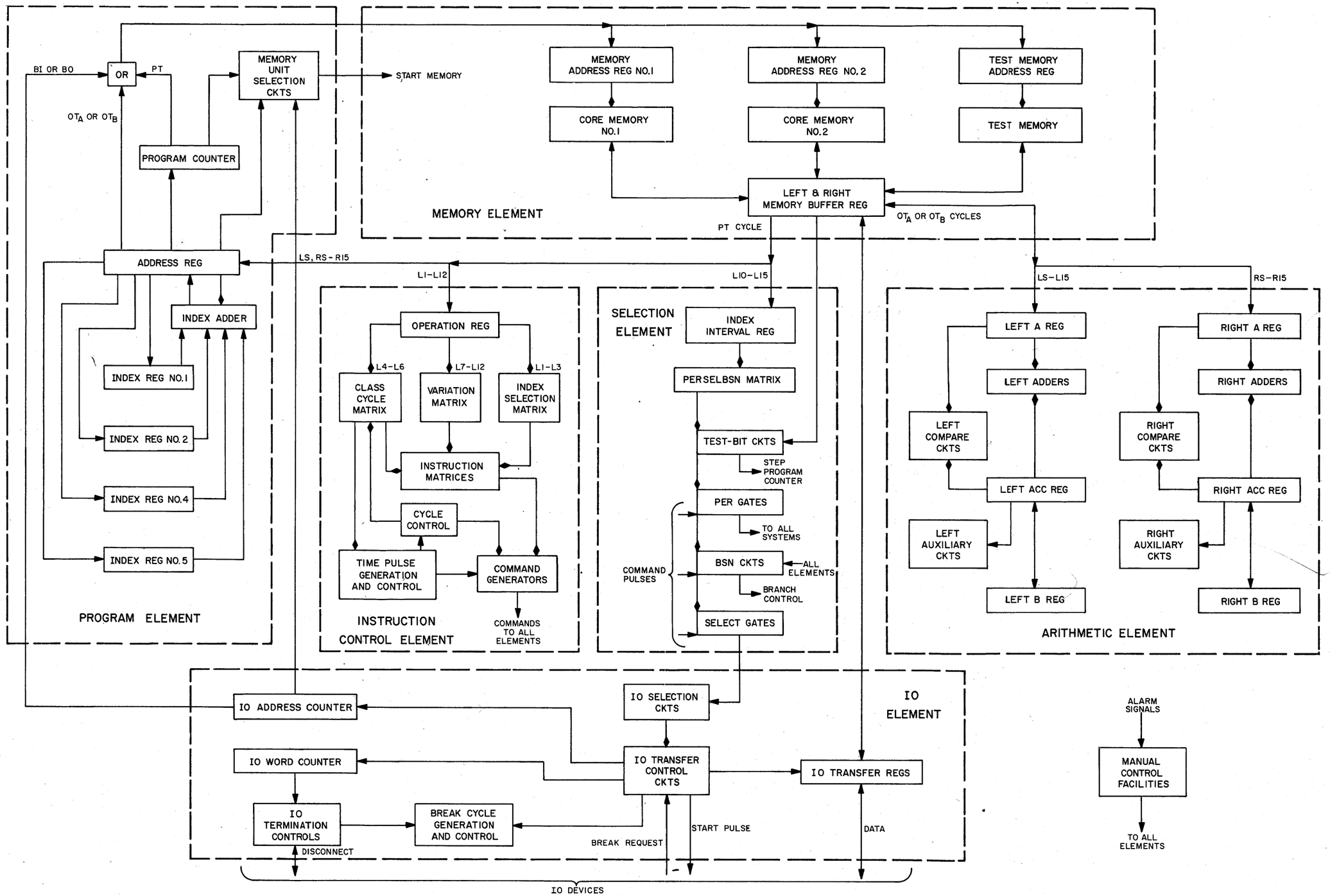


Figure 1-25. Major Circuit Groups of Central Computer System

	PT/OT	0	1	2	3	4	5	6	7	8	9	10	11	0	1	2	3	4	5	6	7	8	9	10	11	0	1	2	3	4	5	6	7	8	9	10	11
COMMON COMMANDS	PT	PAUSE							42 47 52 92			55		31	41 53					31 66* 77 101 266*	31	47				31											
MISCELLANEOUS—CLASS COMMON COMMANDS																																					
CLEAR AND SUBTRACT WORD COUNTER (CSW)(020)		NO PAUSE					153				240 157			NO OT												NO OT											
OPERATE (PER)(01-)													131		71										131												131
LOAD B REGISTER (LDB)(030)		NO PAUSE							114 115 117 124 212				131		71						84 284 40				131												131
EXTRACT (ETR)(004)		NO PAUSE							114 115 117 124 212				131		21 71 230										131												131
SHIFT LEFT AND ROUND (SLR)(024)		NO PAUSE		16 20 21 230			60 260	84 214 215 284			130 134			NO OT												NO OT											
PROGRAM STOP (HLT)(000)		NO PAUSE		21 230				19 219					161 270												134	131											131
COMPARE MASKED BITS (CMM)(040)		NO PAUSE		21 230				19 219																													
COMPARE DIFFERENCE MASKED BITS (CDM)(041)		NO PAUSE		2 4 21 81 89 202 204 230 281 289		63 263	60 260																														
COMPARE RIGHT HALF WORDS (CMR)(042)		NO PAUSE		21 230				19 219					161																								
COMPARE DIFFERENCE RIGHT HALF WORDS (CDR)(043)		NO PAUSE		2 4 21 81 89 202 204 230 281 289		63 263	60 260																														
COMPARE LEFT HALF WORDS (CML)(044)		NO PAUSE		21 230				19 219					161																								
COMPARE DIFFERENCE LEFT HALF WORDS (CDL)(045)		NO PAUSE		2 4 21 81 89 202 204 230 281 289		63 263	60 260																														
COMPARE FULL WORDS (CMF)(046)		NO PAUSE		21 230				19 219					161																								
COMPARE DIFFERENCE FULL WORDS (CDF)(047)		NO PAUSE		2 4 21 81 89 202 204 230 281 289		63 263	60 260																														
TEST ONE BIT (TOB)(05-)		NO PAUSE									98				71 102										131												131
TEST TWO BITS (TTB)(054-)		NO PAUSE		21 230				19 219		98 97	98		161		71 102						114 115 117 124 212				131												131

*APPLICABLE IN *slr, cdm, cdr, cdl,* & *cdf* INSTRUCTIONS ONLY

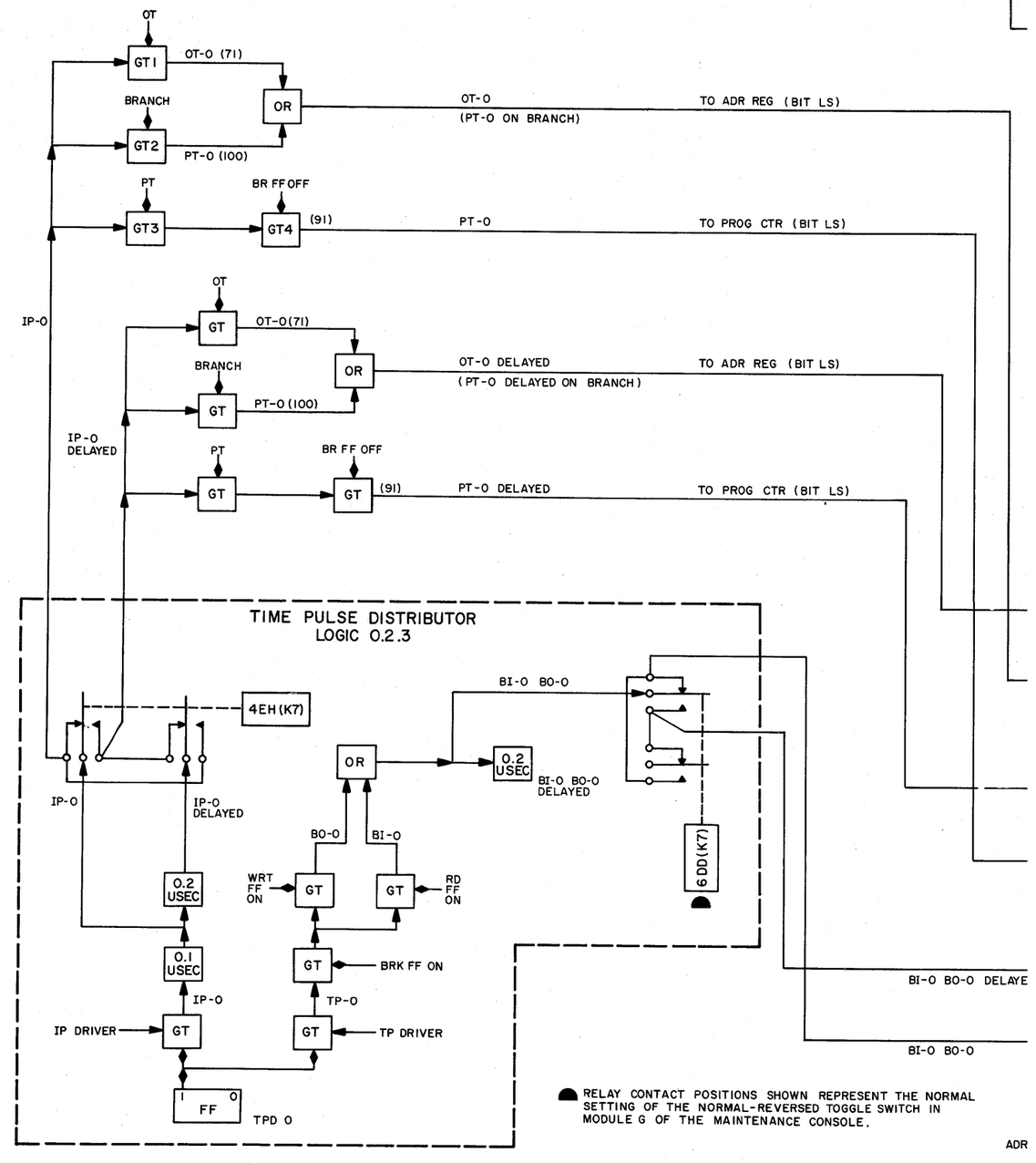
Figure 2–15. Miscellaneous-Class Instructions, Command Sequence Chart

	PT					OTA											OTB											PT					
	7	8	9	10	11	0	1	2	3	4	5	6	7	8	9	10	11	0	1	2	3	4	5	6	7	8	9	10	11	0	1	2	3	4	5	6
COMMON COMMANDS	42 47 52 92			55		31	41 53					31	47			55		31	41 53					31	47			55		31	41 53 91					31 66 † 77 101 266 †
STORE-CLASS COMMON COMMANDS			114 115 117 124 212		131		71						43				132*		71		47								161 167							
FULL STORE (FST) (324)					132*	← NO OT$_A$											→			17 121															
LEFT STORE (LST) (330)												21 230	51							17 228															
RIGHT STORE (RST) (334)												21 230								23 121															
STORE ADDRESS (STA) (340)												21								23 228															
RIGHT ADD ONE (AOR) (344)												21 210 230	51	69			202 204 281 289		14	23 121															
EXCHANGE (ECH) (350)												21 230	51							17 121				10 210			64 264				2 4 81 89 202 204 281 289					
DEPOSIT (DEP) (360)							21 230	19 219		88 288	25 225		19 51 219		25 225					17 121															

* OCCURS AT PT 11 IN FST
† AFFECTS ONLY AOR

Figure 2-16. Store-Class Instructions, Command Sequence Chart

START MEM
65AJAI (O



ADR

START MEM 1(256)² 65AJAI (0-2.1.4)

START MEM 2(64²) 10ACB2 (0.1.4)

6DD(K8)

0.3 USEC

0.14 USEC

OR 1

OR 6

TO ADR REG (BIT LS)

IRANCH)

TO PROG CTR (BIT LS)

YED

AYED ON BRANCH)

TO ADR REG (BIT LS)

YED

TO PROG CTR (BIT LS)

BO-0

BI-0 BO-0 DELAYED

6DD(K7)

RELAY CONTACT POSITIONS SHOWN REPRESENT THE NORMAL SETTING OF THE NORMAL-REVERSED TOGGLE SWITCH IN MODULE G OF THE MAINTENANCE CONSOLE.

GT

GT10

GT9

GT5

OR3

OR

FF

LS

FROM LM BFR

BI-0 BO-0 DELAYED

BI-0 BO-0

ADR REG

START MEM 2 PT-0

PROG CTR (12 WAY) OR

* BITS RS-R11 FROM PROG CTR

IO ADR CTR (12WAY) OR

** BITS RS-R11 FROM IO ADR CTR

CAC OT-3

TP-11 CLR

OR

FF

START TEST MEMORY 4 DSA1 (0.1.3)

GT8

TP-3

GT7

PARITY CK

TP-10+

(0.1.1)

PARITY CK-START TM

START MEM 1 BI-0 BO-0

START MEM 1 PT-0

START MEM 1 OT-0 (PT-0)

CAC OT-0

OR 2

START MEM 2 BI-0 BO-0

START MEM 2 OT-0 (PT-0)

ADR REG (12 WAY) OR 5

TO ZERO SIDE OF MAR 2 FF'S (R4-R15)

TO ZERO SIDE OF MAR 1 FF'S (R5-R15)

TO ZERO SIDE OF TMAR FF'S(R12-R15)

OR 4

GT6

R1-R3

R5-R10

R13-R14

RS

R4

R11

R12

R15

FROM RMBFR

ADR REG

PROG CTR

IO ADR CTR

NOTE:
ALL LOGIC REFERENCED TO 0.4.1 EXCEPT AS OTHERWISE INDICATED

Figure 4-2. Memory Unit and Address Selection

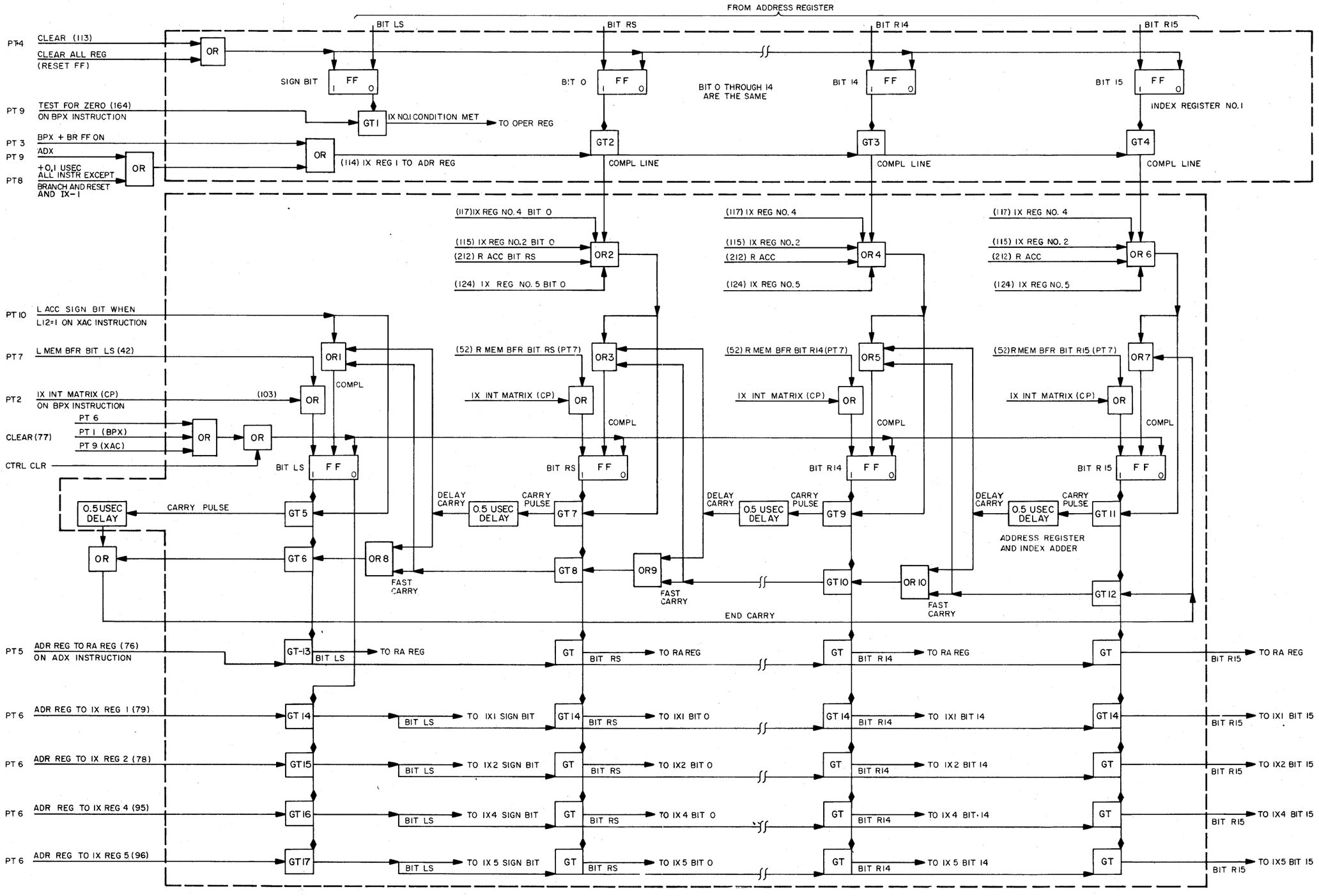


Figure 4-3. Indexing Control, Logic Block Diagram

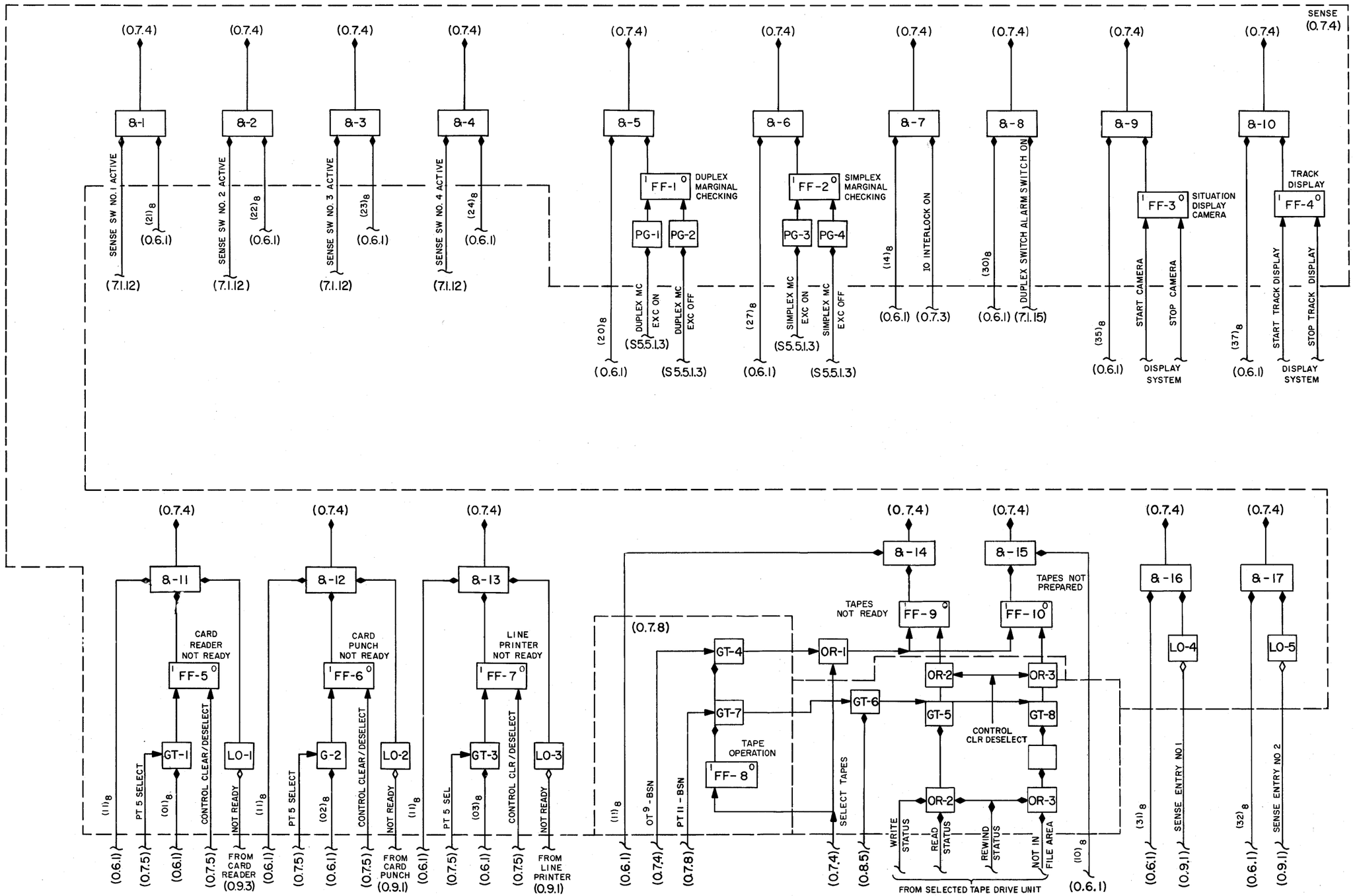


Figure 5–9. Sense Circuits, Group 1, Simplified Logic Diagram

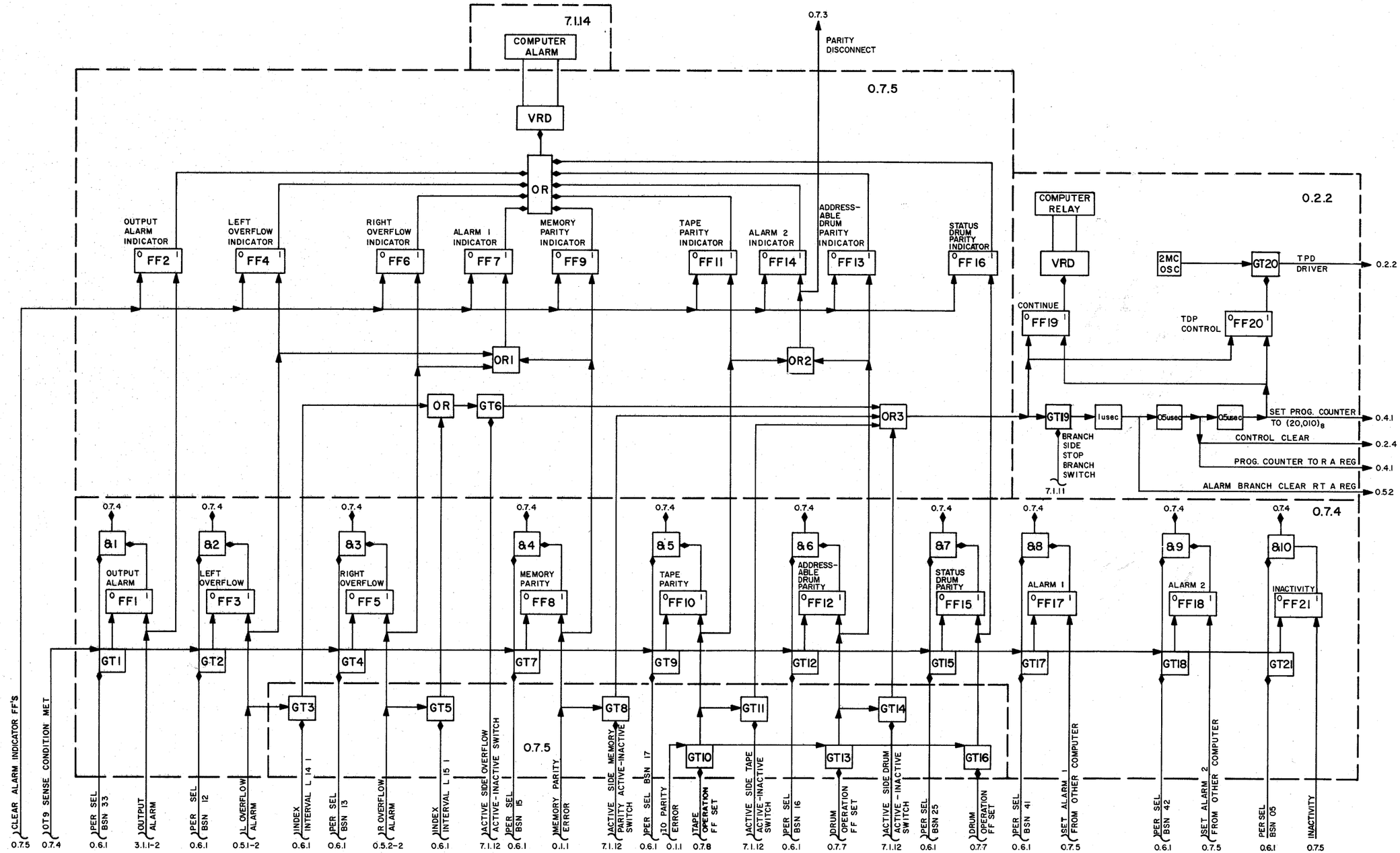


Figure 5-11. Sense Circuits, Group 3, Simplified Logic Diagram

(0.6.1)
IO WORD COUNTER (0.7.3)
FF
RD, WRT ZERO TAPE AND CARD MACH
OR-1
PTI SELECT TAPES
000 011 (03)
000 010 (02)
000 001 (01)
OR-2
CARD MACH OPER
CARD READER
CARD PUNCH
LINE PRINTER
GT-16
WARNING LIGHT CTRLS (0.7.9)
W.L.
000 100 (10)
GT-19
MISC IO (0.7.7)
BURST TIME CTRS
010 001 (21)
GT-21
MANUAL INPUT MATRIX
000 110 (06)
GT-46
GT-18
GT-17
GT-15
SELECT (0.7.5)
156
GT-30
GT-29
(0.2.3) IP 5
(0.3.2) SEL PT
(0.3.2) SEL SDR OT
(0.6.1) 000 001 (01) CARD READER
(0.6.1) 000 010 (02) CARD PUNCH
(0.6.1) 000 011 (03) LINE PRINTER

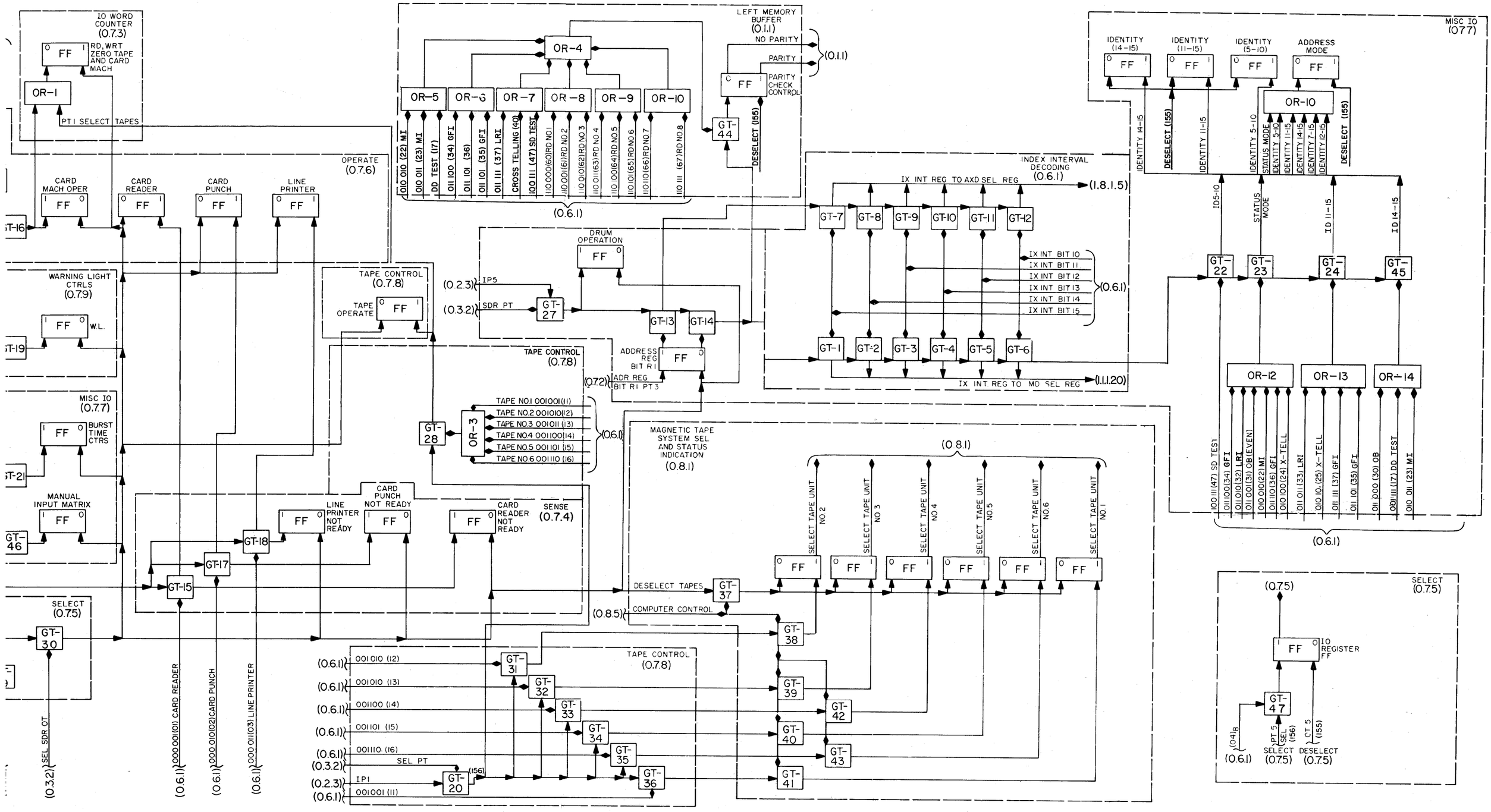
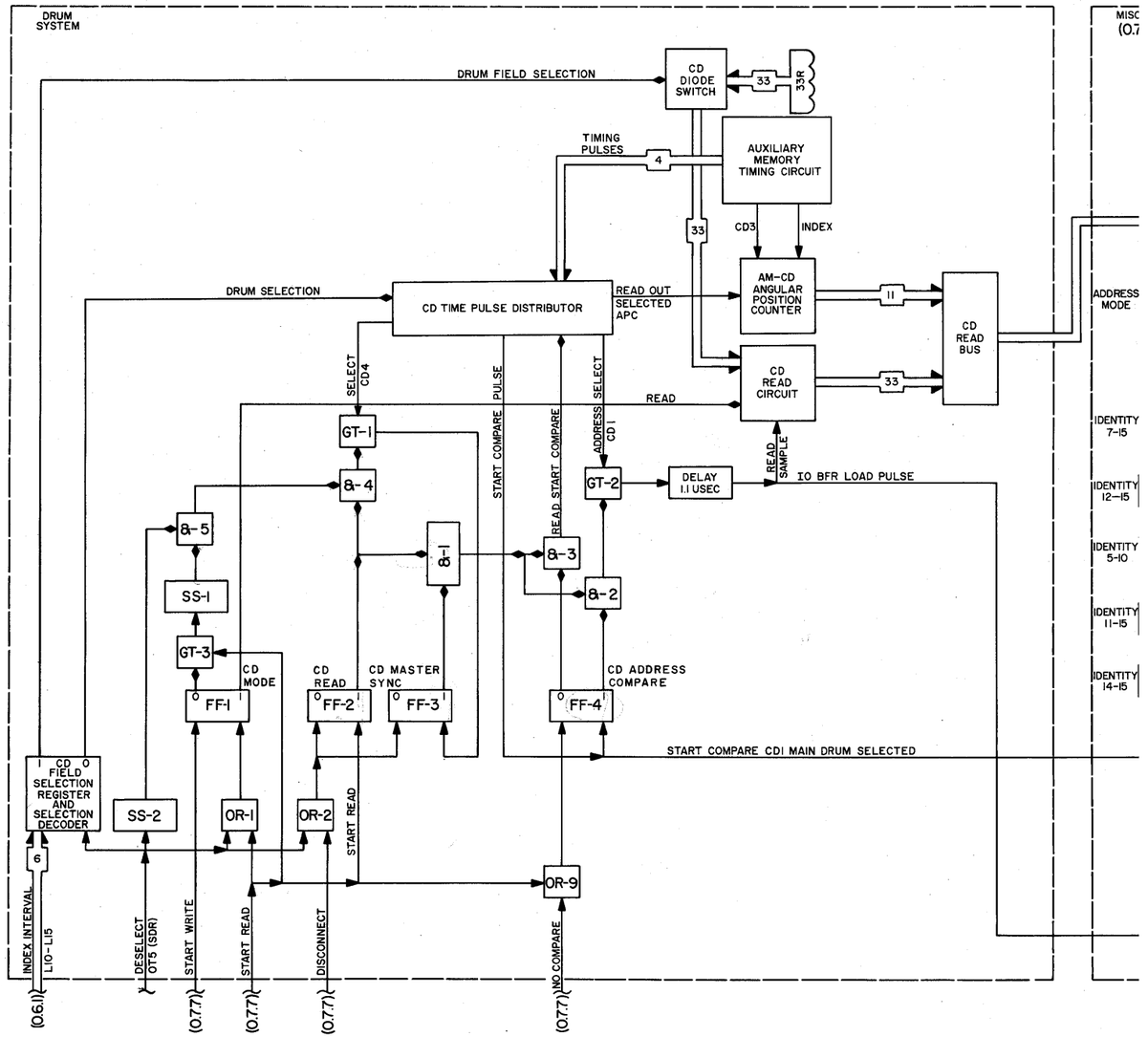


Figure 5–14. IO Unit and Drum Field Selection, Simplified Logic Diagram


DRUM SYSTEM
DRUM FIELD SELECTION
CD DIODE SWITCH
33
33R
TIMING PULSES
4
AUXILIARY MEMORY TIMING CIRCUIT
33
CD3
INDEX
DRUM SELECTION
CD TIME PULSE DISTRIBUTOR
READ OUT SELECTED APC
AM-CD ANGULAR POSITION COUNTER
11
CD READ BUS
CD READ CIRCUIT
33
SELECT CD4
READ
START COMPARE PULSE
READ START COMPARE
ADDRESS CD1
READ SAMPLE
GT-1
8-4
GT-2
DELAY 1.1 USEC
IO BFR LOAD PULSE
8-5
8-1
8-3
8-2
SS-1
GT-3
CD MODE
CD READ
CD MASTER SYNC
CD ADDRESS COMPARE
FF-1
FF-2
FF-3
FF-4
START COMPARE CDI MAIN DRUM SELECTED
CD FIELD SELECTION REGISTER AND SELECTION DECODER
SS-2
OR-1
OR-2
START READ
OR-9
6
INDEX INTERVAL
LIO-LI5
DESELECT OT5 (SDR)
START WRITE
START READ
DISCONNECT
NO COMPARE
(0.6.1)
(0.7.7)
(0.7.7)
(0.7.7)
(0.7.7)
MISC (0.7
ADDRESS MODE
IDENTITY 7-15
IDENTITY 12-15
IDENTITY 5-10
IDENTITY 11-15
IDENTITY 14-15

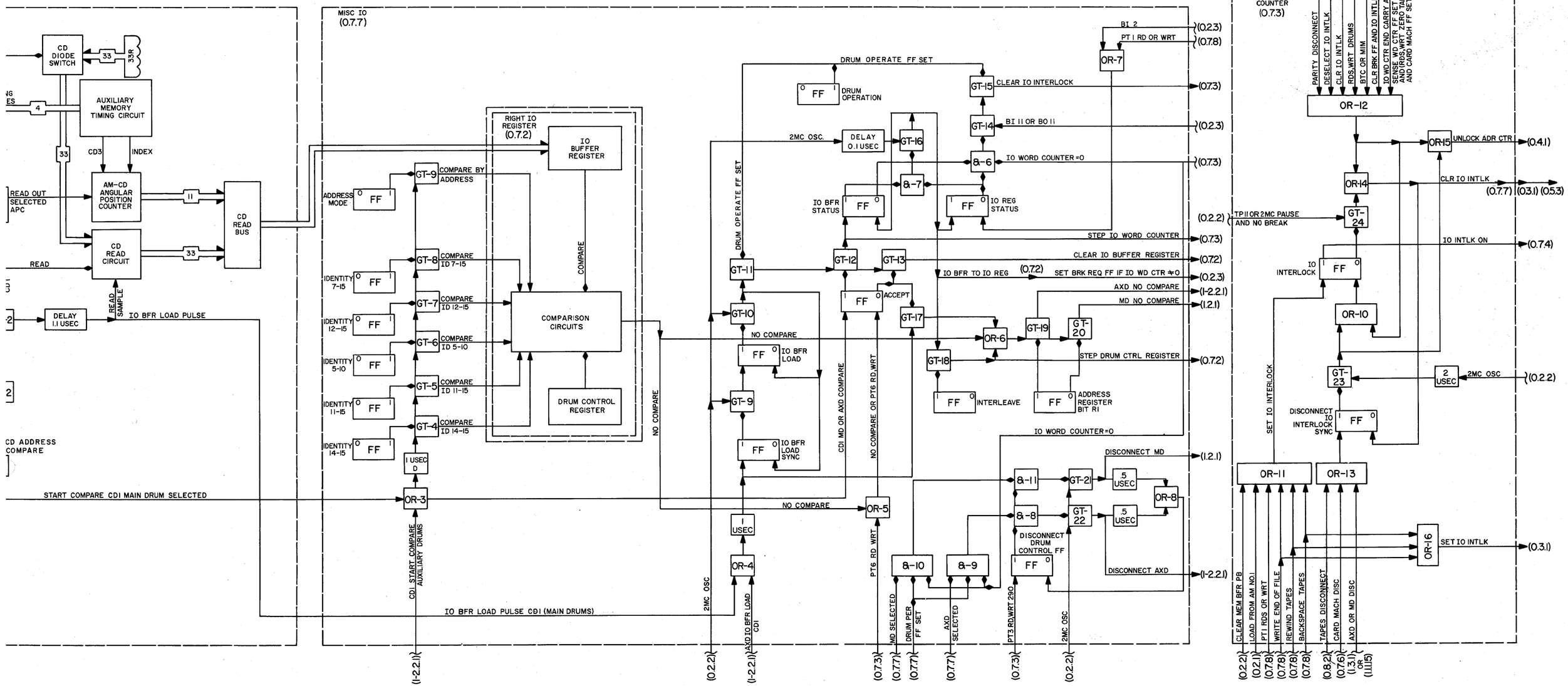
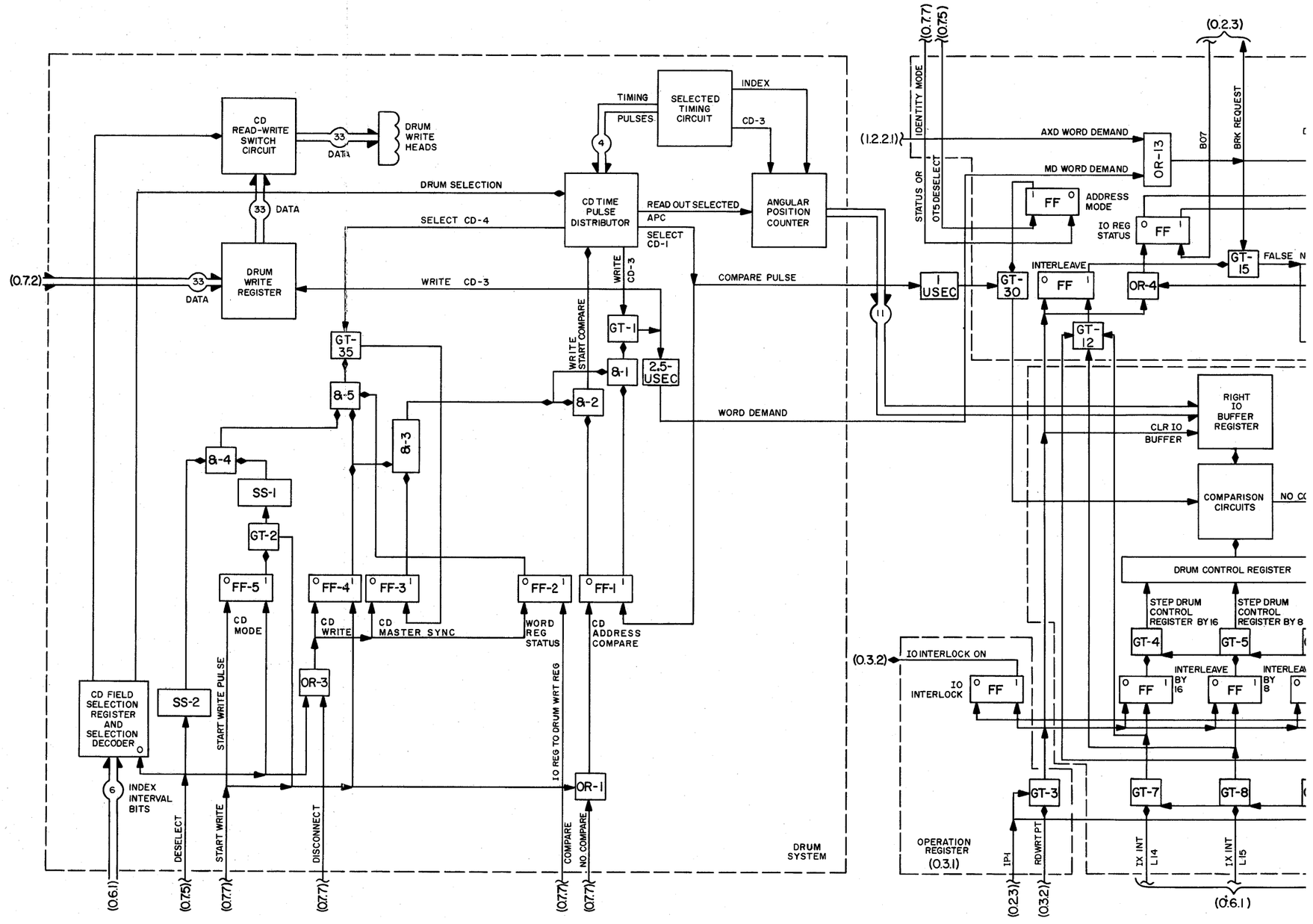


Figure 6-1. Drum Read Circuits, Simplified Logic Diagram


CD READ-WRITE SWITCH CIRCUIT
33 DATA
DRUM WRITE HEADS
DRUM SELECTION
33 DATA
SELECT CD-4
(0.7.2)
33 DATA
DRUM WRITE REGISTER
WRITE CD-3
TIMING PULSES
4
SELECTED TIMING CIRCUIT
INDEX
CD-3
CD TIME PULSE DISTRIBUTOR
READ OUT SELECTED
APC
SELECT CD-1
ANGULAR POSITION COUNTER
COMPARE PULSE
WRITE CD-3
WRITE START COMPARE
GT-35
GT-1
8-1
2.5-USEC
8-5
8-2
8-3
8-4
SS-1
GT-2
FF-5
CD MODE
FF-4
CD WRITE
FF-3
CD MASTER SYNC
FF-2
WORD REG STATUS
FF-1
CD ADDRESS COMPARE
OR-3
CD FIELD SELECTION REGISTER AND SELECTION DECODER
SS-2
START WRITE PULSE
6 INDEX INTERVAL BITS
IO REG TO DRUM WRT REG
OR-1
(0.6.1)
DESELECT
(0.7.5)
START WRITE
(0.7.7)
DISCONNECT
(0.7.7)
COMPARE
(0.7.7)
NO COMPARE
(0.7.7)
DRUM SYSTEM
(0.7.7)
(0.7.5)
IDENTITY MODE
STATUS OR OTS DESELECT
(1.2.2.1)
(0.2.3)
AXD WORD DEMAND
MD WORD DEMAND
OR-13
BO7
BRK REQUEST
FF
ADDRESS MODE
IO REG STATUS
FF
GT-15
FALSE N
INTERLEAVE
FF
GT-30
OR-4
1 USEC
11
GT-12
WORD DEMAND
RIGHT IO BUFFER REGISTER
CLR IO BUFFER
COMPARISON CIRCUITS
NO C
DRUM CONTROL REGISTER
STEP DRUM CONTROL REGISTER BY 16
STEP DRUM CONTROL REGISTER BY 8
GT-4
GT-5
(0.3.2)
IO INTERLOCK ON
IO INTERLOCK
FF
INTERLEAVE BY 16
FF
INTERLEAVE BY 8
FF
GT-3
GT-7
GT-8
OPERATION REGISTER (0.3.1)
IPH
(0.2.3)
RDWRT PT
(0.3.2)
IX INT L14
IX INT L15
(0.6.1)

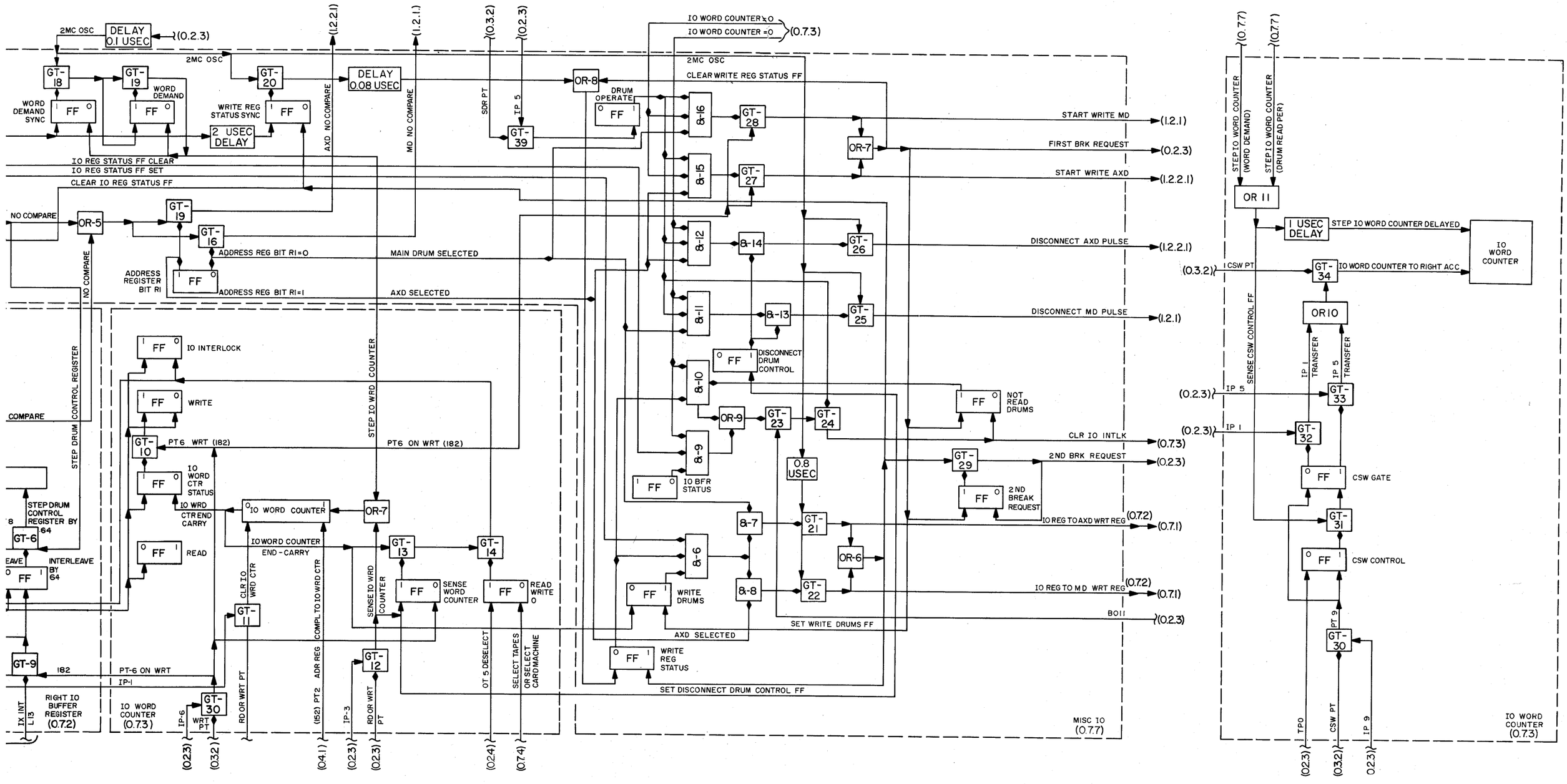


Figure 6–2. Drum Write Circuits, Simplified Logic Diagram

(0.9.3)
(0.9.1)
(0.2.3)
OPERATE
(0.7.6)
START CARD READER
START CARD PUNCH
START LINE PRINTER
CARD MACH 2 ND
BRK REQ
DISCONNECT CARD
READER
IO WORD
(0.7
BO 2
BI 2
BO 7
BI 7
BI 9
RYD-1
RYD-2
RYD-3
SS-1
SS-2
SS-3
GT-12
GT-11
GT-10
GT-9
GT-8
GT-4
GT-6
GT-2
8-1
CARD
READER
CARD
PUNCH
CARD
PRINTER
CARD
MACHIN
FF
CARD READER (1ST) BREAK
REQUEST
GT-1
GT-3
GT-5
GT-7
OR-1
PT5 (SEL)
000001
000002
000003
(0.7.5)
(0.6.1)

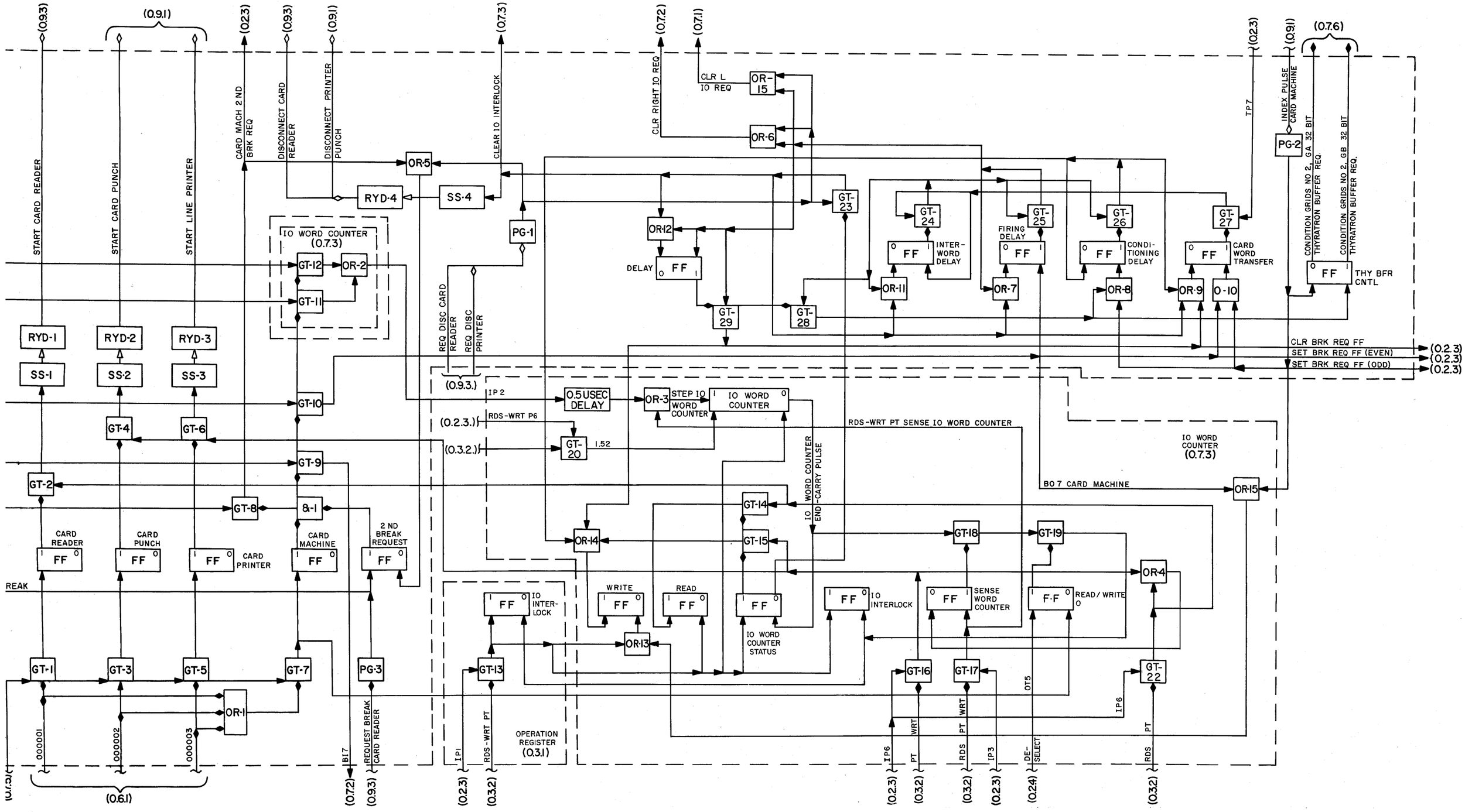
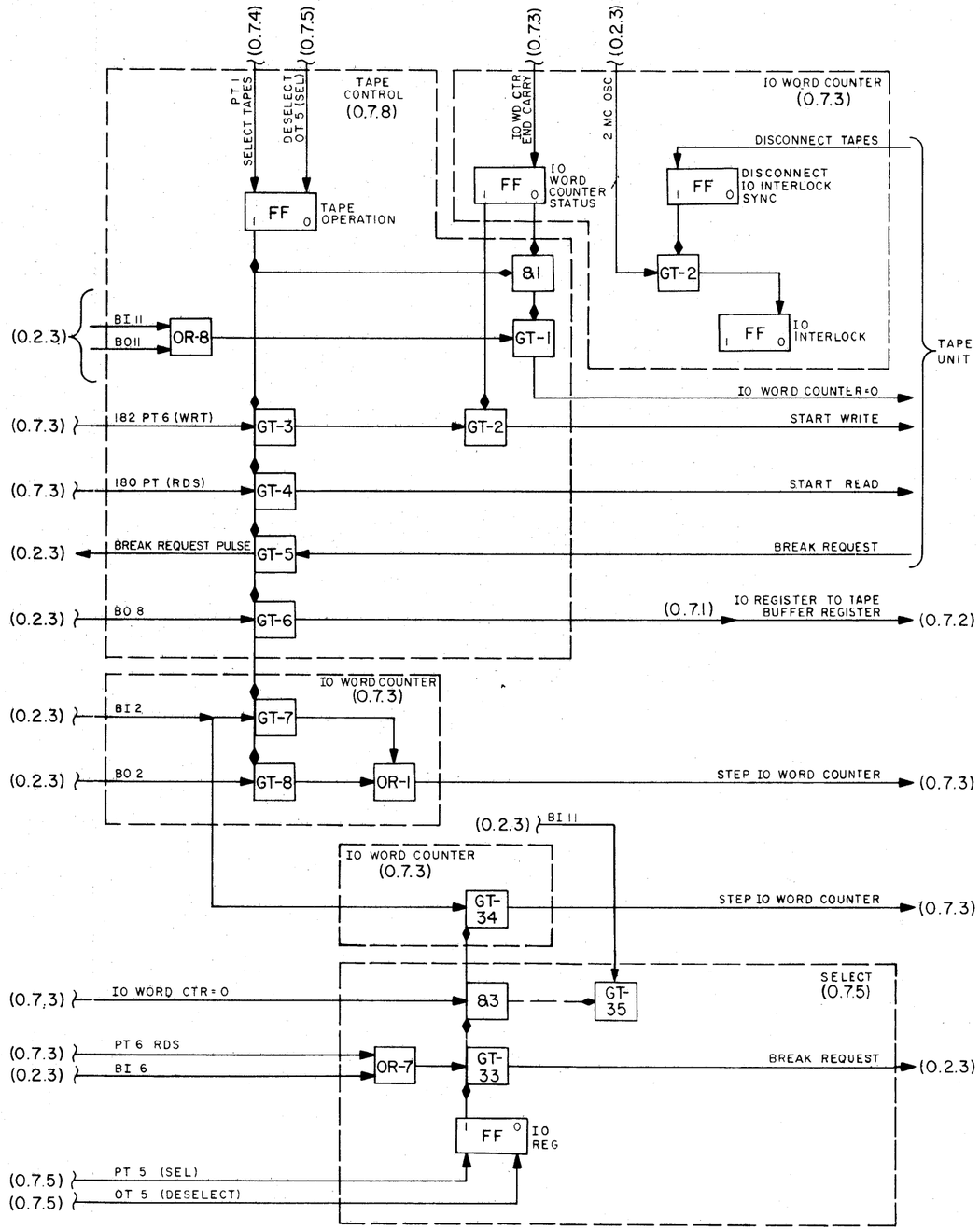


Figure 6–4. Card Machines Read and Write Operations, Simplified Logic Diagrams


TAPE CONTROL (0.7.8)
PT SELECT TAPES (0.7.4)
DESELECT OT 5 (SEL) (0.7.5)
FF TAPE OPERATION
IO WORD COUNTER (0.7.3)
IO WD CTR END CARRY (0.7.3)
2 MC OSC (0.2.3)
FF IO WORD COUNTER STATUS
DISCONNECT TAPES
FF DISCONNECT IO INTERLOCK SYNC
GT-2
FF IO INTERLOCK
TAPE UNIT
B1
GT-1
(0.2.3) BI 11
BO 11
OR-8
IO WORD COUNTER=0
(0.7.3) I82 PT 6 (WRT)
GT-3
GT-2
START WRITE
(0.7.3) I80 PT (RDS)
GT-4
START READ
(0.2.3) BREAK REQUEST PULSE
GT-5
BREAK REQUEST
(0.2.3) BO 8
GT-6
(0.7.1) IO REGISTER TO TAPE BUFFER REGISTER (0.7.2)
IO WORD COUNTER (0.7.3)
(0.2.3) BI 2
GT-7
(0.2.3) BO 2
GT-8
OR-1
STEP IO WORD COUNTER (0.7.3)
(0.2.3) BI 11
IO WORD COUNTER (0.7.3)
GT-34
STEP IO WORD COUNTER (0.7.3)
SELECT (0.7.5)
(0.7.3) IO WORD CTR=0
B3
GT-35
(0.7.3) PT 6 RDS
(0.2.3) BI 6
OR-7
GT-33
BREAK REQUEST (0.2.3)
FF IO REG
(0.7.5) PT 5 (SEL)
(0.7.5) OT 5 (DESELECT)
MISC IO (0.7.7)
(0.2.3) BI 11
GT-13
(0.7.3) IO WD CTR=0
B2
IO WORD COUNTER (0.7.3)
(0.2.3) BI 2
GT-12
REQUEST BREAK MANUAL INPUT MATRIX OR BURST TIME CTR (2.2.2) (3.1.2)
GT-11
OR-2
(0.7.3) PT6 (RDS) I80
GT-9
MANUAL INPUT MATRIX
FF
(0.7.5) DESELECT OT 5
GT-15
(0.7.4) PT 5 SELECT
000 110

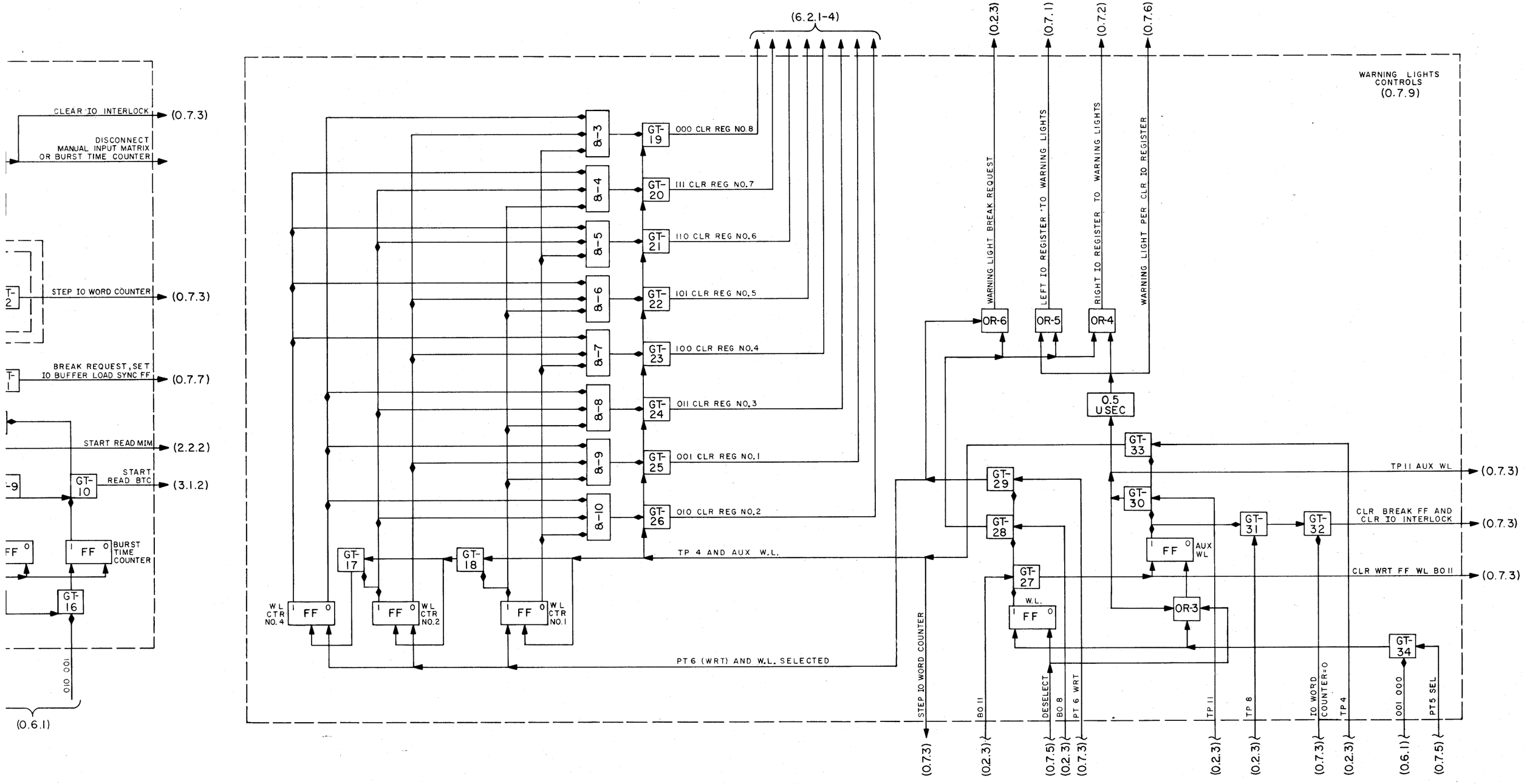


Figure 6-5. Magnetic Tape Units and Miscellaneous IO Units Control Circuits, Simplified Logic Diagram

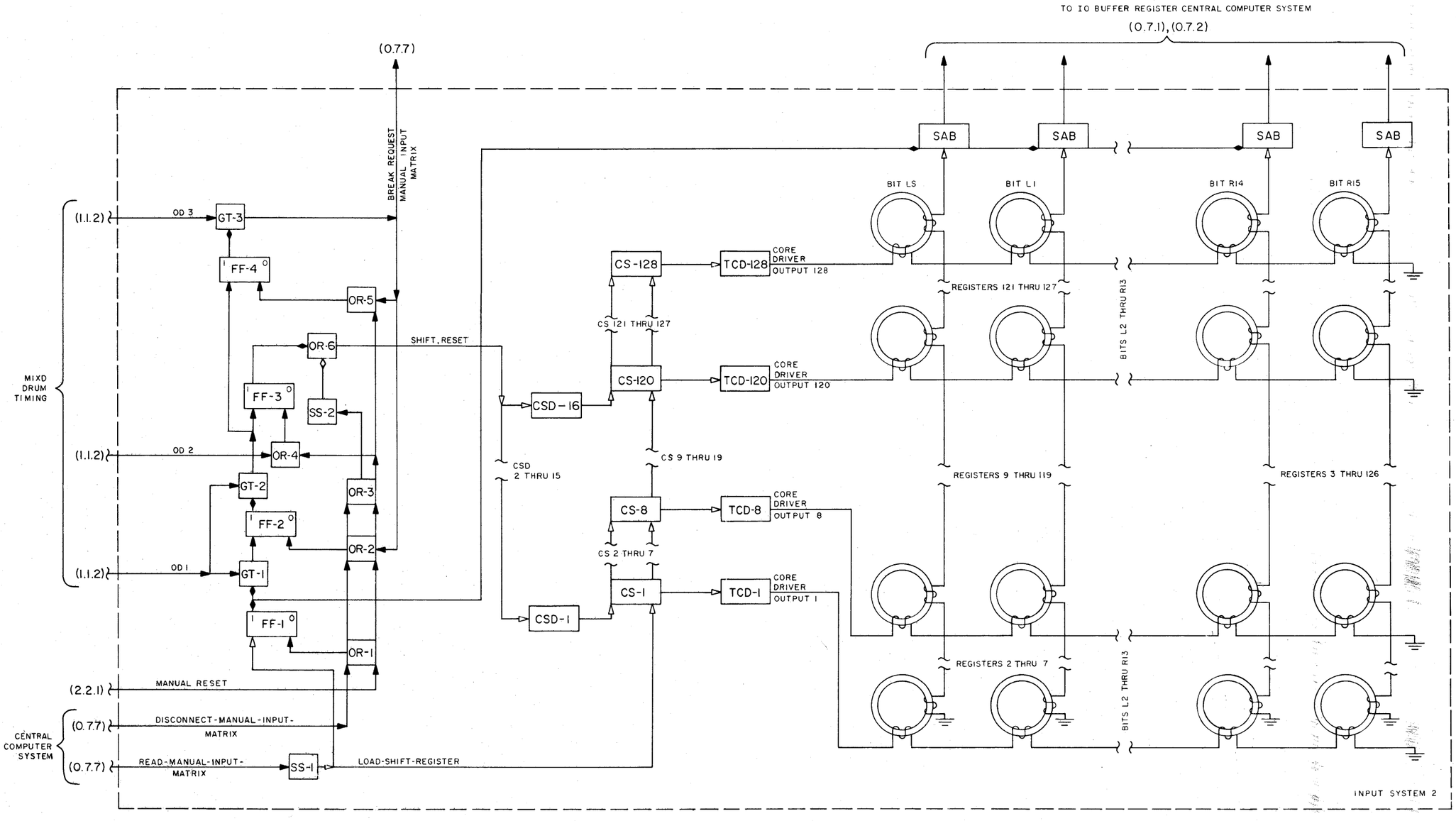
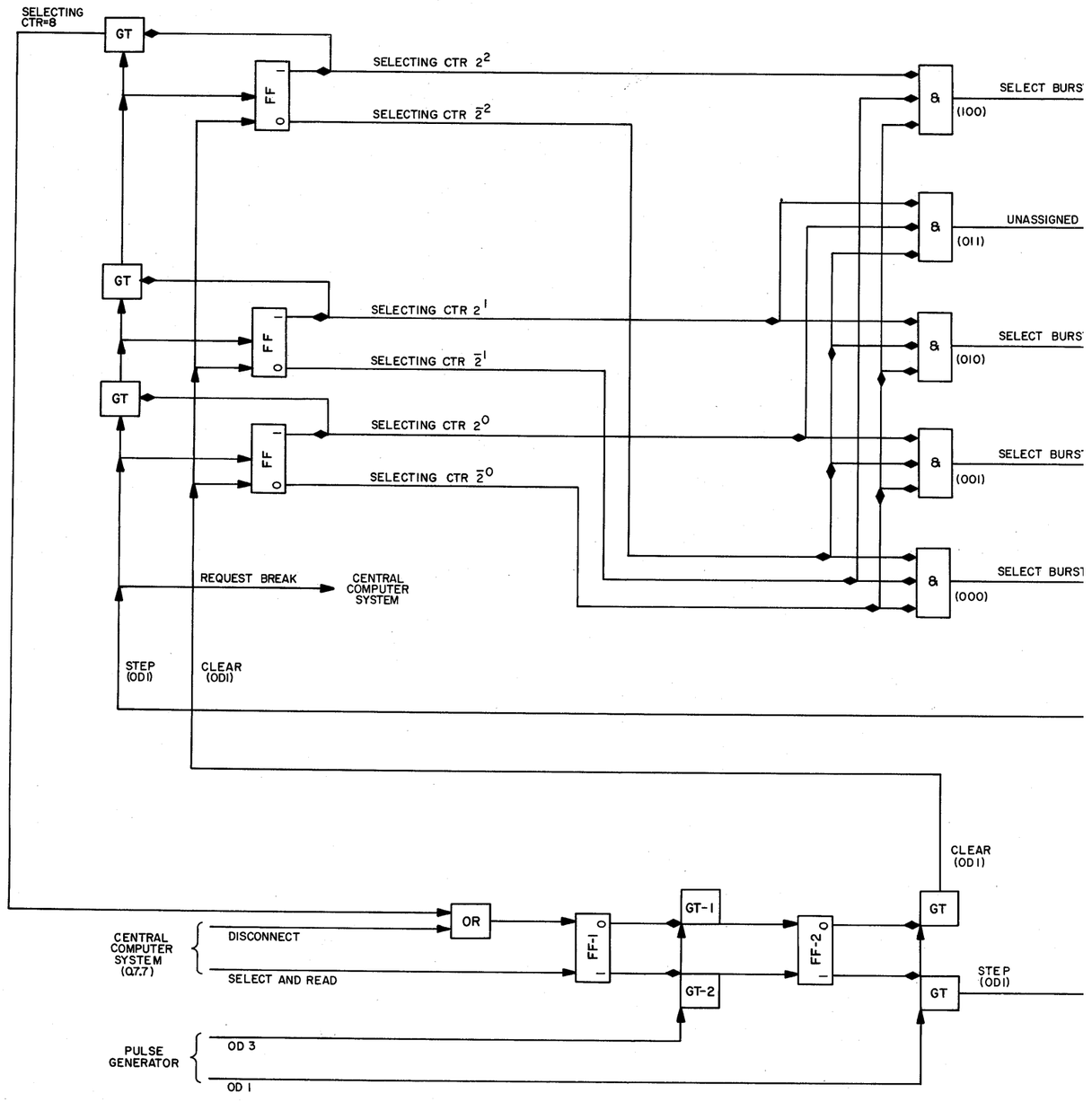


Figure 6–6. Manual Input Matrix, Simplified Logic Diagram


SELECTING CTR=8
GT
FF
1
0
SELECTING CTR 2^2
SELECTING CTR $\bar{2}^2$
SELECTING CTR 2^1
SELECTING CTR $\bar{2}^1$
SELECTING CTR 2^0
SELECTING CTR $\bar{2}^0$
&
SELECT BURST
(100)
UNASSIGNED
(011)
SELECT BURST
(010)
SELECT BURST
(001)
SELECT BURST
(000)
REQUEST BREAK
CENTRAL COMPUTER SYSTEM
STEP (OD1)
CLEAR (OD1)
CLEAR (OD1)
OR
CENTRAL COMPUTER SYSTEM (Q7,7)
DISCONNECT
SELECT AND READ
FF-1
FF-2
GT-1
GT-2
STEP (OD1)
PULSE GENERATOR
OD 3
OD 1

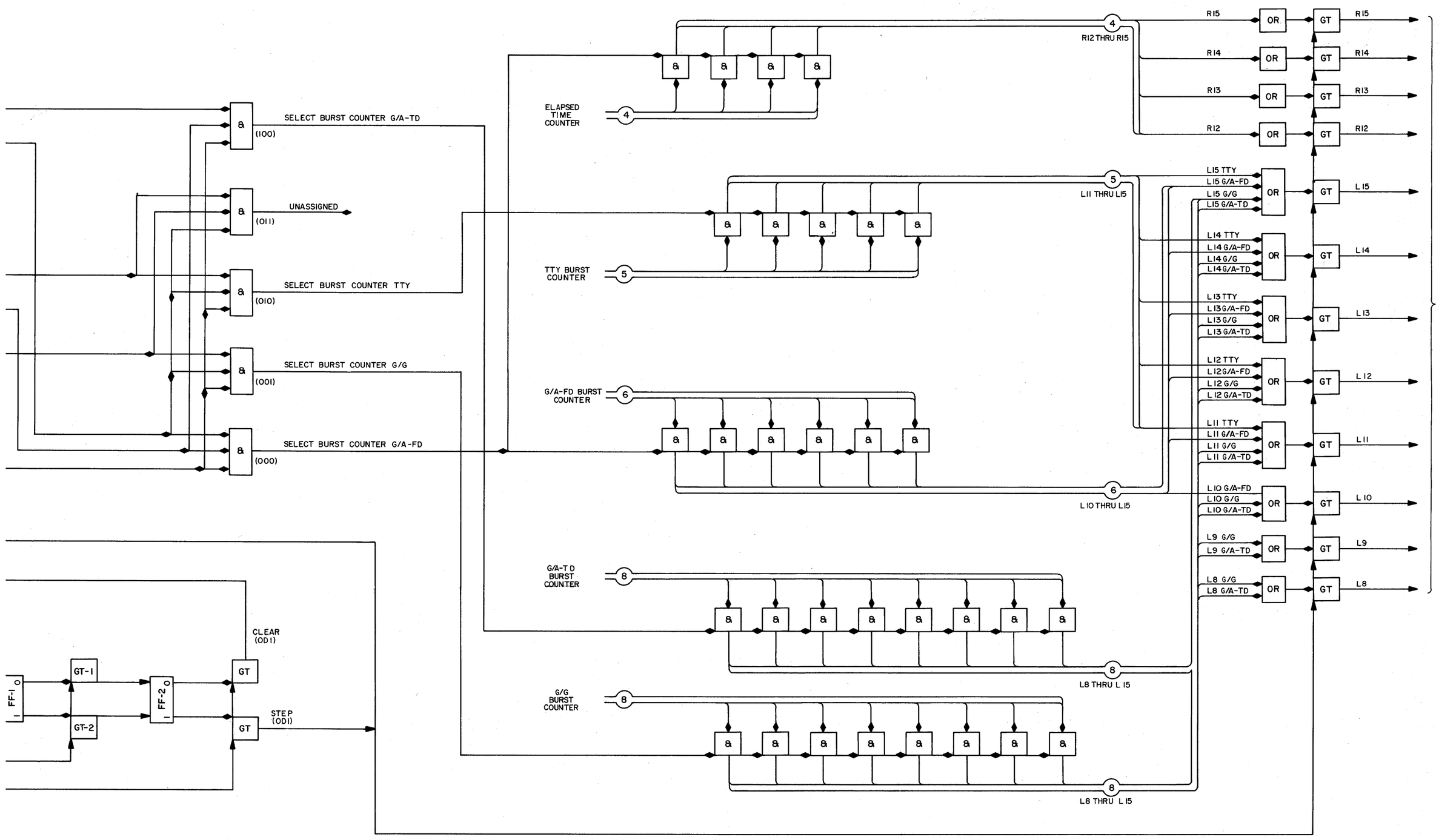


Figure 6-7. Output System, Computer Section, Simplified Logic Diagram