UNIVAC®

File

Computer

model 1

BASIC PROGRAMMING

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

BASIC PROGRAMMING

File

UNIVAC®

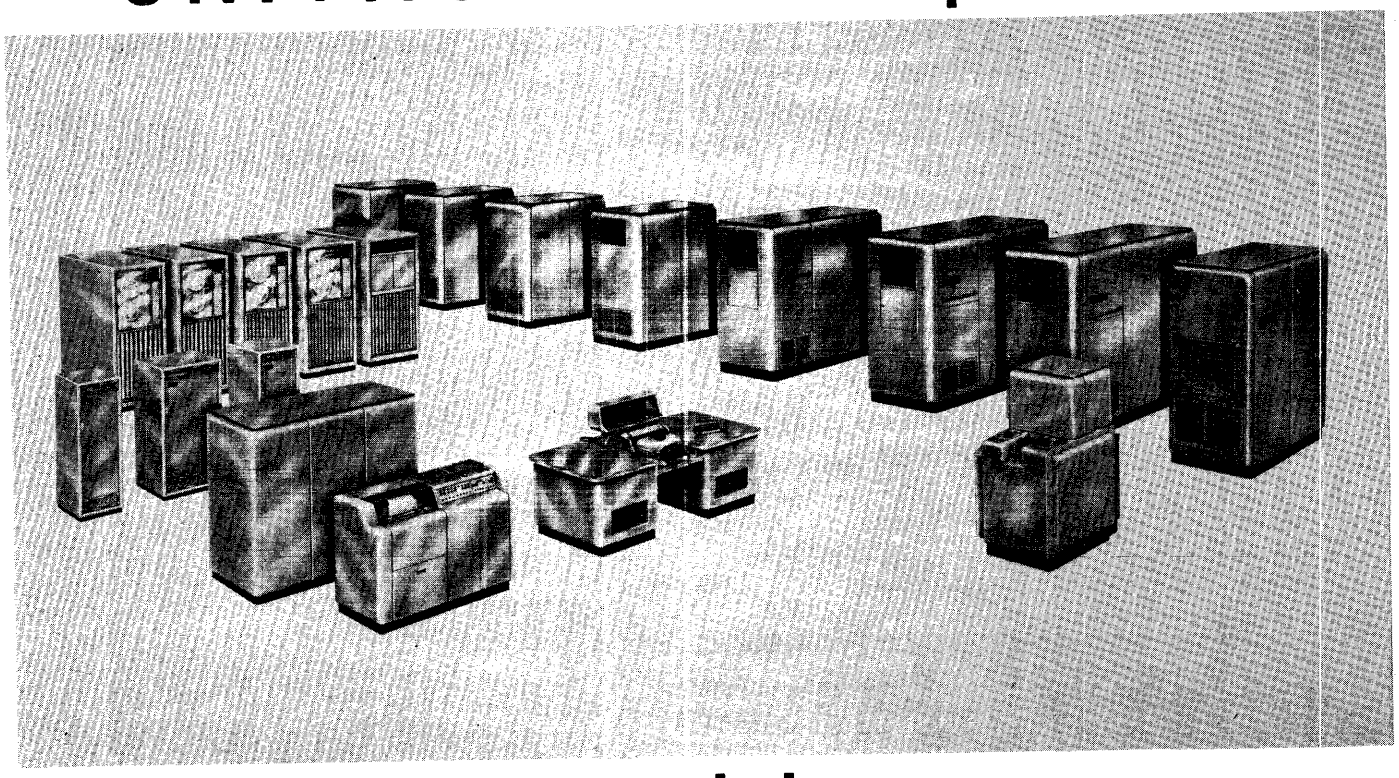
ANOTHER SERVICE OF . . . MANAGEMENT SERVICES AND OPERATIONS RESEARCH DEPARTMENT

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

Computer

DATA AUTOMATION SYSTEM

UNIVAC® File-Computer



model 1

CONTENTS

chapter 1

INTRODUCTION

DATA PROCESSING AREAS IN BUSINESS

The first step in a study of electronic computers is to survey the areas of business operations wherein a computer may become a useful managerial tool. These areas are called *data processing* areas. In its day-to-day functioning, a manufacturing concern is composed of myriad channels through which money and material flow in fulfillment of the company's obligations to its stockholders, employees, vendors, customers and the government. From a data processing point of view, these areas are concerned with management's attempts to record, measure and effectively control this flow. Because of its broad yet familiar activities the manufacturing company's activities will be considered. Figure 1-1 is a generalized block diagram of a typical manufacturing company and its environment.

A TYPICAL MANUFACTURING ORGANIZATION
AND ITS DATA PROCESSING ENVIRONMENT

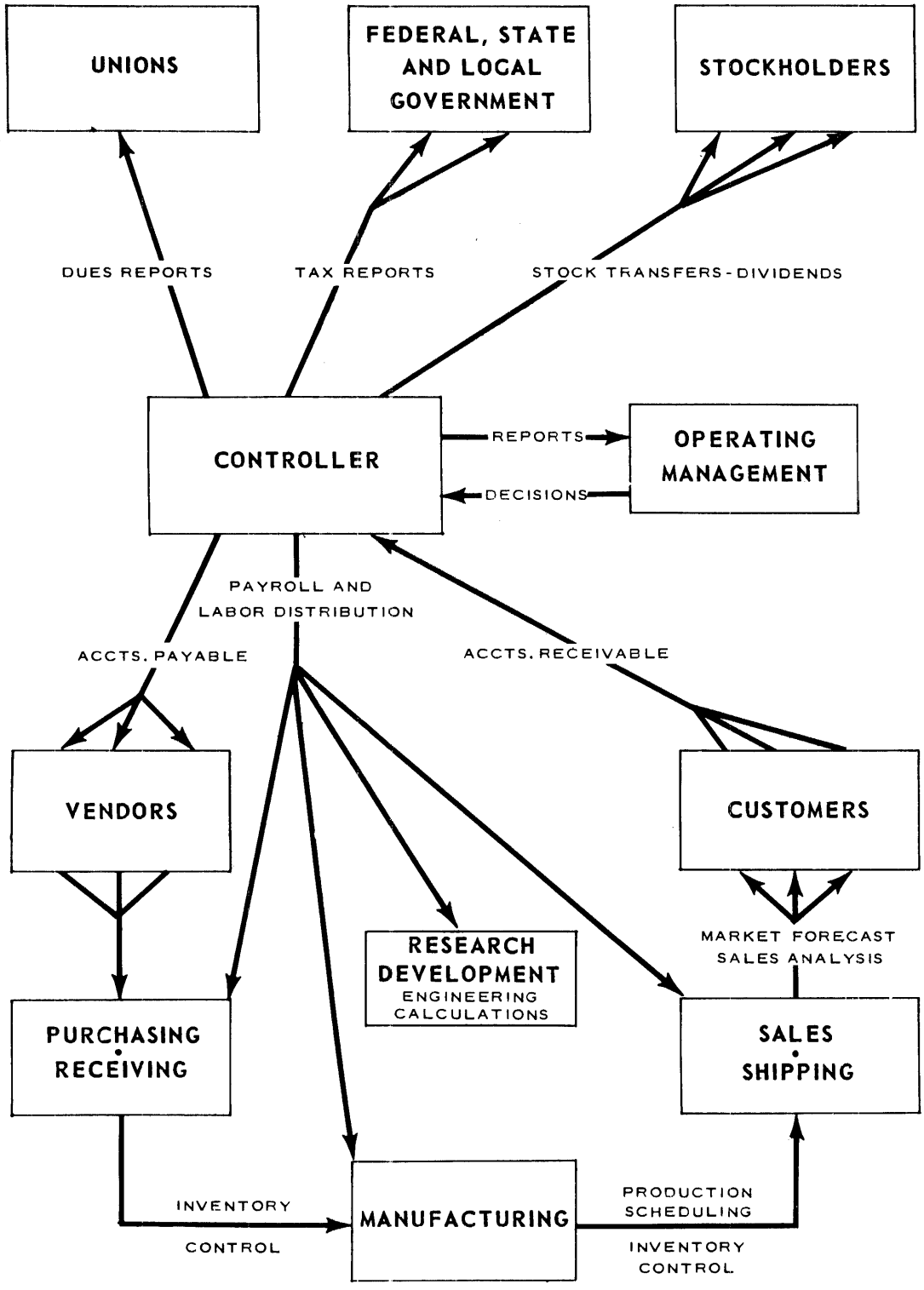


FIGURE 1-1

The most common data processing areas have been indicated on the chart. A very brief description of each is listed below.

Market Forecasting and Sales Analysis

To attempt to find the beginning of the movement through the channels pictured would be to search for the beginning of a circle, because of the multitudinous cross-references and interdependencies which exist. From the point at which planning for the next year commences, however, a certain sequence does follow.

At that point the big question is, "How good will the new year be?" The answer can be found by making a reliable sales forecast to serve as a basis upon which all operational planning will be laid.

The past sales history is essential to such a forecast. Thus, many concerns break down their sales as often as once per week according to the products sold, the regions in which they were sold, the dollar values of the sale, the percentage gross and/or net profit obtained and other significant criteria. In addition to predicating any immediate action which needs to be taken, such reports, if compiled over a period of years, will yield information on the seasonal and regional fluctuations of the sales of various products.

A further study of such a sales analysis may bring out some revealing correlation between the concern's sales and the general business trends and cycles, customer activities and similarly relevant factors. Such correlations are not always easy to find; but once discovered, they offer the means of making a reliable forecast of the sale of each product in each marketing area. An evaluation of the market forecast will affect the budget and production levels to be maintained during the year.

Production Scheduling

The sales forecast and any adjustments to it which may be necessary as the year progresses are the sources of the production orders. The production orders indicate the date of completion and size of each batch of every product to be manufactured. Referencing these orders against a bill of materials listing is then the basis of the production scheduling operation. This listing contains the material, machines and time required for the completion of each phase in the manufacture of the product. Working backwards from the "due date" it is possible to list the times at which materials and machines must be available if the due date is to be met. Proper planning is essential since any mis-scheduling of machine requirements may result in extra production

expense for overtime on the one hand, or time and money loss because of idle machinery and idle manpower on the other. In addition to yielding a machine schedule, the bill of materials listing yields the requisitions for the total raw material requirements and the time in the production line at which they must be available.

Inventory Control

From the bill-of-materials, information is also obtained for inventory control. As a by-product of the machine scheduling, the quantities of raw materials needed during each manufacturing phase are also determined. These raw material requirements are used for the publication of requisitions. In addition, they are compared to the current inventory level of the material and posted to it. If the reorder level is reached, production or purchasing orders, depending on whether the material is processed within the company or purchased, are issued in order to replenish the stock. Proper use of reorder levels can offer considerable savings by accurate control of the minimum inventory level to be maintained. Accurate inventory control is essential in reducing capital investment and storage obsolescence costs of large inventories, or the costs of emergency reorders and delays resulting from shortages.

Accounts Payable

The accounts payable operation is initiated by the receipt of an invoice from the vendor. This invoice is first checked for amounts billed against quantities received and priced against the current price list. Then, although immediate payment of all accurate invoices is possible, payment is usually postponed temporarily to allow further use to be made of the available cash. Such unpaid invoices are listed on the accounts payable ledger. Cash balances and the efficient use of any discount privileges determine the time for selection from this ledger for payment. Checks are produced and appropriate entries made in the vendor's account. Information may also be extracted for such things as general ledger and property accounting, and reports on vendor activity.

Payroll and Labor Distributions

This area is commonly the most highly mechanized data processing area in business today. In spite of the fact that all payrolls are designed primarily to produce paychecks, the variety in important payroll details caused by

unusual or individual labor contracts, differing local and state regulations and plant policies preclude a complete uniformity of description. With this precautionary statement in mind, consider payroll data processing to be divided into three parts: determination of gross pay, computation of net pay from the gross, and labor distribution.

Determination of gross pay may be a trivial operation in the case of a salaried payroll. In most cases, however, the determination of gross pay is an involved process. Gross pay is often based upon the number of hours worked in each of several hourly rate categories (regular and overtime factors) during the pay period. This may be modified in many plants to include bonus or efficiency payments determined by the output of groups of workers or by a piecework schedule. The net pay calculation involves the computation of tax deductions imposed by state, local and federal governments and such other deductions, usually variable in amount from one pay period to the next, as specified by union contracts and fringe benefits or employee options. The end product of the net pay calculations is a series of paychecks (or pay slips if payment is by cash) and various payroll registers listing gross and net pay and the several deductions. In addition to these, the individual earnings record must be updated for end-of-quarter and end-of-year government tax reports.

The labor distribution phase is used by management to establish product costs and selling prices. Gross pay and hours-worked data for each employee, established in the gross pay phase, are distributed to each product, account or activity he has engaged in during the pay period. These are then summarized to produce labor costs for each of the distributed categories.

Tax Reports, Union Dues Reports, Etc.

Under present labor-management practices, management assumes many of the employee's obligations to his environment. Taxes, union dues and various voluntary deductions are withheld. The necessity arises for the firm to make reports to the government, union dues reports to the unions, hospitalization and insurance reports, etc. The information for such reports is available from the payroll processing itself.

Year-to-date totals of gross pay, income tax withheld and FICA tax are sufficient for the preparation of W-2 forms. Similarly a compilation from the employee files and payroll processing results is all that is necessary in the preparation of most other reports.

Accounts Receivable

The accounts receivable operation commences when a shipping document is received. Products listed on this document are priced and the shipment is extended to produce the invoice sent to the customer. At the same time, the total dollar charge is posted to the customer's records on the accounts receivable ledger.

This ledger is often scanned daily. Cash receipts and any earned discounts are credited to it. Appropriate information is entered into the customers' credit history. Aging accounts are extracted, checked, their credit history examined and appropriate action is taken. At the end of the month, the information present is compiled to form monthly statements, which also may be sent to the customer.

Stock Dividends and Transfers

Data processing is also necessary in connection with stock dividends and transfers. Stockholder listings must be periodically maintained to assure that they reflect the latest results of all stock issues, cancellations and transfers. When a dividend is declared, it is then only necessary to select the owners as of that date from the listing and multiply the dividend rate by the number of their shares to make the proper disbursement. Similarly a scanning of this list is sufficient when it is necessary to print and distribute the proxy ballots for the annual stockholders meeting. Year-to-date dividends paid and other information on this listing may be employed in the preparation of the year-end state and federal tax reports and of any statistical reports desired.

ELEMENTS OF A DATA PROCESSING SYSTEM

Before turning directly to the study of the characteristics of the Univac File-Computer Model I System, it is advisable to review the basic elements of a general data processing system.

Consider a business which keeps a record of its stock in a ledger. Each day a clerk is supplied with a form which indicates the number of items sold, listed in stock number order. On the basis of this information, the man brings the inventory up to date by writing a new column in the ledger.

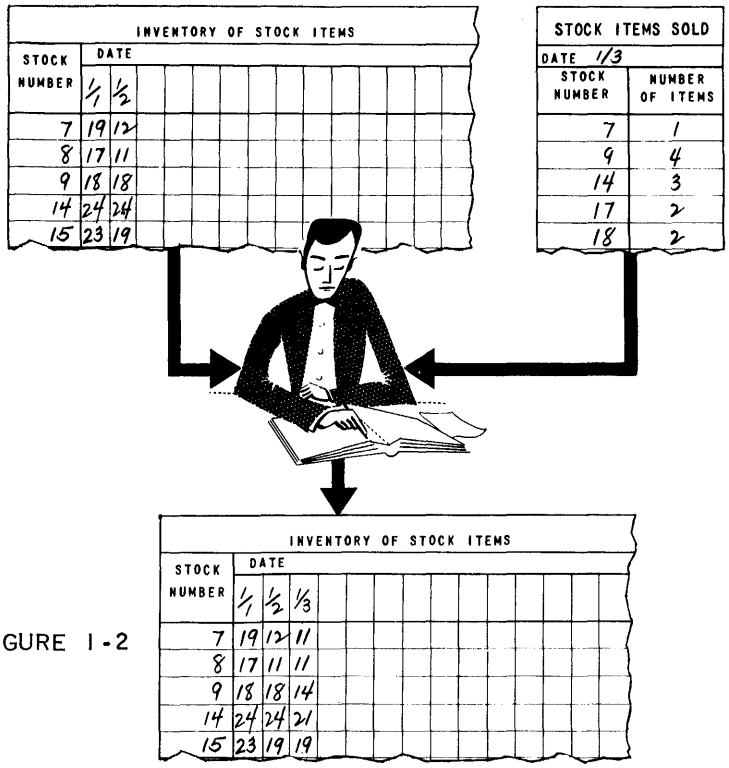


FIGURE 1-2

Even in such a simple example as this may be found the fundamental elements of a data processing system: variable information, master file data, a data processor, reports or other printed output. The list of stock items sold is the variable information; the stock ledger contains the master file data; the inventory clerk serves as the data processor; and the output of the data processing system is the updated stock ledger.

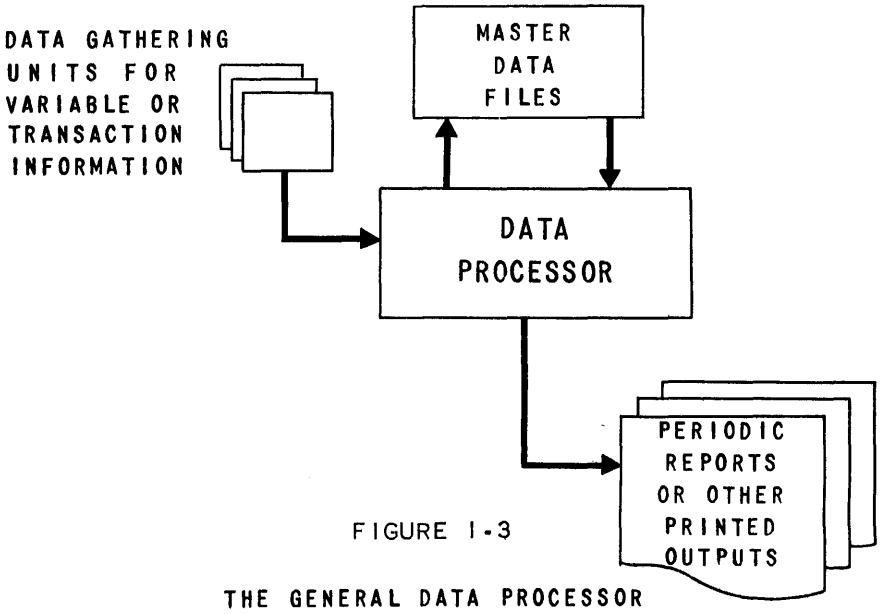


FIGURE 1-3

THE GENERAL DATA PROCESSOR

A data processing system is best described by its outputs. These are the various reports, summaries, statistics, bills, checks, invoices, etc. required by management or government, or the every day facts and figures which are a necessary part of the detailed operation of the company. The information required on these printed outputs, and the time intervals at which this data must be supplied are the two factors which establish the general requirements of the other three elements.

The inputs to the data processing system, from which the output data are to be compiled, usually consist of two types of information: master data files that remain essentially unchanged from one reporting cycle to the next or which change in known or fixed ways; and variable or transaction information which is produced by the day to day activities of the business.

Master File data are the permanent information records containing identifying and historical facts about the individual, account, item, product or service being reported. Examples of master file data are: names and addresses, employee badge numbers, account numbers, current credits and debts, running inventories, etc.

Variable information is data introduced into the data processing system reflecting current operations. It is generated by human activity and is thus essentially unpredictable. Examples of variable information are: the hours worked by an employee, receipts, expenditures, sales, shipments, etc. Since the transactions producing these variables are often physically dispersed (coming from different divisions, departments or branch offices) some means for gathering the data for injection into the system is required.

The data processor is the converter of master and variable data into the output reports. It must also post changes, when necessary, to the master data files. These changes are introduced to the data processor through essentially the same data gathering units which are used for the variable information. The data processor may be:

a clerical staff laboriously making thousands of detailed entries per working day in the manner of the inventory clerk in Figure 1-2;

an electronic computer such as the Univac File-Computer which performs the same data-handling operations at electronic speeds;

any of the many combinations of manual, key driven or punched card data processors between these two extremes.

In order to be able to produce the desired outputs, the data processing system must be able to:

Read documents
Record documents and reports
Sort and classify data
Calculate
Make simple decisions

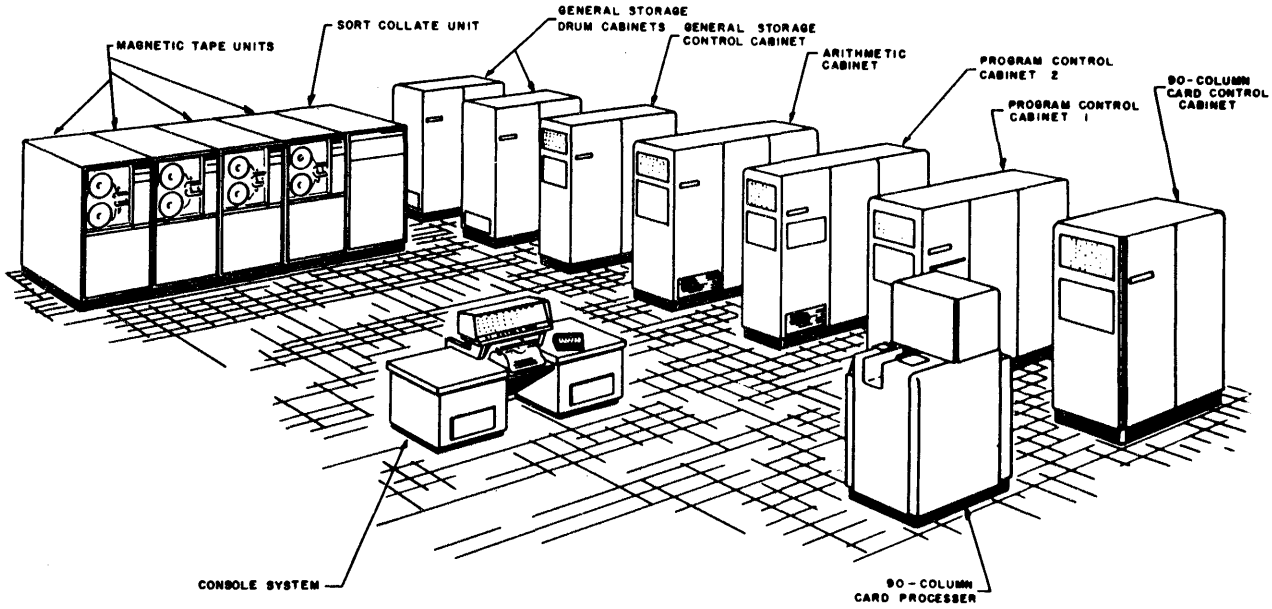


FIGURE 1-4

ELEMENTS OF THE UNIVAC FILE-COMPUTER, MODEL I SYSTEM

Comparison of Manual and Electronic Data Processing

How are the basic elements of a data processing system reflected in an electronic computer application? The following comparison of a manual system with an electronic computing system, is intended to point out the similarities and differences in overall approach to the accomplishment of the same data processing task. In this instance, the manual system is exemplified by an inventory clerk, and the computer is the Univac File-Computer, Model I.

The data processing application consists of the updating of a stock ledger by posting to it the number of items of each stock number sold the previous day, as shown in Figure 1-5.

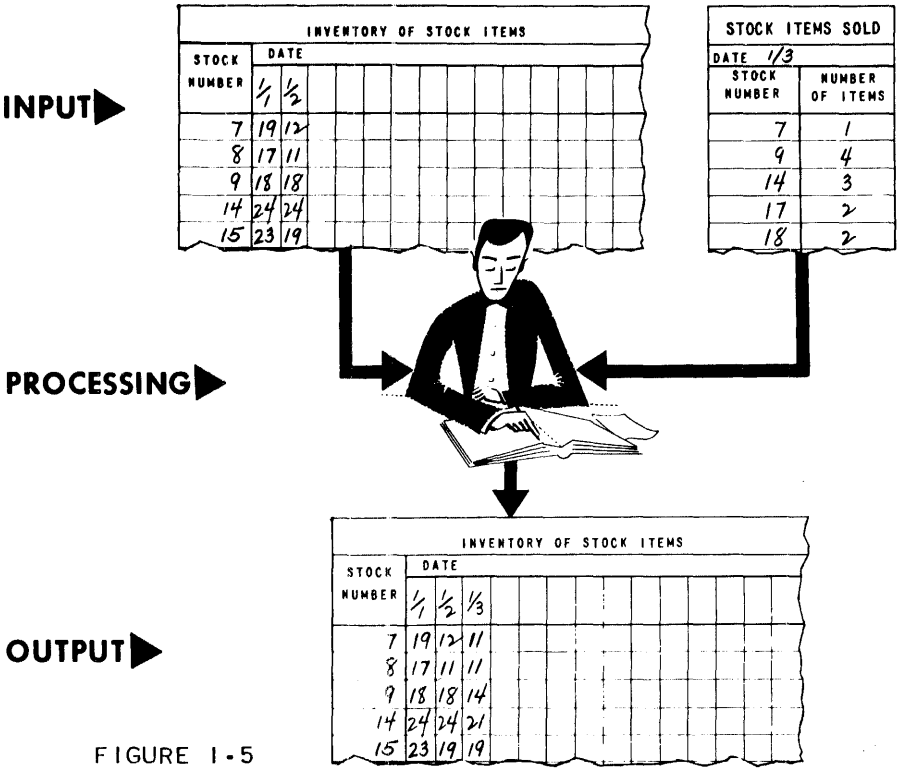


FIGURE 1-5

First, the inputs to the system must be prepared for either the inventory clerk or the computer:

> The inventory clerk must get the stock ledger and take it to his desk. The list of stock items sold must also be prepared for his use.
>
> Although input data might be prepared in one of several different ways for the versatile UFC-I system, a typical inventory application might be as follows: As part of the previous day's inventory processing, the stock inventory was unloaded from the large-capacity, random access storage of the UFC-I onto a Univac magnetic tape. This "stock ledger" is now re-loaded onto the general storage magnetic drums. The list of stock items sold is key-punched into either 80 or 90 column punched cards and is ready for input to the computer.

Second, to do the processing, the clerk or the computer must go through certain steps:

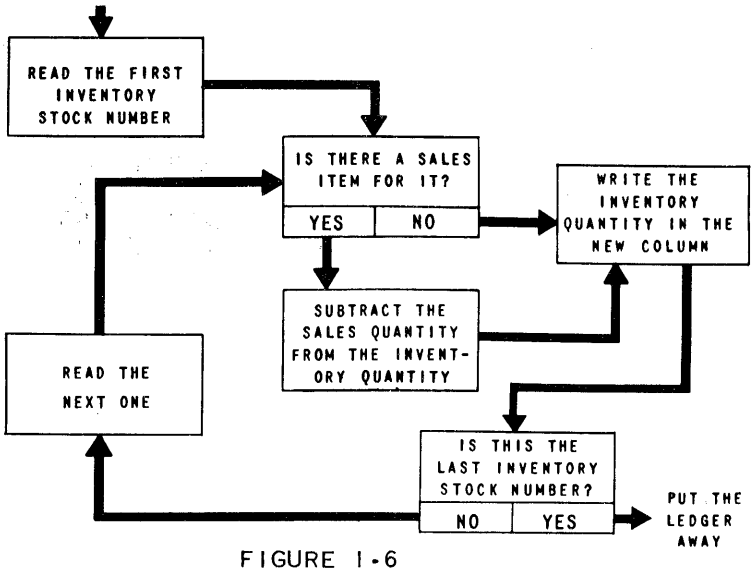
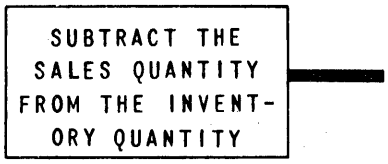


FIGURE 1-6

Both must be able to do arithmetic:

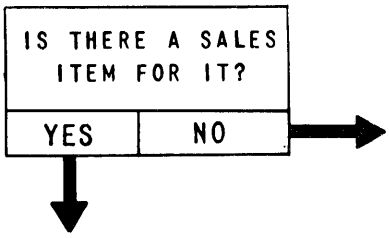
FIGURE 1-7



The UFC-I is capable of performing all of the arithmetic operations of addition, subtraction, multiplication and division at electronic speeds. It will also check each one of these processes for complete accuracy by performing a reverse arithmetic operation — without the necessity for complicated programming techniques.

Both must be able to make logical decisions:

FIGURE 1-8



Any of the logical decisions which the inventory clerk must make in handling the data may also be made by the UFC-I through suitable programming. The UFC-I repertory of instructions contains several decision-making commands, such as comparisons and jump instructions, and the plugboard provides devices for decision-making, such as branches, selectors, and the code distributor.

Both the inventory clerk and the computer must be able to remember information:

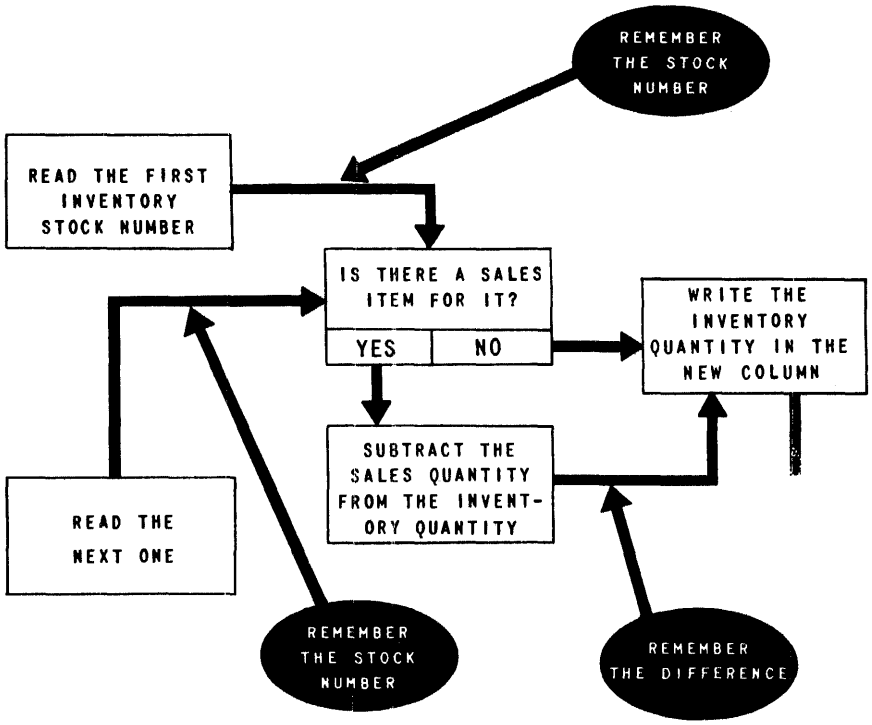


FIGURE 1-9

The clerk is required to remember the stock number, quantities, and results for only a short period of time; the stock ledger "remembers" the information permanently as it becomes one of the official records of the business.

The UFC-I also has means of remembering information for varying lengths of time. The various registers and magnetic core buffers in the system provide temporary storage for information and controlling data; the magnetic drums provide permanent storage for data as long as it is required in the data processing system; punch card files and magnetic tapes provide historical records for future reference.

Both the clerk and the computer must perform the necessary steps in the proper sequences:

In the manual system, the clerk is probably following a work procedure developed by a systems and procedures department and interpreted for the clerk by his supervisor.

The UFC-I follows a "program" or sequence of computer operations designed by systems analysts and programmers to accomplish this data processing task. All decisions and actions to be taken by the computer must be considered in minute detail by those persons developing the computer program to insure that provision is made for all predictable variations.

Two methods of controlling the computer program are used in the UFC-I system: an internally-stored series of instruction words, and a 48-step plugboard providing program steps, substeps, selectors, branches, etc. Typically, these two types of programming are used in combination to achieve the most efficient overall program.

Finally, all of the items are posted, the clerk puts the ledger away and goes about his other duties. But what of the UFC-I system? As indicated above, the inventory records might be unloaded from the magnetic drums for use as input data the following day. It is much more likely, however, that further processing of the data, such as the preparation of sales statistics or posting of accounts receivable information would be accomplished before the updated inventory is unloaded from the drums.

A further advance in integrated data processing might also be used in this instance, if the programmer wishes to take advantage of the large capacity general storage drums of the UFC-I system. The transaction of selling an item from stock results in a kind of chain reaction throughout the data processing system. Every area of the business record-keeping system which is provided for in a pre-selected area in the general storage drums and which is effected by the sale of an item receives a posting at the time the sales entry is processed.

For example: The sale of one item from stock results in the creation of an account receivable as well as the necessity to record a sale. Inventory is depleted by one unit, possibly requiring the initiation of a requisition to buy or a manufacturing order to produce a unit to replenish stock.

Outstanding Features of the Univac File-Computer, Model I System

The Univac File-Computer, Model I, is a medium sized, general purpose, digital, electronic computing system. It is one of the Remington Rand family of Univac computers, which also includes the Univac I, Univac II and Univac Scientific. The UFC-I system possesses unique features which contribute to

its data processing versatility. Among these, the most outstanding are described briefly below.

Flexibility of Input/Output Equipment

Any grouping of input/output devices up to ten in number may be connected to the central computer at the same time. These devices include the UFC-I Console, Inquiry Typewriter, 90-Column Punched Card System, 80-Column Punched Card System, High Speed Printer, High Speed Paper Tape System, Magnetic Tape Unit and special purpose equipments such as those used in the Airline Reservation System.

Large Capacity Random-Access Storage

The UFC-I may incorporate at the present time from one to ten large-capacity, magnetic drums in a system, allowing for random access to stored data. This ability of the computer to accept data in any order eliminates the need for prior sorting of input information. When the correct address of stored data is not known, it may be located through a channel search without interruption of other computer operations.

Time-Sharing Features

Through the demand stations associated with each input/output device, the central computer and several input/output devices may function independently, except during intervals when control information is being exchanged by the computer and the I/O device.

General storage operations, involving the large-capacity, general-storage drums, may be carried on simultaneously with central computer operations and with input/output operations. The "busy" or "not busy" condition of both the general storage system and the various input/output devices may be determined without interrupting any operation currently in progress.

During execution of internally-stored instructions the next instruction is located and readied for execution while the current instruction is in progress.

Internal and External Programming

The UFC-I operates as an *internally* programmed computer through a series of sequentially stored instruction words. It operates as a *plugboard*-programmed computer through the wiring of program steps on a 48-step main program plugboard which incorporates other plugboard-controlled devices, such as selectors, branches, etc.

Most programs developed for the UFC-I incorporate both internally-stored and plugboard-defined instructions in a single inter-related program to exploit the strongest features of each type of programming.

Checking Features

During all data transmission operations, a redundant parity check is conducted by the computer. The computer stops automatically at the point in the program where a parity error occurs.

All arithmetic operations of the computer may be checked for accuracy by a reverse operation; i.e., addition is checked by subtraction. In internally defined programs, checking is performed automatically unless suppressed by the programmer. In plugboard defined programs, a check or no-check decision is effected by the process wiring.

Alpha-Numeric Operation

The UFC-I handles any of the 63 Univac code characters with equal facility, regardless of whether the character is a number, letter, or special symbol. No special programming procedure is required to handle alphabetic information or special symbols.

Automatic Data Translation

Each input device in the UFC-I System automatically translates data from its own language (punched card, paper tape etc.) to the language (Univac code) of the UFC-I central computer. Each output device translates data from the language of the central computer to its own language. Therefore, no central computer time is lost in data translation.

Three-Address Logic

Three-address logic is the three-part principle of computer instruction which includes:

(a) The address (or storage location) from which the first operand is obtained.

(b) The address from which the second operand is obtained.

(c) A third address where the result is to be stored.

In UFC-I internal programming, each 12-character instruction word may contain a) the addresses of *two* operands, b) the address at which the result is to be stored, c) the basic process to be performed, and d) a sub-instruction which may modify or extend the basic operation. Up to 850 of these powerful instruction words may be stored in the high speed drum. The 48-step program control plugboard also operates basically in three-address logic.

The Computer Word

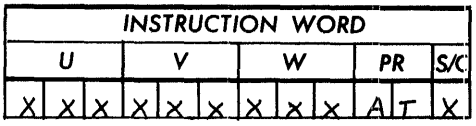
A computer word consists of twelve 7-bit Univac characters, of which the least significant character is usually the sign (+ or -) of the value contained in the word. Two kinds of computer words are utilized by UFC-I: the stored-data word and the internal instruction word.

A stored-data word is composed of any combination of 12 alphabetic or numeric characters stored in a word address within the computer. When this word address is called for in the program, the entire 12 characters will be processed by a single command.

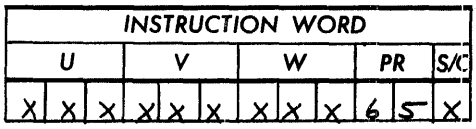
The arithmetic registers, extensively used as intermediate storage locations during computer processes, are the same length as the standard computer word, with the sign position occupying the least significant digit.

The internal instruction word differs from the stored-data word; it must always follow a specific format, as in the following diagram:

Process Instruction Word



Transfer Control Instruction Word



The U, V, and W portions of the instruction word usually represent storage addresses, or, more specifically, the "contents" of particular storage locations identified by these addresses.

In a process instruction word, the operation code is a process code which identifies exactly the process (or operation) which the computer is to perform in the execution of the instruction.

In a transfer of control to plugboard instruction word (transcop), the operation code is a number ranging from 51-98 which indicates the transfer of program control to a particular plugboard step for a sequence of one or more plugboard-defined program steps.

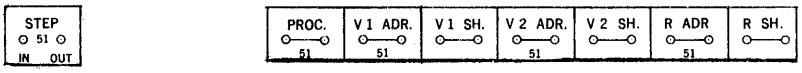
In either case, the "S/C" or special character defines a secondary operation which extends or modifies the basic operation specified by the instruction word.

An instruction word, whether a process or transcop instruction, must contain exactly 12 characters in order to be interpreted by the computer as a valid instruction. Therefore, some legal character is always included in each position of the instruction word. In many cases where some character, or characters, of the U, V, W or S/C portion of the instruction is to be completely ignored, a space code Δ is used.

The Program Step

The main program plugboard of the UFC-I contains hubs for external wiring of 48 program steps. The program step is comparable to the instruction word, as it directs the execution of similar processes to those programmed internally. Fixed wiring within the computer controls the execution of an instruction word; manually plugged wiring controls the execution of a program step on the program control plugboard. Each program step consists of a series of hubs as shown below.

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
51								



Program Step

The wiring of the V_1ADR hub to a storage location serves the same function as the placing of the address of a storage location in the U portion of an instruction word; V_2ADR is comparable to the V address portion; R ADR is comparable to the W address portion. Wiring of the process PROC hub defines the basic operation to be performed, and STEP OUT hub wiring defines the sub-step modification or extension of the basic operation. Whether or not a sub-step is defined by STEP OUT hub wiring, STEP OUT must be wired directly or indirectly to a STEP IN hub or a NEXT INSTRUCTION hub in order for the computer program to continue.

The diagram shown below indicates the comparable relationship of an instruction word and a program step.

U	V	W	PR	S/C
V_1ADR	V_2ADR	R ADR	Proc.	Step Out

Univac File-Computer Code

The internal language of the Univac File-Computer is the seven-bit position Univac representation of alphabetic and numeric characters and certain special symbols.

Within the computer, a character is represented by the presence or absence of a series of seven electrical pulses or magnetic spots. These individual pulses or spots are called "bits" (an abbreviation of "binary digits"). In the accompanying table, and in subsequent references to the Univac code, the presence of a bit is represented by an Arabic 1 and its absence by a 0.

A four-position, excess-3, binary coded decimal system is used to indicate numeric data. "Excess-3" indicates that, for internal computer reasons, a 3 has been added to each number in the binary coded decimal system. In table 1-1, the pulse patterns have already been adjusted to indicate the addition of the excess-3.

In order to represent alphabetic characters and certain special symbols, two additional positions (zone bits) are included as integral parts of the coded character. No special programming is necessary when alphabetic characters and special symbols are employed.

Since the computer is designed to operate only on an odd number of bits per character, the seventh bit position is used as an odd-even check position or parity check position. Thus any character whose excess-3 and zone bits add up to an even number must contain a 1 in the parity bit position. Conversely, any character whose bits already total an odd number must contain a zero in the parity bit position. An odd-even parity check is performed during data transmissions within the computer, as one of the means of insuring complete accuracy of computer operations.

TABLE 1-1

UNIVAC FILE-COMPUTER CODE

Character	Parity Bit	Zone Bits	Excess Three Bits
¡	1	00	0000
Λ	0	00	0001
-	0	00	0010
0	1	00	0011
1	0	00	0100
2	1	00	0101
3	1	00	0110
4	0	00	0111
5	0	00	1000
6	1	00	1001
7	1	00	1010
8	0	00	1011
9	1	00	1100
'	0	00	1101
&	0	00	1110
(	1	00	1111
r	0	01	0000
.	1	01	0001
.	1	01	0010
;	0	01	0011
A	1	01	0100
B	0	01	0101
C	0	01	0110
D	1	01	0111
E	1	01	1000
F	0	01	1001
G	0	01	1010
H	1	01	1011
I	0	01	1100
*	1	01	1101
¢	1	01	1110
@	0	01	1111

Character	Parity Bit	Zone Bits	Excess Three Bits
t	0	10	0000
"	1	10	0001
]	1	10	0010
)	0	10	0011
J	1	10	0100
K	0	10	0101
L	0	10	0110
M	1	10	0111
N	1	10	1000
O	0	10	1001
P	0	10	1010
Q	1	10	1011
R	0	10	1100
$	1	10	1101
~	1	10	1110
?	0	10	1111
∑	1	11	0000
Æ	0	11	0001
:	0	11	0010
+	1	11	0011
/	0	11	0100
S	1	11	0101
T	1	11	0110
U	0	11	0111
V	0	11	1000
W	1	11	1001
X	1	11	1010
Y	0	11	1011
Z	1	11	1100
¢	0	11	1101
=	0	11	1110
Not Used*	1	11	1111

* 1111111 is a delete code for some UFC-1 I/O units.

SYNOPSIS OF PROGRAMMERS' MANUAL

In this introductory Chapter 1, the outstanding features of the Univac File-Computer, Model I, have been described only briefly, as they will be dealt with in greater detail in the chapters that follow. The programmer is being alerted here to the flexibility, the versatility, and the capabilities of the UFC-I. He would be well advised to exploit them to their fullest extent in the programs he devises.

Chapter 2 , "Program Control Storage System" describes the addressable memory locations which comprise the central operating memory of the computer and the data transmissions which result when these locations are referred to.

Chapter 3 describes "Program Control", that section of the computer which directs the interpretation, execution, and sequence of the stored instructions. There are three types of program control that may be used for programming on the UFC-I: internal control, external control, and a combination of internal and external. Internal control executes internally stored instruction words; external control executes plugboard wired instructions. The methods by which the programmer uses internal or external control or a combination of both, are outlined in Chapter 3 , together with the distributors, switches, hubs, and indicators of chief concern to the programmer.

In Chapter 4, "Repertory of Instructions", detailed analysis is given of the 27 instructions which the UFC-I performs, together with the sub-instructions or substeps which extend or modify the basic operations.

Chapter 5 discusses the "General Storage System", which is the large capacity random access, storage system of the UFC-I. The components of the general storage system and the general storage operations which utilize these components, are explained.

Chapter 6, Input/Output System , describes the wide variety of input/output devices which may be incorporated into any UFC-I system, but places major emphasis on the demand stations and their associated control lines because these are the dual keys to the time-sharing features of the input/output system.

Chapter 7 , "Arithmetic Section", includes basic facts concerning the arithmetic and logical processes of the computer.

Chapter 8 , "Timing", describes the basic concepts and supplies approximate times for the operation of the UFC-I central computer.

Appendix: Appended to this manual is a group of supplements which will aid the programmer in understanding and applying the text material. It is suggested that the programmer acquaint himself at the outset with the glossary programming forms, and program control plugboard facsimile. He will also need to refer to them frequently during his study of this manual.

chapter 2

PROGRAM CONTROL STORAGE SYSTEM

INTRODUCTION

Components of the Program Control Storage System

The program control storage system is the central operating memory of the UFC-I system. It is of particular interest to the programmer because it contains all the addressable storage locations of the central computer's operating memory (See Table 2-1), and the necessary circuitry for locating these memory locations and executing data transmissions.

PROGRAM CONTROL STORAGE LOCATIONS WHICH ARE WORD, FIELD, AND BLOCKETTE ADDRESSABLE*

LOCATION	ABBREVIATION	PROGRAM CONTROL STORAGE ADDRESSES		
		Addresses of Word Locations (12 characters)	Addresses of Field Locations (Number of characters specified by Field Selection Pattern)	Addresses of Blockette Locations (120 characters)
Input/Output Tracks (High Speed Drum)	I/O Tracks	000-009 010-019 020-029 --- 090-099	00A-00V (not I & O) 01A-01V " " 02A-02V " " --- 09A-09V " "	00Z 01Z 02Z --- 09Z
Block Transfer Buffer	BTB	100-109	10A-10V (not I & O)	10Z
Factor Storage Tracks (High Speed Drum)	FS Tracks	110-119 120-129	11A-11V (not I & O) 12A-12V " "	11Z 12Z
Intermediate Storage Tracks (High Speed Drum)	IS Tracks	130-139 140-149 150-159 --- 970-979	13A-13V (not I & O) 14A-14V " " 15A-15V " " --- 97A-97V " "	13Z 14Z 15Z --- 97Z
General Storage Buffer	GSB	980-989	98A-98V (not I & O)	98Z

*Each of these locations has a 120-character capacity and can store 10 computer words or up to twenty fields.

PROGRAM CONTROL STORAGE LOCATIONS WHICH HAVE ONLY ONE ADDRESS

LOCATION	ABBREVIATION	ADDRESS	CAPACITY
Register A	RA	990	12 characters
Register B	RB	991	
Register C	RC	992	
Register D	RD	993	
Code Distributor Register	CDR	994	1 character
General Storage Address Register	GSAR	995	7 digits
Instruction Revolver (High Speed Drum)	IRV	996	12 characters
Program Address Counter	PAK	997	3 digits
Shift Revolver (High Speed Drum)	SRV	998	12 characters
High Speed Drum Pattern	ISP	99W	120 characters (only parity bit of each character is used)
Block Transfer Buffer Pattern	BTP	99X	120 bits (120 characters are sent to these locations, but only the parity bits of the characters transmitted are stored).
General Storage Buffer Pattern	GSP	99Y	

TABLE 2-1

Functions of the Progr ontrol Storage System

The program control storage system has two main functions:

When the computer program specifies memory locations from which information is to be acquired and other locations at which the results of operations are to be stored, the program control storage system carries out these data transmissions. This function is similarly performed whether the computer program is defined by internally stored instruction words or by plugboard wired program steps.

When an internally stored program is being run, the *next* instruction to be executed is located during the execution of the *current* instruction. This acquisition of successive instruction words is the second important function of the program control storage system.

Source and Destination References

The programmer will later find statements in this manual to the effect that certain storage locations are addressable either as "sources", "destinations" or both. Understanding of the proper use of these storage locations is necessary in order to avoid common programming errors.

In source references, data is obtained from a storage location specified by the programmer and automatically placed in a temporary memory location by the computer. In the procurement of operands (which are defined as "the contents of a computer location, used in arithmetic and logical operations"), the address of the source is usually derived from the U or V section of an instruction word, or the locations wired to V_1ADR or V_2ADR on the plugboard. The process portion of the instruction determines the location (usually an arithmetic register) at which the data will be held while it is being added, subtracted, or manipulated in some other manner.

In destination references, the process portion of the instruction determines from which temporary memory location (usually arithmetic register C or D) the result is to be obtained. The programmer determines, by the coding of the W portion of the instruction word, or result address R ADR wiring, at which location the result is to be stored.

Example:

INTERNAL:

INSTRUCTION WORD											
U			V			W			PR		S/C
/	2	0	/	/	3	/	2	5	A	D	A

PLUGBOARD:

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
51	FS#2-0		+ C	FS#/-3		FS#2-5		

The above example shows the coding for similar instruction words and program steps. The source of the first operand is specified by the U address (120) of the instruction word or the V_1ADR address (FS#2-0) of the program step. These two are actually the same word location, expressed in terms of internal and external programming. The add process determines that arithmetic register A will be the temporary storage location of the first operand.

The source of the second operand is specified by the V address (113) of the instruction word or the V_2ADR address (FS#1-3) of the program step. The add process determines that arithmetic register B will be the temporary storage location of the second operand.

During the process of addition (and checking), the add process determines that the result of the process, the sum, will be accumulated (automatically) in arithmetic register D. From this temporary storage location the result will be transferred to the destination address, which is specified by the W address (125) of the instruction word or the R ADR address (FS#2-5) of the program step.

All instructions do not require three storage references; some require two and a few, only one.

PROGRAM CONTROL STORAGE ADDRESS FORMAT

A three-character code is used to address locations in program control storage when they are to be used in an instruction word. In this format, the *two* characters at the left (higher order) are always numeric, and the *one* character at the right (low order) may be either numeric or alphabetic.

To address any of these storage locations through plugboard wiring, each addressable location is identified by a label, and three hubs are provided for each location. In this manual, whenever internal and external addresses refer to the same computer location, the proper addressing technique for both internal and external programming is shown in the appropriate segments of programming charts.

Word Addresses

A computer word is composed of twelve characters, of which the low-order character is usually the sign. The 120 characters on a track of the high speed drum are divided into ten words which are addressed 0 through 9. As the drum revolves in a counterclockwise direction, word 9 refers to the first (low order) group of twelve characters on a track, word 8 to the second group of characters, and word 0 to the tenth (highest order) group of twelve characters.

Word addresses are also applicable in the block transfer buffer and the general storage buffer.

The internal address of a word location always includes the identification of the track or buffer in the two higher order characters, and the identification of the word in the low order character.

External addresses of word locations are identified by groups of plugboard hubs labelled I/O WORD 0-9, FS#1 WORD 0-9, FS#2 WORD 0-9, GSB WORD 0-9, and BTB WORD 0-9.

Field Addresses

Because the length of names, addresses, part numbers, quantities, dollars and cents, seldom falls exactly into twelve-character words, the UFC-I provides a method of setting up data fields of varying lengths to facilitate the programmer's handling of this data.

As many as 20 fields may be defined in a track on the high speed drum, in the block transfer buffer, or in the general storage buffer.

The internal address of a field always includes the identification of the track or buffer in the two higher order characters, and the identification of the field in the low order character. A field is defined by any letter A-V, excluding I and O.

The external addressing of fields is accomplished by wiring of the plugboard hubs labelled I/O FIELD A-V, FS#1 FIELD A-V, FS#2 FIELD A-V, BTB FIELD A-V and GSB FIELD A-V.

Blockette Addresses

A blockette of information in the UFC-I is the group of 120-characters which may be contained in 1) a track of the high speed drum, 2) in the block transfer buffer, or 3) in the general storage buffer. Any of the above locations are blockette-addressable by the letter Z in the low order position in the source or destination reference of the instruction word or program step. The same locations are addressable on the plugboard by the wiring of the hubs labelled I/O-Z, FS#1-Z, FS#2-Z, GSB-Z and BTB-Z.

Single Address Locations

The program control storage system includes twelve storage locations, each of which has only one address. These locations are: arithmetic registers

(A,B,C,D), code distributor register, general storage address register, instruction revolver, program address counter, shift revolver, and the three field selection patterns.

The internal address of each of these locations contains the numbers 99 in the high order positions whereas a number in the low order position designates one of the registers or revolvers, and a letter in the low order position designates a field selection pattern.

For external addressing purposes, a set of three hubs is available on the main program plugboard for each of these locations.

COMPONENTS OF THE PROGRAM CONTROL STORAGE SYSTEM

High Speed Drum

Description

The high speed drum is physically located within program control cabinet #1 of the UFC-I system. It revolves at a speed of 12,000 rpm, or one revolution per five milliseconds. Since it could require either 0 milliseconds minimum, or 5 milliseconds maximum, to locate a particular piece of data, the "average access time" to a location on the high speed drum is 2.5 milliseconds locating time plus the time required to read or write the number of characters desired.

Data is stored on the drum surface sequentially (bit by bit) as magnetized spots. A number of dual purpose read/write heads are located in the housing of the drum. Each communicates with a narrow band on the drum surface as the drum revolves. The narrow band associated with each read/write head is called a track and contains 840 bit positions, thus allowing 120 characters of the 7 bit per character Univac code to be stored per track. From one to 120 characters can be recorded or read from a track, depending on the address used in the reference. Recorded data is not affected by the removal of computer power, nor by the reading of information from the track.

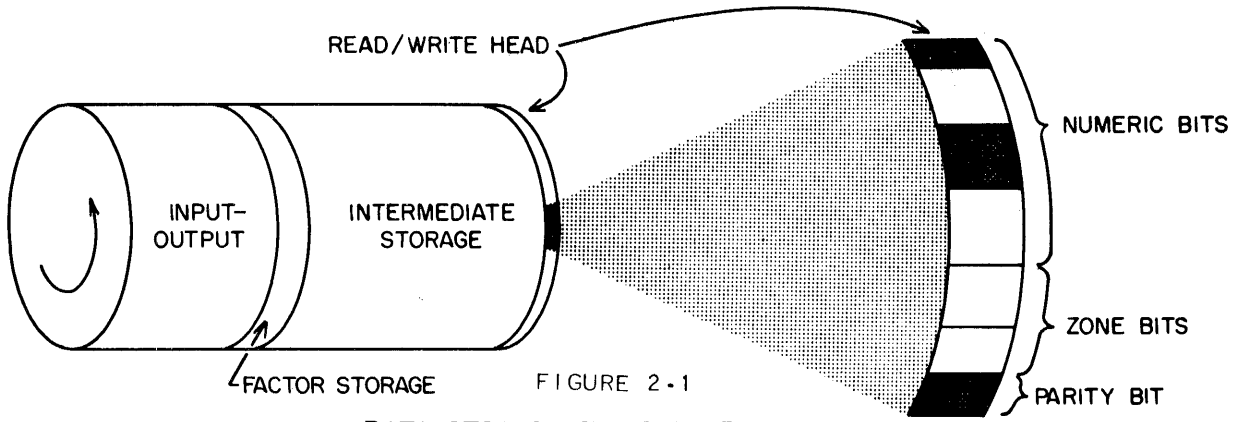


FIGURE 2-1

DATA STORAGE ON HIGH SPEED DRUM

The assignment of tracks on the high speed drum is as follows: twenty are designated as input/output tracks; two are designated as factor storage tracks; eighty-five are designated as intermediate storage tracks, and are used for the storage of constants, intermediate results, and instruction words; one track is designated as the high speed drum field selection pattern track. Also, the instruction revolver and the shift revolver to be studied later are physically located on the high speed drum.

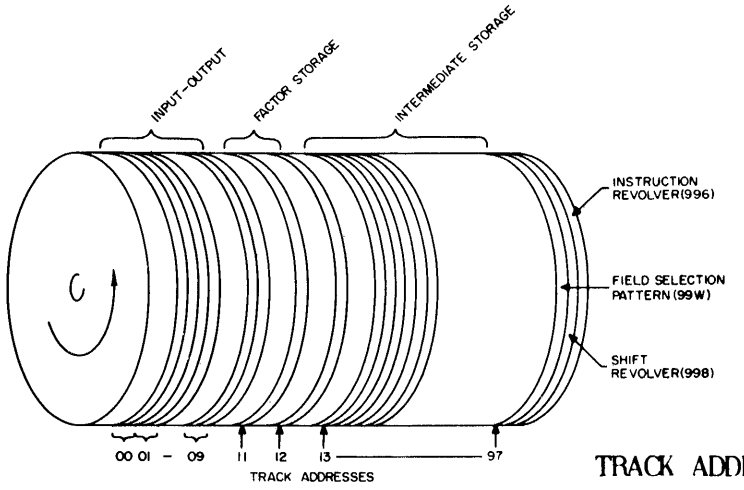


FIGURE 2-2

TRACK ADDRESSES ON HIGH SPEED DRUM

Addressing the High Speed Drum

General Addressing Structure

A three-character address is used to locate specific positions on the drum. The two high order characters of the address specify the track, and are always numeric. The low order character of the address determines that either a section or an entire track is to be used. If only a section is to be ad-

dressed, a word or a field will be specified. A word in this position is indicated by a number; a field is indicated by an alphabetic character except for the character Z which indicates the entire 120 characters in the specified track.

Input/Output Tracks

The high speed drum includes 20 input/output tracks arranged in pairs. Each pair is addressed as if it were a single track, with numbers 00 through 09 representing the first through the tenth pair of tracks.

This addressing method is possible because the computer is in direct communication with only one track of each pair at a time, while the input/output device associated with that pair of tracks is in communication with the other. Switching commands enable the computer and input/output device to exchange positions in relation to the I/O track. For example, (see Figure 2-3), the computer is in contact with the left hand track and the I/O device is in contact with the right hand track. After the switching command is executed, the I/O device is in contact with the left hand track, and the computer is in contact with the right hand track.

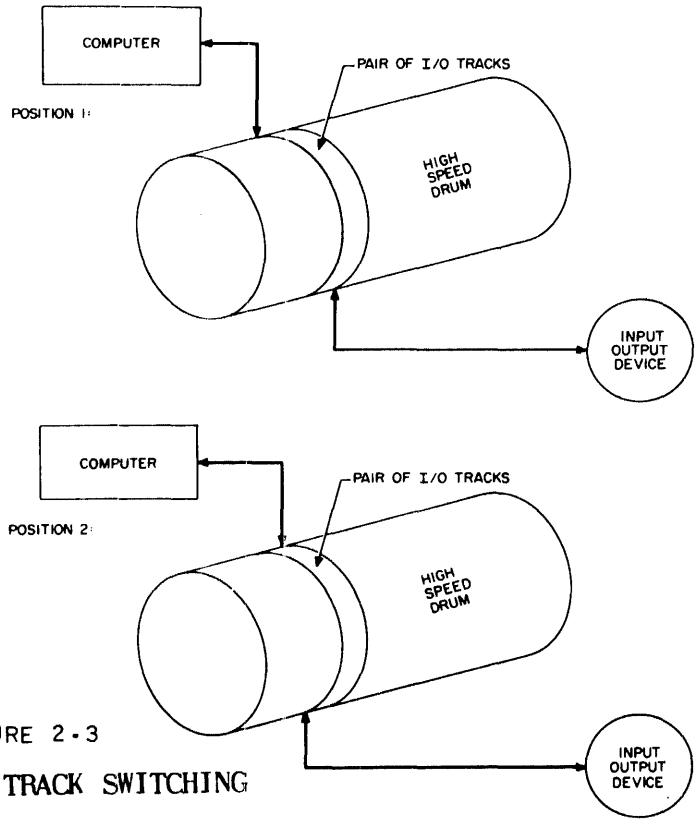


FIGURE 2-3

INPUT/OUTPUT TRACK SWITCHING

Since the plugboard provides only one set of hubs for the ten word addresses and one set of hubs for the 20 possible fields in any input/output track, the input/output track of only that unit currently "on demand" is addressable from the plugboard. When a particular input/output unit is placed "on demand", the computer is in direct contact with that unit, and with the input/output track directly associated with it. (The "demand" concept is discussed in Chapter VI.)

Internally, input/output tracks are directly available to program control storage when addressed by the U, V, or W address of an instruction word. Any I/O track may be addressed internally at any time.

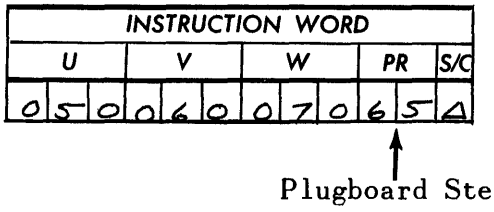
Externally, input/output tracks are available in either of two situations:

When an input/output unit has been assigned to the I/O tracks referred to, and that I/O unit has been placed "on demand";

When a transcop (transfer control to plugboard) instruction word or any instruction word containing a breakpoint initiates a plugboard sequence of program steps and two conditions are met: a) the U, V, or W section of the instruction word refers to an input/output track, and b) the V_1ADR, V_2ADR, or R ADR hubs of program steps are wired to the appropriate U ADR, V ADR, or W ADR hubs.

For example: assume that during a plugboard sequence, the programmer wishes to address word 0 in I/O tracks 05, 06 and 07, without placing each of these I/O units on demand. The transcop instruction word which transfers control to the plugboard (step #65 in this example) may be coded as follows:

TRANSCOP INSTRUCTION WORD:



Plugboard Step #65

During the execution of the series of program steps beginning with step #65, the plugboard wiring of a V_1ADR hub to a U ADR hub will supply the contents of storage location 050; similar wiring to V ADR or W ADR hubs would provide access to storage locations 060 and 070 respectively.

Note that this wiring need not follow any specific pattern. V_1ADR may be wired to U ADR, V ADR or W ADR at the option of the programmer. V_2ADR and R ADR may also be wired to any one of the hubs U ADR, V ADR or W ADR at the option of the programmer. (See also the example under "Intermediate Storage Tracks" below.)

Factor Storage Tracks

Tracks addressed internally as 11 and 12 are wired on the plugboard as factor storage #1 (FS#1) and factor storage #2 (FS#2) respectively.

An important use of the factor storage tracks is for the storage of constants which may be addressed by either instruction words or program steps, since these two tracks are always directly available to program control storage.

Intermediate Storage Tracks

Tracks numbered 13 through 97 may be used for storage of intermediate results, constants, and instruction words. In most instances, a portion of this general area is reserved for the internally stored program. Up to 850 instruction words may be stored in this area.

Intermediate storage "IS" tracks are available to program control storage directly by addressing them in the U, V, or W address of an instruction word.

These IS tracks are available to program control storage on the plugboard only through the wiring of the V_1 address (V_1ADR), V_2 address (V_2ADR) or R address (R ADR) hubs to the U address (U ADR), V address (V ADR), or W address (W ADR) hubs in any combination.

The tracks referred to in each of these cases are determined by the IS track address included in the U, V, or W address of the transfer of control (transcop) instruction word which initiates that sequence of program steps.

Following is an example of internal and plugboard addressing of high speed drum tracks:

INTERNAL:

INSTRUCTION WORD											
U			V			W			PR		S/C
0	5	0	7	9	1	1	1	4	A	D	A

To address the same storage locations on the plugboard, two conditions must be met:

(a) Input/output unit 05 must be "on demand".

(b) The following transcop instruction word must precede step 73.

INTERNAL:

INSTRUCTION WORD										
U			V			W			PR	S/C
4	4	4	7	9	1	4	4	4	7	3	4

PLUGBOARD:

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
73	I/O-0		+ C	V ADR		FS #1-4		74

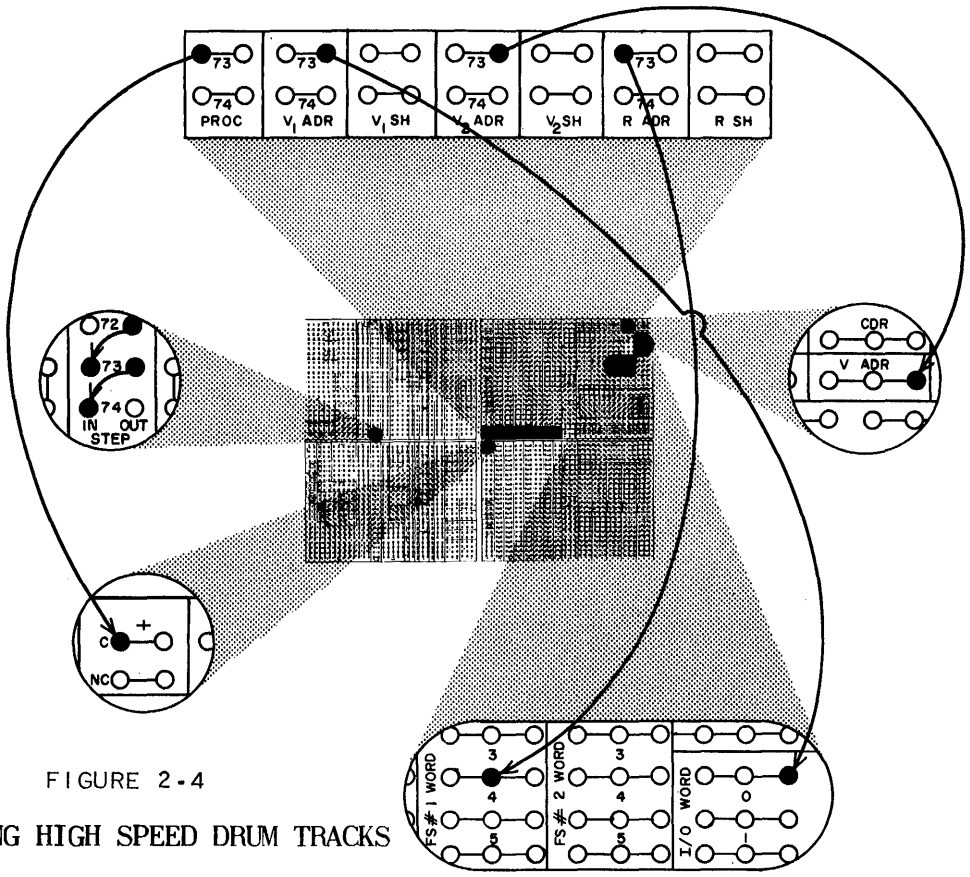


FIGURE 2-4

ADDRESSING HIGH SPEED DRUM TRACKS

The coding of the above program step indicates that the program comes into step #73 from step #72, although our example states that the program comes into step #73 from a transcop instruction word.

This illustrates the ability of the transcop instruction to tranfer control to any plugboard step the programmer desires, regardless of the plugboard program which may also utilize the same step.

High Speed Drum Field Selection Pattern Track

The high speed drum field selection pattern track is addressable by the symbol 99W as a destination address only. This track would normally be addressed for the purpose of transferring a predetermined field selection pattern to this track.

Construction of a pattern for the field selection pattern track, referred to as "high speed drum intermediate storage field selection pattern" (ISP), is accomplished by loading a series of characters without parity bits to indicate the body of a field, and any character which contains a parity bit to indicate the ending character of a field.

For example: referring to the Univac Code Chart (Table 1-1) the programmer will note that the character 4 (0000111), L (0100110), and Δ (0000001) do not contain parity bits and therefore could be used to fill out the body of the field. The characters 2 (1000101), A (1010100), and + (1110011) contain parity bits and could be used to denote the end of a field.

When a field pattern is transferred to the high speed drum field selection pattern track ISP, the entire character is stored on the ISP track, but only the parity bits are used for field selection. Therefore, in building a field pattern only the parity bits of the characters need be considered. Since only twenty fields can be addressed on any track (A through V, excluding I and O), a field pattern should not define more than twenty fields.

A number of these field patterns may be stored on the high speed drum or on the general storage drums, and may be transferred to the ISP when needed. A pattern on the ISP controls the field addressing of all other tracks on the high speed drum. However, word addressing is still applicable even though a field pattern is present.

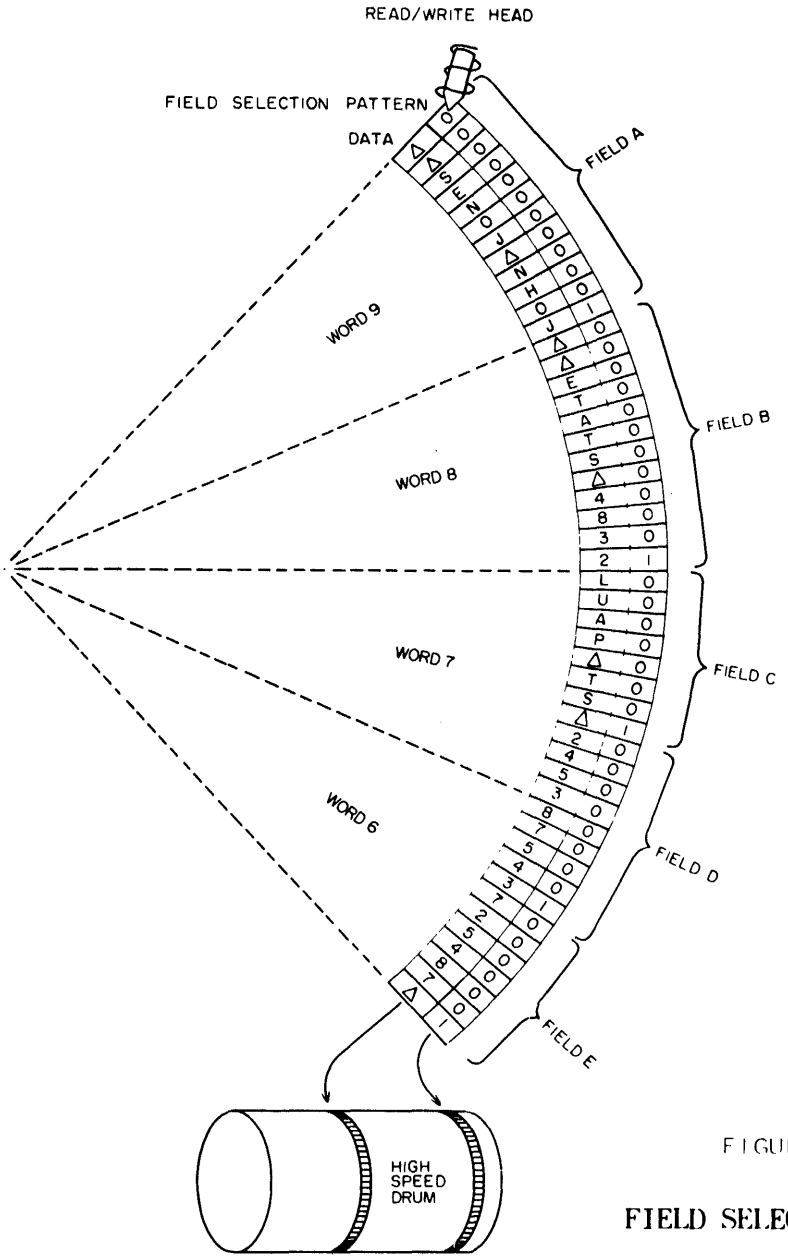


FIGURE 2-5

FIELD SELECTION PATTERN

Field Addresses on the High Speed Drum

In an instruction word, to address fields on any input/output, factor storage, or intermediate storage track, the track desired is signified by the two high-order (numeric) characters of the address, while the field is signified by the low-order character, which must be a letter from A through V, excluding I and O.

To address these fields in a program step, I/O FIELD A-V, FS#1 FIELD A-V an FS#2 FIELD A-V hubs are provided on the main program plugboard.

Example:

(1) 05A identifies the first field of track number 05.

(2) 47V identifies the twentieth field of track number 47.

INTERNAL:

INSTRUCTION WORD											
U			V			W			PR		S/C
0	5	A	4	7	V	1	2	M	A	D	A

PLUGBOARD:

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
51								52
52	I/O A		+C	V ADR		FS#2-M		53

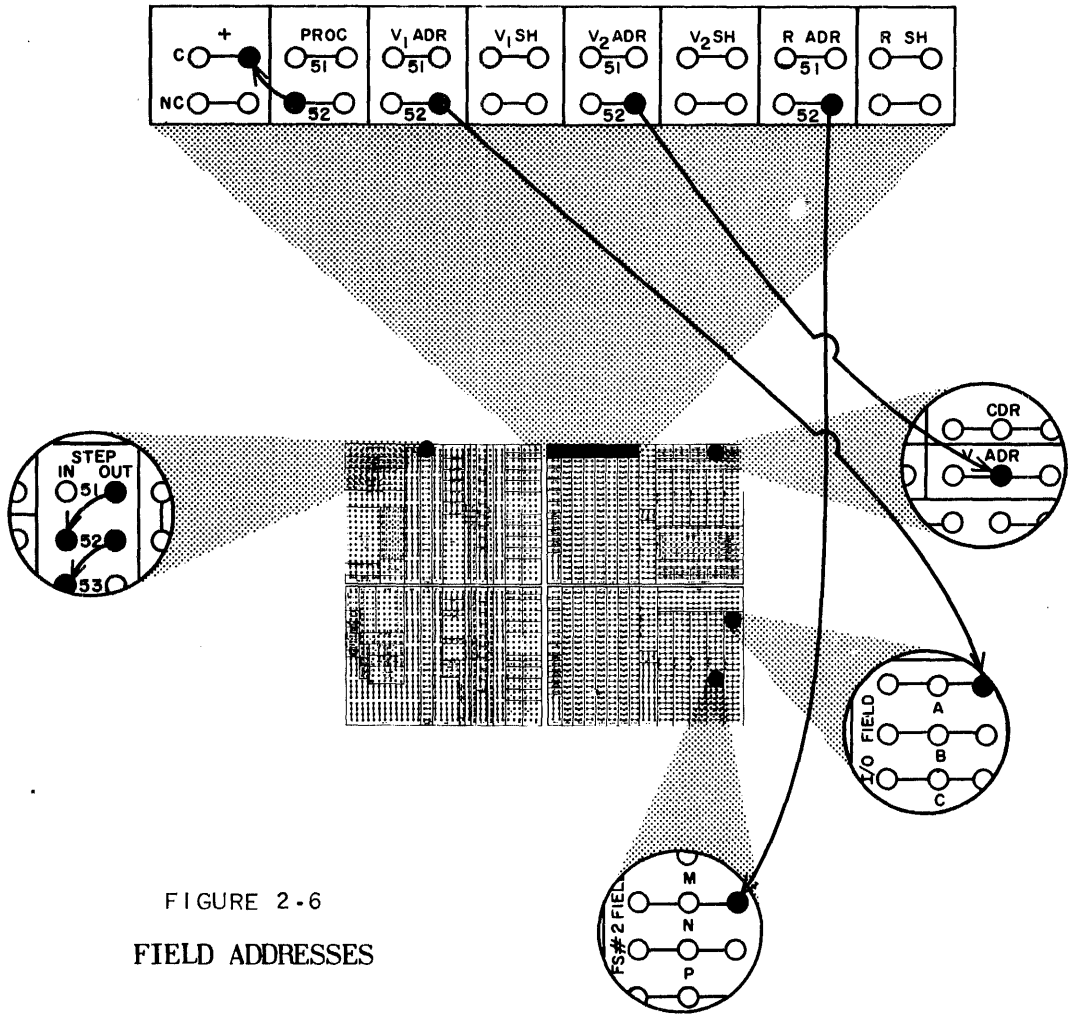


FIGURE 2-6

FIELD ADDRESSES

The above example has been designed to include a field in an input/output track, a field in an intermediate storage track, and a field in a factor storage track in order to emphasize the point that any field (which is a portion of the data on a track) is addressable under the same conditions as the track which contains the field. Therefore, in this example:

For internal programming, all of the above fields are directly addressable at any time.

For external programming: 1) I/O unit 5 must be on demand in order for field A of track 5 to be available; 2) a transcop instruction word (or some other method of loading the instruction revolver IRV_c) must make the field address 47V available to the V ADR hub; 3) field M of a factor storage track is available at all times.

Word Addresses on the High Speed Drum

To address word locations on any input/output, factor storage, or intermediate storage track internally, the track desired is signified by the two high order characters of the address, whereas the word is signified by the low order character.

To address these words in a program step, I/O WORD 0-9, FS#1 WORD 0-9, and FS#2 WORD 0-9 hubs are available on the main program plugboard.

032 indicates track number 03, word number 2.
119 indicates track number 11, word number 9.
870 indicates track number 87, word number 0.

INTERNAL:

INSTRUCTION WORD											
U			V			W			PR		S/C
0	3	2	1	1	9	8	7	0	A	D	A

PLUGBOARD:

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
76	I/02		+ C	FS #1-9		W. ADR		77

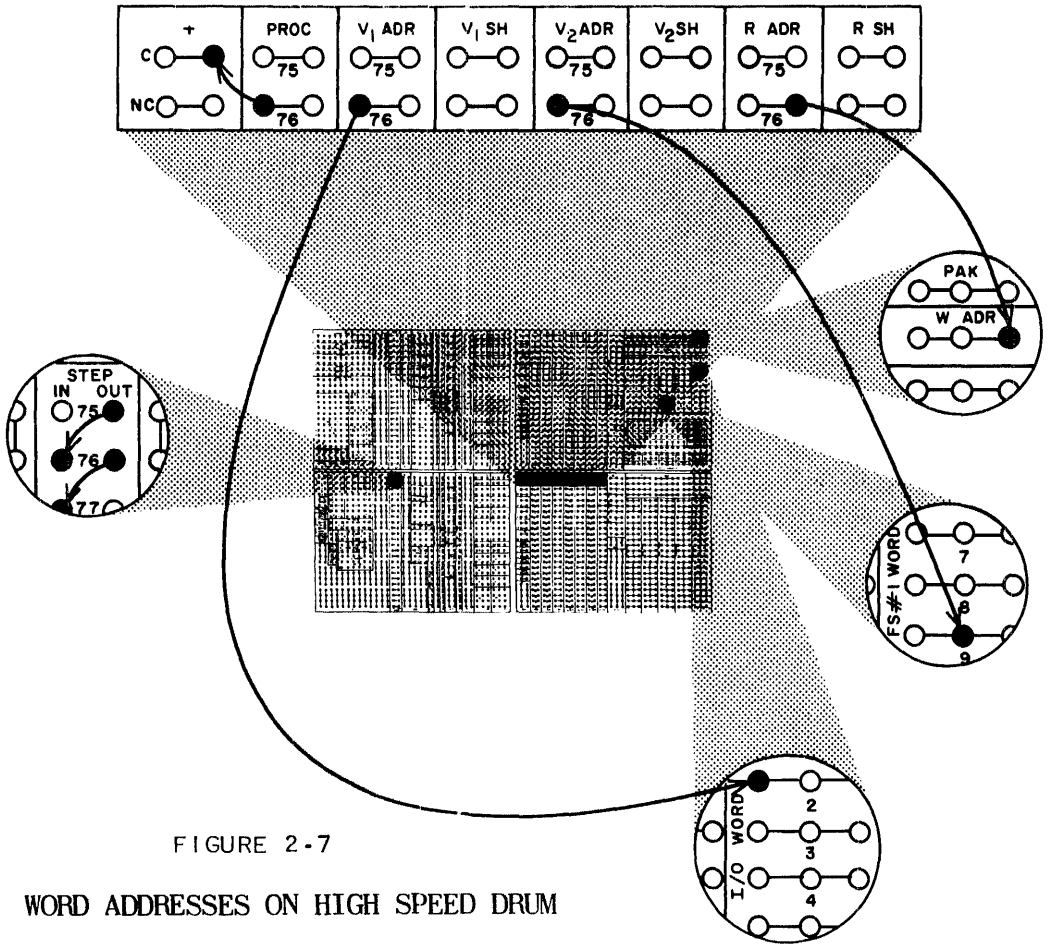


FIGURE 2-7

WORD ADDRESSES ON HIGH SPEED DRUM

The above example has been purposely designed to include a word location in an input/output track, in a factor storage track, and in an intermediate storage track, in order to emphasize the point that any word location is addressable under the same conditions as the track which contains the word. Therefore, in the above example:

For internal programming, all of the above word locations are directly addressable at any time.

For external programming: 1) I/O unit 3 must be on demand in order for word 2 of track 3 to be available; 2) word 9 of factor storage #1 is available at all times; 3) word location 870 is only available through the W ADR hub when a transcop instruction (or some other method of loading the instruction revolver IRV_c) precedes the program step in which 870 is desired.

Blockette Addressing of the High Speed Drum

Input/output tracks, factor storage tracks, and intermediate storage tracks are all blockette addressable. Internally, the entire 120 characters which may be contained in one of these tracks is directly addressable at all times by the use of the letter Z in the low order position of the address. Externally, the plugboard hubs I/O-Z, FS#1-Z and FS#2-Z serve the same purpose. Blockette addressability of these tracks is exactly the same concept as expressed in the discussion of high speed drum tracks earlier in this chapter.

An example of the usefulness of blockette addressability is given below, where an instruction to buffer transfer 01Z to 12Z results in the entire contents of channel 01 being moved to channel 12.

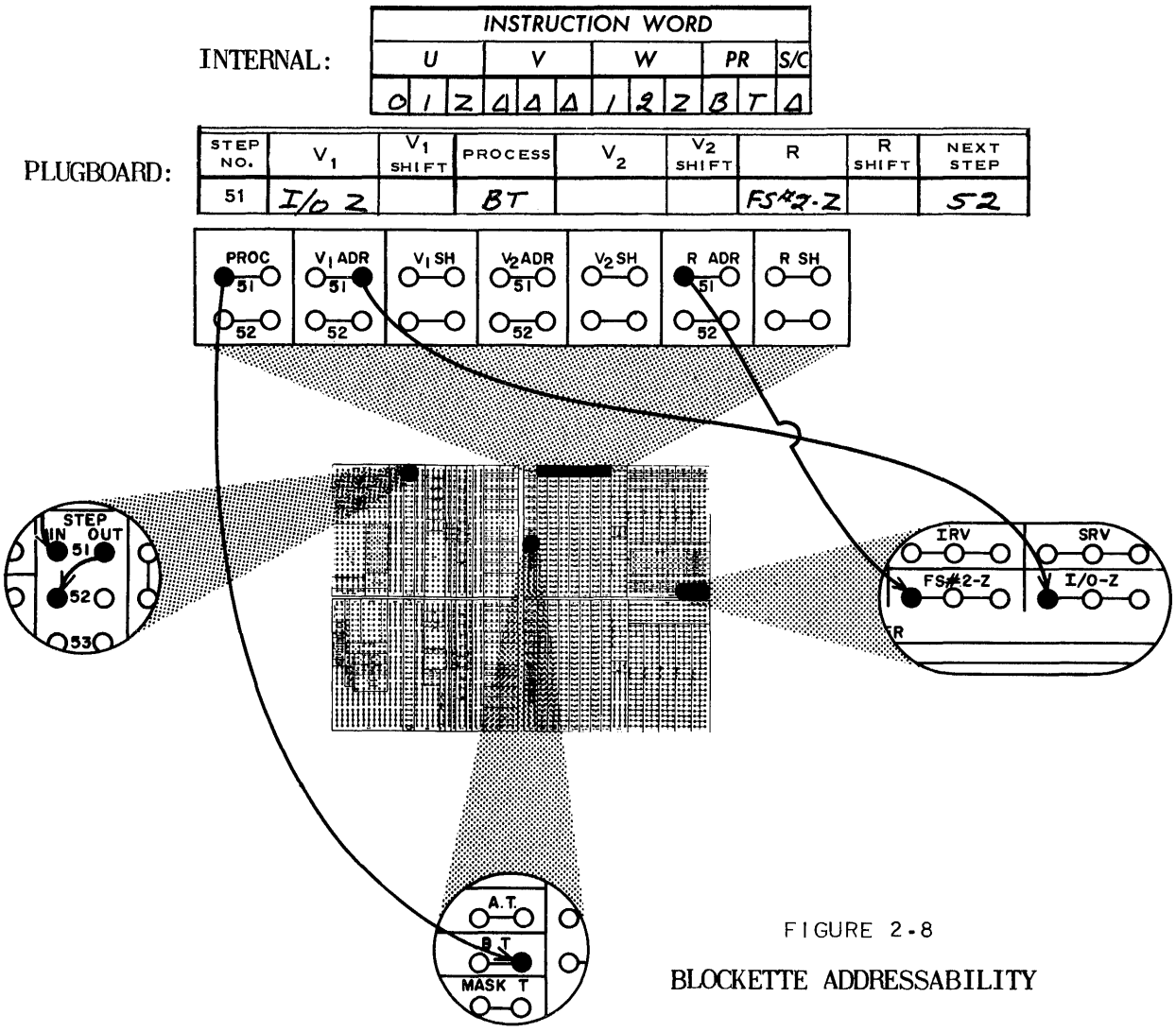


FIGURE 2-8

BLOCKETTE ADDRESSABILITY

*V_1ADR-I/O-Z wiring assumes that I/O unit 1 is on demand.

Block Transfer Buffer

Description

The block transfer buffer (BTB) is a rapid access magnetic core memory which stores up to 120 seven bit Univac characters. In its function as an addressable memory location available to program control storage, the BTB is word, field, and blockette addressable. Addresses in the block transfer buffer are directly available at all times to program control storage through the U, V or W portion of instruction words or through wiring of the BTB WORD 0-9, BTB FIELD A-V and BTB-Z hubs on the plugboard.

As its name implies, BTB is used as the intermediate storage location during the "buffer transfer" operation, which transfers blocks of data from one program control storage location to another. Up to 120 characters may be transferred by a single buffer transfer command.

Addressing Structure

(1) The address of the BTB is 10, supplemented by a letter or number designating the field or word, or by the letter Z which designates the entire 120 characters of the buffer.

(2) Word addresses follow the same pattern as words on the high speed drum. Each number from 0 through 9 represents a twelve-character word in the buffer.

Examples of word addresses in BTB:

100 indicates BTB, word 0
108 indicates BTB, word 8

(3) Field addresses are defined in the same manner as fields on the high speed drum. Each letter A through V, excluding I and O, indicates one of the twenty fields which may be set up in the buffer by the block transfer buffer field selection pattern (BTP).

Block Transfer Buffer Field Selection Pattern

The address of the block transfer buffer field selection pattern is 99X or the BTP hubs on the plugboard. The pattern may be thought of as an additional row of 120 bits, associated with the block transfer buffer. The method

of constructing BTP is similar to that used for the high speed drum field selection pattern. Characters without a parity bit indicate the body of a field; any character which contains a parity bit indicates the end of a field. Since only twenty fields can be addressed, a pattern should not define more than that number.

The BTP is addressable as a destination *only*, for receiving a complete 120-character field pattern.

An example of loading a pre-determined field pattern into the block transfer field selection pattern 99X follows:

INTERNAL:

INSTRUCTION WORD											
U			V			W			PR		S/C
0	5	Z	4	4	4	9	9	X	B	T	4

PLUGBOARD:

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
51								52
52	I/O-Z		BT			BTP		53

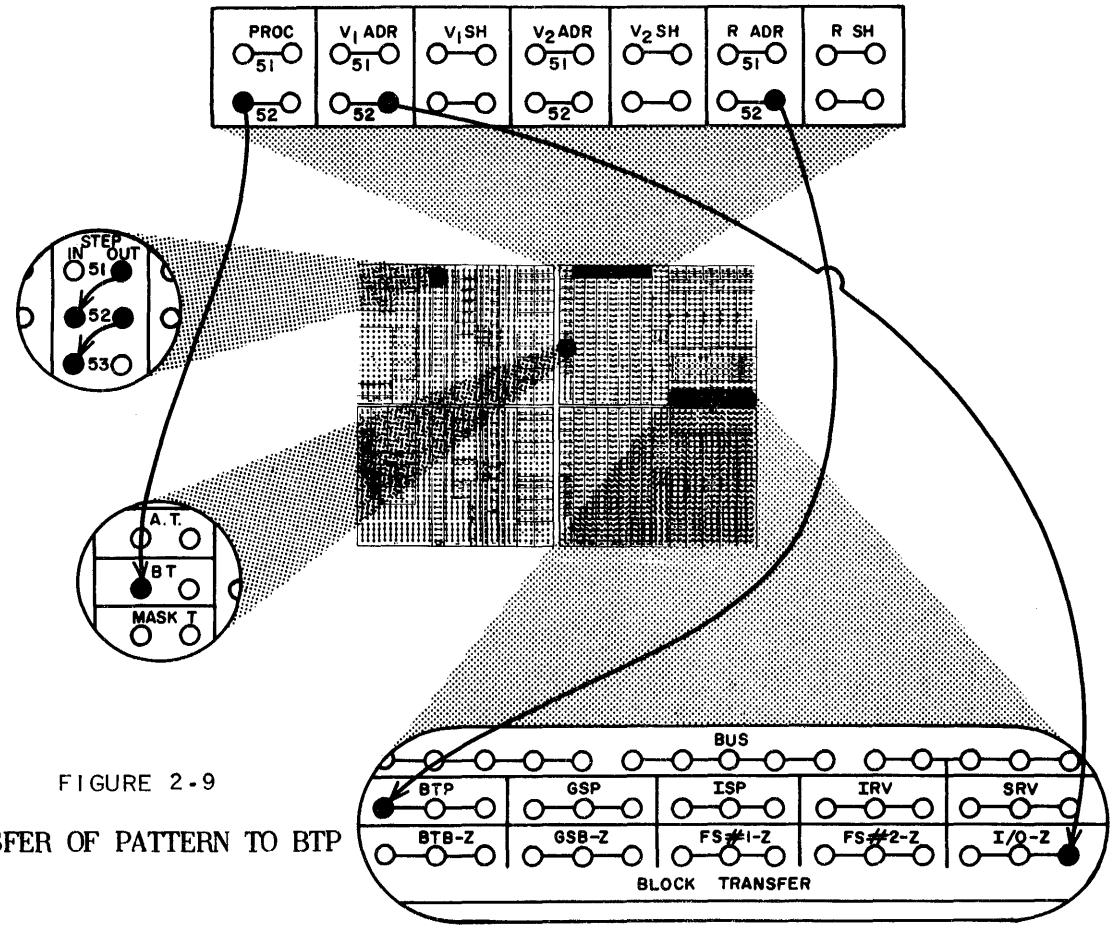


FIGURE 2-9

TRANSFER OF PATTERN TO BTP

General Storage Buffer (GSB)

Description

The general storage buffer is a rapid access magnetic core memory which stores up to 120 seven bit Univac characters. The function of the general storage buffer as an intermediate storage location between the program control storage system and the general storage drums is discussed in Chapter 4, "General Storage System".

In its function as an addressable memory location available to program control storage, the GSB is addressable by word, field or blockette.

Addresses in the general storage buffer are directly available to program control storage (except during the times the general storage buffer is engaged in certain general storage operations) through the U, V, and W address of instruction words, or through wiring of the GSB WORD 0-9, GSB FIELD A-V and GSB-Z hubs on the plugboard.

Addressing Structure

The address of GSB is 98, supplemented by a letter or number designating the field or word, or by the letter Z which designates the entire 120 characters of the buffer.

Word addresses follow the same pattern as words on the high speed drum. Each number from 0 through 9 represents a twelve character word in the buffer.

Examples of word addresses in GSB

980 indicates GSB, word 0
985 indicates GSB, word 5

Field addresses are defined in the same manner as fields on the high speed drum. Each letter A through V, excluding I and O, indicates one of the twenty fields which may be set up in the buffer by the general storage buffer field selection pattern GSP.

See Figure V-6 for a representation of the addressing structure of the general storage buffer.

General Storage Buffer Field Selection Pattern (GSP)

The general storage buffer field selection pattern may be addressed internally by the symbol 99Y, or externally by wiring the GSP hubs on the plugboard. The pattern may be thought of as an additional row of 120 bits associated with the general storage buffer.

The method of constructing GSP is similar to that used for the high speed drum field selection pattern. Characters without a parity bit indicate the body of a field; a character which contains a parity bit indicates the end of a field. A pattern should not define more than twenty fields within the buffer, since only twenty fields can be addressed (A through V, excluding I and O). The pattern may be addressed only as a destination and 120 characters must be transferred to it.

General Storage Address Register (GSAR)

The general storage address register (GSAR) is a special purpose memory location which has as its prime function the holding of the general storage drum address in operations involving the finding of storage locations in the general storage drum system (See Chapter 5).

In its role as a component of the program control storage system, GSAR is available to program control storage through code 995 in the U, V, or W address of an instruction word or through wiring of the GSAR hub on the plugboard.

When GSAR is addressed in a source reference, a space code (Λ) is automatically generated by the computer. The space code, followed by the seven digits held by GSAR is transferred to the temporary storage location specified by the process. (A space code in the sign position is treated by the computer as a plus (+) in arithmetic operations.)

When GSAR is addressed in a destination reference, the sign and seven low order digits are transferred from the temporary storage location specified by the process and loaded into GSAR, the sign being automatically shifted off.

Program Address Counter (PAK)

The program address counter (PAK) is a special purpose memory location which holds the address of the next instruction word to be located during execution of instruction words.

In its role as a component of the program control storage system, PAK is available to program control storage by two different methods of addressing: 1) code 997 may be entered in the U, V, or W address of an instruction word, or 2) the PAK hub may be wired on the plugboard.

When PAK is addressed in a source reference, a space code is automatically generated by the computer. The space code, followed by the three digits held by the PAK, is transferred to the temporary storage location specified by the process. (A space code in the sign position is treated by the computer as a plus (+) in arithmetic operations.)

When PAK is addressed in a destination reference, the sign and three low order digits are transferred from the temporary storage location specified by the process and loaded into PAK, the sign being automatically shifted off.

Code Distributor Register (CDR)

The code distributor register (CDR) is a single character register normally used to store control characters which will effect the distribution of information by the code distributor. It is discussed in its primary role of determining program variance in Chapter 3.

In its role as a component of the program control storage system, CDR is available to program control storage by two different methods of addressing: 1) code 994 may be entered in the U, V, or W address of an instruction word, 2) the CDR hub may be wired on the plugboard.

When CDR is addressed in a source reference, a space code (Δ) is automatically generated by the computer. The space code, followed by the single character held in the CDR, is transferred to the temporary storage location specified by the process. (A space code in the sign position is treated by the computer as a plus (+) in arithmetic operations.)

When CDR is addressed as a destination, the sign and a single character are transferred from the temporary storage location specified by the process and loaded into CDR, the sign being automatically shifted off.

Arithmetic Registers A, B, C, D

Arithmetic registers A, B, C, D are used during most of the processes which the UFC-I executes. They may be used as intermediate storage locations for operands and results as specified by the particular process involved, or they

may be programmed sources or destinations at the choice of the programmer.

As units of the program control storage system, the arithmetic registers are addressable as follows:

(1) When called for in the U, V, or W portions of an instruction word, the address of Register A (RA) is 990; Register B (RB) is 991; Register C (RC) is 992; and Register D (RD) is 993.

(2) Plugboard addressing of these registers is accomplished by wiring the hubs marked RA, RB, RC, and RD.

(3) Any of the four arithmetic registers may be referred to as either a source or a destination. In either case, exactly twelve characters is involved, usually a sign in the low order position, called the sign position, and eleven alpha-numeric characters in the remaining positions.

Instruction Revolver (IRV)

The instruction revolver is addressed internally by the code 996, or by wiring of the plugboard hub IRV. Although referred to in the singular, the instruction revolver is actually two 12 character revolvers.

For purposes of clarity the instruction revolver which holds the *current* instruction being executed is referred to as IRV_c. The instruction revolver which holds the *next* instruction to be executed is referred to as IRV_n.

The instruction revolver may be addressed as either a source or a destination, supplying or receiving exactly 12 characters in either case. When the instruction revolver is specified as the source in a program control storage reference, the contents of IRV_c is supplied. When the instruction revolver is specified as the destination, the values supplied are loaded into IRV_n.

The principal function of the instruction revolver is discussed in Chapter 3, in its relation to the acquisition and execution of instruction words.

Shift Revolver (SRV)

The shift revolver is a 12 character revolver, the function of which is to hold shift words used to control shifting operations. It may be addressed only as a destination.

In an instruction word, the shift revolver is addressable by the code 998 in the W portion of the instruction word. On the plugboard, the shift revolver is addressed by wiring the SRV hub to an R ADR hub.

Note that:

(a) The special instruction word "Load Shift L" loads the shifts specified in the instruction word itself (See Chapter 3).

(b) Shift instructions included in the U, V, and W portions of a transcop instruction are available on the plugboard through U, V, and W SHIFT hubs.

(c) Use of the shift revolver and available internal and plugboard shifts are discussed in Chapter 3.

chapter 3

PROGRAM CONTROL

INTRODUCTION

Program Control is the section of the UFC-I central computer which directs the interpretation, execution, and sequence of computer instructions. It is the main control for the entire system.

Program Control executes two types of computer instructions: *instruction words* and *program steps*.

Internal program control executes internally stored instruction words by coordinating and controlling the actions of the computer components discussed below, under "Components Used During Internal Program Control". The discussion does not attempt to include all of the components of the computer that are involved in internal control, but only those of particular concern to the programmer.

External program control, which uses many of the same components as internal program control, executes plugboard defined program steps through the coordination of the computer components discussed under "Components Used During External Program Control".

Normally, a combination of internally stored and plugboard defined programs is the most efficient method of accomplishing a given data-processing task, since the best features of each type of program may be utilized as the problem changes in nature.

The techniques by which the programmer is able to effect a transfer of control from an internally stored program to a plugboard defined program, and vice versa, are discussed under "Components Used to Effect Combination Control".

COMPONENTS USED DURING INTERNAL PROGRAM CONTROL

Operation Pulse/Enable Distributor (OED)

The operation pulse/enable distributor is a group of circuits which controls the time sequence of every computer instruction and supplies the principal control pulses and/or enables which are required to carry out each part of an instruction. It is included in this chapter primarily for the purpose of illustrating the execution of a computer instruction.

OED operates in segments numbered 0-13, and at each setting of OED, one or more parts of the instruction are executed. During the execution of most instruction words, OED cycles either from 0-13 or from 2-13 depending upon whether or not the next instruction to be executed was loaded during the processing of the previous instruction word. During execution of most program steps, OED cycles from 3-12.

Table 3-1 shows a *typical* OED cycle. The specific sequence of OED steps that is carried out during the execution of each instruction word or program step varies with different instructions. A complete analysis of the OED cycle for each instruction will be included in a later publication.

TYPICAL OED CYCLE

OED Time	Operation Sequence
*0	Instruction Word (IW) address to SAR
*1	IW obtained and loaded into IRV_n
2	IRV switches; IRV_n becomes IRV_c Process and Special Character of IW set up in Program Control
3	V_1 address to SAR V_1 shift to SK
4	V_1 obtained and loaded into RA
5	V_2 address to SAR V_1 shift in RA
6	V_2 obtained and loaded into RB V_2 shift to SK
7	PAK advanced by one Next IW address to SAR V_2 shifted in RB
8	Next IW obtained and loaded into IRV_n R shift to SK Process (check in addition and subtraction)
9	R address to SAR R shifted in RD (or RC)
10	R to storage
11	Check (in multiplication and division)
12	Step Out hub emits (program step only)
13	Subinstruction initiated.

* If this IW is already loaded into IRV_n, OED cycles from 2-13; if not, OED cycles from 0-13.

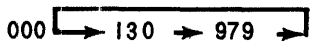
TABLE 3-1

Program Address Counter (PAK)

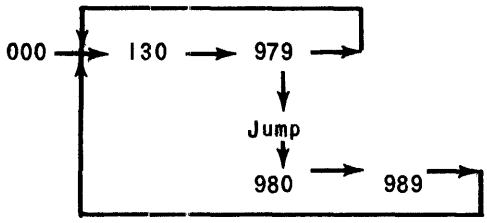
The Program Address Counter (PAK) is a three-character shift register addressable internally by the code 997. Instruction words are always obtained from the address specified by the contents of PAK. It is the device used by program control to "remember" where to get the next instruction word in the program. Unless PAK is modified by the program, it always advances by one, and the next instruction word is taken from the word location next in sequence.

PAK can be set manually to any value in the range of addresses 000 through 999. Operation of a MASTER CLEAR switch automatically sets PAK to 000. Thereafter, PAK is advanced by one or reset depending on the instructions executed by program control. If the first instruction word in the program is to be obtained from any other location than 000, the correct address of the first instruction word must be set up in PAK after the MASTER CLEAR button has been pressed.

Where the contents of PAK are not modified by the program, PAK advances sequentially one word location at a time through word locations as described below, cycling repeatedly through the 130-979 area where instruction words are usually stored.



Where the PAK is manually set to a value within the range of addresses 980-989, or where a jump is programmed to one of these locations, PAK counts to 989 and then jumps back to 130. The complete range of addresses through which PAK cycles is as follows:



When Program Control attempts to load the contents of the storage location specified by PAK into the instruction revolver, it must find there a valid instruction word as defined above. Therefore, PAK values in the range 990-999 should normally be avoided in programming since the memory locations included in this range may not, and some cases, cannot, contain a valid instruction word. The CDR, for example, is addressed by the code 994, which is within the 990-999 range. However, the CDR is a single character register and could not possibly contain a valid instruction word.

Effect of PAK Modifications on a Program

Where the contents of PAK are modified in any of the ways described below, the cycling of PAK will be affected as in Figure 3-1.

(1) PAK may be modified by addressing it as the destination in either an instruction word or a program step. In either case, Program Control executes the instruction stored at the word location specified by the *modified* contents of the PAK as the next instruction.

(2) Special instructions, Unconditional Jump and Test Demand In unconditionally modify PAK with the same effect as the above. Other instructions: Channel Search Probe, Test Incoming Control, Demand In, Jump on Plus, Jump on Negative, Jump on Zero result in modification of PAK conditionally. (See Chapter 4.)

PAK OPERATION WITH PROGRAM MODIFICATION
OF PAK SETTING

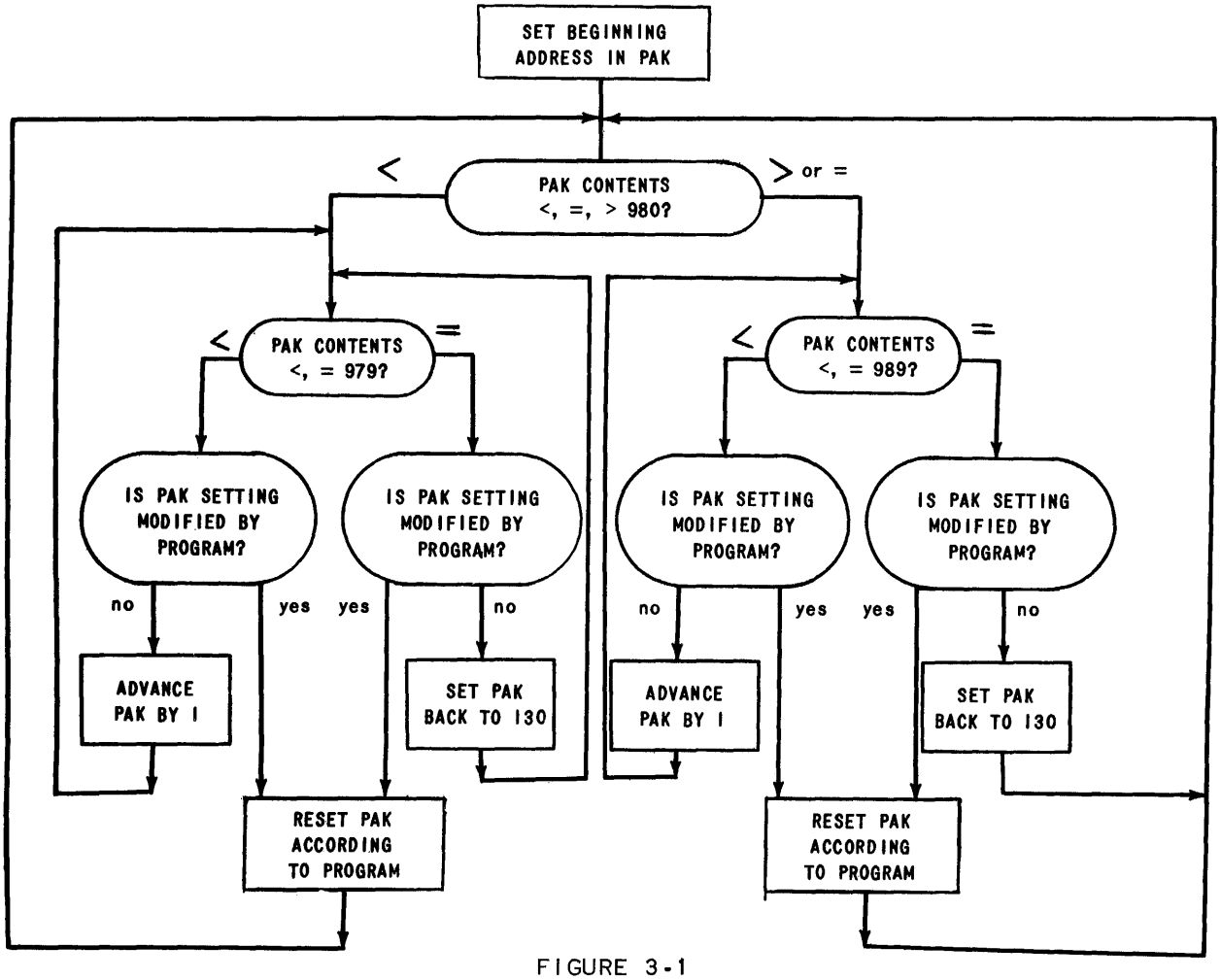


FIGURE 3-1

(3) When program control executes a TRANSCOP (Transfer control to plugboard), (51-98) instruction, it initiates a sequence of program steps.

EFFECT OF PLUGBOARD CONTROLLED PROGRAM
ON PAK SETTINGS

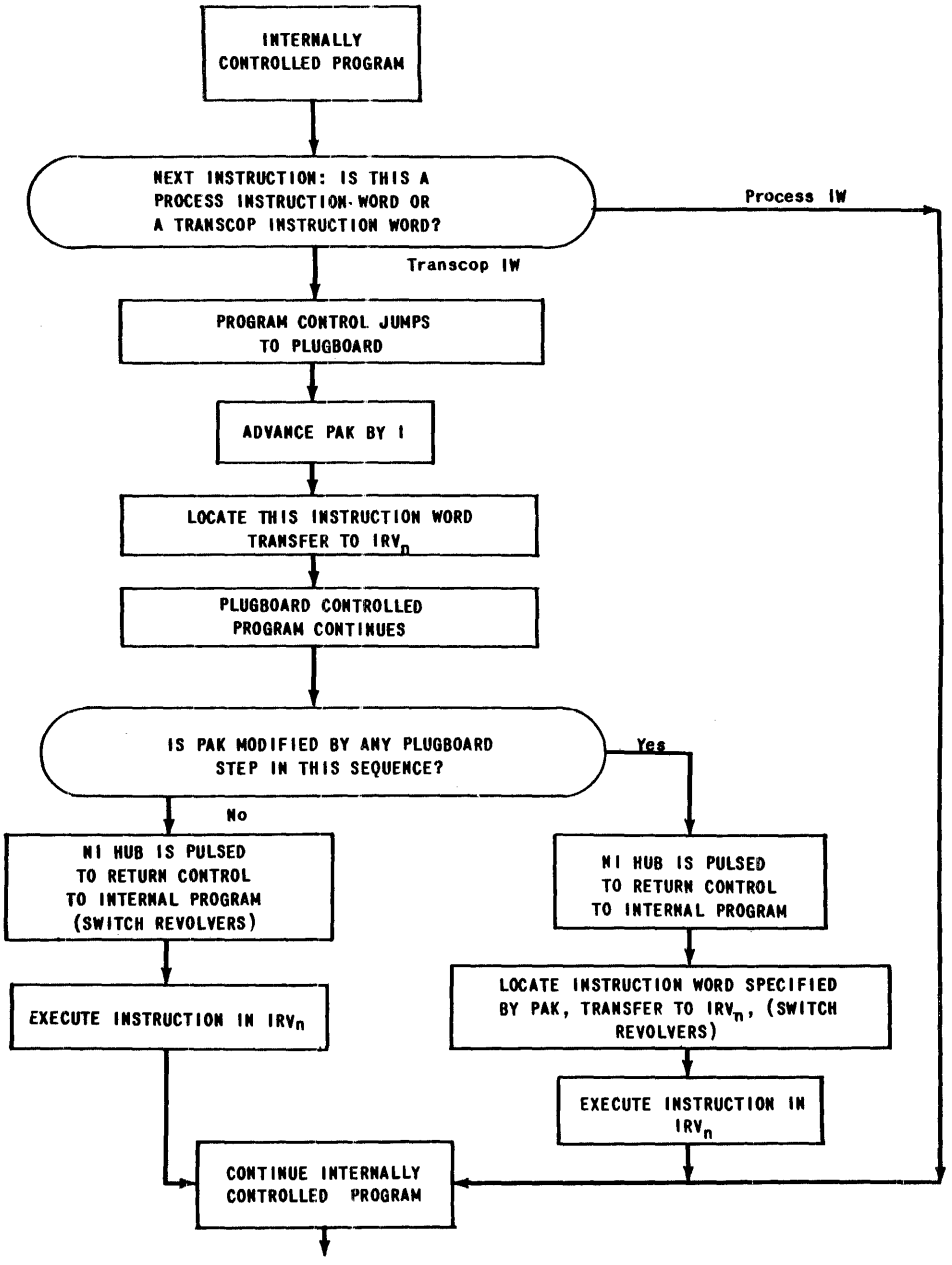


FIGURE 3-2

During the first program step, PAK is advanced by one, and the instruction word specified by the contents of PAK is loaded into IRV_n.

When next instruction (NI) is pulsed, control returns to the internal program. The next instruction executed depends upon whether or not the PAK was sent data during any step during the plugboard sequence:

(1) If PAK *was not* sent data, the next instruction executed is the instruction word loaded during the first program step.

(2) If PAK *was* sent data, the instruction word specified by the contents of PAK (at the time NI is pulsed) is obtained, and that instruction becomes the next instruction executed in the program.

Storage Address Register (SAR)

The Storage Address Register is a three character non-addressable register which temporarily holds the following "storage addresses" during the execution of computer instructions:

SAR receives from PAK and holds temporarily the address of the location from which the next instruction word is to be obtained during OED 7 of the current instruction word.

SAR holds temporarily the addresses of the locations from which V_1 and V_2 are obtained, and the address of the location at which R is stored during the execution of both internal and external programs.

During the execution of input/output instructions, SAR successively holds the U and V sections of the instruction word. The interpretation of U and V determines which input/output unit is called for, whether track switch will occur, and which computer-I/O control lines will be activated.

Instruction Revolver (IRV)

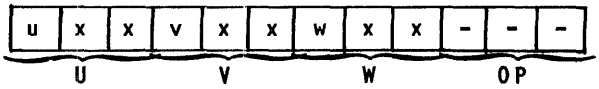
The instruction revolver (IRV), internally addressable by code 996, actually consists of two 12-character revolvers, since both have the same address. One revolver, IRV_c contains the current instruction word, and the other, IRV_n, contains the next instruction word to be executed.

Program control obtains from IRV_c the U, V, W, and OP sections of the current instruction word as it executes that instruction word. Normally, while one instruction is being executed, the next instruction word is loaded into IRV_n. Upon completion of the current instruction word, program control switches to IRV_n, finding there the next instruction word to be executed. This method of operation provides a valuable time-saving factor because the next instruction may be found and loaded into one revolver, IRV_n, while the other revolver, IRV_c, is occupied by the current instruction word.

When PAK is modified, the next instruction word is not immediately loaded into IRV_n, and program control does not switch revolvers. First it must gain access to the instruction word specified by the modified contents of PAK, load it into IRV_n, and then switch to that revolver. Thus, the instruction specified by the modified PAK becomes the next instruction to be executed.

Shift Revolver (SRV)

The shift revolver (SRV) is a rapid access storage location of 12 characters, which contains the shifting instructions to be followed as the V_1 and V_2 operands are loaded into their respective arithmetic registers and the results are sent to a destination. SRV can be loaded by the internal instruction LS, or by the transfer process, addressing 998 as the destination. The format for all shift words held in SRV is as follows:



(1) u, v, and w indicate the type of shift to be executed. Available types of shift, and the code which initiates each type include:

0 = No shift on this operand.

1 = Right end around. All characters including the sign position move to the right, and the characters shifted off re-enter the register at the left end.

2 = Left end off. All characters except the sign position move to the left, with zeros inserted in the vacated positions: characters shifted off are lost.

3 = Right end off. All characters except the sign position move to the right, with space codes inserted in the vacated positions; characters shifted off are lost.

(2) xx in the two character position at the right of u, v, and w indicate the number of character positions to be shifted, ranging from one character to 11 characters.

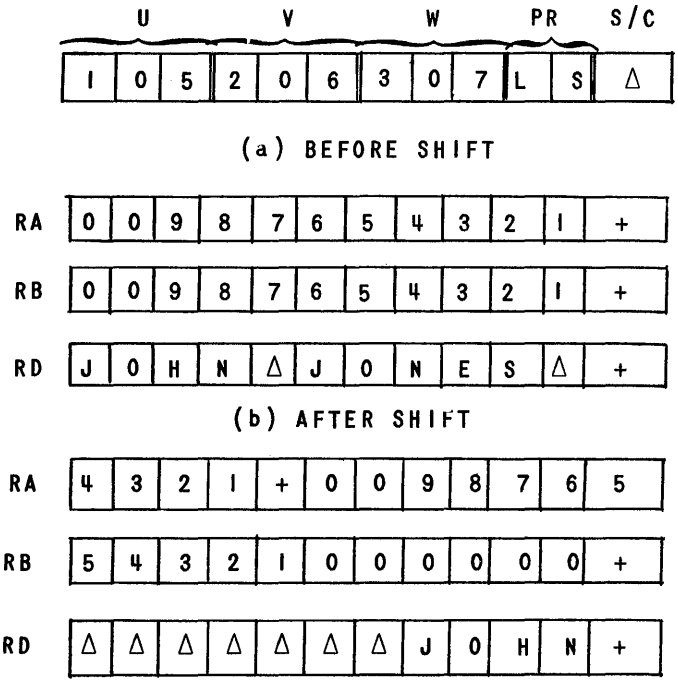
(3) The operation (OP) section of a shift word is not significant in the shift revolver itself. However, in the example below, the code 32 or LS in the OP section of the instruction word loads the U, V, W pattern into the shift revolver.

Example:

We will assume that arithmetic registers A, B and D contained the characters shown in the following diagram "Before Shift".

Shift instruction has previously been loaded into SRV.

After execution of the shifts, the registers would appear as shown in the diagram "After Shift".



SRV is cleared each time access to a new instruction word is obtained, except where the instruction just executed loaded the SRV, either by transfer or by the LS internal command. In effect, the shift pattern loaded into the shift revolver is "used up" by the next instruction word. The following, therefore, should be observed:

(1) SRV should be loaded *immediately before* a process instruction word which is to use the shift word held in SRV.

(2) Where a plugboard sequence requires reference to the SRV through U, V, W SHIFT operations, the SRV must be loaded by the instruction word *immediately preceding* the transcop instruction which relinquishes control to the plugboard.

Shift Counter (SK)

The shift counter is a three character non-addressable register, which specifies the type of shift to be performed and the number of places to be shifted during the execution of both internally stored and plugboard defined instructions. When a shift is to be executed the number of places to be shifted is automatically loaded into SK. As the data is shifted in the arithmetic register SK counts backward. When it reaches zero, the shift terminates.

INSTRUCTIONS IN WHICH SHIFTS CAN BE PROGRAMMED TO OCCUR

Computer Instructions in which the following Processes are executed:	Quantities for which Shifts can be Programmed	Arithmetic Registers Involved:	
Add	V_1, V_2, and R	V_1 in RA	
Add & Check			
Subtract			
Subtract & Check			
Mask Transfer		V_2 in RB	
Substitute U			
Substitute V			
Substitute W		R in RD	
Multiply Store Upper	V_1, V_2, and R*	V_1 in RA	
		V_2 in RB	
Multiply Store Lower		R in both RC & RD or in RC or RD*	
Multiply, Store Upper & Check	V_1 and V_2**	V_1 and RA	
Multiply, Store Lower & Check		V_2 in RB	
Divide, Store Quotient	V_1, V_2, and R	V_1 in RC***	
		V_2 in RB	
		R in RD	
Divide, Store Quotient & Check	V_1, V_2 **	V_1 in RC***	
		V_2 in RB	
Divide, Store Remainder	V_1, V_2, and R	V_1 in RC***	
		V_2 in RB	
		R in RC	
Divide, Store Remainder & Check	V_1, V_2**	V_1 in RC***	
		V_2 in RB	
Compare	V_1 and V_2	V_1 in RA	
		V_2 in RB	
Suppress Left Zeros	V_1 and R	V_1 in RA	
Left Normalize		R in RD	
Arithmetic Transfer	Three separate shifts can be programmed. The Source Data (or the Source Data plus Space codes) are manipulated.	All shifting done in RD	V_1 Shift time
			V_2 Shift time
			R Shift time

TABLE 3-2

* In left and right shifts for R in Multiply processes, Registers C and D are shifted as a single (22-character) register. In right end around shifts for R in Multiply processes, however, only the register from which the result is stored is shifted.

** If a shift for R is programmed, the shift will be performed but will cause an ARITHMETIC CHECK ERROR.

*** V_1 is not actually shifted in RA, although the effect in the Divide algorithm is the same as if V_1 were also shifted in RA. The shift for V_1 is performed in RC to provide a correct remainder should the (u-v+n+1)<1 condition be detected at process time (See Chapter 7).

Process Register (PR)

The process register (PR) is a two-character register, *not addressable* by the programmer. It contains either the low order four bits of process or transfer control codes from the operation section of the current instruction word, or the number of the plugboard step currently being executed.

When process instruction words are initiated, PR receives the process code, a number < 50.

When transcop instruction words are initiated, PR receives the transfer control code, always a number > 50. This directs program control to the next plugboard step to be executed.

Code Distributor Register (CDR)

The code distributor register (CDR) is a one-character register addressable internally by the code 994. In common usage it is addressed in W portions of instruction words for the purpose of loading a desired control character into CDR. At some point later in the program, CDR is probed and the code distributor (CD) functions as a multiple position plugboard switch.

A complete description of the code distributor and the code distributor register, together with their use, is included in this chapter under "Components Used During External Program Control".

Branch Storage

Branch storage, which is not addressable internally, is a temporary memory location the setting of which is determined by the result of arithmetic or comparison operations. During execution of instruction words, the setting of branch storage is used to set conditional storage as described under "Conditional Storage" below. Conditional storage must be set in the instruction word which sets branch storage if a + or - condition is desired, since branch storage is cleared to 0 just before the process of each instruction is initiated.

(1) In arithmetic operations:

Where the result, R, is > 0, branch storage is set to +;
Where the result, R, is = 0, branch storage is set to 0;
Where the result, R, is < 0, branch storage is set to -.

(2) In comparison operations:

Where U > V branch storage is set to +;
$V_1 > V_2$

Where U = V branch storage is set to 0;
$V_1 = V_2$

Where U < V branch storage is set to -.
$V_1 < V_2$

(Note: For further detail, see "Branch Storage Settings", Chapter 7.)

Conditional Storage

Conditional storage provides a method of varying an internal program when the sequence of operations is determined by the algebraic sign of the result of an arithmetic or compare operation. The result of the operation first establishes the condition +, -, or 0 in branch storage. This condition may then be used to set conditional storage. When conditional jump instructions are executed, the setting of conditional storage becomes the deciding factor in determining the path which the program will follow (see "Repertory of Instruction").

Conditional storage may be considered a more permanent storage location than branch storage. When any of the appropriate special characters appear in the special character position of an instruction word, conditional storage is "set" to the +, -, or 0 condition prevailing in branch storage at the time the special character position is examined. Each setting of conditional storage prevails until another "Set Conditional Storage" is called for by an instruction word.

"Set Conditional Storage" is accomplished by placing a 2, 3, 4, 9, B, C, D, or I in the special character position of the instruction word. When the character 9 or I is used, only "set conditional storage" will be accomplished. When any other of the characters listed is used, breakpoints are also examined as shown in Table 4-2 "Valid Subinstructions", Chapter 4.

If any process instruction word contains a set conditional storage character, and initiates an arithmetic or comparison operation, the result of that operation will determine the setting of conditional storage.

If a process instruction word contains a set conditional storage character but does *not* initiate an arithmetic or comparison operation, conditional storage will be set to the zero (0) condition, since this is the "cleared" condition of branch storage as described in "Branch Storage" above.

Special Character Register (SR)

The special character register is a one-character, non-addressable register which receives the contents of the special character S/C portion of the instruction word when it is obtained from the instruction revolver (IRV_C). The contents of SR determine which extension or modification of the basic operation is to be carried out as the instruction word is executed. An extension or modification of an instruction word may be either (1) a signal, such as received from a "special character out" plugboard hub, or (2) a subinstruction, such as "channel search", which affects a series of events. (See Chapter 4 for valid S/C characters and descriptions.)

Signals and subinstructions take effect prior to the execution of the next instruction word except:

(1) Where the subinstruction is "suppress check", in which case the S/C takes effect during the processing of the current arithmetic operation.

(2) Where SR received a character from a transcop instruction, in which case the signal or subinstruction is delayed until after the plugboard sequence initiated by the TC instruction has been completed and the NI hub has been signalled.

Breakpoints

When it is desirable to interrupt a program at the end of a particular instruction word or at some point in a sequence of plugboard steps, the following conditions must be fulfilled:

(1) The appropriate BREAKPOINT SELECTOR button, or combination of BREAKPOINT SELECTOR buttons on the computer control cabinet is depressed:

BREAKPOINT SELECTOR 1 = Examine breakpoint 1.
BREAKPOINT SELECTOR 2 = Examine breakpoint 2.
BREAKPOINT SELECTOR 3 = Examine breakpoint 3.
CLEAR = Ignore all breakpoints.

(2) The appropriate letter or number is included in the special character S/C position of the instruction word.

When a process instruction word containing a breakpoint character in the S/C position of the word is executed and the appropriate BREAKPOINT SELECTOR is depressed, the program is interrupted at the end of the execution of the instruction word, and the corresponding "breakpoint hub" (1, 2, or 3) on the plugboard emits a pulse.

When a transcop instruction word containing a breakpoint character in the S/C position of the word is executed and the appropriate BREAKPOINT SELECTOR is depressed, control is transferred to the plugboard and the entire sequence of plugboard operations initiated by the transcop instruction is executed. Then the program is interrupted and the corresponding breakpoint hub (1, 2, or 3) on the plugboard emits a pulse. Special characters and the breakpoint pulses they initiate are given in the table "Valid Subinstructions", Chapter 4.

COMPONENTS USED DURING EXTERNAL PROGRAM CONTROL

Operation Pulse/Enable Distributor (OED)

The OED is a group of circuits which controls the time sequence and supplies the principal control pulses and/or enables required to carry out each part of a program step or instruction word.

Typical steps in the OED are discussed above under "Components Used During Internal Control".

Program Address Counter (PAK)

The program address counter, previously discussed in its vital role as a component of internal program control, is also addressable on the plugboard, and may be modified during external program control.

The following example shows the modification of PAK by the addition of some value obtained from word 2 of factor storage #1. The STEP OUT hub of step 76 is wired to NEXT INSTRUCTION which transfers control to the internal program where the modified contents of PAK designate the address of the next instruction to be executed.

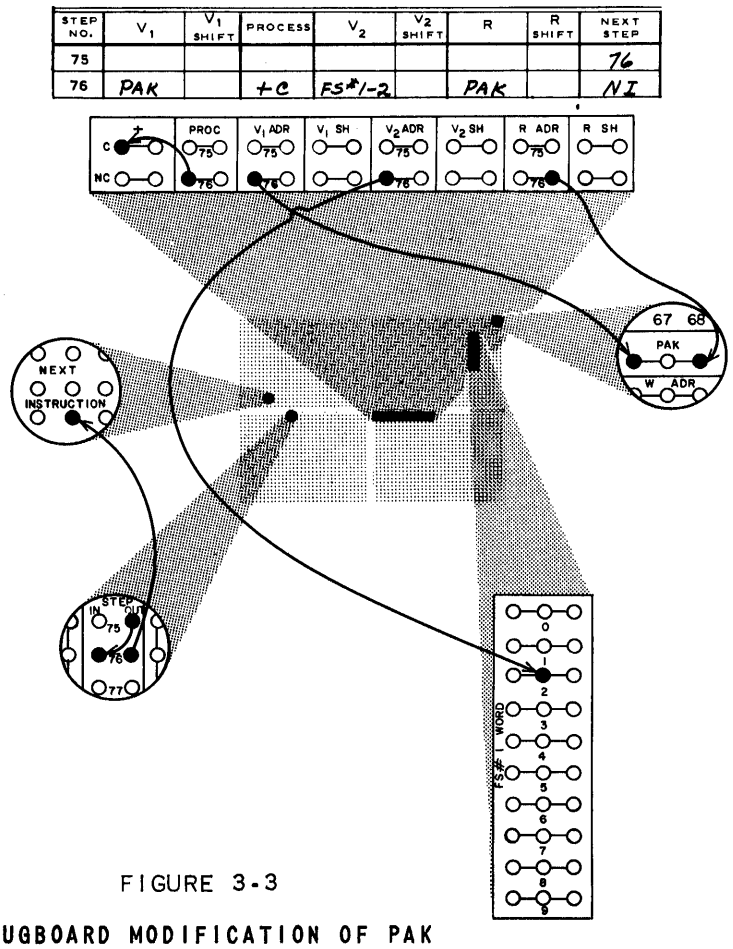


FIGURE 3-3

PLUGBOARD MODIFICATION OF PAK

Storage Address Register (SAR)

During plugboard defined programs, SAR holds the addresses of the locations from which V_1 and V_2 are obtained, and address of the location at which R is stored.

Instruction Revolver (IRV)

The instruction revolver, although not actively engaged during external program control, is addressable from the plugboard either as a source or as a destination. When addressed as a source, the contents of IRV_c are supplied; when addressed as a destination, the result of the operation is stored at IRV_n.

In the following example, the contents of IRV_c have been modified by the addition of a value obtained from field A of the block transfer buffer. The modified instruction has been stored at IRV_n. When the NEXT INSTRUCTION hub is signalled, instruction revolvers will be switched and the next instruction to be executed will be the one just modified in program step 74.

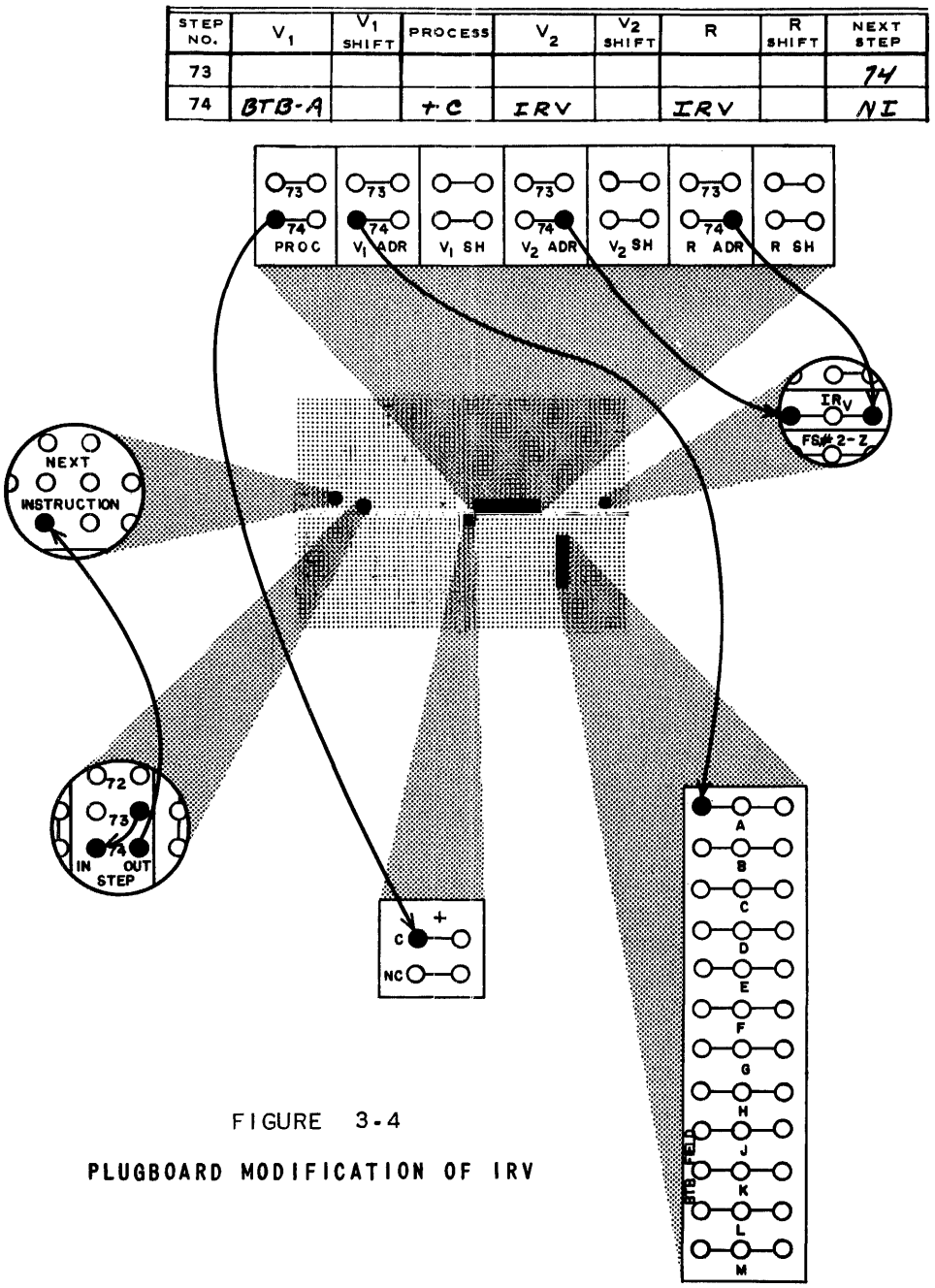


FIGURE 3-4

PLUGBOARD MODIFICATION OF IRV

Shift Revolver (SRV)

The shift revolver is addressable on the plugboard by wiring one of the "SRV" hubs as a destination address or one of the U, V, or W SHIFT hubs.

A shift pattern may be "loaded" into the shift revolver (SRV) by a plugboard step. This pattern must be used during the execution of the *first* instruction word after internal program control takes command, since the shift revolver (SRV) is cleared each time a new instruction word is used.

After a shift pattern has been loaded into the shift revolver, either by the internal command "LS", by a transfer instruction, or by a plugboard step as described above, the contents of the shift revolver are accessible to the programmer through two sets of "U, V, W SHIFT" hubs on the plugboard. Wiring of V_1SH, V_2SH, or R SH to any one of these hubs will produce the shift specified by this portion of the shift revolver. Any combination of V_1SH, V_2SH, R SH and U, V, W SHIFT hubs may be wired. For example:

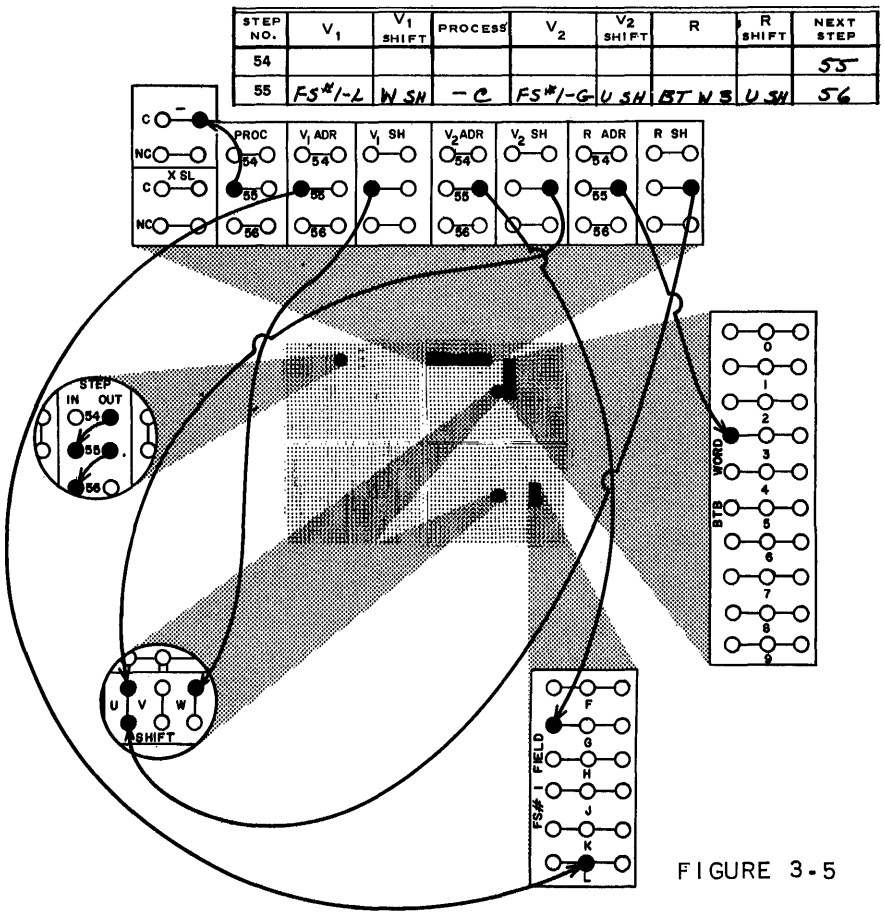


FIGURE 3-5

SHIFTING OF PLUGBOARD ACCORDING TO SHIFT REVOLVER

In the above example, the contents of FS #1-L will be shifted in accordance with the W section of the shift revolver (SRV); FS #1-G will be shifted in accordance with the U section of SRV; the result will be shifted in accordance with the U section of SRV before it is stored at BTB word 3.

Shift Counter

The shift counter performs the same basic function during external program control as it performs during internal program control discussed above.

Shift Hubs

Shifting of arithmetic registers is accomplished on the plugboard by wiring V_1SH, V_2SH, or R SH to the desired shift hubs.

The shifts available externally to the programmer are the same as those discussed previously in this chapter under "Components of Internal Program Control-Shift Revolver" and in Table 3-2.

Appropriate wiring to facilitate shifting by the use of shift hubs is shown in the following diagram:

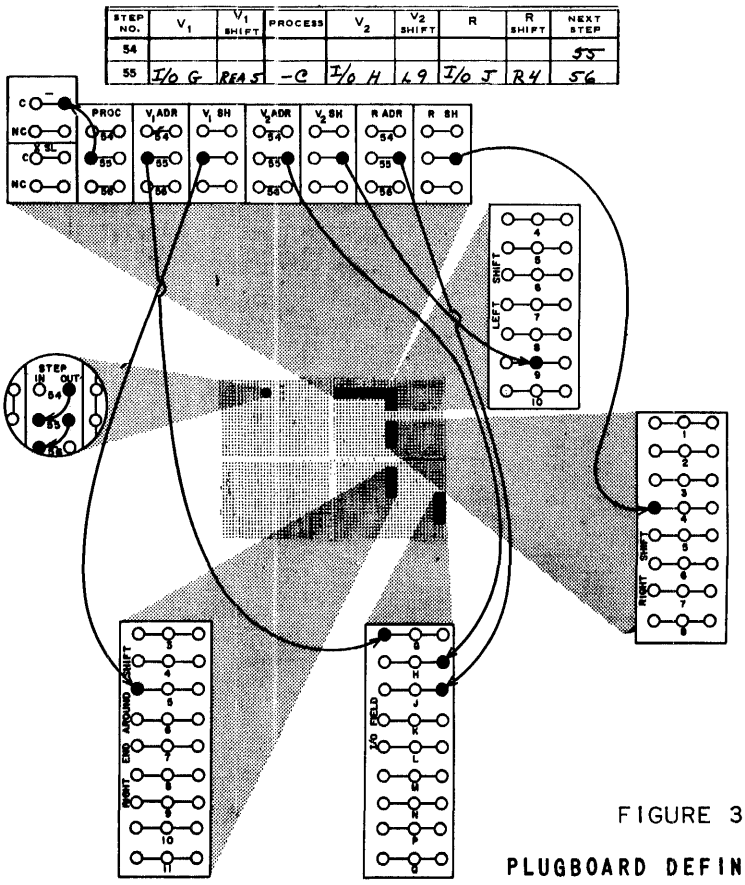


FIGURE 3-6

PLUGBOARD DEFINED SHIFTS

Code Distributor

The code distributor provides an effective means of varying a computer program, depending on conditions brought to the central computer by an input device, or conditions which develop during the processing of information.

The code distributor is composed of a single-character register, the code distributor register (CDR), and a multiple translator which performs the decision-making function.

The code distributor register (CDR) must first be "loaded" by transferring a single character to it. The current necessary to continue the program will be received from the appropriate hub of the code distributor, depending upon the wiring of the code distributor and the character held by CDR.

Although 63 characters comprise the Univac code, the code distributor functions only when the code distributor register holds one of the 40 characters shown in the chart below:

USABLE CODE DISTRIBUTOR CHARACTERS

ZONE	XS3 BITS										
	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	
00	0	1	2	3	4	5	6	7	8	9	Group 1
01	;	A	B	C	D	E	F	G	H	I	Group 2
10	)	J	K	L	M	N	O	P	Q	R	Group 3
11	+	/	S	T	U	V	W	X	Y	Z	Group 4

The three basic methods of using the code distributor are explained and diagrammed as follows:

A. CDR Pulse

When one of the 40 usable characters has first been loaded into the code distributor register, and one of the CDR PULSE IN hubs is probed, the code distributor emits a pulse from one of the ten CDR PULSE OUT hubs labeled 0-9. This pulse directs the program to continue along the path specified by the wiring of this active hub.

The CDR PULSE OUT hub which will be activated can be determined from the above chart. Since the code distributor in this method of operation interprets only the four low order (or XS3) bits of a character (those at the top of the vertical columns), there is no distinguishable difference among the characters in any one column. The characters 2, B, K, or S are identical in that 0101 represents the pattern of the low order four bits for each of them. Therefore, if any one of the characters 2, B, K or S is in the CDR when CDR PULSE IN is probed, a pulse will be emitted from the 2 hub. This function makes it possible for pulses to be switched to one of ten routes, depending on the character held by the code distributor register and the wiring of hubs 0 through 9.

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP	REMARKS
51								51	
52	I/O 4		A.T.			CDR		CDR PULSE IN	WORD 4 OF I/O UNIT ON DEM. → CDR

CDR PULSE IN: STEP 52			
NO	OUT	NO	OUT
0	STEP 53	3	STEP 59
1	STEP 55	4	N.I
2	STEP 57	5	

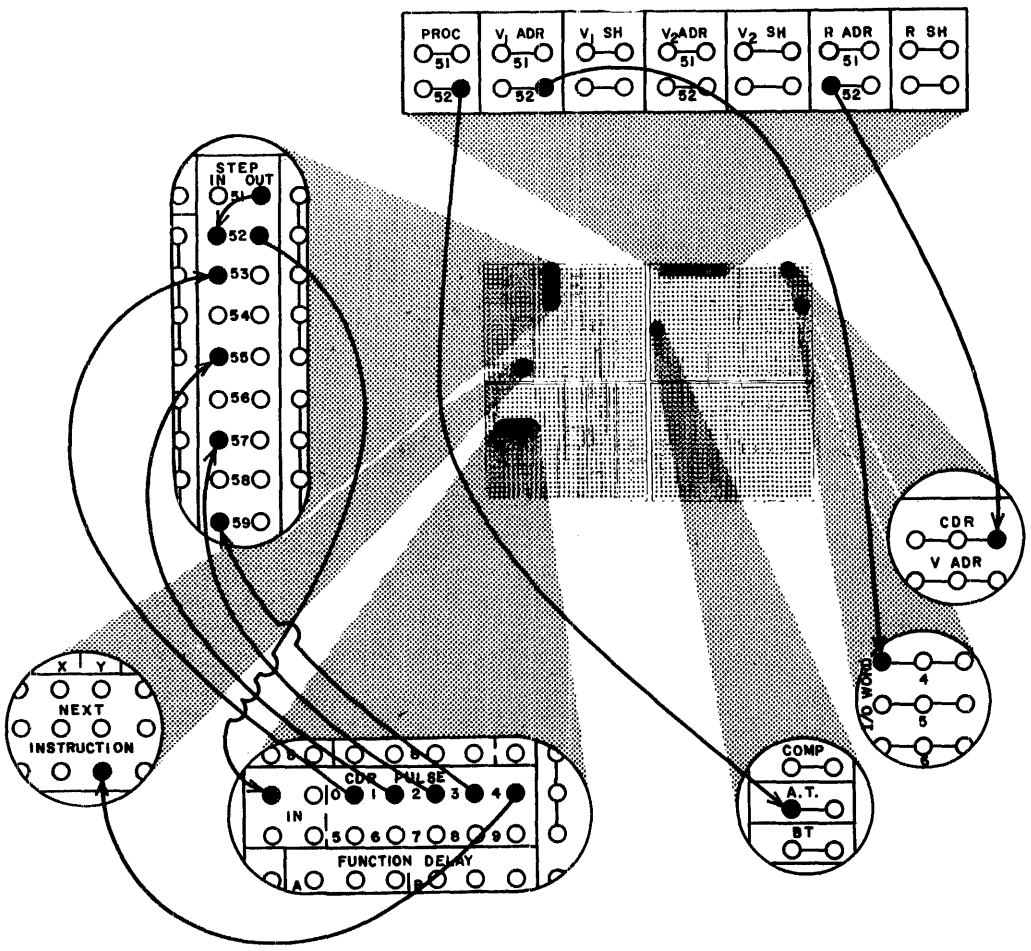


FIGURE 3-7

CDR PULSE TO SELECT PROGRAM STEP
FROM TRANSACTION CODE

The above wiring diagram illustrates the use of CDR PULSE to select one of several alternative program paths (in this case alternative program steps) depending on a transaction code in an input punched card.

(1) Assuming that punched cards represent five different types of transactions being processed against inventory records, the transaction codes in effect are:

Code		Transaction
0	=	Stock Receipt
1	=	Stock Withdrawal
2	=	Purchase Order
3	=	Price Change
4	=	Physical Inventory Adjustment

(2) Step 52 transfers a code from input word 004 into the code distributor register CDR.

(3) The STEP OUT of step 52 probes CDR PULSE IN, and a pulse is emitted from the 0, 1, 2, 3, or 4 hub, depending on the code in the CDR. In the above situation, if a 3 were loaded into CDR in step 52, a pulse would be emitted by hub CDR PULSE 3 and the program would continue by executing step 59.

B. CDR Group

When one of the 40 usable characters has first been loaded into the code distributor register CDR, and any one of four CDR IN hubs (GROUP 1, GROUP 2, GROUP 3, GROUP 4) is probed by a d-c level current, a d-c enable current is available from one (and only one) of the ten CDR OUT hubs associated with the CDR IN GROUP hub probed.

In this method of operation, the code distributor recognizes only the four lower order or XS3 bits (the four bits at the top of the vertical columns in the usable character chart). Therefore, when any of the characters 4, D, M, or U is loaded into CDR and CDR IN GROUP 2 is probed, a d-c enable current is received from CDR OUT hub D.

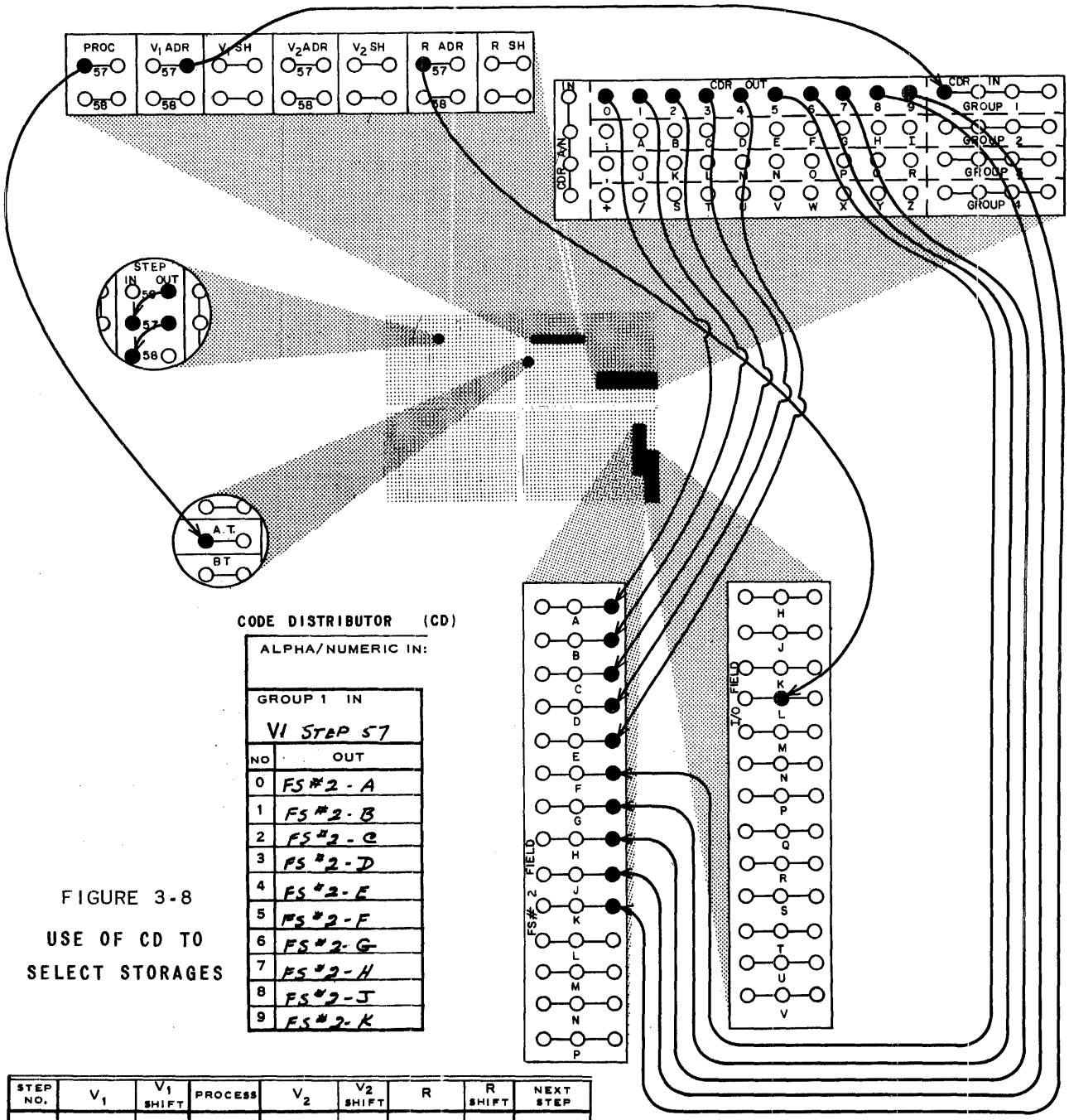
Summary of CDR GROUP usage:

If a CDR IN GROUP 1 hub is probed, one of the ten CDR OUT hubs (0-9) will have a d-c output.

If a CDR IN GROUP 2 hub is probed, one of the ten CDR OUT hubs (;-I) will have a d-c output.

If a CDR IN GROUP 3 hub is probed, one of the ten CDR OUT hubs ()-R) will have a d-c output.

If a CDR IN GROUP 4 hub is probed, one of the ten CDR OUT hubs (+-Z) will have a d-c output.



CODE DISTRIBUTOR (CD)

ALPHA/NUMERIC IN:

GROUP 1 IN V1 STEP 57	
NO	OUT
0	FS#2 - A
1	FS#2 - B
2	FS#2 - C
3	FS#2 - D
4	FS#2 - E
5	FS#2 - F
6	FS#2 - G
7	FS#2 - H
8	FS#2 - J
9	FS#2 - K

FIGURE 3-8

USE OF CD TO SELECT STORAGES

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
56								57
57	CDR-IN		A.T.			I/O-K		58

C. CDR Alphanumeric (A/N)

When one of the 40 usable characters has first been loaded into the code distributor register CDR, and the ALPHANUMERIC IN hub is probed by a d-c level current, a d-c enable current will be available from only one of the 40 CDR OUT hubs, that hub which corresponds exactly to the character held by the code distributor register.

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
74								75
75	I/O - 0		A.T.			CDR		76
76	I/O - 1		+C	CDR - A/N IN		CDR - A/N IN		77

CODE DISTRIBUTOR (CD)							
ALPHA/NUMERIC IN: V2 STEP 76; R STEP 76							
GROUP 1 IN		GROUP 2 IN		GROUP 3 IN		GROUP 4 IN	
NO	OUT	NO	OUT	NO	OUT	NO	OUT
0		;		,		+	
1		A	GSB - A	J	GSB - J	/	
2		B	GSB - B	K	GSB - K	S	GSB - S
3		C	GSB - C	L	GSB - L	T	GSB - T
4		D	GSB - D	M	GSB - M	U	
5		E	GSB - E	N	GSB - N	V	
6		F	GSB - F	O		W	
7		G	GSB - G	P	GSB - P	X	
8		H	GSB - H	Q	GSB - Q	Y	
9		I		R	GSB - R	Z	

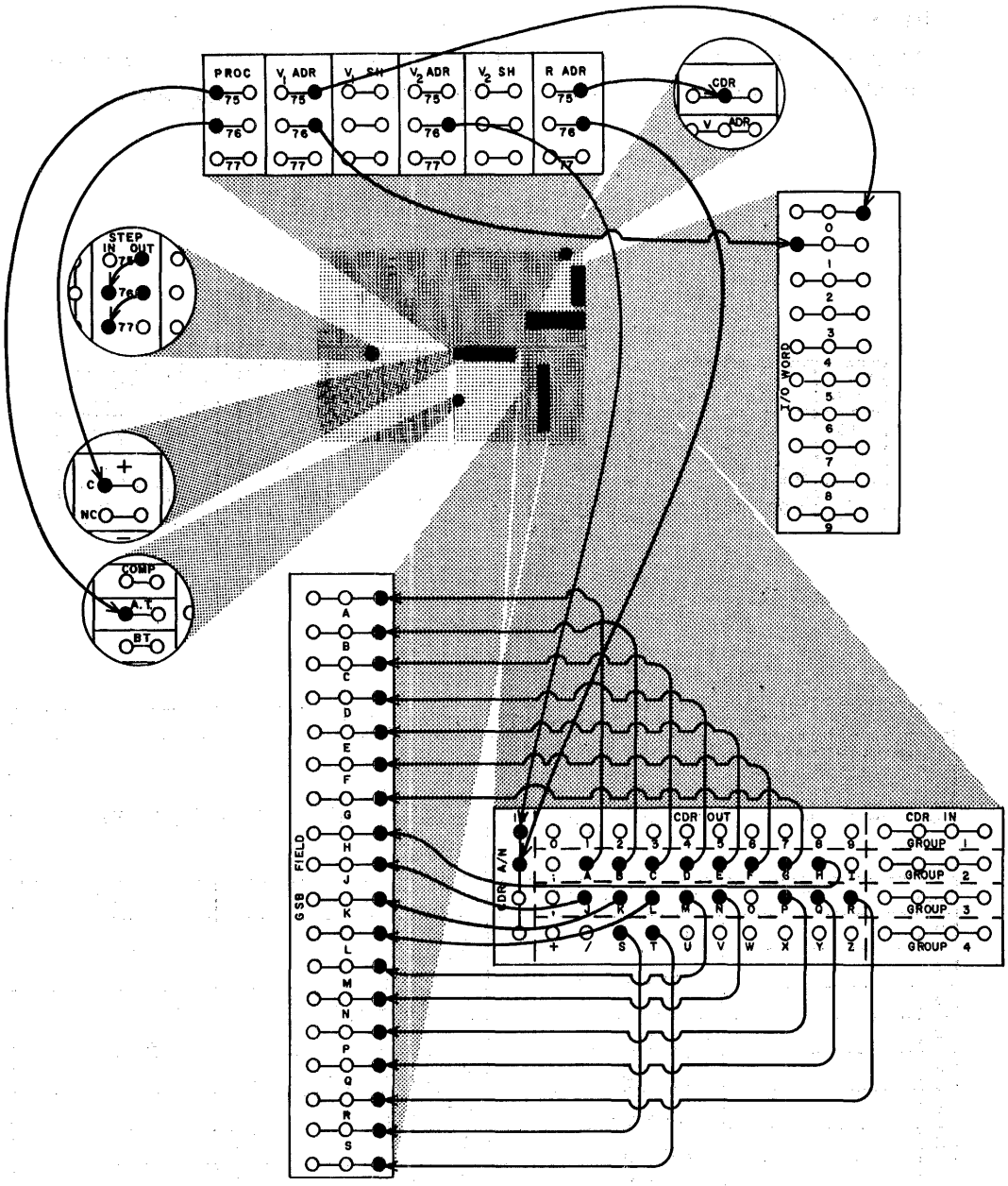


FIGURE 3-9

CD TO SELECT STORAGES BY CDR ALPHANUMERIC

In this method of operation, the code distributor recognizes all six bits (ignoring parity bit) of the 40 usable characters. Therefore, a d-c enable current will be available from CDR OUT hub "D" only on the condition that a "D" was previously loaded into the code distributor register.

Thus the code distributor probed by CDR A/N provides a choice of none or one of 40 different routes for the program to follow.

The above diagram, Figure 3-9, illustrates a method of using the code distributor to select storages (in this case fields in the general storage buffer) depending upon an alphabetic code in the code distributor register.

(1) Assuming that the inventory records for stock carried at 18 branch warehouses are stored on the general storage drums, a file entry is maintained on the drum for each stock number, with the quantity on hand at each of the 18 warehouses stored in a separate field. Each stock receipt or withdrawal is identified by a stock number which includes an alphabetic character which identifies the branch warehouse involved.

(2) In step 75, the alphabetic character which identifies the branch warehouse is loaded into the code distributor register.

(3) In step 76, a value, such as a stock receipt quantity, is added to a branch warehouse inventory quantity located through the code distributor; and the sum of this calculation is stored in the field, again through the use of the code distributor.

D. Multiple choices from a single code:

The use of the code distributor has been simplified in the above explanation by dividing usage into three separate concepts, but in actual practice, the code distributor can become a much more powerful decision-making device, if the three are integrated, as indicated in the example below. This example illustrates the ability of the code distributor to select more than one variable factor, depending on a single code held in the code distributor register:

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
51	FS #1-A		A.T.			CDR		52
52	GSB-F		+ C	CDR IN GROUP 1		GSB-G		53
53	GSB-H		CDR IN GROUP 4	GSB-J		GSB-K	CDR IN GR 3	54

CODE DISTRIBUTOR (CD)							
ALPHA/NUMERIC IN:							
GROUP 1 IN V2 STEP 52		GROUP 2 IN		GROUP 3 IN R SH STEP 53		GROUP 4 IN PR STEP 53	
NO	OUT	NO	OUT	NO	OUT	NO	OUT
0		;		,		+	
1	FS #1-C	A		J	W. SHIFT	/	+ CHECK
2	FS #1-B	B		K	RT SH. 3	S	- CHECK
3	FS #1-D	C		L	RT SH 4	T	X, SL CHECK
4	FS #1-E	D		M	RT SH 5	U	÷ SQ CHECK
5	FS #1-F	E		N	RT SH 6	V	- NO CHECK
6	FS #1-G	F		O	RT SH 7	W	+ NO CHECK
7		G		P		X	
8		H		Q		Y	
9		I		R		Z	

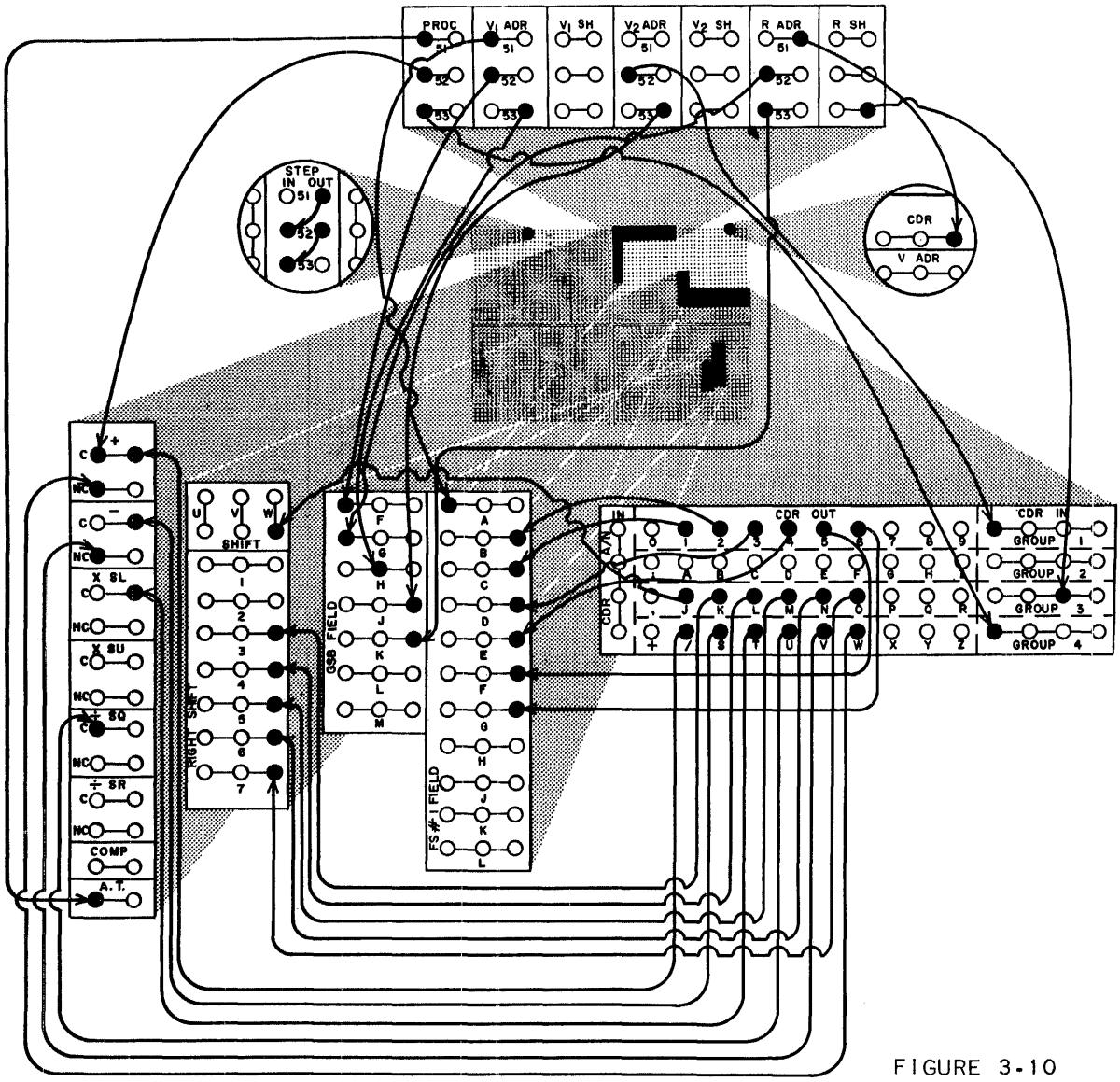


FIGURE 3-10

CDR TO SELECT STORAGE, PROCESS & SUBJECT
FROM A TRANSACTION CODE

(1) Step 51 transfers, into the code distributor register, a number ranging from 1 to 6.

(2) V_2ADR of step 52 probes CDR IN GROUP 1, and receives a d-c enable from one of the CDR OUT hubs numbered 1 through 6. This determines the field in factor storage #1, FS #1 FIELD B-G, from which the V_2 operand is to be obtained.

(3) Process PROC of step 53 probes CDR IN GROUP 4 to determine which one of 6 arithmetic processes is to be performed.

(4) Result shift, RSH of step 53 probes CDR IN GROUP 3 to determine which one of 6 shifts is to be performed on the result.

In this example, a single code (4) stored in the code distributor register directs (a) the V_2 address to field FS #1-E, (b) the process of step 53 to the "divide, store quotient" (÷SQ) process, and (c) the right shift of step 53 to RIGHT SHIFT 5.

Selectors

Selectors provide a means of varying the computer program depending on conditions which may arise in the input data or during the processing of that data by the computer. The current to be controlled is brought to the selector by wiring of the "common" (C) hub of the selector.

(1) Nonselect (NS) is the normal or dormant position of a selector. Current will flow from the "common" (C) hub through the "nonselect" (NS) hub whenever the selector is in this position.

(2) Select (S) is the "picked up" position of a selector. Current will flow from the "common" (C) hub through the "select" (S) hub whenever the selector is in this position.

The activation or "pick up" of a selector to the "select" position is accomplished by an electromagnet which pulls the movable switch arm to make contact with the select side of the switch. Current to activate this electromagnet may be supplied by a device called a "program select", or by use of the current from either a computer input control line or "Selector hold B+".

The wiring necessary to "pick up" a selector by a "program select" is diagrammed and explained below:

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
51								52
52								IN PS 2
53								54
54								55

PROGRAM SELECTS (PS)				
NO	IN	DELAY OUT	DROP OUT	B+
1				
2	Our STEP 52	IN STEP 53		T 13

T #	PICK-UP FROM	GROUND	SELECT	COMMON	NON-SELECT
13	P.S.2	CG			

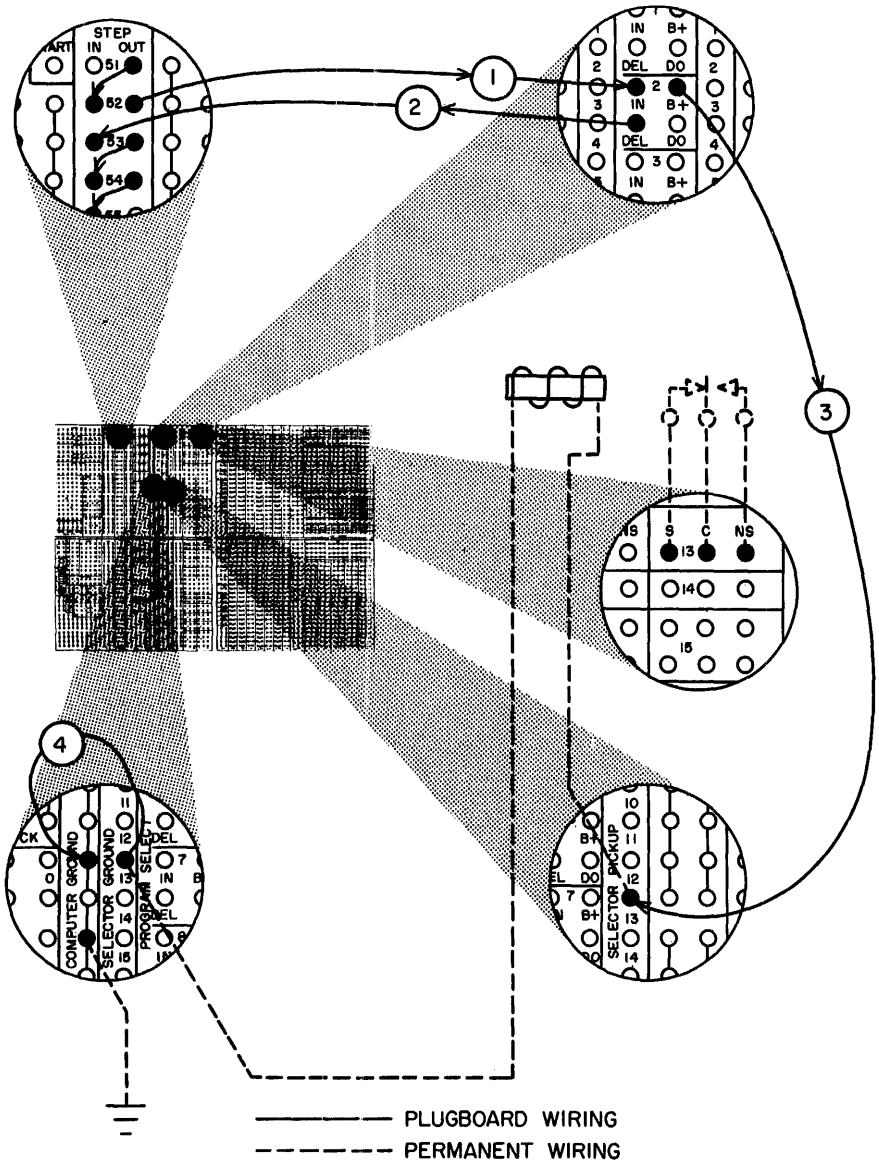


FIGURE 3-11 SELECTORS

We will assume that after completion of step 52, selector 13 is to be picked up before step 53 is initiated. In this case:

(1) The STEP OUT of step 52 is wired to the IN hub of program select 2. Other sources of a pulse to activate a program select include SPECIAL CHARACTER OUT and OUT of a BRANCH.

(2) DELAY OUT (DEL) of program select 2 is wired to IN of step 53 to continue the program. Because a delay of approximately 15 milliseconds is required to set any selector to the select (S) position, a DELAY OUT is used to delay the program 15 milliseconds while the selector is being picked up for use in step 53. If the selector were to be used in a later step (at least 15 milliseconds later), the delay would not be necessary, and the STEP OUT could be wired to both STEP IN of a later step and the IN hub of program select 2.

(3) The B+ of program select 2 is wired to selector pickup 13, to energize the electromagnet which moves selector 13 to the "select" (S) position.

(4) SELECTOR GROUND 13 is wired to COMPUTER GROUND to complete the B+ circuit. (COMPUTER GROUND may be used to ground a selector at any time.) DEMAND GROUND may be used only when the associated demand station is "on demand" by the computer. (See Chapter 6.)

Branching

The plugboard BRANCH is a method of varying the sequence of a computer program depending on the result of arithmetic and comparison operations. When any of these operations is completed, branch storage is "set" to a plus (+), minus (-), or zero (0) condition. If branch storage is probed the program continues through the branch on the path determined by this setting of branch storage.

The setting of branch storage is determined as follows:

(1) Arithmetic operations:
Where result R is > 0, program continues through + branch.
Where result R is = 0, program continues through 0 branch.
Where result R is < 0, program continues through - branch.

(2) Comparison operations:

Where $V_1 > V_2$, program continues through + branch.
Where $V_1 = V_2$, program continues through 0 branch.
Where $V_1 < V_2$, program continues through - branch.

Each arithmetic or comparison operation, whether internal or plugboard, sets branch storage to the condition appropriate for the result of that particular step. Branch storage is only a temporary memory location, however, since it is always cleared to the 0 condition just before the process of each instruction is initiated.

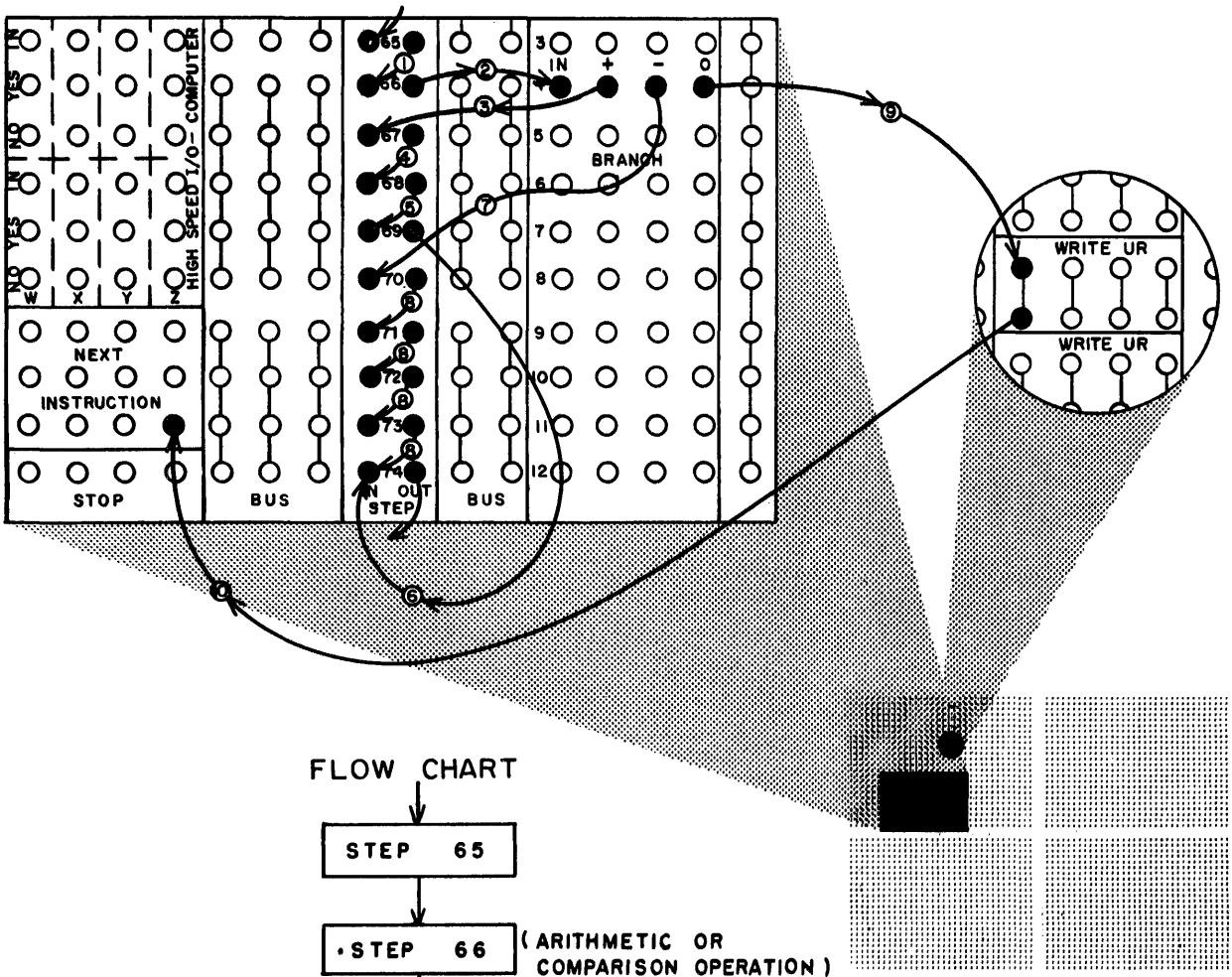
(Note: For further detail, see "Branch Storage Settings" in Chapter 7.)

The following example illustrates the use of a plugboard branch to vary the path of a program.

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
65								66
66								BR4
67								68
68								69
69								74
70								71
71								72
72								73
73								74

BRANCHING (BR)				
NO	IN FROM	+	−	0
1				
2				
3				
4	OUT STEP 66	IN STEP 67	IN STEP 70	W1

WRITE URA (W)		
NO	IN	OUT
1	OBR 4	NI
2		
3		
4		



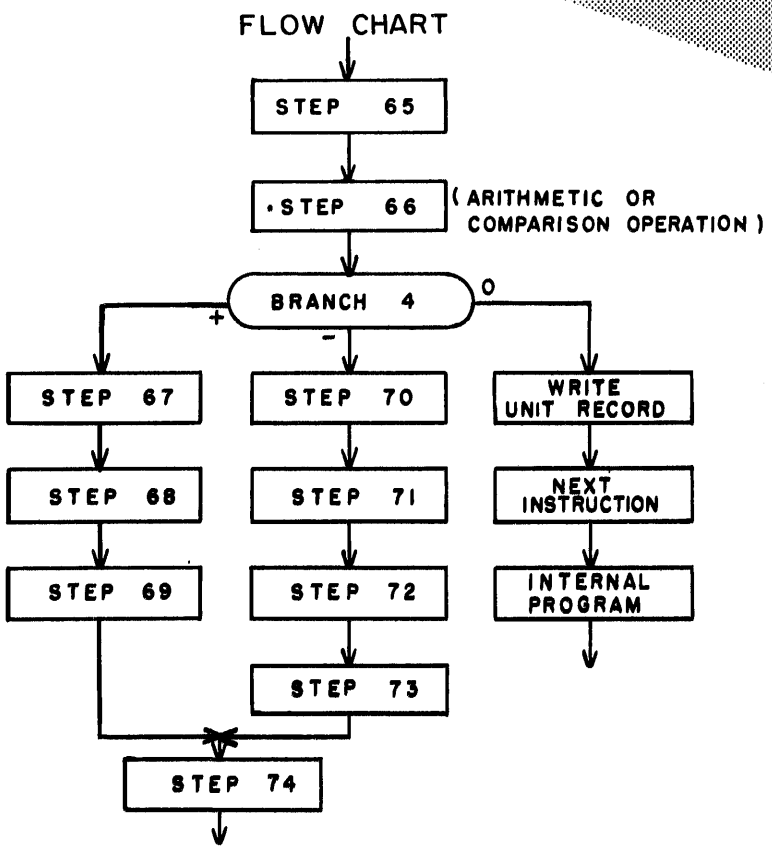


FIGURE 3-12

PLUGBOARD BRANCH WIRING

We will assume that in step 66 a value, V_1 is compared to a second value, V_2 and branch storage is set. As indicated in the accompanying flow chart:

(1) A plus (+) condition directs the program to the sequence of steps 67, 68, 69 and 74.

(2) A minus (-) condition directs the program to the sequence of steps 70, 71, 72, 73 and 74.

(3) A zero (0) condition directs the program to execute the general storage operation "WRITE UNIT RECORD", then to proceed to the NEXT INSTRUCTION hub, which directs program control to leave the plugboard and to begin execution of internally stored instruction words.

Explanation of Plugboard Branch Wiring

(1) The "OUT" of step 65 is wired to "IN" of step 66.

(2) The "OUT" of step 66 is wired to "IN" of Branch 4.

(3) "+" of Branch 4 is wired to "IN" of step 67.

(4) "OUT" of step 67 is wired to "IN"of step 68.

(5) "OUT" of step 68 is wired to "IN" of step 69.

(6) "OUT" of step 69 is wired to "IN" of step 74.

(7) "-" of Branch 4 is wired to "IN" of step 70.

(8) "OUT" of steps 70, 71, 72, and 73 are wired to "IN" of steps 71, 72, 73, and 74, respectively.

(9) "0" of Branch 4 is wired to "WRITE UR".

(10) "WRITE UR" is wired to "NEXT INSTRUCTION".

Alternate Switches

Alternate switches provide a means whereby a program sequence may be varied manually, i.e., by direct intervention of the operator, who sets the switches to the SELECT (S) or NONSELECT (NS) position.

Alternate switches are similar to selectors in two ways: the factor to be varied is usually wired to the COMMON (C) hub of the alternate switch, and the alternative courses of action are defined by the wiring of the SELECT (S) and NONSELECT (NS) hubs.

In the diagram below, the setting of alternate switch #1 determines whether the program will begin at step 57 or step 61. When the program reaches step 65, the setting of alternate switch #6 determines whether the program will continue by executing step 66 or an instruction in the internal program through the NEXT INSTRUCTION hub.

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
57								58
58								59
59								60
60								61
61								62
62								63
63								64
64								65
65								CON. ALT. SW. 6
66								67
67								68
68								69
69								70
70								71
71								72
72								73
73								74
74								75

START TO:
COM. ALT. SW. 1

ALTERNATE SWITCHES (ALT.SW.)			
NO	SELECT	COMMON	NON-SELECT
1	IN STEP 57	START	IN STEP 61
2			
3			
4			
5			
6	IN STEP 66	OUT STEP 65	N I

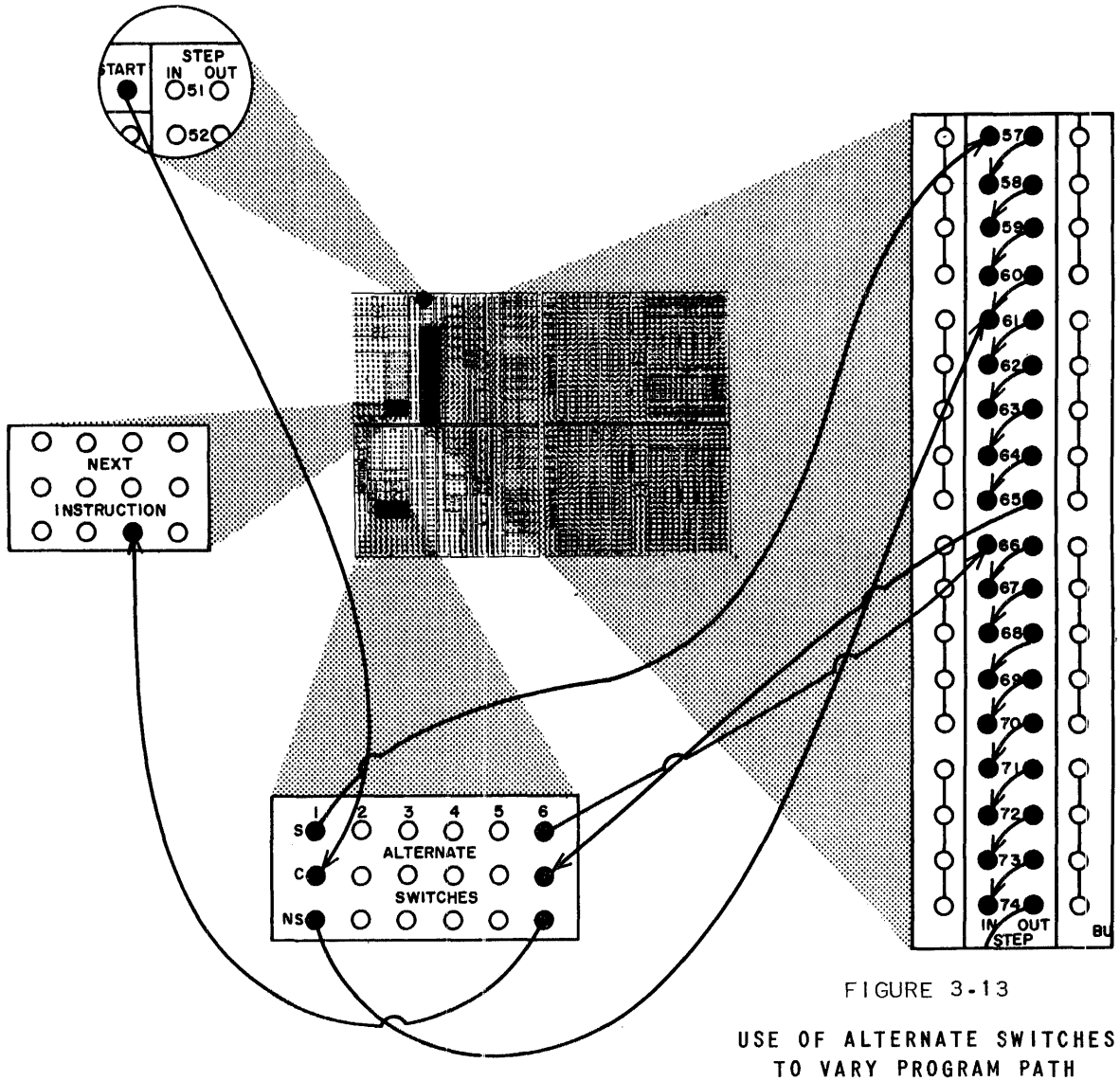


FIGURE 3-13

USE OF ALTERNATE SWITCHES
TO VARY PROGRAM PATH

Function Delay

The FUNCTION DELAY hubs provide a method of insuring that two conditions are fulfilled before the program is allowed to proceed along the path designated by the OUT of function delay.

Four groups of FUNCTION DELAY hubs, A, B, C, D, are provided, each containing an IN1, an IN2, and an OUT hub. When a predetermined condition has been met, a pulse travels by plugboard wiring to one of the IN hubs; when a second condition is met, a pulse travels to the other IN hub; and when both conditions have been met, the OUT hub emits a pulse which is wired to the plugboard hub from which the program is to continue.

Note that in the use of function delay, the predetermined conditions can be met in any sequence. That is, IN1 need not be pulsed before IN2. It is, however, imperative that *both conditions* are met, and both IN hubs are pulsed, before the OUT hub will emit the pulse required to continue the program.

When a function delay is "set" by the receipt of an IN pulse at either IN1 or IN2, it will remain set until the other IN hub receives a pulse, or until the MASTER CLEAR button on the control panel is depressed by the operator.

FUNCTION SEQUENCE (FS)			
NO	SET	PROBE	OUT
1			
2			
3			
4			

FUNCTION DELAY (FD)			
NO	IN_1	IN_2	OUT
A			
B			
C			
D			

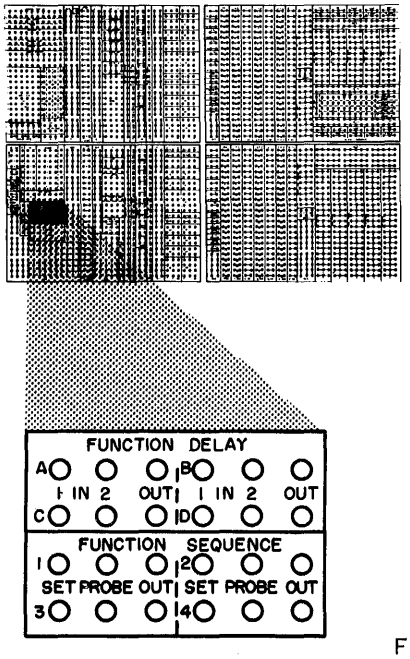


FIGURE 3-14

FUNCTION DELAY
&
FUNCTION SEQUENCE

Function Sequence

The FUNCTION SEQUENCE hubs provide a method of insuring that certain conditions are fulfilled in a predetermined sequence before the program is allowed to proceed along the path designated by the OUT hub of a function sequence.

Four groups of FUNCTION SEQUENCE hubs, numbered 1, 2, 3 and 4, are provided, each group containing a SET hub, a PROBE hub, and an OUT hub. When the first condition has been met, a pulse travels by plugboard wiring to the SET hub, and "sets" the function sequence in preparation for the probe. When the second condition is fulfilled, a pulse travels to the PROBE hub. When both conditions have been fulfilled, function sequence has been "set" and "probed", the OUT hub emits the pulse necessary to continue the program.

Note that in the use of function sequence the SET and PROBE must occur in that sequence in order for the OUT hub to be activated.

When a function sequence is set, it remains set until the probe is received, or until the MASTER CLEAR button on the control panel is depressed by the operator.

Condition Compare

When the condition compare sub-step is initiated by a wiring of the CONDITION COMPARE hub, the next compare process is modified so that where a space code and zero appear in corresponding positions in V_1 and V_2, the zero is considered greater than the space code. (Normally, a comparison of zero and space is simply ignored.)

Clear BTB to Ignores

When the CLEAR BTB TO IGNORES hub receives a pulse, an ignore code (i) is placed in each of the 120 character positions of the block transfer buffer BTB.

CONDITION COMPARE (C/C)		
NO	IN	OUT
1		
2		
3		
4		

CLEAR BLOCK TRANSFER BUFFER (CLBTB)		
NO	IN	OUT
1		
2		
3		
4		

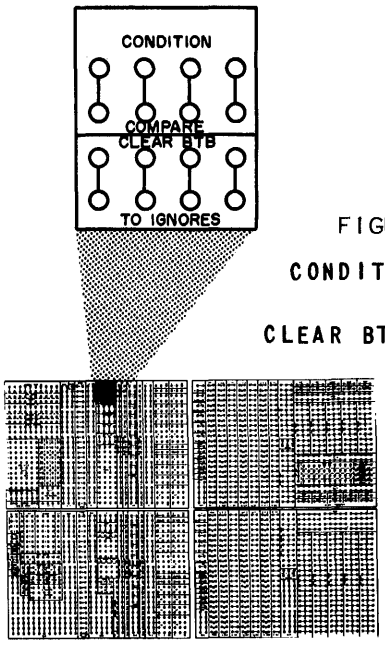


FIGURE 3-15
CONDITION COMPARE
&
CLEAR BTB TO IGNORES

Selector Hold B+

The selector hold B+ (SEL HOLD B+) hubs emit a constant B+ current as long as the computer is in operation. This current may be wired to indicators, indicator switch, or selector pick up hubs only. Once an indicator light comes on or a selector is picked up by SEL HOLD (B+), the light or selector remains in this new condition until the computer is turned off. It is possible, however, to control the B+ current by wiring a selector between the B+ source (SEL HOLD B+) and the selector or indicator that is to be used. This intervening selector acts as a switch to interrupt and control the B+ current.

Caution must be exercised in the wiring of SEL HOLD (B+), since serious damage to the computer may result from wiring a B+ hub to a destination that cannot accept this powerful current. Only those hubs listed above are capable of receiving B+ current without damage to the computer. (See Diagram Figure 3-16 for use of Selector Hold B+).

Program Indicator Lights

By applying B+ current (from a PROGRAM SELECT B+, SELECTOR HOLD B+, or LOW SPEED I/O-COMPUTER CONTROL LINES) to the appropriate INDICATOR hubs 1 - 6 on the program control plugboard, any or all of the program indicator lights on the Console and Program Control Cabinet #1 may be illuminated.

Indicator lights may be used to signal the condition (select or nonselect) of selectors, to indicate that a particular subroutine is being performed, or to convey other types of information to the operator concerning the progress of the program. (See Diagram Figure 3-16 for use of Program Indicator Lights.)

Indicator Switch

The indicator switch, mounted on the console and on control cabinet #1, is normally in a closed position, allowing current to flow through the switch. When it is depressed by the operator, the current flow is interrupted until the operator releases the switch.

Indicator switches may be used to extinguish indicator lights or to interrupt a B+ current, allowing one or more selectors to drop out to the nonselect position. (See Diagram Figure 3-16 for use of Indicator Switch.)

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
61								62
62								BR 1

BRANCHING (BR)				
NO	IN FROM	+	−	0
1	OUT STEP 62	IN STEP 63	IN STEP 63	IN PS5

PROGRAM SELECTS (PS)				
NO	IN	DELAY OUT	DROP OUT	B+
1				
2				
3				
4				
5	0 BR 1	IN ST. 63; DO. PS5	DEL. OUT PS5	T7; IND 6

INDICATORS (IND)	
NO	FROM
1	
2	
3	
4	
5	
6	SEL. T7
IND. SWITCH IN:	SEL. HOLD (BP)
IND. SWITCH OUT:	COM. T7

T #	PICK-UP FROM	GROUND	SELECT	COMMON	NON-SELECT
7	PS5; SEL. T7	CG	IND 6; PU. T7	IND. SWITCH	

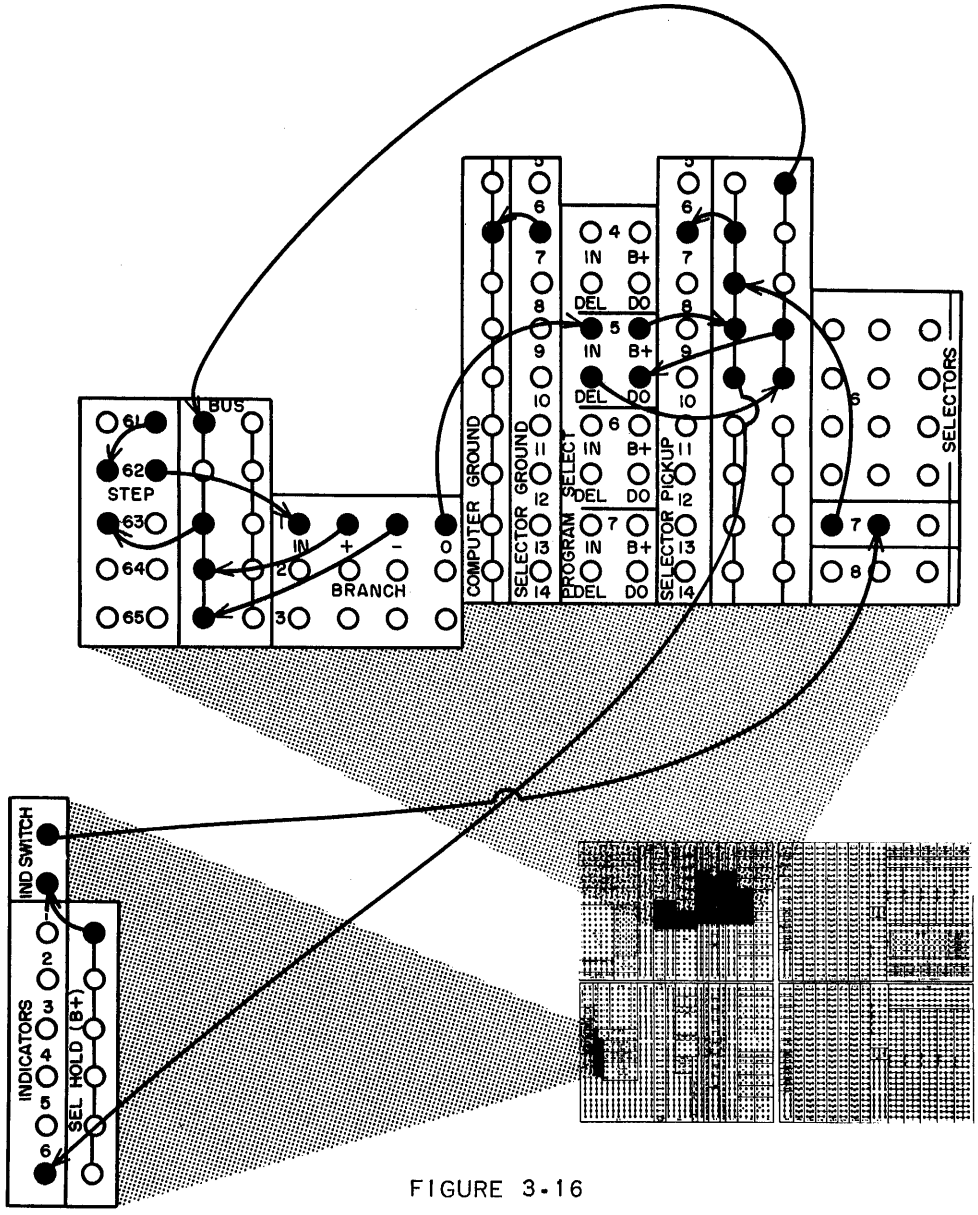


FIGURE 3-16

USE OF SELECTOR HOLD B+
INDICATORS & INDICATOR SWITCH

In the above example, program step 62 resulted in a zero (0) condition in BRANCH 1, causing SELECTOR 7 to be picked up (moved to the select position) and causing INDICATOR 6 to be illuminated. This alerted the operator to the 0 condition in BRANCH 1. By depressing the INDICATOR SWITCH on the control panel, the internal connection between the IND SWITCH hub will be broken, the indicator will be extinguished and SELECTOR 7 will drop out (to the non-select position.)

Other Plugboard Wiring Techniques Demonstrated by this Diagram.

The + and - hubs of BRANCH 1 are both wired through a BUS to the IN hub of STEP 63 to provide the pulse necessary to continue the program.

The 0 hub of BRANCH 1 is wired to the IN hub of PROGRAM SELECT 5. After a 15 millisecond delay (which allows SELECTOR 7 to be picked up), a pulse will be emitted by the delayed out DEL hub of PROGRAM SELECT #5. This pulse will travel to the bus where it is used to accomplish two functions:

(1) To drop out PROGRAM SELECT 5, and

(2) To continue the program through STEP 63.

The B+ hub of PROGRAM SELECT 5 is wired to accomplish three functions:

(1) To provide B+ current to SELECTOR PICKUP 7,

(2) To provide B+ current to the select side of SELECTOR 7 and,

(3) To light INDICATOR 6.

Bus Hubs

Bus hubs are groups of 3, 4, 5, or 6 hubs, wired together internally, which allow the programmer to wire two or more out hubs to one or more destinations. For example, when a particular storage location is referred to many times in a program, that storage may be wired to a bus hub, and the other hubs in the bus will provide access to that storage through the bus. Many sets of buses are provided on the program control plugboard. (See Diagram Figure 3-16 for use of Bus Hubs.)

Unibus Hubs

A unibus is a special type of bus which allows current to flow only in one direction. Each of the eight unibuses contains four IN hubs and one OUT hub, providing pulses from four possible sources which are directed to a common destination. Unibuses are used to insure that no back circuit (reverse current), which might adversely affect the program, can occur. (See Diagram Figure 3-17.)

Out Expander

Out expander hubs are special purpose hubs which amplify and multiply pulses where a single pulse is to be wired to several destinations. When an IN hub is pulsed, an amplified pulse is emitted by both OUT hubs.

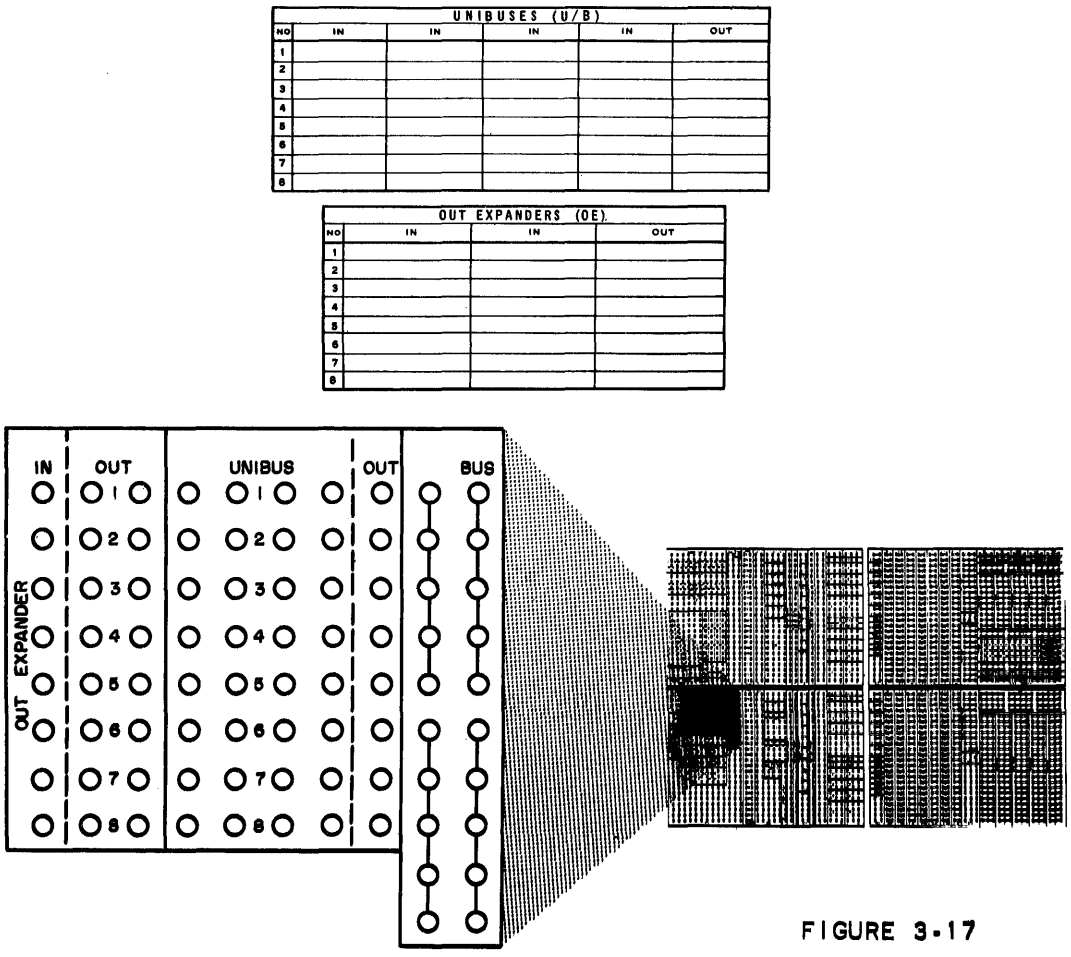


FIGURE 3-17

BUSES, UNIBUSES, & OUT EXPANDERS

Error Hubs

When any one of the following error conditions exists within UFC-I, the appropriate error hub emits a pulse and the computer "hangs up" (the program is interrupted) at the point in the program at which the error is detected.

(1) Parity error:

In order to insure complete accuracy of operation, the UFC-I performs a parity check as each character is transmitted from one register or memory location to another. If any check detects an even number of binary "ones" in any character, computer operation stops and the parity error (PAR ERROR) hub on the program control plugboard emits a pulse.

(2) Arithmetic errors:

Divide Overflow error occurs when the programmed V_1 left shift in a divide operation is so large as to cause the number of quotient digits to overflow the capacity of arithmetic register D if the division were to be performed.

Add/Subtract Overflow error occurs when the result of an add or subtract operation causes a "carry past" the highest order digit position of arithmetic registers C and D.

Normalize Overflow error occurs when a V_1 operand of zero is detected during the left normalize process.

Arithmetic Check error occurs when the result of one of the following arithmetic processes does not prove when checked: Add, Subtract, Multiply-Store Lower, Multiply-Store Upper, Divide-Store Quotient, and Divide-Store Remainder.

(3) General Storage Program errors (See Chapter 5 - General Storage System):

Any one of the following errors causes the computer operation to stop and a pulse to emitted from the GS PROG ERROR hubs:

(a) Alpha-Zone error: when any character containing zone bits other than 00 is sent to any character position of the GSAR.

(b) Inactive Drum Section error: when a drum section not included in the computer system is referred to in a general storage address.

(c) Odd Angular Address error: when the AA (two lower order digits) of the GSAR contain an odd number.

(d) Unit Record Identifiers All Ignores error: when a channel search operation is attempted using an identifier composed completely of ignore codes.

Step Repeat

When an error hub is wired to a STEP REPEAT hub, and the error hub emits a pulse, the instruction during which the error occurred is automatically repeated. If the error is eliminated after repeated execution of the step, the program continues. If not, the operator will have to intervene. This allows the central computer to continue operations even though certain *machine errors* have occurred. The parity error and check error hubs are usually wired to a step repeat hub. This insures automatic recovery from temporary or momentary machine failures.

Step Clear

The STEP CLEAR hubs can be wired to continue computer operation even though a programming error has been detected. Normally it is not practicable to continue the main program after an error condition has been detected. Instead, a subroutine, designed for pinpointing the error, should be initiated. For example, the subroutine might call for a type-out of the contents of certain registers or memory locations to enable the programmer to find the instruction in which the error occurred.

STEP CLEAR (SC)		
NO	IN	OUT
1		
2		
3		
4		

STEP REPEAT (SR)	
NO	FROM
1	
2	
3	
4	

ERROR SIGNALS	
TYPE	TO
PARITY	
÷ O'FLOW	
± O'FLOW	
N O'FLOW	
ARITH.	
GS PROG	

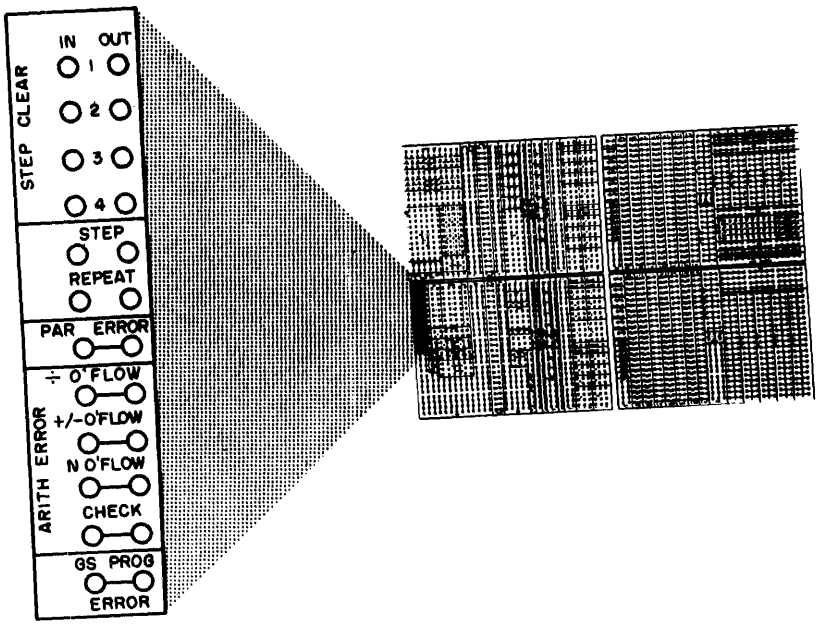


FIGURE 3-18

ERROR HUBS

Start

The START hub emits a pulse when the "Start" button is pressed after the "Master Clear" button has first been pressed.

To initiate a plugboard defined program, the START hub may be wired to the STEP IN hub of the program step at which the program is to begin.

To initiate an internally stored program, the START hub is wired to the NEXT INSTRUCTION hub. The program will begin by the execution of the instruction word stored at the location specified by the initial setting of PAK. (See Diagram Figure 3-19.)

Stop

When the STOP hub receives a pulse, computer operation stops. If a hub which is wired indirectly to the STOP hub is Y-wired through a bus to the IN hub of a step, the computer will start (when operation is resumed) at this *program step.*

If the hub which is wired to the STOP hub is not wired to a STEP IN hub, the computer will resume operations (when the START button is depressed) with the next instruction word set up in program control. (See Diagram Figure 3-19.)

NOTE: To avoid operating and debugging complications, it is always advisable to wire the STOP hub when a stop is desired.

Special Character Out

When any of the special characters Q - Y appear in the special character position of an instruction word, a pulse is emitted by the corresponding SPECIAL CHARACTER OUT hub on the plugboard. This pulse may be used to initiate an operation such as Condition Compare or Clear BTB to Ignores, which is not directly available to the internally stored program. Special Character Out must not be wired to initiate a program step, however, since the internally stored program is not interrupted when a special character out pulse is received. (See Diagram Figure 3-19.)

Console B+

The Console B+ (CNSL B+) hub emits a B+ current when the Console-Normal switch on the typewriter control panel is set to "console". Thus B+ current will normally be used to pick up a selector. The Console B+ current flows during all console operations of the typewriter. (See Diagram Figure 3-19.)

START TO:

BREAKPOINTS (B/P)	
1	
2	
3	

SPECIAL CHARACTER OUTS (SCO)					
NO	TO	NO	TO	NO	TO
Q		T		W	
R		U		X	
S		V		Y	

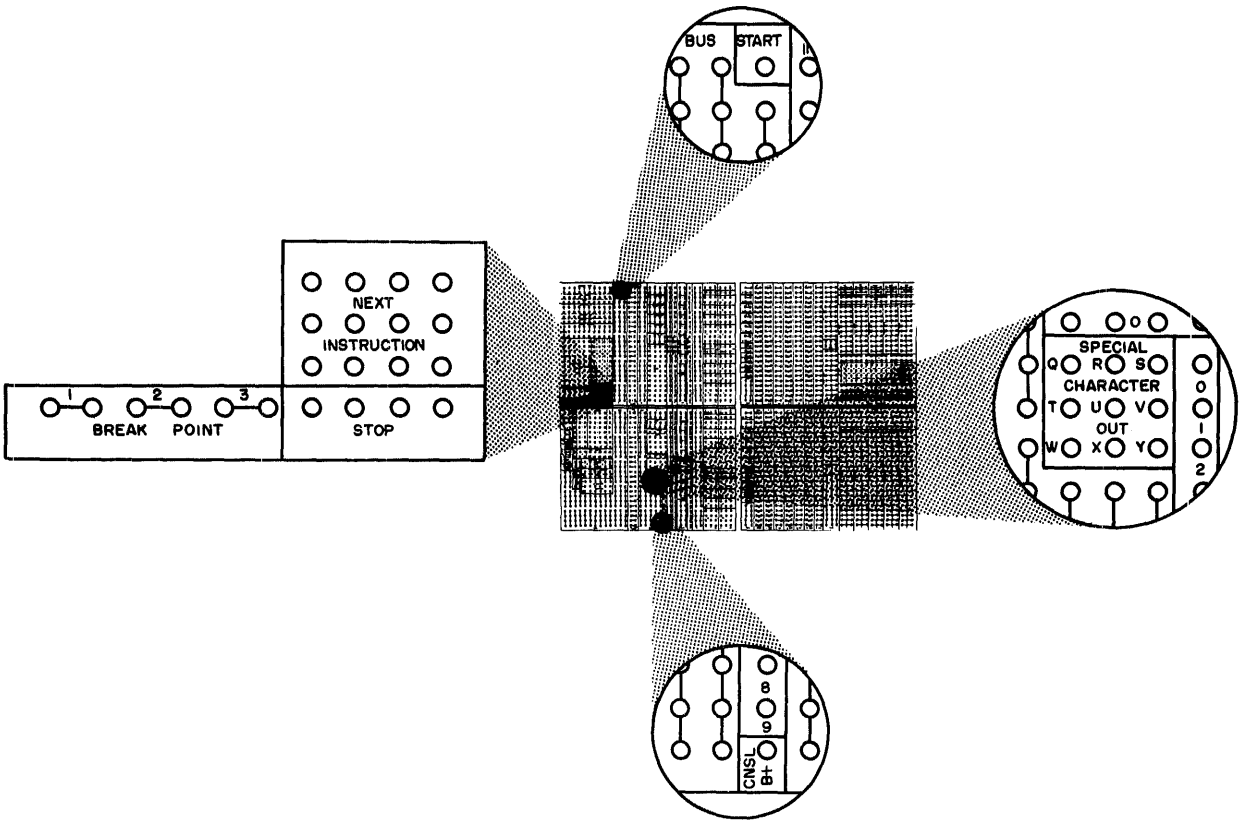


FIGURE 3-19

START, STOP, SPECIAL CHARACTER OUT,
NEXT INSTRUCTION, BREAKPOINT, CNSL B+ HUBS

COMPONENTS USED TO EFFECT COMBINATION CONTROL

Transfer from Internal to External Program Control

A. Via Transcop Instruction Word

When program control executes a transcop instruction word, it interrupts the internally stored program and begins execution of the plugboard defined program step designated by the process (PR) section of the transcop instruction.

B. Via Breakpoint

When an instruction word containing a breakpoint special character is executed and the breakpoint selector is set in order to allow a pulse to be emitted from the corresponding breakpoint hub on the program control plugboard, the next operation in the program depends on the wiring of the BREAKPOINT hub.

(1) If the BREAKPOINT hub is wired to a STEP IN hub, the execution of the internally stored program is interrupted and execution of program steps begins.

(2) If the BREAKPOINT hub is wired to a STOP hub, the next instruction word is set up in program control before the computer stops. Operation can be resumed from that point in the internally stored program by the operator's use of the START button. (See Figure 3-19.)

C. Via ERROR-STEP CLEAR Wiring

When an error condition is detected, the corresponding ERROR hub on the plugboard emits a pulse. When this ERROR hub is wired to a STEP CLEAR IN hub, program control is cleared of the current instruction. If the STEP CLEAR OUT hub is wired to a STEP IN hub, a plugboard-defined error analysis subroutine is initiated. Portions of this subroutine might be internally stored, in which case the NEXT INSTRUCTION hub could be wired, transferring control to the internally stored program.

Transfer from External to Internal Program Control

A. Via NEXT INSTRUCTION

When a NEXT INSTRUCTION hub on the program control plugboard receives a pulse, the plugboard defined program is interrupted and program control is transferred to the internally stored program. (See Figure 3-19.)

B. Via STEP OUT-STOP Wiring

When the OUT of a program step is wired directly to the STOP hub and a pulse is received by the STOP hub, the computer stops with the next instruction word set up in program control. When the START button is pressed, the computer resumes operation by executing internally stored instructions rather than plugboard defined program steps. (See Figure 3-19.)

chapter 4

REPERTORY OF INSTRUCTIONS

INTRODUCTION

The repertory of instructions performed by the UFC-I consists of 27 operations, plus the subinstructions of substeps which extend or modify basic operations.

In this chapter, instructions are discussed under five headings:

(1) Arithmetic and Logical instructions

(2) Jump instructions

(3) Special Purpose instructions

(4) Input/Output instructions

(5) Transfer of Control instruction

Arithmetic and Logical Instructions

*AD	Add	SU	Substitute U
*SB	Subtract	SV	Substitute V
*MU	Multiply, Store Upper	SW	Substitute W
*ML	Multiply, Store Lower	*SZ	Suppress Left Zeros
*DQ	Divide, Store Quotient	*LN	Left Normalize
*DR	Divide, Store Remainder	*AT	Arithmetic Transfer
*CP	Compare	*MK	Mask Transfer

The arithmetic and logical instructions are discussed as a group because 1) the programming techniques are similar; 2) the arithmetic registers, A, B, C, and D, are used during all of these operations; 3) the sources, destinations, and programmed shifts applicable to all of them are similar; and 4) the computer uses the same basic circuitry for the execution of all of these instructions.

All of the above operations may be programmed internally; those prefixed by an (*) also have plugboard hubs available for external programming. When any of the Add, Subtract, Multiply or Divide operations are executed, the results may be checked for accuracy. When any of these operations are performed as part of the execution of an instruction word, the result will be checked automatically by a reverse arithmetic process, unless the check is suppressed by the initiation of a "check suppress" subinstruction.

When any of these operations is defined in a program step, the decision to check or not check is put into effect by wiring of the process (PROC) hub to either the check (C) or no check (NC) hub of the appropriate operation.

Jump Instructions

UJ	Unconditional Jump	JN	Jump on Negative
		JP	Jump on Positive
		JZ	Jump on Zero

Jump instructions of the UFC-I consist of one unconditional jump, UJ, and three conditional jumps, JN, JP, and JZ.

All of these jump instructions are "returnable jumps"; that is, during the execution of the jump instruction itself, a means is provided to return at some later point in the program to any instruction in the program.

* Operations that have plugboard hubs available for external programming.

Special Purpose Instructions

*BT	Buffer Transfer	*CC	Channel Clear
LA	Load General Storage Address Register	SP	Search Probe
LS	Load Shift Revolver		

All of the above special purpose instructions may be programmed internally; those prefixed by an asterisk (*) are also available through plugboard wiring.

Input/Output Instructions

*TI	Test Incoming Control
*TD	Test Demand In
*DE	Demand

Input/Output instructions provide the means of controlling the input/output units of the UFC-I system through the integration of the above instructions and the appropriate groupings of plugboard hubs (See Chapter 6).

Transfer of Control Instruction

TC Transcop (Transfer Control to Plugboard)

The Transcop instruction is the instruction most often used to transfer control of the program from the internally stored program to the plugboard.

In the detailed description of each of the instructions in the repertory, storages referred to during internal control are identified as U, V, and W; storages referred to by plugboard wiring are identified by V_1, V_2, and R.

When a subinstruction is referred to, it defines the extension or modification initiated by a character in the S/C position of an instruction word. A substep refers to a comparable operation initiated by the wiring of the STEP OUT hub to a hub which initiates this action on the plugboard.

For internal programming, the next instruction to be executed is taken from the address specified by the PAK, except when control is transferred to a program step designated by a transcop instruction.

For external programming, the next instruction to be executed is determined by the wiring of the appropriate STEP IN plugboard hub, except in the case

* Plugboard hubs are available for external programming.

of wiring a NEXT INSTRUCTION hub, when control is transferred to the internal program.

Shifts referred to in this repertory are shown in detail in Table 3-2, Chapter 3.

Special character codes and the subinstructions which they initiate are defined in Table 4-2.

A summary of the initial and final contents of the arithmetic registers RA, RB, RC, and RD in relation to the instructions which use those registers are shown in Table 4-3.

INSTRUCTION DEFINITIONS

Arithmetic Instructions

Add (AD): Add to the contents of the U (or V_1) address, the contents of the V (or V_2) address and store the sum at the W (or R) address.

Subtract (SB): Subtract from the contents of the U (or V_1) address the contents of the V (or V_2) address and store the remainder at the W (or R) address.

Multiply, Store Upper (MU): Multiply the contents of the U (or V_1) address by the V (or V_2) address and store the higher order product digits at the W (or R) address.

Multiply, Store Lower (ML): Multiply the contents of the U (or V_1) address by the contents of the V (or V_2) address and store the lower order product digits at the W (or R) address.

Divide, Store Quotient (DQ): Divide the contents of the U (or V_1) address by the contents of the V (or V_2) address and store the quotient at the W (or R) address.

Divide, Store Remainder (DR): Divide the contents of the U (or V_1) address by the contents of the V (or V_2) address and store the remainder at the W (or R) address.

Logical Instructions

Compare (CP): Compare the contents of the U (or V_1) address with the contents of the V (or V_2) address -

If: $\left\{\begin{matrix} U > V \\ V_1 > V_2 \end{matrix}\right\}$ branch storage is set to the plus (+) condition.

$\left\{\begin{matrix} U < V \\ V_1 < V_2 \end{matrix}\right\}$ branch storage is set to the minus (-) condition.

$\left\{\begin{matrix} U = V \\ V_1 = V_2 \end{matrix}\right\}$ branch storage is set to the zero (0) condition.

Substitute U (SU): Substitute the contents of the U section of the V address for the contents of the U section of the U address and store the result at the W address.

Substitute V (SV): Substitute the contents of the V section of the V address for the contents of the V section of the U address and store the result at the W address.

Substitute W (SW): Substitute the contents of the W section of the V address for the contents of the W section of the U address and store the result at the W address.

Suppress Left Zeros (SZ): Replace by space codes (Δ) all zeros to the left of the most significant digit in the contents of the U (or V_1) address and store the result at the W (or R) address. Ignore the V (or V_2) address.

Left Normalize (LN): Shift the contents of the U (or V_1) address to the left in Register A until the first significant digit is in the high order position of RA. Store the normalized word at the W (or R) address. Ignore the V (or V_2) address. The normalizing shift count is formed in Register B.

Arithmetic Transfer (AT): Transfer the contents of the U (or V_1) address to the W (or R) address via Register D. Ignore the V (or V_2) address.

Mask Transfer (MK): Mask the contents of the U (or V_1) address with the contents of the V (or V_2) address and transfer the result to the W (or R) address. A zero or a space code (Δ) in the mask causes the corresponding U (or V_1) character to be transferred to the W (or R) address; if the mask contains any other character, an ignore code will be transferred to the W (or R) address in the appropriate character position.

Jump Instructions

Unconditional Jumps (UJ): Jump unconditionally to the instruction word specified by the W address of the current instruction. Transfer the U section of the current instruction word to the W section of the word at the V address.

Jump on Negative (JN): If conditional storage is minus (-), jump to the instruction word specified by the W address of the current instruction. Transfer the U section of the current instruction word to the W section of the word at the V address.

If conditional storage is not minus, take the next instruction from the address specified by PAK.

Jump on Plus (JP): If conditional storage is plus (+), jump to the instruction word specified by the W address of the current instruction. Transfer the U section of the current instruction word to the W section of the word at the V address.

If conditional storage is not plus, take the next instruction from the address specified by PAK.

Jump on Zero (JZ): If conditional storage is zero (0), jump to the instruction word specified by the W address of the current instruction. Transfer the U section of the current instruction word to the W section of the word at the V address.

If conditional storage is not zero, take the next instruction from the address specified by PAK.

Special Purpose Instructions

Buffer Transfer (BT): Transfer the contents of the U (or V_1) address to the W (or R) address via the block transfer buffer. Ignore the V (or V_2) address.

Load General Storage Address Register (LA): Load the general storage address register (GSAR) with the two lowest order digits of U, the three digits of V, and the two higher order digits of W.

Load Shift Revolver (LS): Load the shift revolver (SRV) with the contents of the current instruction revolver (IRV_C).

Channel Clear (CC): Clear the track or buffer specified by the W (or R) address to space codes (Δ). Ignore the U (or V_1) and V (or V_2) addresses.

Search Probe (SP): Channel search storage is probed to determine whether a previously initiated channel search operation is completed.

Input/Output Instructions

Test Demand In (TD): Test the designated I/O unit to determine whether it is ready or not ready to receive instructions.

Demand In (DE): Place the designated I/O unit on demand, track switch if specified, and exchange computer ⇔ I/O control information.

Test Incoming Control (TI): Test high speed I/O - computer control line storage for incoming control information; track switch if specified.

Transfer of Control Instruction

Transcop: Transfer control to the plugboard step designated by the process section of this instruction word.

ANALYSIS OF INSTRUCTIONS

Arithmetic and Logical Instructions

Add

Operation Code: 14S
Mnemonic Code: AD
Descriptive Code: (U) + (V) → W
Description:

Contents of U (or V_1) address are loaded into RA; shift performed if required.

Contents of V (or V_2) address are loaded into RB; shift performed if required.

Contents of RA and RB are added, forming the sum in RD; check performed if indicated.

Sum in RD is shifted if required and stored at the W (or R) address.

Sub-instruction or sub-step is initiated.

Subtract

Operation Code: 22S
Mnemonic Code: SB
Descriptive Code: (U) - (V) → W
Description:

Contents of U (or V_1) address are loaded into RA; shift performed if required.

Contents of V (or V_2) address are loaded into RB; shift performed if required.

Contents of RB are subtracted from the contents of RA, forming the difference in RD; check performed if indicated.

Difference in RD is shifted if required, and stored at the W (or R) address.

Sub-instruction or sub-step is initiated.

Add - Check; Add - No Check Subtract - Check; Subtract - No Check		
OPERATION SEQUENCE	**INSTRUCTION WORD**	**PROGRAM STEP**
In Register A, place the contents of the location specified by:	U-address	V_1 ADDRESS wiring
Shift the contents of Register A in accordance with:	Contents of the U-section of the Shift Revolver	V_1 SHIFT wiring
In Register B, place the contents of the location specified by:	V-address	V_2 ADDRESS wiring
Shift the contents of Register B in accordance with:	Contents of the V-section of the Shift Revolver	V_2 SHIFT wiring
Add (Subtract) the contents of Register B to (from) the contents of Register A, forming the sum (difference) separately in both Registers C and D.	PR = AD (SB)	PROCESS to: + C or + NC (- C or - NC) wiring
Determine that checking is not to be suppressed in this instruction, and check the addition (subtraction). (Use sum (difference) in Register C for the check operation.)	S/C is not one of the following: B, C, D, E, F, G, H, or I	
- - - - - - - - - or - - - - - - - - -	- - - - - - or - - - - - -	- - - - - - - - -
Determine that checking is to be suppressed in this instruction.	S/C is one of the following: B, C, D, E, F, G, H, or I	
Shift the contents of Register D in accordance with:	Contents of the W-section of the Shift Revolver	R SHIFT wiring
Store the final contents of Register D at the location specified by:	W-address	R ADDRESS wiring
Initiate sub-instruction(s) or sub-step(s) in accordance with:	S/C *	STEP OUT wiring

* If S/C = F, the only sub-instruction (suppress check) specified by this instruction has been completed prior to this time. If S/C = B, C, D, F, G, H, or I, the other sub-instruction or instructions specified by these values are initiated at this time.

Add - Check; Add - No Check
Subtract - Check; Subtract - No Check

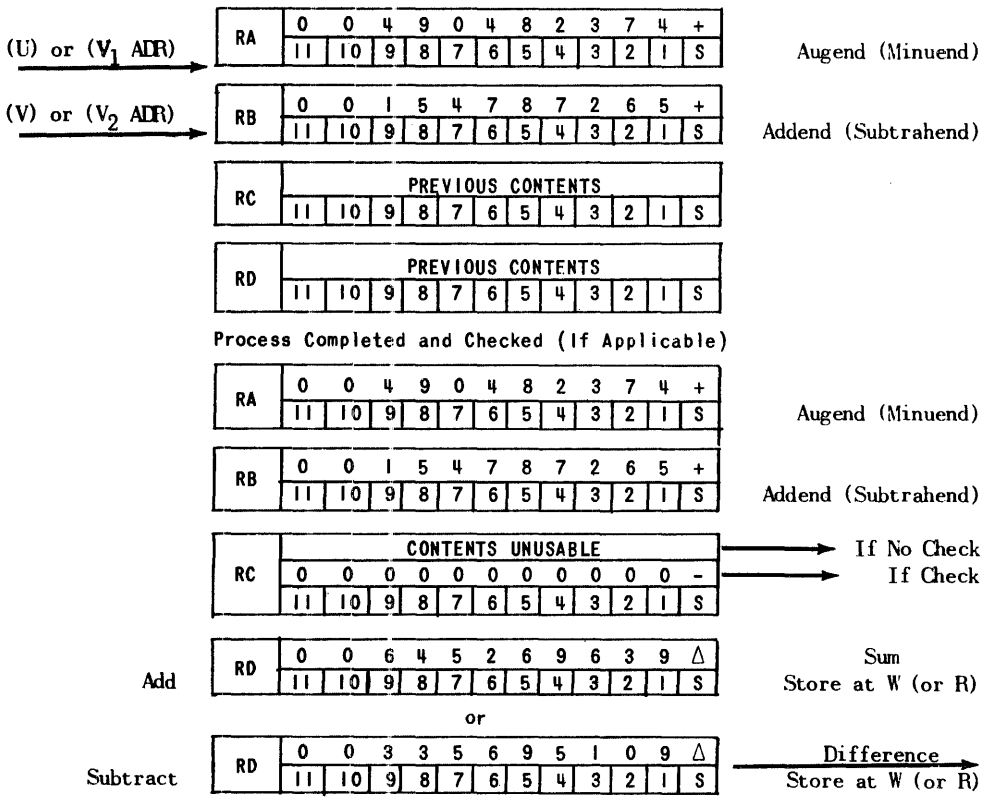
PROGRAMMED SHIFTS

Quantities that can be shifted	Arithmetic Registers Involved	Types of Shift Permitted		
		Left Shift	Right Shift	Right End Around Shift
V_1	V_1 in Register A	yes	yes	yes
V_2	V_2 in Register B	yes	yes	yes
R	R in Register D	yes	yes	yes

(See Table 4-1 for permissible sources of operands, destinations for results and applicable data transmission rules.)

CONTENTS OF ARITHMETIC REGISTERS

Operands Loaded:



Multiply, Store Upper

Operation Code: 44S
Mnemonic Code: MU
Descriptive Code: (U) X (V) = (RC) → W
Description:

Contents of U (or V_1) address are loaded into RA; shift performed if required.

Contents of V (or V_2) address are loaded into RB; shift performed if required.

Contents of RA are multiplied by the contents of RB, forming the product in RC and RD.

Product in RC and RD is shifted, if required*, and contents of RC are stored at the W (or R) address.

Sub-instruction or sub-step is initiated, including check if indicated.

Multiply, Store Lower

Operation Code: 43S
Mnemonic Code: ML
Descriptive Code: (U) X (V) = (RD) ⟶ W
Description:

Contents of U (or V_1) address are loaded into RA; shift performed if required.

Contents of V (or V_2) address are loaded into RB; shift performed if required.

Contents of RA are multiplied by the contents of RB, forming the product in RC and RD.

Product in RC and RD is shifted, if required*, and contents of RD are stored at the W (or R) address.

Sub-instruction or sub-step is initiated, including check if indicated.

* Note special shift conditions in Table 3-2, Chapter 3.

Multiply, Store Lower - Check; Multiply, Store Lower - No Check Multiply, Store Upper - Check; Multiply, Store Upper - No Check		
OPERATION SEQUENCE	**INSTRUCTION WORD**	**PROGRAM STEP**
In Register A, place the contents of the location specified by:	U-address	V_1 ADDRESS wiring
Shift the contents of Register A, in accordance with:	Contents of the U-section of the Shift Revolver	V_1 SHIFT wiring
In Register B, place the contents of the location specified by:	V-address	V_2 ADDRESS wiring
Shift the contents of Register B, in accordance with:	Contents of the V-section of the Shift Revolver	V_2 SHIFT wiring
Multiply the contents of Register A by the contents of Register B. Determine the sign of the product in the usual algebraic manner. Form a 22 character product as follows: Form the sign of the product and the 11 lower order digits of the product in Register D. Form the sign of the product and the 11 higher order digits of the product in Register C.	PR = ML (MU)	PROCESS to: XSL C or XSL NC (XSU C or XSU NC)
Determine that checking is <u>not</u> to be suppressed in this instruction.	S/C is not one of the following: B, C, D, E, F, G, H, or I	
- - - - - - - - - - or - - - - - - - - - -	- - - - - - - or - - - - - -	- - - - - - - - - -
Determine that checking <u>is</u> to be suppressed in this instruction.	S/C is one of the following: B, C, D, E, F, G, H, or I	
If applicable*, shift the contents of Registers C or D in accordance with:	Contents of the W-section of the Shift Revolver	R SHIFT wiring
Store the contents of Register C or D at the location specified by:	W-address	R ADDRESS wiring
If applicable, check multiplication. Initiate other sub-instruction(s) or sub-step(s) in accordance with:	S/C	STEP OUT wiring

* Applicable only in no-check operations.

Multiply, Store Upper - Check; Multiply, Store Upper - No Check
Multiply, Store Lower - Check; Multiply, Store Lower - No Check

PROGRAMMED SHIFTS

Quantities that can be shifted	Arithmetic Registers Involved	Types of Shift Permitted		
		Left Shift	Right Shift	Right End Around Shift
V_1	V_1 in Register A	yes	yes	yes
V_2	V_2 in Register B	yes	yes	yes
Check -	(No result shift permitted)	-	-	-
No Check R	R in both Register C & D	yes	yes	yes

(See Table 4-1 for permissible sources of operands, destinations for results, and applicable data transmission rules.)

Contents of Arithmetic Registers

Operands Loaded:

(U) or (V_1 ADR) →

	11	10	9	8	7	6	5	4	3	2	1	S	
RA	0	0	9	4	7	8	9	2	3	4	7	+	Multiplicand

(V) or (V_2 ADR) →

	11	10	9	8	7	6	5	4	3	2	1	S	
RB	0	0	5	7	2	8	2	3	0	4	1	+	Multiplier

	11	10	9	8	7	6	5	4	3	2	1	S
RC	Previous Contents											
RD	Previous Contents											

Process Completed:

	11	10	9	8	7	6	5	4	3	2	1	S	
RA	0	0	9	4	7	8	9	2	3	4	7	+	Multiplicand
RB	0	0	5	7	2	8	2	3	0	4	1	+	Multiplier
RC	0	0	0	0	5	4	2	9	7	4	5	Δ	Product High Order Digits → If Store Upper, Store at W (or R) address
RD	7	6	7	4	9	1	6	7	2	2	7	Δ	Product Low Order Digits → If Store Lower, Store at W (or R) address

Process Completed - Check Completed:

	11	10	9	8	7	6	5	4	3	2	1	S	
RA	0	0	9	4	7	8	9	2	3	4	7	+	Multiplicand
RB	0	0	5	7	2	8	2	3	0	4	1	+	Multiplier
RC	0	0	0	0	0	0	0	0	0	0	0	-	
RD	0	0	0	0	0	0	0	0	0	0	0	-	

Divide, Store Quotient

Operation Code: 48S
Mnemonic Code: DQ
Descriptive Code: (U) ÷ (V) = (RD)⟶W
Description:

Contents of U (or V_1) address are loaded into RA and RC; shift performed if required.

Contents of V (or V_2) address are loaded into RB; shift performed if required.

Contents of RA are divided by the contents of RB, forming the remainder in RC and the quotient in RD.

Quotient in RD is shifted if required*, and contents of RD are stored at the W (or R) address.

Sub-instruction or sub-step is initiated, including check if indicated.

Divide, Store Remainder

Operation Code: 49S
Mnemonic Code: DR
Descriptive Code: (U) ÷ (V) = (RC)⟶W
Description:

Contents of U (or V_1) address are loaded into RA and RC; shift performed if required.

Contents of V (or V_2) address are loaded into RB; shift performed if required.

Contents of RA are divided by contents of RB, forming the remainder in RC and the quotient in RD.

Remainder in RC shifted, if required*, and contents of RC stored at the W (or R) address.

Sub-instruction or sub-step initiated, including check if indicated.

* Note special shift conditions in Table 3-2, Chapter 3.

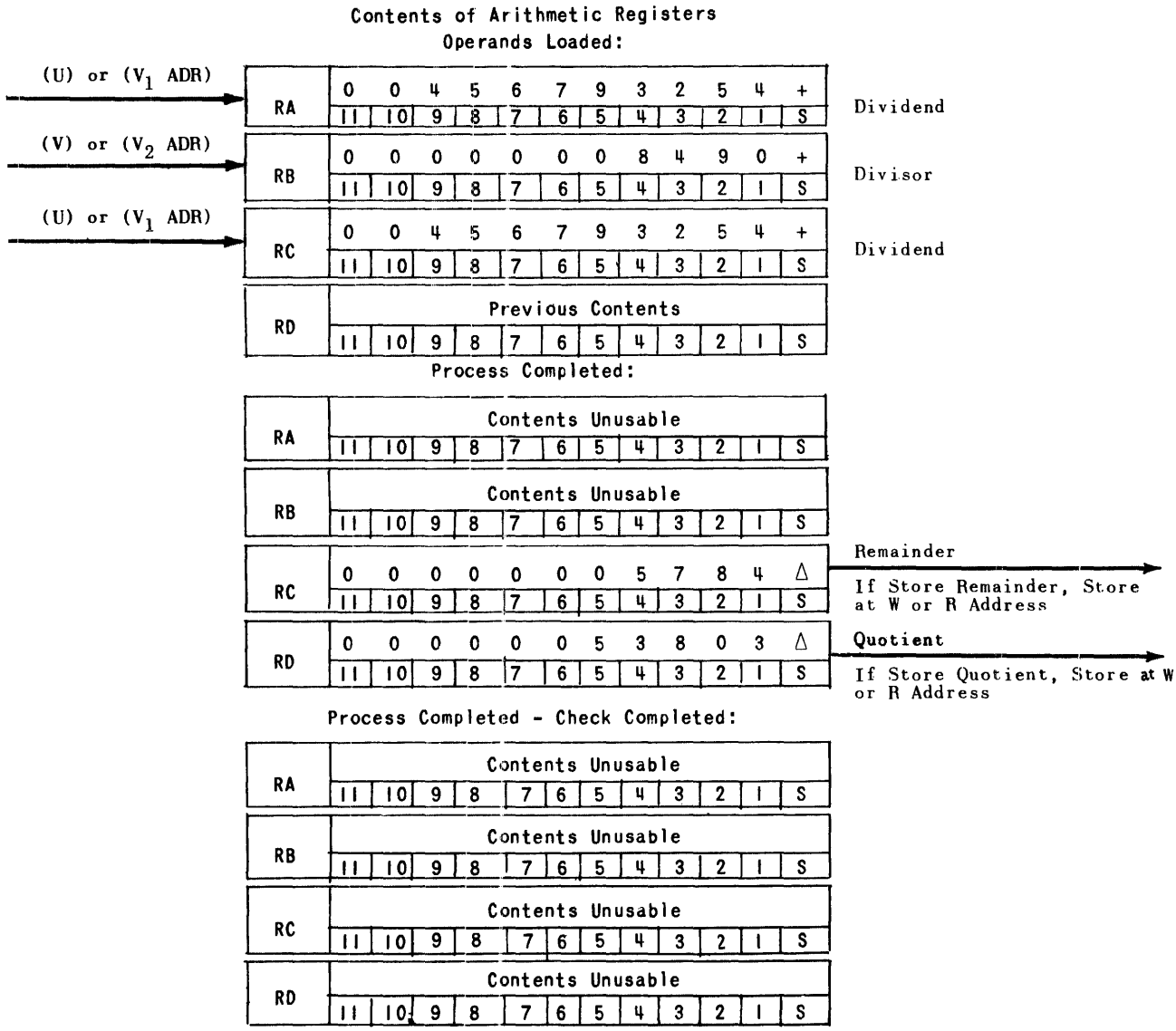
Divide, Store Quotient - Check; Divide, Store Quotient - No Check Divide, Store Remainder -Check; Divide, Store Remainder -No Check		
OPERATION SEQUENCE	**INSTRUCTION WORD**	**PROGRAM STEP**
In Registers A and C, place the contents of the location specified by:	U-address	V_1 ADDRESS wiring
Shift the contents of Registers A and C in accordance with:	Contents of the U-section of the Shift Revolver	V_1 SHIFT wiring
In Register B, place the contents of the location specified by:	V-address	V_2 ADDRESS wiring
Shift the contents of Register B in accordance with:	Contents of the V-section of the Shift Revolver	V_2 SHIFT wiring
Divide the contents of Register A by the contents of Register B. Determine the sign of the quotient in the usual algebraic manner. Form a quotient and remainder as follows: Form up to a 12 character quotient in Register D. (11 characters and the correct sign) Form up to a 12 character remainder in Register C. (Sign of remainder same as sign of dividend)	PR = DQ (DR)	PROCESS to: + SQC or + SQNC (+ SRC or + SRNC) wiring
Determine that checking is <u>not</u> to be suppressed in the instruction.	S/C is not one of the following: B, C, D, E, F, G, H, or I	
---------- or ----------	---------- or ----------	----------
Determine that checking <u>is</u> to be suppressed in the instruction.	S/C is one of the following: B, C, D, E, F, G, H, or I	
If applicable,* shift the contents of Register C or D in accordance with:	Contents of the W-section of the Shift Revolver	R SHIFT wiring
Store the contents of Register C or D at the location specified by:	W-address	R ADDRESS wiring
Initiate sub-instruction(s) or sub-step(s) in accordance with:	S/C	STEP OUT wiring

* Applicable only in no-check operations.

Divide, Store Quotient - Check; Divide, Store Quotient - No Check
Divide, Store Remainder - Check; Divide, Store Remainder - No Check
PROGRAMMED SHIFTS

Quantities that can be shifted	Arithmetic Registers Involved	Types of Shift Permitted		
		Left Shift	Right Shift	Right End Around Shift
V_1	V_1 in Registers A & C	yes	yes	yes
V_2	V_2 in Register B	yes	yes	yes
Check -	(No result shift permitted)	-	-	-
No Check R	R in Register C or D	yes	yes	yes

(See Table 4-1 for permissible sources of operands, destinations for results, and applicable data transmission rules.)



Compare

Operation Code: 37S
Mnemonic Code: CP
Descriptive Code: (U) : (V) Set Branch Storage
Description:

Contents of U (or V_1) address are loaded into RA; shift performed if required.

Contents of V (or V_2) address are loaded into RB; shift performed if required.

Contents of RA are compared with the contents of RB:

If (U) > (V) Branch storage is set to the plus (+) condition.
(V_1) > (V_2)

If (U) < (V) Branch storage is set to the minus (-) condition.
(V_1) < (V_2)

If (U) = (V) Branch storage is set to the zero (0) condition.
(V_1) = (V_2)

Sub-instruction or sub-step is initiated

OPERATION SEQUENCE	INSTRUCTION WORD	PROGRAM STEP
In Register A, place the contents of the location specified by:	U-address	V_1 ADDRESS wiring
Shift the contents of Register A in accordance with:	Contents of the U-section of the Shift Revolver	V_1 SHIFT wiring
In Register B, place the contents of the location specified by:	V-address	V_2 ADDRESS wiring
Shift the contents of Register B in accordance with:	Contents of the U-section of the Shift Revolver	V_2 SHIFT wiring
Left end around shift Register A and B. Compare (on a bit by bit basis, highest order character first) the word V_1 held in Register A with the word V_2 held in Register B to determine the relative magnitude of these two words: If $V_1 > V_2$, set Branch storage to + If $V_1 = V_2$, set Branch storage to 0 If $V_1 < V_2$, set Branch storage to -	PR = CP	PROCESS to COMP wiring
Initiate sub-instruction(s) or sub-step(s) in accordance with:	S/C	STEP OUT wiring

Compare

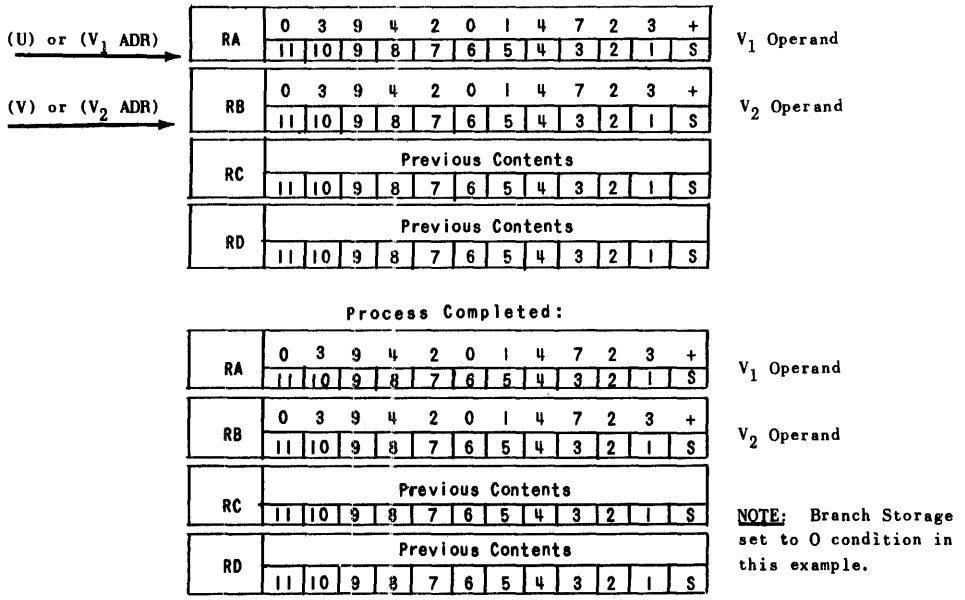
PROGRAMMED SHIFTS

Quantities that can be shifted	Arithmetic Registers Involved	Types of Shift Permitted		
		Left Shift	Right Shift	Right End Around Shift
V_1	V_1 in Register A	yes	yes	yes
V_2	V_2 in Register B	yes	yes	yes

(See Table 4-1 for permissible sources of operands, destinations for results, and applicable data transmission rules.)

Contents of Arithmetic Registers

Operands Loaded:



Process Completed:

NOTE: Branch Storage set to 0 condition in this example.

Substitute U

Operation Code: 24S
Mnemonic Code: SU
Descriptive Code: Sub U
Description:

Contents of the U address are loaded into RA; shift performed if required.

Contents of the V address are loaded into RB; shift performed if required.

Contents of the U section of RB are substituted for the contents of the U section of RA; modified contents of RA are sent to RC and RD.

Contents of RD are shifted, if required, and stored at the W address.

Sub-instruction is initiated.

Substitute V

Operation Code: 25S
Mnemonic Code: SV
Descriptive Code: Sub V
Description:

Contents of the U address are loaded into RA; shift performed if required.

Contents of the V address are loaded into RB; shift performed if required.

Contents of the V section of RB are substituted for the contents of the V section of RA, modified contents of RA are sent to RC and RD.

Contents of RD are shifted, if required, and stored at the W address.

Sub-instruction is initiated.

Substitute W

Operation Code: 26S
Mnemonic Code: SW
Descriptive Code: Sub W
Description:

Contents of the U address are loaded into RA; shift performed if required.

Contents of the V address are loaded into RB; shift performed if required.

Contents of the W section of RB substituted for the W section of RA; modified contents of RA are sent to RC and RD.

Contents of RD shifted, if required, and stored at the W address.

Sub-instruction is initiated.

Substitute U Substitute V Substitute W	
OPERATION SEQUENCE	**INSTRUCTION WORD**
In Register A, place the contents of the location specified by:	U-address of this instruction word
Shift the contents of Register A in accordance with:	Contents of the U section of the Shift Revolver
In Register B, place the contents of the location specified by:	V-address of this instruction word
Shift the contents of Register B in accordance with:	Contents of the V section of the Shift Revolver
Right-end-around shift the contents of Registers A and B; send a modified copy of the contents of Register A to Registers C and D as follows:	
Substitute U Instruction Replace the contents of the U section of Register A by the contents of the U section of Register B, when forming the modified copy of the contents of Register A in Registers C and D.	PR = SU
Substitute V Instruction. Replace the contents of the V section of Register A by the contents of the V section of Register B, when forming the modified copy of the contents of Register A in Registers C and D.	PR = SV
Substitute W Instruction Replace the contents of the W section of Register A by the contents of the W section of Register B, when forming the modified copy of the contents of Register A in Registers C and D.	PR = SW
Shift the contents of Register D in accordance with:	Contents of the W section of the Shift Revolver
Store the final contents of Register D at the location specified by:	W-address of this instruction word
Initiate sub-instruction(s) in accordance with:	S/C

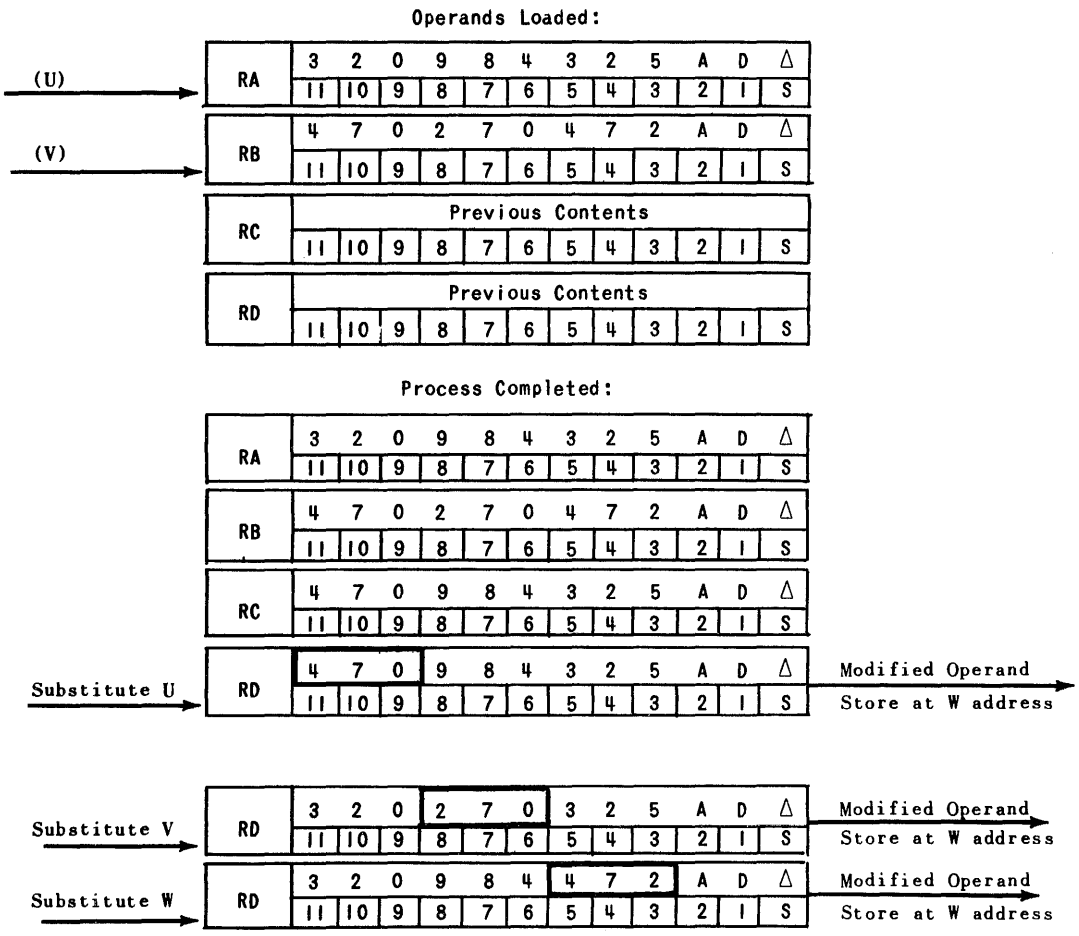
Substitute U
Substitute V
Substitute W

PROGRAMMED SHIFTS

Quantities that can be shifted	Arithmetic Registers Involved	Types of Shift Permitted		
		Left Shift	Right Shift	Right End Around Shift
U	U in Register A	yes	yes	yes
V	V in Register B	yes	yes	yes
W	W in Register D	yes	yes	yes

(See Table 4-1 for permissible sources of operands, destinations for results, and applicable data transmission rules.)

Contents of Arithmetic Registers



Suppress Left Zeros

Operation Code: 29S
Mnemonic Code: SZ
Descriptive Code: (U) Less Zeros → W
Description:

Contents of U (or V_1) address are loaded into RA; shift performed if required.

Space codes (Δ) replace all zeros to the left of the most significant digit of RA, and result is transferred to RD.

Result shifted if required and contents of RD stored at the W (or R) address.

Sub-instruction or sub-step is initiated.

OPERATION SEQUENCE	INSTRUCTION WORD	PROGRAM STEP
In Register A, place the contents of the location specified by:	U-address	V_1 ADDRESS wiring
Shift the contents of Register A in accordance with:	Contents of the U-section of the Shift Revolver	V_1 SHIFT wiring
Left Shift the contents of Register A until the most significant character of the word held in Register A is located in Register A's most significant character position: Keep track of the number of shifts required to do this, and then right shift the contents of Register A the same number of places, substituting space codes (Δ) in Register A's higher-order character positions. Transmit the final contents of Register A to Register D.	PR = SZ	PROCESS to SLZ wiring
Shift the contents of Register D in accordance with:	Contents of the W-section of the Shift Revolver	R SHIFT wiring
Store the final contents of Register D at the location specified by:	W-address	R ADDRESS wiring
Initiate sub-instruction(s) or sub-step(s) in accordance with:	S/C	STEP OUT wiring

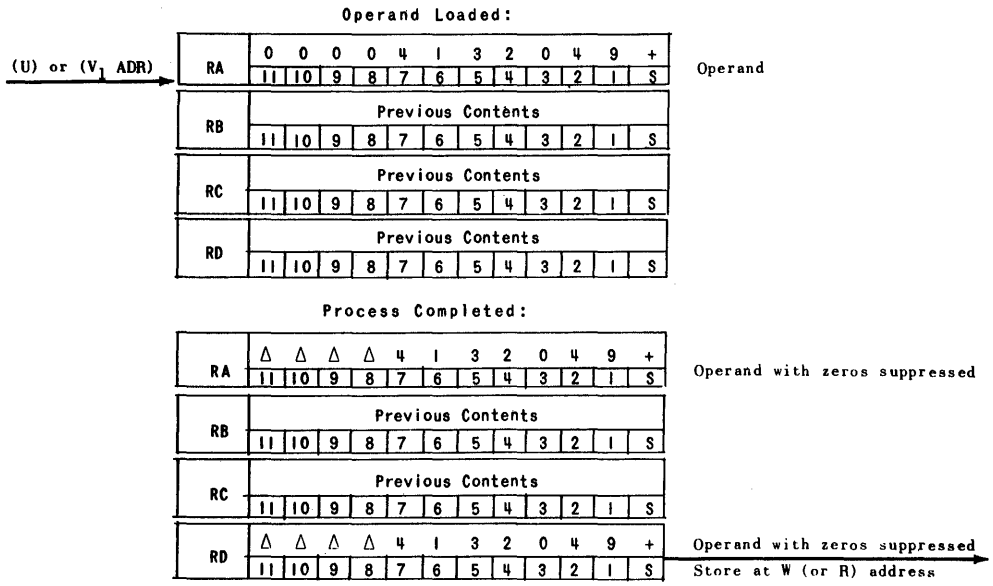
Suppress Left Zeros

PROGRAMMED SHIFTS

Quantities that can be shifted	Arithmetic Registers Involved	Types of Shift Permitted		
		Left Shift	Right Shift	Right End Around Shift
V_1	V_1 in Register A	yes	yes	yes
R	R in Register D	yes	yes	yes

(See Table 4-1 for permissible sources of operands, destinations for results, and applicable data transmission rules.)

Contents of Arithmetic Registers



Left Normalize

Operation Code: 35S
Mnemonic Code: LN
Descriptive Code: Normalize (U) ⟶ W; RB = Shift Count
Description:

Contents of the U (or V_1) address are loaded into RA; shift performed if required.

Contents of RA are shifted to the left until the first significant digit is in the high order position of RA, forming the normalizing shift count in register B.

Normalized word transferred to RD; shift performed if required, and contents of RD stored at the W (or R) address.

Sub-instruction or sub-step is initiated.

OPERATION SEQUENCE	INSTRUCTION WORD	PROGRAM STEP
In Register A, place the contents of the location specified by:	U-address	V_1 ADDRESS wiring
Shift the contents of Register A in accordance with:	Contents of the U-section of the Shift Revolver	V_1 SHIFT wiring
Left shift the contents of Register A until the most significant character of the word held in Register A is in Register A's most significant character position. Count the number of shifts required to do this. Place this count in Register B; and send the (normalized) contents of Register A to Register D.	PR = LN	PROCESS to NORM wiring
Shift the contents of Register D in accordance with:	Contents of the W-section of the Shift Revolver	R SHIFT wiring
Store the final contents of Register D at the location specified by:	W-address	R ADDRESS wiring
Initiate sub-instruction(s) or sub-step(s) in accordance with:	S/C	STEP OUT wiring

Left Normalize

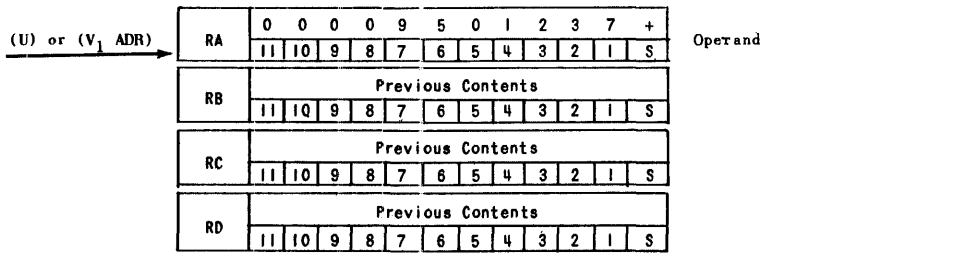
PROGRAMMED SHIFTS

Quantities that can be shifted	Arithmetic Registers Involved	Types of Shift Permitted		
		Left Shift	Right Shift	Right End Around Shift
V_1	V_1 in Register A	yes	yes	yes
R	R in Register D	yes	yes	yes

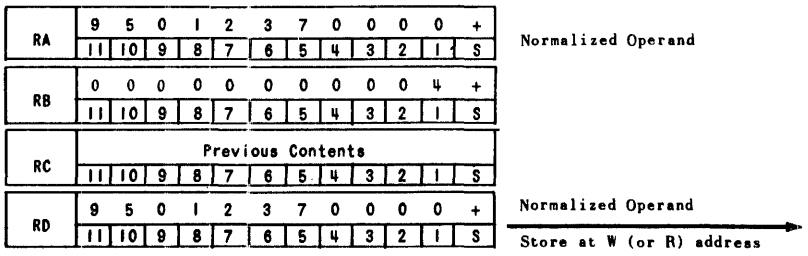
(See Table 4-1 for permissible sources of operands, destinations for results, and applicable data transmission rules.)

Contents of Arithmetic Registers

Operand Loaded:



Process Completed:



Arithmetic Transfer

Operation Code: 13S
Mnemonic Code: AT
Descriptive Code: (U)⟶RD⟶W
Description:

Contents of U (or V_1) address are loaded into RD; shift performed if required.

Contents of RD are shifted again if required (V_2 shift time).

Contents of RD are shifted a third time, if required (R shift time), and result stored at the W (or R) address.

Sub-instruction or sub-step is initiated.

OPERATION SEQUENCE	INSTRUCTION WORD	PROGRAM STEP
Transfer data (1 to 12 characters) from one storage location to another via Register D as follows:	PR = AT	PROCESS to AT wiring
In Register D, place the contents of the location specified by:	U-address	V_1 ADDRESS wiring
Shift the contents of Register D in accordance with:	Contents of the U-section of the Shift Revolver	V_1 SHIFT wiring
Shift the contents of Register D in accordance with:	Contents of the V-section of the Shift Revolver	V_2 SHIFT wiring
Shift the contents of Register D in accordance with:	Contents of the W-section of the Shift Revolver	R SHIFT wiring
Store the contents of Register D at the location specified by:	W-address	R ADDRESS wiring
Initiate sub-instruction(s) or sub-step(s) in accordance with:	S/C	STEP OUT wiring

Arithmetic Transfer

PROGRAMMED SHIFTS

Quantities that can be shifted	Arithmetic Registers Involved	Types of Shift Permitted		
		Left Shift	Right Shift	Right End Around Shift
V_1 Source Data held in Register D	RD in V_1 Shift Time	yes	yes	yes
	RD in V_2 Shift Time	yes	yes	yes
	RD in R Shift Time	yes	yes	yes

(See Table 4-1 for permissible sources of operands, destinations for results, and applicable data transmission rules.)

Contents of Arithmetic Registers

Operand Loaded:

		11	10	9	8	7	6	5	4	3	2	1	S
RA	Previous Contents												
RB	Previous Contents												
RC	Previous Contents												
RD		2	0	0	3	0	0	4	0	0	M	L	Δ
		11	10	9	8	7	6	5	4	3	2	1	S

(U) or (V_1 ADR) →

Process Completed:

		11	10	9	8	7	6	5	4	3	2	1	S
RA	Previous Contents												
RB	Previous Contents												
RC	Previous Contents												
RD		2	0	0	3	0	0	4	0	0	M	L	Δ
		11	10	9	8	7	6	5	4	3	2	1	S

Result → Store at W (or R) address

Mask Transfer

Operation Code: 42S
Mnemonic Code: MK
Descriptive Code: Mask (U) with (V) → W
Description:

Contents of U (or V_1) are loaded into RA; shift performed if required.

Contents of V (or V_2) are loaded into RB; shift performed if required.

Contents of RA masked with the contents of RB; the character in RA is transferred to RC and RD if the corresponding RB character is a zero or space code (Δ), otherwise an ignore code (i) is transferred to RC and RD. Result is shifted in RD, if required, and contents of RD stored at the W (or R) address.

Sub-instruction or sub-step is initiated.

OPERATION SEQUENCE	INSTRUCTION WORD	PROGRAM STEP
In Register A, place the contents of the location specified by:	U-address	V_1 ADDRESS wiring
Shift the contents of Register A in accordance with:	Contents of the U-section of the Shift Revolver	V_1 SHIFT wiring
In Register B, place the contents of the location specified by:	V-address	V_2 ADDRESS wiring
Shift the contents of Register B in accordance with:	Contents of the V-section of the Shift Revolver	V_2 SHIFT wiring
Right-end-around shift Registers A and B. Beginning with the sign position, examine each of the 12 characters of the word (V_2) held in Register B. If the (V_2) character is a zero, transmit the corresponding V_1 character to a correspondingly significant character position in Registers C and D.	PR = MK	PROCESS to MASK T wiring
---------- or ----------	---------- or ----------	----------
If the (V_2) character is not zero, transmit an Ignore code (i) to the correspondingly significant character position in Registers C and D.		
Shift the contents of Register D in accordance with:	Contents of the W-section of the Shift Revolver	R SHIFT wiring
Store the final contents of Register D at the location specified by:	W-address	R ADDRESS wiring
Initiate sub-instruction(s) or sub-step(s) in accordance with:	S/C	STEP OUT wiring

Mask Transfer

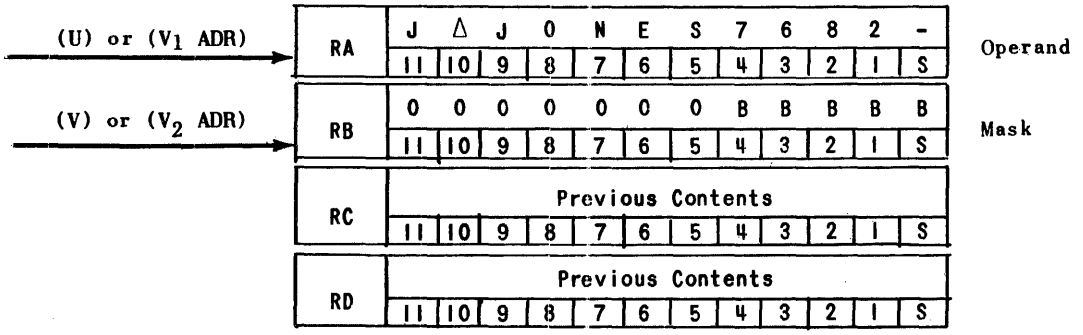
PROGRAMMED SHIFTS

Quantities that can be shifted	Arithmetic Registers Involved	Types of Shift Permitted		
		Left Shift	Right Shift	Right End Around Shift
V_1	V_1 in Register A	yes	yes	yes
V_2	V_2 in Register B	yes	yes	yes
R	R in Register D	yes	yes	yes

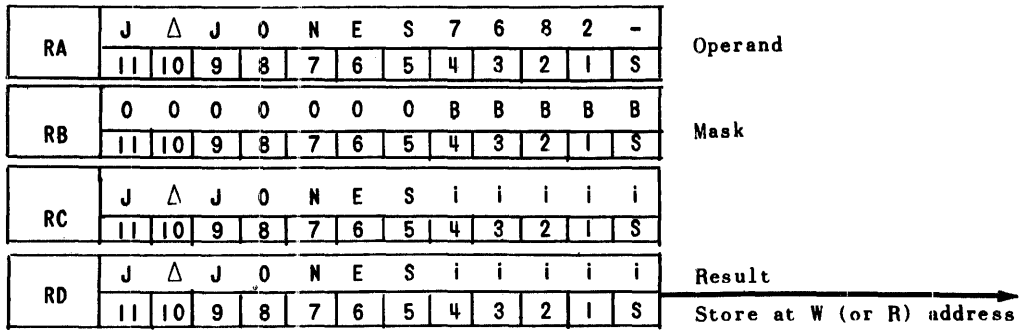
(See Table 4-1 for permissible sources of operands, destinations for results, and applicable data transmission rules.)

Contents of Arithmetic Registers

Operands Loaded:



Process Completed:



PERMISSABLE SOURCES AND APPLICABLE DATA TRANSMISSION RULES

Source	Source Address Specifies	Rules
High Speed Drum Tracks (except ISP) Block Transfer Buffer General Storage Buffer	Any word location 0-9 (12 characters) or Any field location A-V (excluding I and O) containing from 1 up to 12 characters. (If more than 12 characters are specified, only the lower order 12 of the field are obtained.)	1 or 3 or 4
Arithmetic Registers A, B, C, D	Exactly 12 characters	3 or 5
Instruction Revolver		3
General Storage Address Register	A space(Δ) and the 7 characters in the General Storage Address Register.	2
Program Address Counter	A space(Δ) and the 3 characters in the Program Address Counter.	2
Code Distributor Register	A space(Δ) and the 1 character in the Code Distributor Register.	2

PERMISSABLE DESTINATIONS AND APPLICABLE PROGRAMMED DATA TRANSMISSION RULES

Destination	Destination Address Specifies	Rules
High Speed Drum Tracks (except ISP) Block Transfer Buffer General Storage Buffer	Any word location 0-9 (12 characters) or Any field location A-V (excluding I and O) from 1 up to 119 characters (Z or 120 character blockettes not permitted).	6 or 8 or 9
Arithmetic Registers A, B, C, D	Exactly 12 characters	8 or 10
Instruction Revolver		8
Shift Revolver		8
General Storage Address Register	Sign and 7 characters (sign is automatically shifted off as General Storage Address Register is loaded).	7
Program Address Counter	Sign and 3 characters (sign is automatically shifted off as Program Address Counter is loaded).	7
Code Distributor Register	Sign and 1 character (sign is automatically shifted off as Code Distributor Register is loaded).	7

TABLE 4-1

Data Transmission Rules

Arithmetic Transfer	Divide, Store Quotient	Substitute W
Add	Divide, Store Remainder	Suppress Left Zeros
Subtract	Mask Transfer	Left Normalize
Multiply, Store Upper	Substitute U	Compare
Multiply, Store Lower	Substitute V	

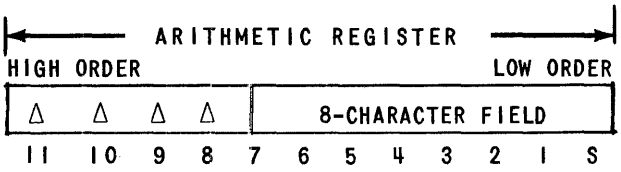
Introduction

Storage locations that may be addressed to obtain operands (called "Permissible Sources") and storage locations that may be addressed as destinations for the storage of results (called "Permissible Destinations") are given in the tables on the preceding page. If any storage locations other than those listed in the tables are specified in any of the instructions listed above, the program is automatically stopped.

The source (or sources) and the destination specified in an instruction need not contain the same number of characters.

Source Rules

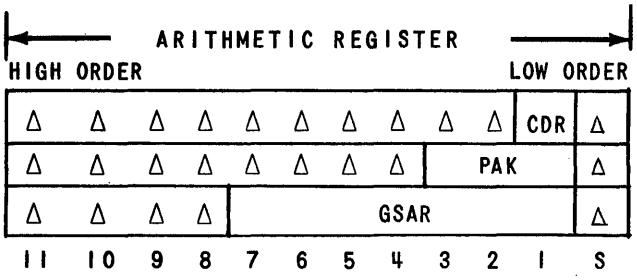
(1) If the source address specifies less than 12 characters and the block transfer buffer, general storage buffer, or high speed drum is involved, a copy of the contents of the source address is loaded into the lower order character positions of the arithmetic register, and the higher order character positions of the arithmetic register (those not receiving source data) are filled with space codes (Δ).



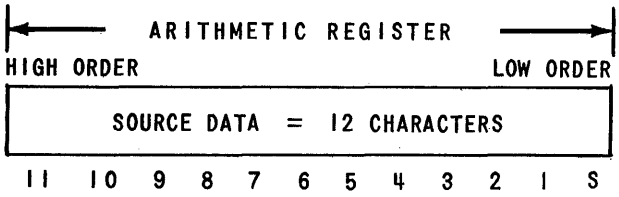
(2) If the source address specifies GSAR, PAK or CDR, a space code (Δ)* is automatically sent to the arithmetic register as the first (or low order) character of the transfer; a copy of the contents of GSAR, PAK or CDR is then sent to the other lower order character

* A space code in the sign position is treated by the computer as a plus in arithmetic operations.

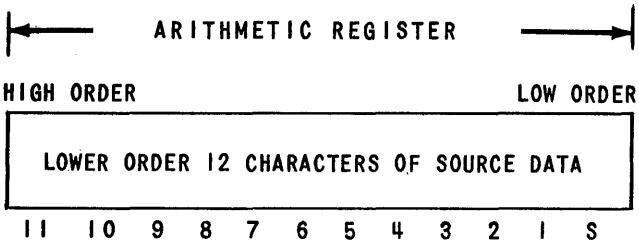
positions of the arithmetic register, and the higher order character positions of the arithmetic register are filled with space codes (Δ).



(3) If the source address specifies 12 characters, the arithmetic register is loaded with an exact copy of the contents of the source address.



(4) If the source address specifies more than 12 characters, only the 12 lower order characters of the source address are loaded into the arithmetic register. Higher order characters of the source data are not sent to the arithmetic register.

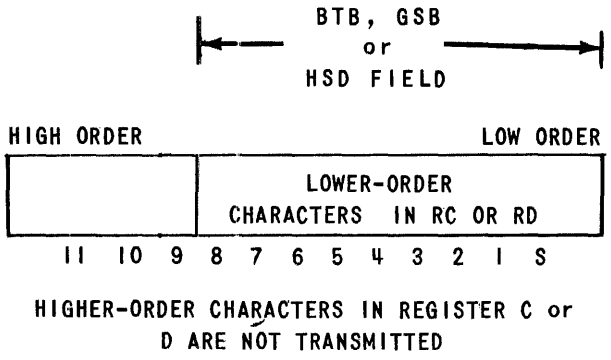


(5) Where the same arithmetic register that is to receive the source data is also referred to as the source, that register performs a right-end-around shift of 12 characters.

Destination Rules

(6) If the destination address specifies less than 12 characters, and a BTB, GSB or HSD field is involved, data is transferred out of

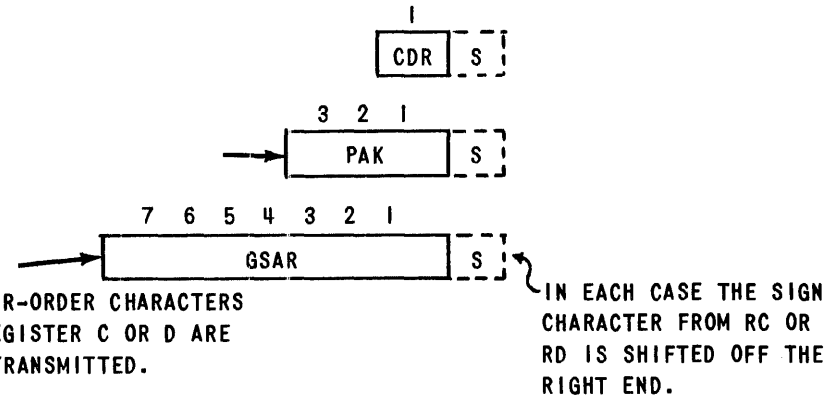
the arithmetic register C or D until the number of characters specified by the destination has been transmitted; the transfer operation then terminates.



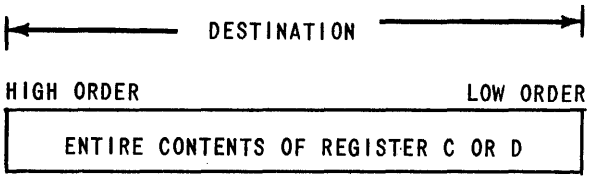
HIGHER-ORDER CHARACTERS IN REGISTER C or D ARE NOT TRANSMITTED

Note: The data transmitted is shifted out the low-order end of RC or RD, sent to the destination, and also re-entered into RC or RD at the high order end. Although the transfer of data to a destination terminates when the capacity of the destination is filled, the recirculation (or end around shift) of RC or RD does not. Twelve characters are always shifted out of RC or RD, and they are left in the same status as at the beginning of the transfer.

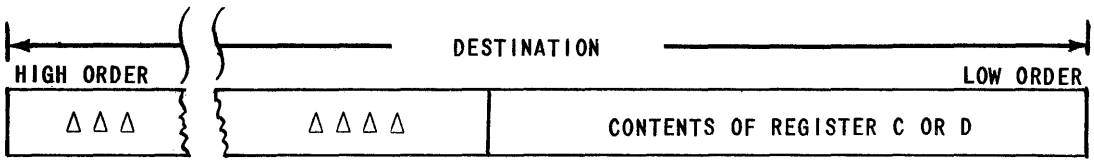
(7) If the destination address specifies GSAR, PAK or CDR, the sign character in register C or D and a number of characters equal to capacity of GSAR, PAK, or CDR are transferred out of register C or D, and shifted into these registers from the high order end. When the last character transmitted (the high order character) is shifted into these registers, the sign character is shifted off the low order end. (Note following Rule 6 applies).



(8) If the destination address specifies exactly 12 characters, a copy of the entire contents of register C or D is stored at the destination address.



(9) If the destination address specifies more than 12 characters, a copy of the contents of register C or D is stored in the 12 lower order character positions of the destination and the higher order character positions in the destination are filled with space codes.



(10) Where Register D is referred to as a destination, a right-end-around shift of 12 characters occurs, and the transfer terminates.

Jump Instructions

Unconditional Jump

Operation Code: 41S
Mnemonic Code: UJ
Descriptive Code: U Jump
Description:

Address in the U section of the instruction word is transferred to the W section of the word at the address given in the V section of the instruction word.

Address given in the W section of the instruction word is transferred to PAK and is used as the address of the next instruction.

Sub-instruction is initiated.

OPERATION SEQUENCE	INSTRUCTION WORD
Modify the contents of PAK and cause a jump to occur as follows:	PR = UJ
Transfer the three characters which form:	U-section of this instruction word
to the W-section of the word location specified by:	V-address of this instruction word
Transfer the three characters which form:	W-section of this instruction word
to PAK, and take the next instruction word from the location specified by the modified contents of PAK.	
Initiate sub-instruction(s) in accordance with:	S/C

Jump on Plus

Operation Code: 17S
Mnemonic Code: JP
Descriptive Code: Jump +
Description:

Conditional storage is examined for plus state:

If conditional storage is plus (+), the address in the U section of the instruction word is transferred to the W section of the word at the address given in the V section of the instruction word. The address given in the W section of the instruction word is transferred to PAK and used as the address of the next instruction.

If conditional storage is not plus (either - or 0), the U, V, and W sections of the instruction word are ignored and the next instruction is taken from the address specified by PAK.

Sub-instruction is initiated.

OPERATION SEQUENCE	INSTRUCTION WORD
Examine Conditional Storage for plus (+).	PR = JP
If Conditional Storage is plus:	
Transfer the three characters which form:	U-section of this instruction word
to the W-section of the word location specified by:	V-section of this instruction word
Transfer: to PAK, and use the modified contents of PAK as the address of the next instruction.	W-section of this instruction word
If Conditional Storage is not plus (+): Ignore the U, V, and W sections of this instruction word and take the next instruction word from the location specified by the unaltered contents of PAK.	
Initiate sub-instruction(s) in accordance with:	S/C

Jump on Negative

Operation Code: 15S
Mnemonic Code: JN
Descriptive Code: Jump -
Description:

Conditional storage is examined for minus state:

If conditional storage is minus (-), the address in the U section of the instruction word is transferred to the W section of the word at the address given in the V section of the instruction word. The address given in the W section of the instruction word is transferred to PAK and used as the address of the next instruction.

If conditional storage is not minus, (either + or 0), the U, V, and W sections of the instruction word are ignored, and the next instruction is taken from the address specified by PAK.

Sub-instruction is initiated.

OPERATION SEQUENCE	INSTRUCTION WORD
Examine Conditional Storage for negative (-).	PR = JN
If Conditional Storage is negative:	
Transfer the three characters which form:	U-section of this instruction word
to the W-section of the word location specified by:	V-section of this instruction word
Transfer:	W-section of this instruction word
to PAK, and use the modified contents of PAK as the address of the next instruction.	
If Conditional Storage is not negative (-): Ignore the U, V, W sections of this instruction word and take the next instruction word from the location specified by the unaltered contents of PAK.	
Initiate sub-instruction(s) in accordance with:	S/C

Jump on Zero

Operation Code: 19S
Mnemonic Code: JZ
Descriptive Code: Jump 0
Description:

Conditional storage is examined for zero state:

If conditional storage is zero, the address in the U section of the instruction word is transferred to the W section of the word at the ad-

dress given in the V section of the instruction word. The address given in the W section of the instruction word is transferred to PAK, and used as the address of the next instruction.

If conditional storage is not zero, (either + or -), the U, V, and W sections of the instruction word are ignored, and the next instruction is taken from the address specified by PAK.

Sub-instruction is initiated.

OPERATION SEQUENCE	INSTRUCTION WORD
Examine Conditional Storage for zero (0).	PR = JZ
If Conditional Storage is zero:	
Transfer the three characters which form:	U-section of this instruction word
To the W-section of the word location specified by:	V-section of this instruction word
Transfer:	W-section of this instruction word
To PAK, and use the modified contents of PAK as the address of the next instruction.	
If Conditional Storage is not zero (0): Ignore the U, V, and W sections of this instruction word and take the next instruction word from the location specified by the unaltered contents of PAK.	
Initiate sub-instruction(s) in accordance with:	S/C

Use of Jump Instructions

Jump instructions provide a means of modifying the sequence of operations in an internal program. The kind of decisions which are reflected through plugboard wiring of selectors, branches, etc. may be made internally through the use of jump instructions.

Unconditional and Conditional Jumps

Unconditional Jump UJ is the only unconditional jump instruction in the UFC-I repertory. When this instruction is executed, the jump always occurs; that is, the next instruction is always taken from the word address specified in the W section of the unconditional jump instruction word.

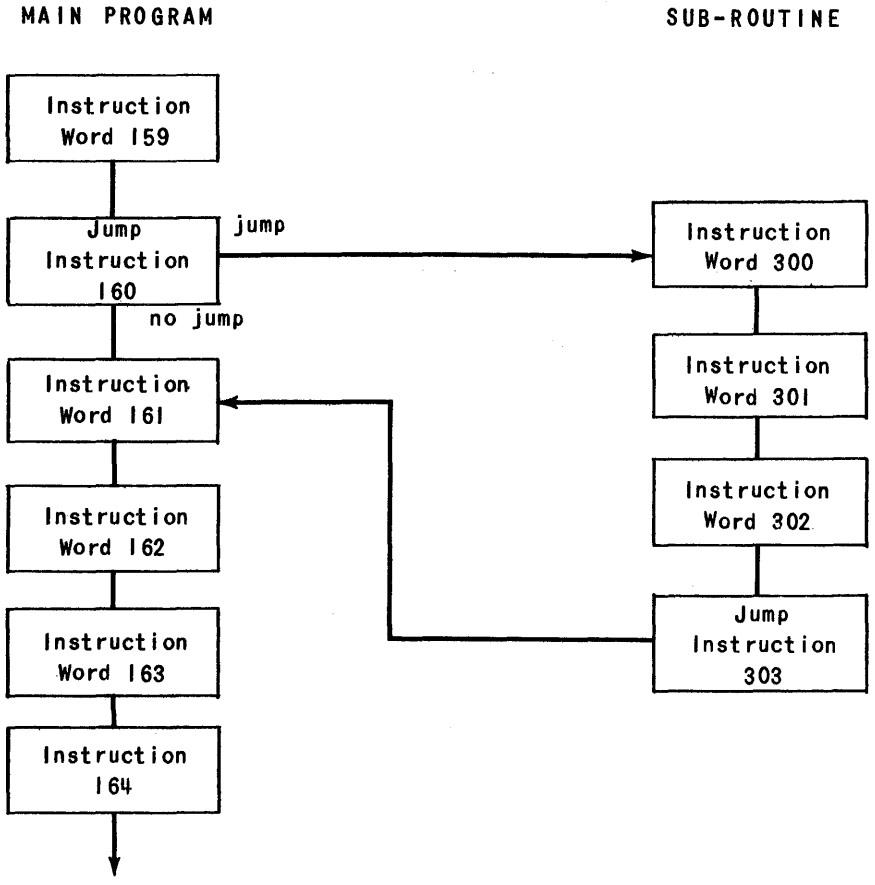
Conditional jump instructions in the UFC-I repertory include Jump on Plus JP, Jump on Negative JN and Jump on Zero JZ. In each of these commands, the state (+, -, 0) of conditional storage (a special purpose memory location discussed in Chapter 3) determines whether or not the jump will occur. If the condition specified by the jump instruction coincides with the state of conditional storage, the jump will be executed. If not, the jump is ignored and the next instruction is taken from the address specified by PAK, which is the next instruction in sequence.

Returnable Jump Feature

All of the jump instructions in the UFC-I repertory are "returnable jumps"; that is, during the execution of the jump instruction, a means is provided to return to any valid instruction word -- after the subroutine initiated by the subroutine has been completed. No additional or complicated programming technique is necessary to exploit the returnable feature built into the UFC-I jump instructions.

A returnable jump usually modifies the main chain of events by jumping to a series of instruction words called a "subroutine". At completion of the subroutine, control returns to the main program through a second jump instruction.

For example:



Jump instruction 160 is coded as follows:

	U	V	W	PR	S/C
160	161	303	300	UJ	Δ

U = the instruction word in the main program to which control returns.

V = the last instruction word of the subroutine.

W = the first instruction word of the subroutine.

Jump instruction 303 is coded as follows:

	U	V	W	PR	S/C
303	Δ Δ Δ	992	161	UJ	Δ

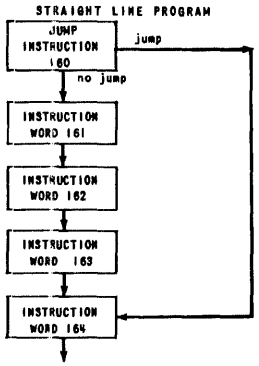
This instruction jumps control back to the main program at instruction word 161.

Note: This method of coding a returnable jump may be applied to any of the four jump commands.

In some cases, particularly in connection with programs utilizing "straight line" coding, the programmer may wish to jump to another part of the main program or subroutine with no thought of returning to the main program. To accomplish this, the programmer may vary his coding of the returnable jump so that the jump is executed in the shortest possible time.

This modified returnable jump is accomplished by the use of a "dummy" address in the V section of the jump instruction. Because Arithmetic Register C (addressed by 992) is used as an intermediate storage location during the execution of jump instructions, the use of RC as the dummy location will result in the fastest execution of a jump instruction.

For example:



Jump instruction 160 =

U	V	W	PR	S/C
ΔΔΔ	992	164	JP	Δ

U = may contain any valid characters

V = dummy address (in this case Register C)

W = address of the instruction at which the program is to continue.

Note: Any of the four jump instructions may be coded in the above manner.

Permissable Destinations

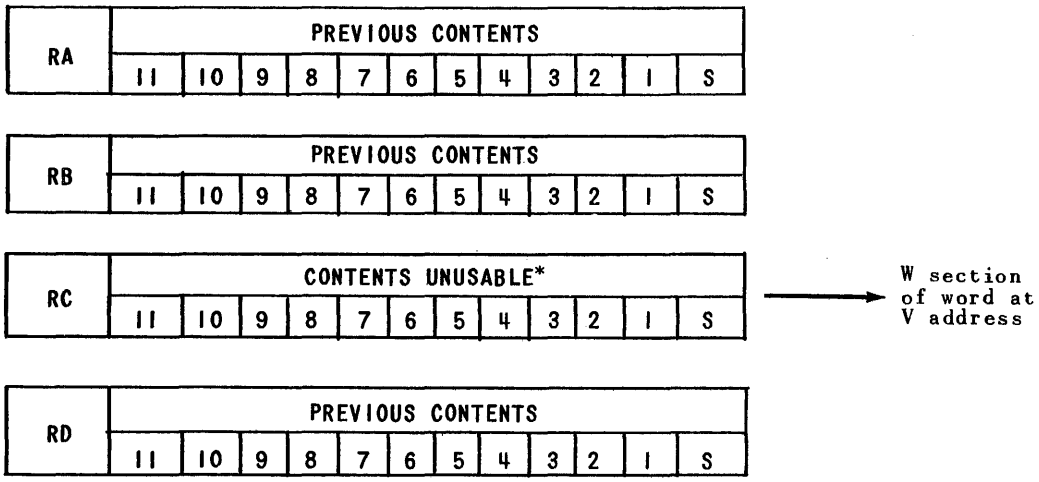
In jump instructions the V section of instruction word specifies a word or field address which is referred to for address modification purposes.

This address may specify:

A word location on the high speed drum, general storage buffer or block transfer buffer.

Register C (where a non-returnable jump is desired).

Contents of Arithmetic Registers after execution of a jump instruction:



*Register C is used as an intermediate storage location for the U section of the jump instruction word *when a jump occurs* and the W section of the word at the V address is modified.

Final contents of Register C are unusable. The address modification of the PAK which occurs during jump instruction is accomplished automatically as the jump instruction is executed.

Special Purpose Instructions

Buffer Transfer

Operation Code: 23S
Mnemonic Code: BT
Descriptive Code: (U) → BTB → W
Description:

Contents of the source address specified by U (or V_1) are transferred to a destination address specified by W (or R) address, via the block transfer buffer. V (or V_2) address is ignored.

Sub-instruction is initiated.

During the first half of a BT operation, the contents of the source address are transferred into the low order positions of the BTB, replacing as many characters as required. Higher order positions of BTB are not disturbed.

During the second half of a BT operation, the number of characters specified by the destination address is transferred from the low order positions of BTB to the destination.

The BT operation may not be used to transfer data from one BTB location to another, since BTB is not word and field addressable during the BT operation.

BUFFER TRANSFER

OPERATION SEQUENCE	INSTRUCTION WORD	PROGRAM STEP
Transfer data (1 up to 120 characters) from one program control storage location to another, via BTB, as follows:	PR = BT	PROCESS to BT wiring
Transfer, to the block transfer buffer, lowest-order character first, the contents of the location specified by:	U-address	V_1 ADDRESS wiring
Load the first character received* into BTB's word 9, character S position. Load the next character received into BTB's word 9, character 1 position, etc. until entire contents of the source are stored in BTB in this manner. When less than 120 characters are loaded into BTB, do not alter the higher-order character positions of BTB (i.e., those which do not receive data).		
Transfer data, out of BTB, to the location specified by:	W-address	R ADDRESS wiring
Send the data to the destination,** lowest order character first, beginning with BTB's word 9, character S position. Store the character from BTB's word 9, character 1 position in the next lowest-order character position of the destination, etc. Transmit out only the number of characters required to fill the capacity of the destination.		
Initiate sub-instruction(s) or sub-step(s) in accordance with:	S/C	STEP OUT wiring

* If BTB is specified as the source, the first half of this instruction is omitted.

** If BTB is specified as the destination, no transfer out of BTB occurs (i.e., the second half of the instruction is omitted).

Data Transmission Rules - Buffer Transfer

Introduction

Storage locations that may be addressed to obtain operands (called "Permissible Sources") and storage locations that may be addressed as destinations for the storage of results (called "Permissible Destinations") are given in the tables on the following page. If any storage locations other than those listed in these tables are specified in a Buffer Transfer, the program is automatically stopped.

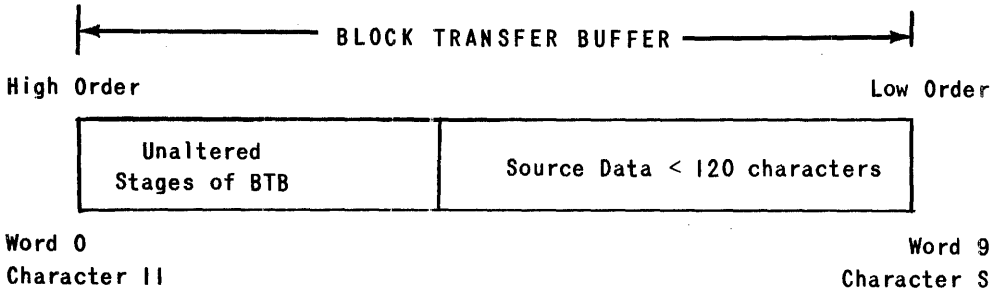
Source and destination references need not contain the same number of characters.

Source Rules

(Numbers preceding rules refer to numbers on the following tables)

(1) If the block transfer buffer is not referred to in either the source or destination address and:

(1a) If the number of characters specified by the source address is less than 120 characters, a copy of the source data is stored in the block transfer buffer beginning with word 9, character S; and the higher-order stages of BTB, those which do not receive source data, are left unaltered.



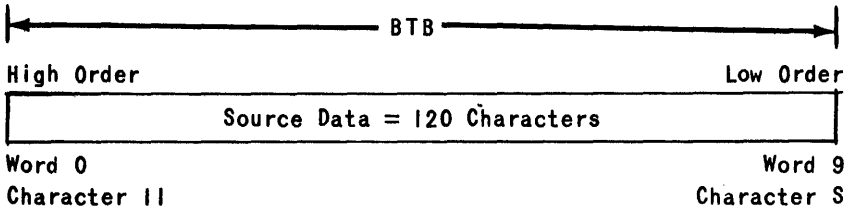
(1b) If the number of characters specified by the source address is 120 characters, an exact copy of the contents of the source address is stored in the block transfer buffer, beginning with BTB's word 9, character S position.

PERMISSIBLE SOURCES AND APPLICABLE BUFFER TRANSFER DATA TRANSMISSION RULES

Source	Source Address Specifies	Rules
Block Transfer Buffer	Same number of characters as destination address	2 or 5
High Speed Drum Tracks (except ISP)	Any word location 0-9 (12 characters) or Any field location A-V (excluding I and O) containing from 1 up to 119 characters or A blockette (Z) 120 characters	1 (1A or 1B)
General Storage Buffer		
Arithmetic Registers A, B, C, D	Exactly 12 characters	1 (1A)

PERMISSIBLE DESTINATIONS AND APPLICABLE BUFFER TRANSFER DATA TRANSMISSION RULES

Destination	Destination Address Specifies	Rules
Block Transfer Buffer	Same number of characters as source address	4 or 5
High Speed Drum Tracks (except ISP)	Any word location 0-9 12 characters) or Any field location A-V (excluding I and O), from 1 up to 119 characters or A blockette (Z) 120 characters.	3 (3A or 3B or 3C)
General Storage Buffer		
Arithmetic Registers A, B, C, D	Exactly 12 characters	3 (3A or 3B)
Instruction Revolver		
Shift Revolver		
High Speed Drum Field Selection Pattern ISP	Exactly 120 characters	3 (3C)
General Storage Buffer Field Selection Pattern GSP		
Block Transfer Buffer Field Selection Pattern BTP		



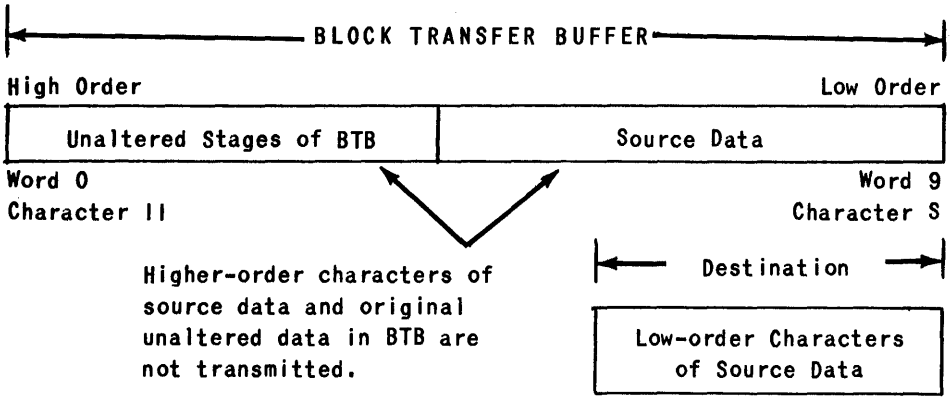
(2) If the block transfer buffer is referred to as the source address, no transmission into BTB occurs during first half of a buffer transfer. During the second half of a buffer transfer, the *data supplied by BTB is always transferred out of BTB beginning with word 9, character S position.* In this case, the number of characters BTB supplies as a source depends on the number of characters specified by the destination address. Data transmitted is always sent from BTB beginning with the lowest order character position of BTB (word 9, character S). The destination address may specify a particular field in the case of field-addressable destinations.

Note: In buffer transfers, BTB is *not* word and field addressable since program control storage ignores the lower-order character in the BTB address.

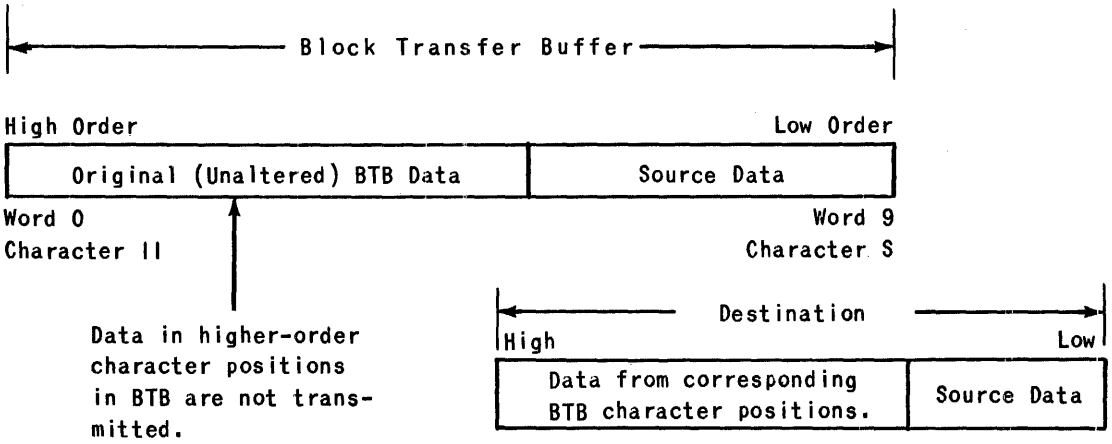
Destination Rules

(3) If the block transfer buffer is not referred to in either the source or destination address, and:

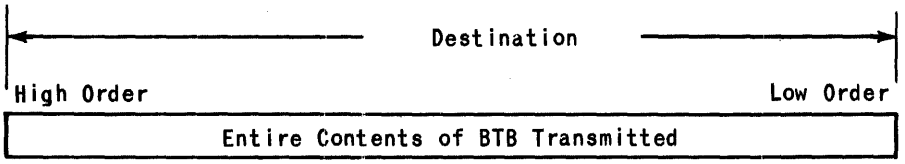
(3a) If the number of characters specified by the destination address is less than the number of characters specified by the source address, not all the source data is sent to the destination. That is, only those character positions in BTB beginning with word 9, character S and extending up to the capacity of the destination supply data.



(3b) If the number of characters specified by the destination address is greater than the number of characters specified by the source address, more characters of BTB data are transmitted out to the destination than were received from the source. The destination's lower-order character positions thus receive a copy of the source data, and the higher-order character positions of the destination receive a copy of the data held in correspondingly significant BTB character positions.



(3c) If 120 characters are specified by the destination address, the entire contents of BTB are transferred to the destination.



(4) If the block transfer buffer is referred to as the destination address, no transmission out of BTB occurs during the second half of a Buffer Transfer. (The note following rule 2 also applies to rule 4.)

BTB Both Source and Destination

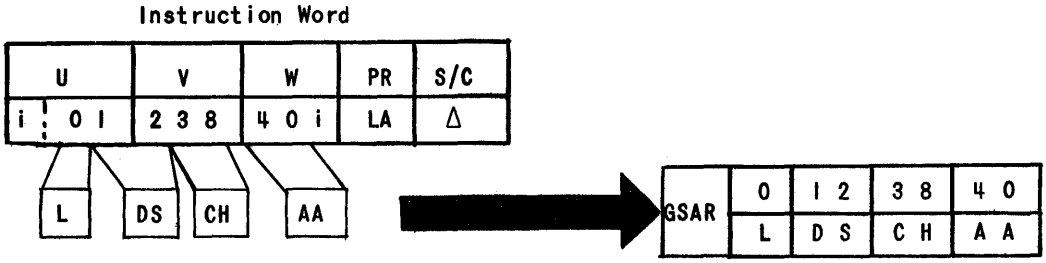
(5) If the block transfer buffer is referred to as both source and destination, nothing happens in BTB. Both halves of the buffer transfer are suppressed, an "end of operation" pulse is immediately produced, and the program continues. *A buffer transfer cannot be used, therefore, to transfer data from one BTB location to another BTB location.*

Load General Storage Address Register

Operation Code: 31S
Mnemonic Code: LA
Descriptive Code: $IRV_c \rightarrow GSAR$
Description:

The general storage address register (GSAR) is loaded with the two lower order digits of U, the three digits of V and the two higher order digits of W.

Sub-instruction is initiated.



In this instruction, the data loaded into the General Storage Address Register (GSAR) is obtained from the instruction itself (from the Instruction Revolver currently being used, IRV_c). As part of the execution of the instruction, program control storage is automatically notified that the general storage address register is the required destination.

The general storage address register may also be loaded by addressing GSAR as the destination in an instruction word or a program step.

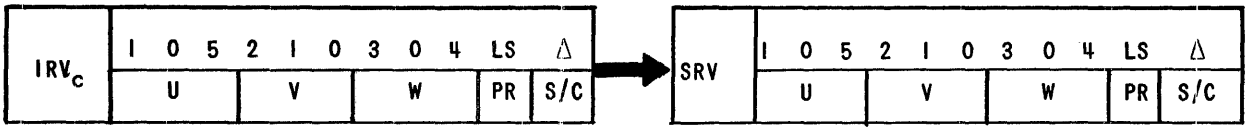
OPERATION SEQUENCE	INSTRUCTION WORD
Load GSAR with 7 characters from this instruction word (i.e., from IRV_c) as follows:	PR = LA
Place the lower order 2 digits of the U section of this instruction word in GSAR's 2 higher order digit positions:	U-section of this IW
Place the 3 digits of the V section of this instruction word in GSAR's next three higher-order digit positions.	V-section of this IW
Place the 2 higher order digits of the W section of this instruction word in GSAR's two lowest-order digit positions:	W-section of this IW
Initiate sub-instruction(s) in accordance with:	S/C

Load Shift Revolver

Operation Code: 32S
Mnemonic Code: LS
Descriptive Code: $(IRV_c) \longrightarrow SRV$
Description:

Contents of the current instruction revolver (IRV_c) are transferred to the shift revolver (SRV).

Sub-instruction is initiated.



In this instruction, the data loaded into the shift revolver (SRV) is obtained from the instruction itself (from current instruction revolver IRV_c). As part of this instruction, program control storage is automatically notified that the shift revolver SRV is the required destination.

The shift revolver may also be loaded by addressing SRV as the destination in an instruction word or a program step.

OPERATION SEQUENCE	INSTRUCTION WORD
Transfer the contents of IRV_c to the Shift Revolver, as follows:	PR = LS
Place the U section of this instruction word in the U-section of the Shift Revolver.	U-section of this IW
Place the V section of this instruction word in the V-section of the Shift Revolver.	V-section of this IW
Place the W section of this instruction word in the W-section of the Shift Revolver.	W-section of this IW
Place the OP section of this instruction word in the lower-order character positions of the Shift Revolver.	OP-section of this IW
Initiate sub-instruction(s) in accordance with:	S/C

Channel Clear

Operation Code: 33S
Mnemonic Code: CC
Descriptive Code: Δ → (W)
Description:

The track or buffer specified by the W (or R) address is cleared to space codes. The U (or V_1) and V (or V_2) addresses are ignored.

Sub-instruction or sub-step is initiated.

The Channel Clear operation includes only one programmed storage reference — a destination. When this instruction is executed, 120 space codes (Δ) are automatically generated by the computer and stored at the destination (W or R address).

OPERATION SEQUENCE	INSTRUCTION WORD	PROGRAM STEP
Place a space code in each of the 120 character positions of the track or buffer specified by:	PR = CC W-address*	PROC to CH C1 wiring R ADDRESS wiring*
Initiate sub-instruction(s) or sub-step(s) in accordance with:	S/C	STEP OUT wiring

*Permissible destinations in a channel clear instruction are:

High Speed Drum: Input/Output Tracks
Factor Storage Tracks
Intermediate Storage Tracks

General Storage Buffer

Block Transfer Buffer

Channel Search Probe

Operation Code: 27S
Mnemonic Code: SP

Descriptive Code: ?CS Storage

Found:	PAK
Busy:	U → PAK
Not Found:	V → PAK
Ignore Found:	W → PAK

Description:

Channel Search Storage is examined to determine whether a previously initiated channel search operation is completed.

If the search is *completed successfully* (found), channel search storage is set to the plus (+) state, and the next instruction is taken from the address specified by the PAK.

If the search is *not completed* (busy) the next instruction is taken from the U address.

If the search is *completed unsuccessfully* (not found) channel search storage is set to minus (-) state and the next instruction is taken from the V address.

If the search is *completed* and the special condition "*ignore found*" occurs, channel search storage is set to the zero (0) state and the next instruction is taken from the W address. (This condition does not occur on channel search unequal operations.)

Sub-instruction is initiated.

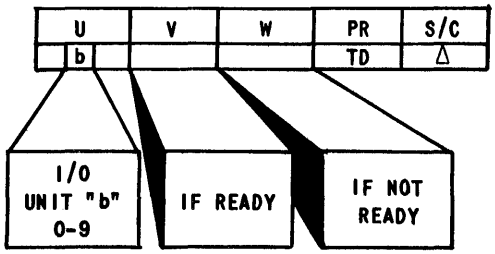
OPERATION SEQUENCE	INSTRUCTION WORD	SEARCH PROBE SUB-STEP	
Test to see if a previously initiated Channel Search Operation is completed:	PR = SP	STEP OUT to CS PROBE wiring	STEP OUT to CS PROBE + WAIT wiring
If the previously initiated search is not completed, proceed with the program and take the next instruction from:	The location specified by the U address	ACTIVE out wiring	Wait until search is completed
If the previously initiated search is completed, examine Channel Search Storage:			
If Channel Search Storage is set to MINUS, take the next instruction from:	The location specified by the V address	MINUS (-) out wiring	MINUS (-) out wiring
If Channel Search Storage is set to ZERO, take the next instruction from:	The location specified by the W address	ZERO (0) out wiring	ZERO (0) out wiring
If Channel Search Storage is set to PLUS, take the next instruction from:	The location specified by PAK	PLUS (+) out wiring	PLUS (+) out wiring
Initiate sub-instruction(s) in accordance with:	S/C		

Input/Output Instructions

Test Demand In

Operation Code: 34S
Mnemonic Code: TD
Description:

I/O unit "b" is tested to determine whether it is ready or not ready for use as follows:



A DEMAND TEST IN signal is sent to the I/O unit specified by the middle digit "b" of the U section of this instruction word.

If I/O unit "b" is READY, the next instruction is taken from the V address of this instruction word.

If I/O unit "b" is NOT READY, the next instruction is taken from the W address of this instruction word.

Sub-instruction is initiated.

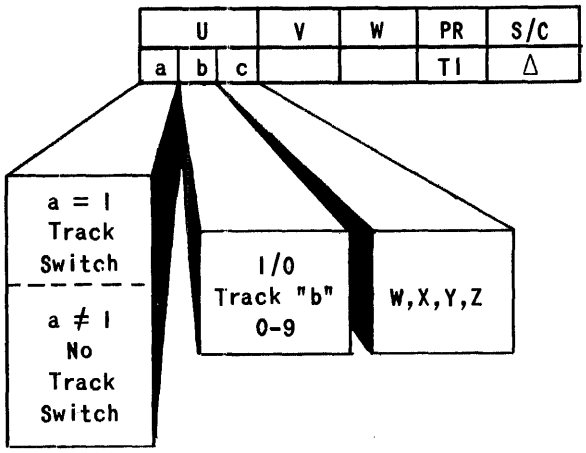
OPERATION SEQUENCE	INSTRUCTION WORD
Determine whether I/O unit "b" is READY or NOT READY for subsequent use, as follows:	PR = TD
Examine:	Middle digit (b) of the U-section of this instruction word.
Send the I/O unit specified by this digit a DEMAND TEST IN signal.	
If I/O unit "b" is READY, take the next instruction word from the location specified by:	V-section of this instruction word.
If I/O unit "b" is NOT READY, take the next instruction word from the location specified by:	W-section of this instruction word.
Initiate sub-instruction(s) in accordance with:	S/C

Test Incoming Control

Operation Code: 39S
Mnemonic Code: TI
Description:

High Speed I/O-Computer Control Line Storage is tested as follows:

The low order character "c" of the U section of this instruction word is examined. If "c" equals W, X, Y or Z, High Speed I/O-Computer Control Line Storage is tested for the corresponding condition:



If W, X, Y or Z condition is found:

The middle digit "b" and high order digit "a" of the U section of this instruction word are examined.

If "a" = 1, track switch is executed on I/O tracks specified by "b".

If "a" ≠ 1, no track switch is executed on I/O tracks specified by "b".

The next instruction is taken from the W section of this instruction word.

Sub-instruction is initiated.

If W, X, Y or Z condition is not found:

Track switch is not executed.

Next instruction is taken from the location specified by PAK.

Sub-instruction is initiated.

OPERATION SEQUENCE	INSTRUCTION WORD
Test High Speed I/O -Computer Control Line Storage for W, X, Y, or Z, as follows:	PR = TI
Examine: If c = W, test High Speed I/O-Computer Control Line Storage for the W-condition: If c = X, test for the X condition If c = Y, test for the Y condition If c = Z, test for the Z condition	W, X, Y, or Z in the lowest order character (c) of the U-section of this instruction word
If the particular W, X, Y, or Z condition tested is found, examine: For track switching: "b" specifies the I/O unit whose associated I/O tracks are to be conditionally switched: If a = 1, track switch If a = 0, do not track switch	Middle digit (b) and highest order digit (a) of the U-section of this instruction word
Take the next instruction word from the location specified by:	W-address of this instruction word.
If the particular W, X, Y, or Z condition tested is not found, do not track switch, regardless of the value of "a". Take the next instruction word from the location specified by the contents of PAK.	
Initiate sub-instruction(s) in accordance with:	S/C

Demand In

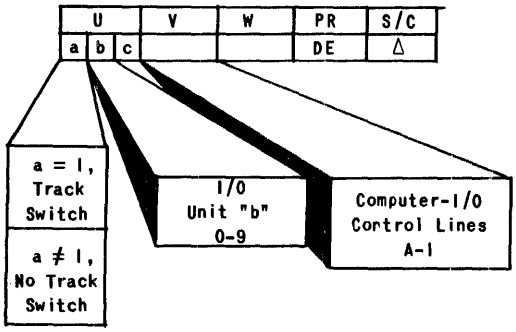
Operation Code: 45S
Mnemonic Code: DE
Description:

I/O unit is placed on demand; track switch is performed conditionally, and computer - I/O unit control information is exchanged as follows:

I/O unit "b" is sent a DEMAND IN signal.

When I/O unit "b" becomes READY, it produces a DEMAND OUT or SPECIAL OUT signal:

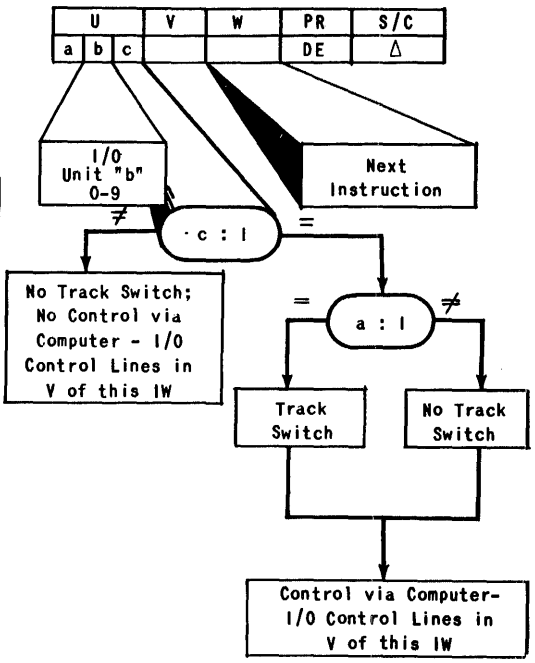
DEMAND OUT:



Next instruction is taken from location specified by PAK.

Sub-instruction is initiated.

SPECIAL OUT:



Next instruction is taken from W address.

Sub-instruction is initiated.

OPERATION SEQUENCE	INSTRUCTION WORD
Place I/O unit "b" on demand conditionally track switch and exchange control information as follows:	PR = DE
Examine:	
Send I/O unit "b" a DEMAND IN signal: i.e., place I/O unit "b" on demand:	Middle digit (b) of the U-section of this instruction word
(LS) I/O-Computer control lines (a-1) from I/O unit "b" are energized	
When I/O unit "b" becomes READY, it produces a DEMAND OUT or a SPECIAL OUT signal. If it produces a SPECIAL OUT it also sends one or more signals over the (HS) I/O-Computer control lines (W,X,Y,Z) to High Speed I/O-Computer Control Line Storage.	
If a DEMAND OUT "b" is produced, track switch if:	Highest order digit (a) of the U-section of this instruction word.
is a "1"; do not track switch if it is a "0". Whether track switching occurs or not, send I/O unit "b" the control information specified by:	V-section of this instruction word.
(via the Computer-I/O control lines), and take the next instruction word from the location specified by the contents of PAK.	
If a SPECIAL OUT is produced, examine:	Lowest-order digit (c) of the U-section of this instruction word:
If c = 0, do not track switch regardless of the value of "a", and do not send the control information (V) to I/O Unit "b".	
If c = 1, track switch if "a" = 1, do not track switch if "a" = 0. Send the control information (V) to I/O Unit "b".	
In any event, if a SPECIAL OUT is produced, take the next instruction from the location specified by:	W-address of this instruction word.
Initiate sub-instruction(s) in accordance with:	S/C

Transfer of Control Instruction

Transcop (Transfer Control to Plugboard)

Operation Code: 51 through 98
Description:

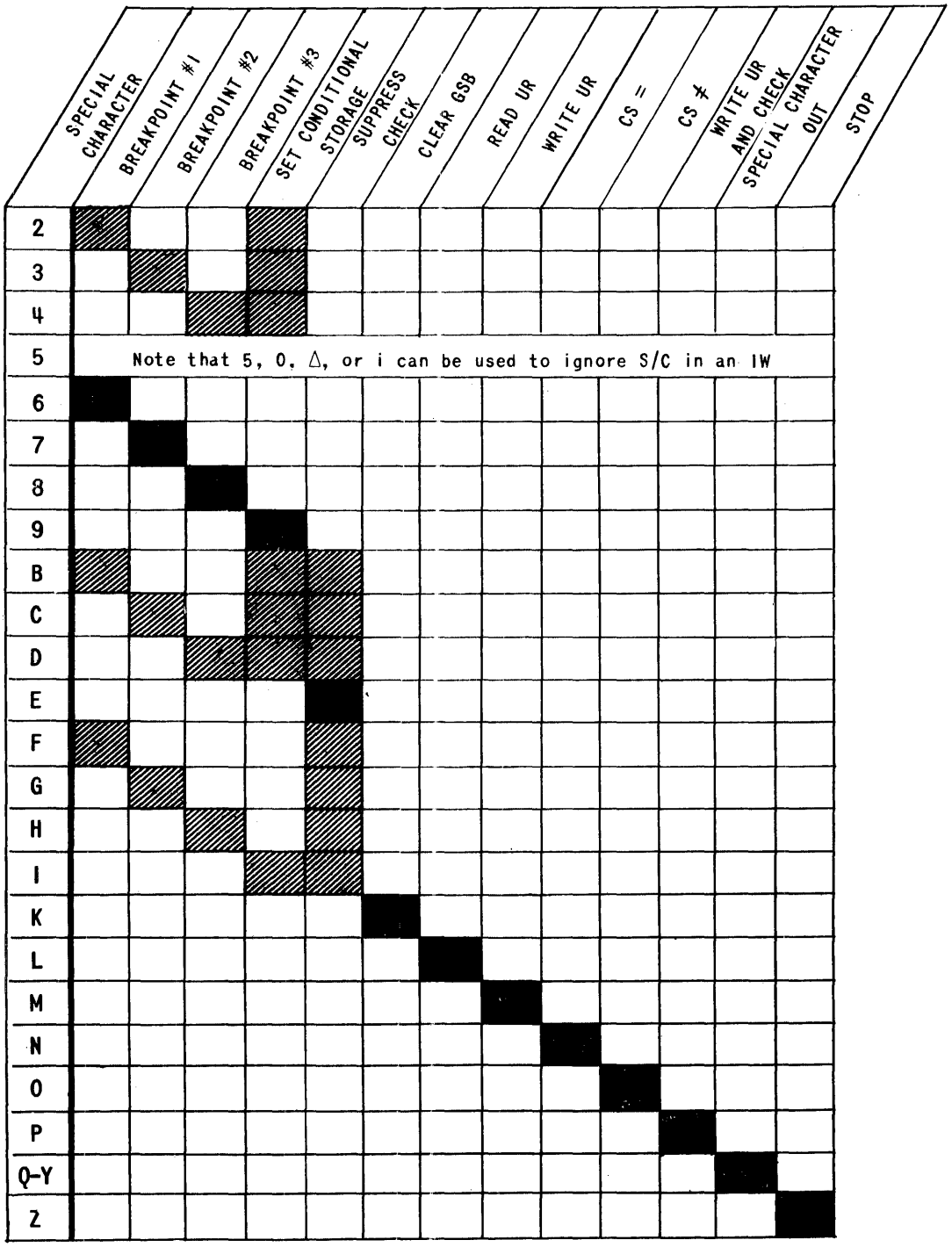
The execution of instruction words is interrupted and control is transferred to the plugboard step identified by the operation code of this instruction word.

This instruction word is retained in IRV_C. U, V, and W portions of this instruction are available during the plugboard sequence through U, V and W ADDRESS HUBS.

When the plugboard sequence is completed and NEXT INSTRUCTION is signalled, the sub-instruction of the transcop word is initiated.

OPERATION SEQUENCE	INSTRUCTION WORD
Interrupt the execution of Instruction Words; retain this instruction Word in IRV_C; transfer Program Control to the Plugboard Step numerically equal to:	PR = 51 - 98
Execute the Program Step wired there.	
A sequence of one or more program steps is thus initiated. The U, V, and W addresses of this instruction word are available to each program step in the TC - initiated plugboard sequence.	
If the plugboard-defined program pulses the NEXT INSTRUCTION hub, program control resumes the internal program, then initiates the sub-instruction(s) specified by S/C in this instruction word.	

SPECIAL CHARACTER CODES



Note: Certain values of S/C specify but one Sub-Instruction; others specify two or three Sub-Instructions. For ready reference, each Sub-Instruction's unique value of S/C is indicated by a shaded area, and cross-hatching is employed when two or three Sub-Instructions are specified by a value of S/C.

TABLE 4-2

INITIAL AND FINAL CONTENTS OF RA, RB, RC, AND RD FOR INSTRUCTIONS WHICH USE ARITHMETIC REGISTERS

INSTRUCTIONS	CONTENTS OF								RESULT FORMED IN	RESULT STORED FROM
	Register A		Register B		Register C		Register D			
	Initial	Final	Initial	Final	Initial	Final	Initial	Final		
Add	Augend	Augend	Addend	Addend	PC	UN	PC	Sum	RD	RD
Add & Check	Augend	Augend	Addend	Addend	PC	Negative Zero	PC	Sum	RD	RD
Subtract	Minuend	Minuend	Subtrahend	Subtrahend	PC	UN	PC	Difference	RD	RD
Subtract & Check	Minuend	Minuend	Subtrahend	Subtrahend	PC	Negative Zero	PC	Difference	RD	RD
Multiply, Store Upper	Multiplicand	Multiplicand	Multiplier	Multiplier	PC	PROD U	PC	PROD L	RC & RD	RC
Multiply, Store Upper & Check	Multiplicand	Multiplicand	Multiplier	Multiplier	PC	Negative Zero	PC	Negative Zero	RC & RD	RC
Multiply, Store Lower	Multiplicand	Multiplicand	Multiplier	Multiplier	PC	PROD U	PC	PROD L	RC & RD	RD
Multiply, Store Lower & Check	Multiplicand	Multiplicand	Multiplier	Multiplier	PC	Negative Zero	PC	Negative Zero	RC & RD	RD
Divide, Store Quotient	Dividend	UN	Divisor	UN	Dividend	Remainder	PC	Quotient	RC & RD	RD
Divide, Store Quotient & Check	Dividend	UN	Divisor	UN	Dividend	UN	PC	UN	RC & RD	RD
Divide, Store Remainder	Dividend	UN	Divisor	UN	Dividend	Remainder	PC	Quotient	RC & RD	RC
Divide, Store Remainder & Check	Dividend	UN	Divisor	UN	Dividend	UN	PC	UN	RC & RD	RC
(Any) Divide Where Divisor = 0	Dividend	UN	Divisor	UN	Dividend	Zero	PC	Zero	RD & RD	RC or RD
(Any) Divide Where Quotient Digit < 1	Dividend	UN	Divisor	UN	Dividend	Dividend (Remainder)	PC	Zero	RC & RD	RC or RD
Mask Transfer	Operand	Operand	Mask	Mask	PC	R	PC	R	RC & RD	RD
Compare	V_1 Operand	V_1 Operand	V_2 Operand	V_2 Operand	PC	PC	PC	PC	Branch Storage Set to	+, 0, −
Suppress Left Zeros	V_1 Operand	R	PC	PC	PC	PC	PC	R	RA & RD	RD
Left Normalize	V_1 Operand	Normalized Operand	PC	Normalized Count	PC	PC	PC	Normalized Operand	RA, RB, & RD	RD
Substitute (U, V, or W)	V_1 Operand	V_1 Operand	V_2 Operand	V_2 Operand	PC	R	PC	R	RC & RD	RD
Jump on Plus Jump on Minus Jump on Zero Unconditional Jump	PC	PC	PC	PC	PC	RC used as intermediate storage for U. Final Contents of RC are unusable.	PC	PC	RC	RC
Arithmetic Transfer	PC	PC	PC	PC	PC	PC	Source Data	Data transmitted	RD	RD

R - Result
PC - Previous contents
UN - Unusable

RA - Register A
RB - Register B
RC - Register C
RD - Register D

PROD U - higher-order characters of product
PROD L - lower-order characters of product

TABLE 4-3

chapter 5

GENERAL STORAGE SYSTEM

INTRODUCTION

Modern business operations demand split-second decisions on the part of managerial and supervisory personnel. In order to make these decisions intelligently, the manager must have access to the very latest facts and figures pertinent to the question at hand. The following are typical situations where the large capacity random access general storage system of the UFC-I contributes to effective decision-making by keeping information readily available:

Inventory control, where inventory balances and other continuing information may be carried in general storage. Immediate posting of random receipts and withdrawals always maintains inventory status up to date.

Production scheduling, where a highly volatile situation exists. Schedule and process changes must be reflected immediately in order to determine shipping dates and quantities.

Sales analysis, where vast amounts of random data must be classified, accumulated, and reduced in volume.

Insurance and transportation problems, where large rate tables may be stored for reference periodically during calculations.

Recapitulation and reporting, in any situation where a mountain of detail must be condensed so that reference, calculation, summarization and reporting may be accomplished in a single operation.

COMPONENTS OF THE GENERAL STORAGE SYSTEM

The general storage system is the large capacity, random access memory of the Univac File-Computer, Model I, system. It is composed of the following principal parts:

(1) General storage drums (GSD). A UFC-I system may include from 1 to 10 magnetic drums, called general storage drums (GSD). Each drum is capable of storing 180,000 7-bit alpha-numeric characters.

(2) General storage address register (GSAR). This 7-digit register holds the address of the general storage location involved in a general storage reference.

(3) General storage buffer (GSB). This 120-character, magnetic core buffer serves as an intermediate storage in data transmissions to and from the general storage drums, and holds the unit record identifier in channel search operations.

(4) Circuitry. The controlling, locating, and synchronizing circuitry necessary for execution of general storage operations is also considered a basic part of the general storage system.

Data organization in general storage. The basic unit of data handled by the general storage system is the "unit record". A unit record may be read from a location in general storage; it may be stored at a location in general storage; it may be searched for in general storage.

A *unit record* is a group of adjoining characters handled by the computer as a unit during general storage operations. A unit record must consist of at least 12 characters (one computer word), but may also consist of any integral multiple of 12 characters up to a maximum of 120 characters.

An "*item*" or "*file-entry*" is a group of facts pertaining to a major unit of a file. Each item, or major unit, is identifiable as unique from all other major units in the file. For example, an individual payroll record may be an item in a payroll file; a part number and its related information may be an item in an inventory file; an account receivable may be an item in a customer file.

"*Unit records*" and "*items*" bear the following relationship to each other in general storage of the Univac File-Computer.

(1) A unit record may be a complete item in itself.
(2) A unit record may be only a portion of an item.
(3) A unit record may contain several complete items.

When files are recorded in general storage, the maximum amount of space necessary for further processing of any item is usually allotted to all items in that file. Thereafter, reference to any item is made on a unit record basis, the length of which is specified by the programmer within the limits previously prescribed. Thus, data held in general storage of UFC-I may be obtained, altered, and stored selectively on a unit record basis.

Time Sharing during General Storage Operations. General storage operations, which provide access to the data on the general storage drums, may be time-shared with operations of the central computer and the input-output units.

Whenever the general storage buffer (GSB) and the general storage address register (GSAR) are engaged in general storage operations, they are not available to program control storage operations of the central computer. However, by suitable programming, operations of the general storage system and the program control storage system may be carried out simultaneously, with a resultant time-saving factor in accomplishing the assigned processing of data.

GENERAL STORAGE DRUMS

Description

The general storage magnetic drums provide the large capacity, random access storage of the UFC-I. A maximum of 10 drums, containing a total of 1,800,000 characters of information, may be included in the system. Segments of this information may be located by drum section, channel, and unit record area, as described below:

(1) Each general storage drum is divided into three sections, each section containing 100 channels of information.

(2) A channel contains the information available to one read/write head around the circumference of the magnetic drum. (See Figure 5-1.)

(3) Each channel consists of (a) space for 600 characters of filed information, (b) a 6-character search control location, important in the channel search operations discussed below, and (c) a dead space, which allows channel search to continue from channel to channel without waiting for a complete drum revolution.

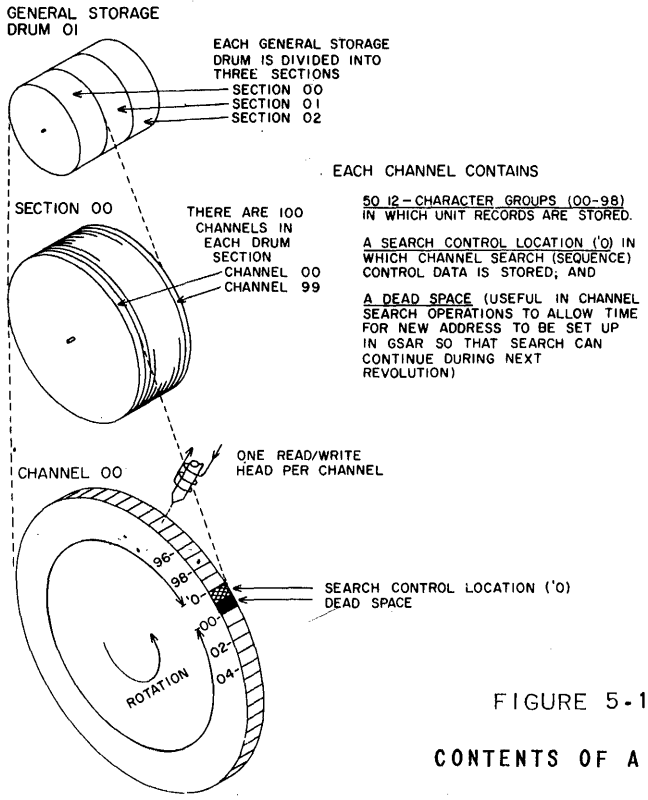


FIGURE 5-1

CONTENTS OF A CHANNEL

Address Structure

Two types of general storage addresses are employed in UFC-I: unit record area (URA) addresses and search control location (SCL) addresses.

A *Unit Record Area* is identified by means of a 7-character format as follows:

L	DS	CH	AA
Unit Record Length	Drum Section	Channel	Angular Address

All characters in the 7-character format *must* be numeric.

The *Unit Record Length Code* specifies the length or number of characters to be called for in a particular general storage reference. This code may be any number from 0 through 9. The code digit, multiplied by 12, defines the number of characters in a unit record area, except in the case of 0 which may be considered as 10, as it designates a 120 character unit record area.

The *Unit Record Selector* (a switch on the general storage control cabinet) performs two important functions in relation to the unit record length code:

(1) If this switch is set to "GSAR", the length code (L) held in the general storage address register (GSAR) specifies the length of the unit record.

(2) With the switch set at any one of the positions 0 through 9, the unit record length is defined by the switch setting, and the "L" code is ignored.

The *General Storage Drum Section* is identified by "DS" in the address format. Each drum is divided into three sections. A ten-drum system contains thirty such sections, designated by addresses 00 through 29. For example, 00, 01 and 02 indicate the three drum sections of the first drum in the system.

Channel Address (CH) specifies a particular 600-character channel of the 100 channels available on each drum section. Addresses of these channels are 00 through 99 in any section of any general storage drum.

Angular Address (AA) specifies the exact location within a partiuclar channel at which a unit record area begins. The variation of these numbers depends on the length of the unit record used.

The initial URA on each channel is addressed by 00, with the angular address increasing by two for each succeeding group of twelve characters. Thus,

successive 12-character unit record areas are addressed by multiples of two, while 24 character unit record areas are addressed by multiples of four. This type of format provides a method of continuous addressing throughout the general storage system for unit record areas of 12, 24, 60 and 120 characters.

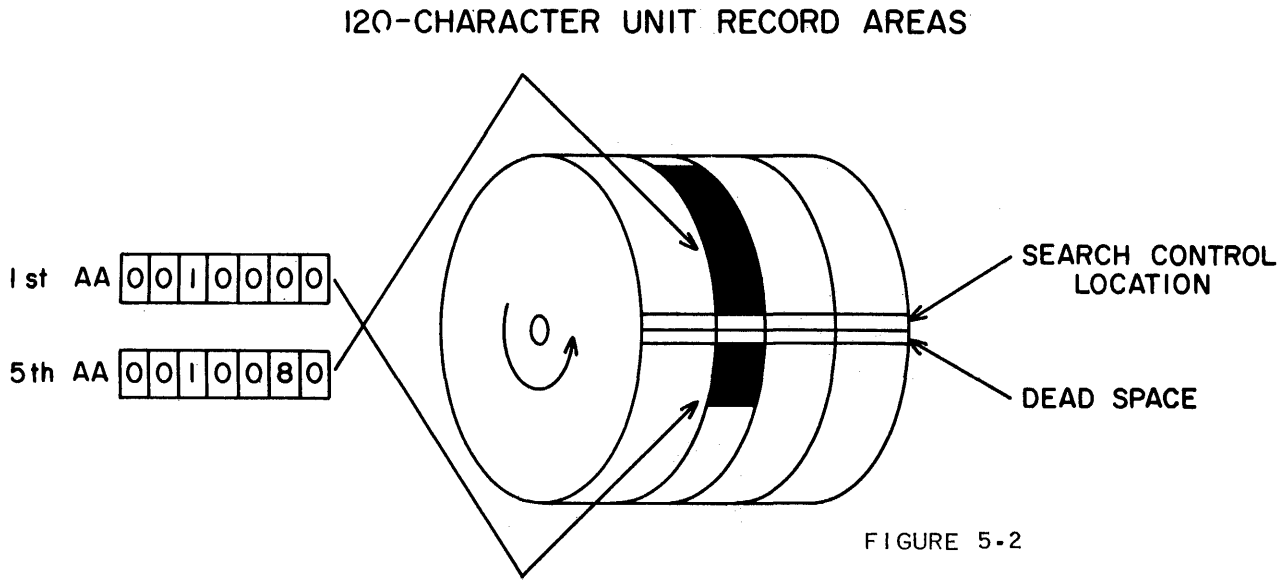


FIGURE 5-2

120-CHARACTER UNIT RECORD AREAS

NUMBER OF UNIT RECORD AREAS
FOR VARIOUS DRUM SYSTEMS

Drums per System	Characters per Unit Record Area									
	12	24	36	48	60	72	84	96	108	120
1	15,000	7,500	4,800	3,600	3,000	2,400	2,100	1,800	1,500	1,500
2	30,000	15,000	9,600	7,200	6,000	4,800	4,200	3,600	3,000	3,000
3	45,000	22,500	14,400	10,800	9,000	7,200	6,300	5,400	4,500	4,500
4	60,000	30,000	19,200	14,400	12,000	9,600	8,400	7,200	6,000	6,000
5	75,000	37,500	24,000	18,000	15,000	12,000	10,500	9,000	7,500	7,500
6	90,000	45,000	28,800	21,600	18,000	14,400	12,600	10,800	9,000	9,000
7	105,000	52,500	33,600	25,200	21,000	16,800	14,700	12,600	10,500	10,500
8	120,000	60,000	38,400	28,800	24,000	19,200	16,800	14,400	12,000	12,000
9	135,000	67,500	43,200	32,400	27,000	21,600	18,900	16,200	13,500	13,500
10	150,000	75,000	48,000	36,000	30,000	24,000	21,000	18,000	15,000	15,000

Variability of Unit Record Addressing

Items or file entries of varying lengths may be stored (a) in different drums within a general storage system, (b) in different sections of a single drum, (c) in different channels of a section, or (d) in different character groups within a channel. For example, payroll, accounts receivable, and accounts payable may be stored in neighboring sections of a drum, as in the following illustration:

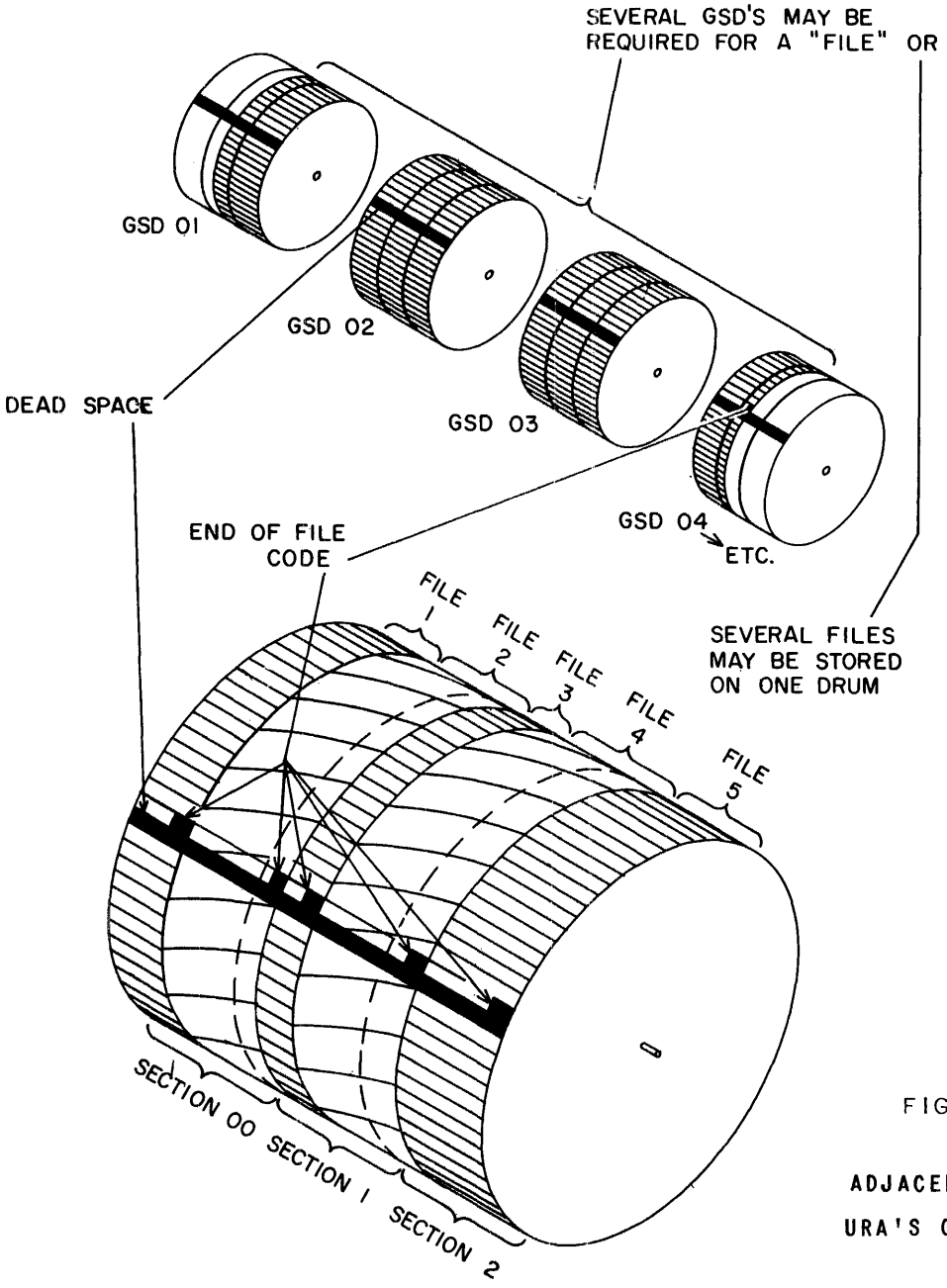


FIGURE 5-3

ADJACENT STORAGE OF URA'S OF VARYING LENGTHS

VALID GENERAL STORAGE UNIT RECORD AREA ADDRESSES

Unit Record Area Length Code (L)	1	2	3	4	5	6	7	8	9	0
Characters per Unit Record Area	12	24	36	48	60	72	84	96	108	120
Complete Unit Record Areas per Channel	50	25	16	12	10	8	7	6	5	5
Characters Utilized per Channel	600	600	576	576	600	576	588	576	540	600
	00	00	00	00	00	00	00	00	00	00
	02									
	04	04								
	06		06							
	08	08		08						
	10				10					
	12	12	12			12				
	14						14			
	16	16		16				16		
	18		18						18	
	20	20			20					20
	22									
	24	24	24	24		24				
	26									
	28	28					28			
	30		30		30					
	32	32		32				32		
	34									
	36	36	36			36			36	
	38									
	40	40		40	40					40
	42		42				42			
	44	44								
	46									
	48	48	48	48		48		48		
	50				50					
	52	52								
	54		54						54	
	56	56		56			56			
	58									
	60	60	60		60	60				60
	62									
	64	64		64				64		
	66		66							
	68	68								
	70				70		70			
	72	72	72	72		72			72	
	74									
	76	76								
	78		78							
	80	80		80	80			80		80
	82									
	84	84	84			84	84			
	86									
	88	88		88						
	90		90		90					
	92	92								
	94									
	96	96								
	98									

When items on the drum are short, and when they may be desired in sequence, several short items may be read by a single read command. In the same manner, these several small items may be restored on the drum by a single write command. For example: a group of ten 12-character items stored in unit record areas 00, 02, 04 through 18 could be called up as a single 120 character unit record by inserting a 0 (120 character URL code) in the unit record length code position of the address beginning at angular address 00 on the channel.

Any size of unit record (of the ten sizes available) may be read out, beginning at any valid 12-character group angular address provided there are enough character positions between the beginning angular address and the end of the channel. For example, unit record length code 0 (120 characters) may be called for beginning at any angular address up to and including 80, which is the last valid 120-character unit record address. In any address greater than 80, the full 120 characters cannot be obtained from the channel.

A *Search Control Location* (SCL) refers to a specific 6-character memory location on each channel of general storage. It is addressed by drum section, channel and the designation '0 (prime zero) in the following format:

-	DS	CH	'0
Unit Record Length (Ignored)	Drum Section	Channel	Search Control Location

The Unit Record length code may be any computer character, since it is ignored in any operations associated with the search control location.

Drum section, and channel within that drum section, identify the channel search control location used as a reference. Each character in the drum section and channel portions of the format must be numeric or an error will result.

Search control location "'0" is the exact code which must be used to differentiate the search control location address from any unit record angular address. The characters '0, when stored in any search control location, act as a stop code, terminating channel search operations when '0 is interpreted in the general storage address register.

During channel search operations, discussed more fully below, an "overflow address" stored in the search control location of each channel directs the computer where to continue the search if the desired unit record is not found in the channel just searched.

The method of loading an overflow address into the search control location is as follows:

(1) Place the 6-character address (to be stored in the search control location) in character positions 1 - 6 (Field A) of the general storage buffer (GSB). For example, if search is to continue at location 030500, these numbers would be loaded into the GSB.

(2) Place the address of the desired search control location in the general storage address register (GSAR). An example might be search control location 0250'0.

(3) Execute a "Write Unit Record" general storage operation. Thus, overflow address 030500 would be loaded into the search control memory location '0 of channel 50, drum section 02.

Internal:

Instruction Location			INSTRUCTION WORD									
			U		V			W			PR	S/C
1	4	0	0	0	A	Δ	Δ	Δ	9	8	A	A	T	Δ
1	4	1	0	0	B	Δ	Δ	Δ	9	9	5	A	T	M

External:

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
51								52
52	00 A		AT			GSB A		53
53	00 B		AT			GSAR		WUR

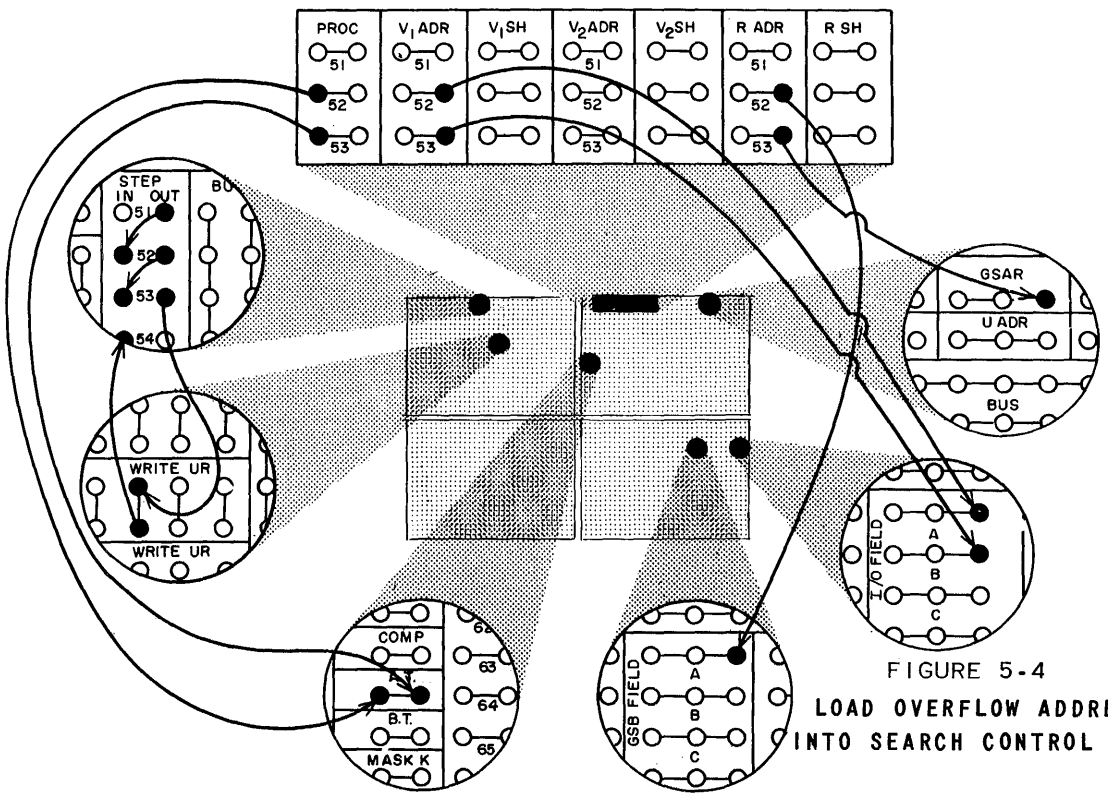


FIGURE 5-4
LOAD OVERFLOW ADDRESS INTO SEARCH CONTROL LOCATION

General Storage Address Register (GSAR)

The General Storage Address Register is a 7-digit register the principal function of which is to hold general storage addresses during general storage operations.

The two types of valid addresses held in GSAR are:

Unit Record Area Addresses
Search Control Location Addresses

L	D	S	C	H	A	A
-	D	S	C	H	'	0

The GSAR performs a secondary function in its role as an addressable unit of the program control storage system. This secondary function was discussed in Chapter 2, "Program Control Storage System".

The General Storage Address Register is addressable internally by the code 995, or by wiring of the GSAR hubs on the main program plugboard.

An address may be loaded into the GSAR by one of several methods:

(1) Internal command LA (see example 1, below).

(2) Internal arithmetic transfer of an address to the GSAR.

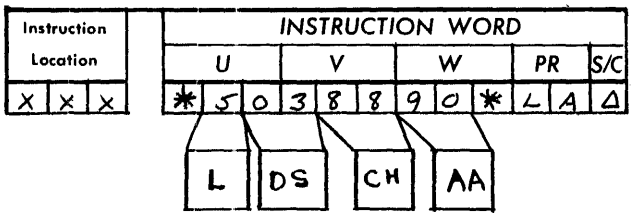
(3) Through modification or calculation of an address, with the GSAR as the destination address R.

(4) Plugboard version of 2 above (see example 2, below).

(5) Plugboard version of 3 above.

Example 1

The operation code LA loads the GSAR with the two lower order digits of the U section, plus the V section and the two higher order digits of the W section of the instruction word.



* Any computer character.

In this example, the address 5038890 will automatically be loaded into GSAR, and general storage operations utilizing GSAR may be initiated.

Example 2

Assuming that the desired general storage address is stored at Field J of the input/output track on demand, the following programming would transfer this address to the general storage address register.

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
63								64
64	I/O-J		AT			GSAR		65

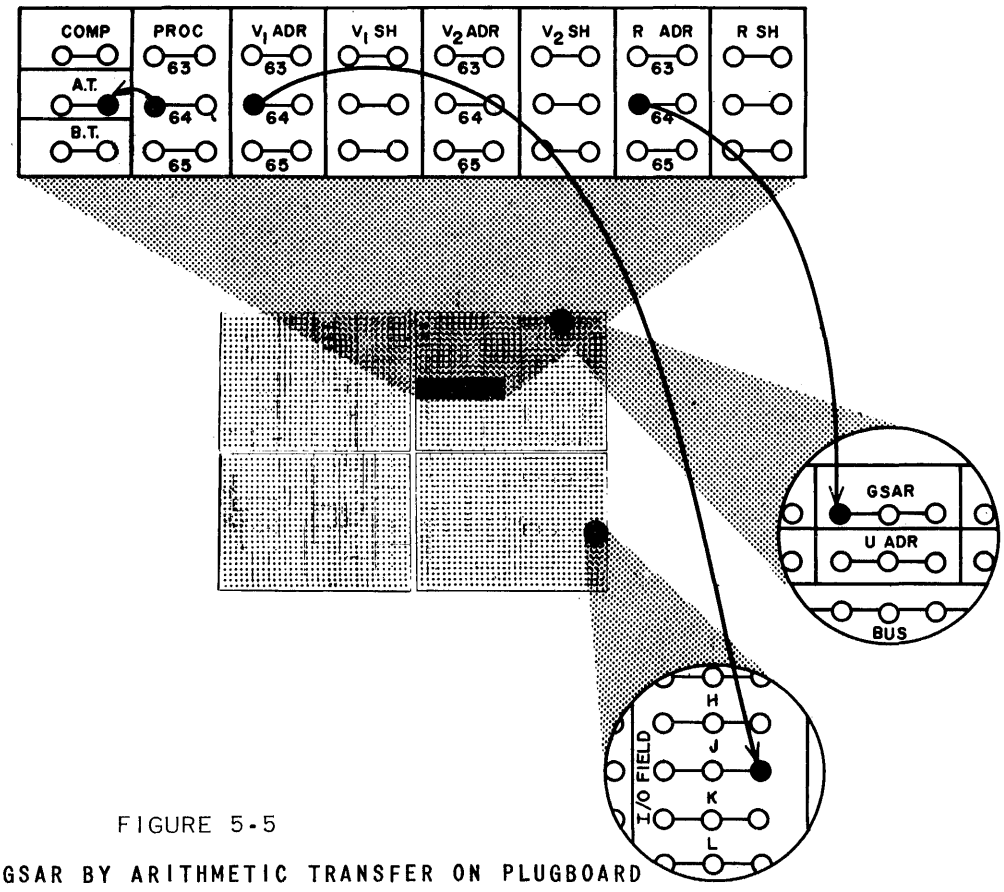


FIGURE 5-5

LOAD GSAR BY ARITHMETIC TRANSFER ON PLUGBOARD

GENERAL STORAGE BUFFER (GSB)

Function

The general storage buffer is a 120 character, magnetic core memory that provides rapid access to general storage drum data. Whenever a unit record is referred to on the drum and read into the GSB, it is held in the buffer while its component words and fields are being processed. Conversely, data to be recorded on the drum is held in the GSB while words and fields are being processed. The GSB, therefore, serves as in intermediate storage location, or "buffer," between the general storage drums and the program control storage system.

General Storage Buffer Addressing

The address of the general storage buffer (GSB) is 98, supplemented by a third number or letter to indicate a word, field, or blockette. Word addresses are designated by 0 through 9. Fields, which can be of any length up to 119 characters and any number up to 20, are designated by letters A through V, omitting I and O. The entire contents of the buffer (120 characters) is designated by the blockette address Z.

For example:
985 designates the general storage buffer word number 5.
98F designates the sixth field in the general storage buffer, field F.

In addressing the GSB, note that field A begins at word 9, character S position. Access to a particular field in GSB requires scanning the buffer beginning at word 9, character S, until the field is located, whereas any buffer word address is directly accessible, regardless of its position in the buffer. Both word 9, character S position and field A, character S position represent the first character in the buffer.

The lengths of fields in the unit record area are determined by the general storage buffer field selection pattern (GSP), which is addressed internally as 99Y. It can be addressed only as a destination, i. e., for storing a 120 character pattern which will control the field addressing of the buffer until replaced by a different pattern. The field pattern may be changed at any time throughout the program as various types and sizes of unit record areas are processed.

Construction of the GSP is similar to the block transfer pattern (BTP) and high speed drum intermediate storage pattern (ISP) discussed under Program Control Storage System.

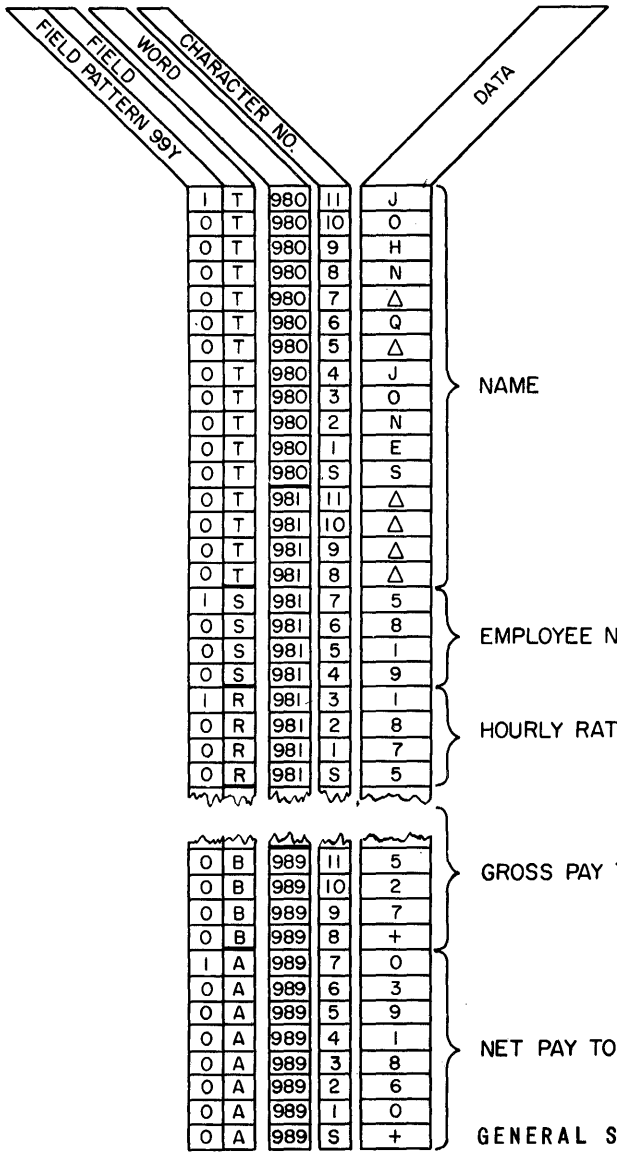


FIGURE 5-6

GENERAL STORAGE BUFFER ADDRESSING

GENERAL STORAGE OPERATIONS

Introduction

The general storage operations described below provide the link between the program control storage system of the central computer and the general storage system which provides the large capacity random access storage. Each of the general storage operations can be time-shared with central computer operations, since each general storage operation is performed as a subinstruction internally or substep on the plugboard. In other words, the operation may may be carried on simultaneously with any program controlled operations which do not require use of the same registers and buffers.

Read Unit Record (R1 - R4)

This operation: (1) locates the specific unit record for which the 7-digit general storage address is held in GSAR, or it locates a specific search control location for which the 6-digit address is held in GSAR, and (2) transfers the unit record to the general storage buffer. "Read Unit Record" is initiated when two conditions are met:

(1) The general storage address of the desired unit record has been placed in the general storage address register.

(2) The letter "L" is placed in the special character "S/C" portion of the instruction word, or the "READ UR" hub on the main program plugboard is signaled.

An example of the proper internal instruction word is:

INSTRUCTION WORD											
U			V			W			PR		S/C
*	1	0	2	0	9	6	2	*	L	A	L

* Any computer character

The process code (PR) is LA, indicating "Load GSAR" with the address 1020962; special character L indicates that Read Unit Record should be executed.

Wiring of the substep "Read Unit Record" on the plugboard is illustrated in the following diagram:

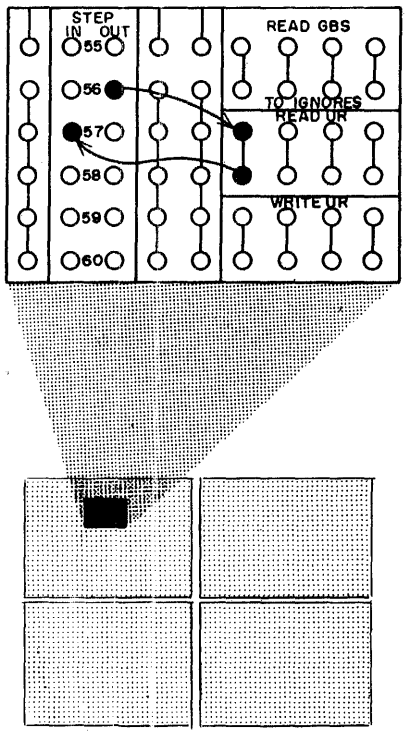


FIGURE 5-7

READ UNIT RECORD

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
56						R1		

READ URA (R)		
NO	IN	OUT
1	FROM ST. 56 TO STEP 57	
2		
3		
4		

Access to the general storage address register and general storage buffer is prevented by an interlock during this operation.

The time required to execute the Read Unit Record operation (assuming that GSAR is already loaded) is indicated in the following table.

Length Code:	1	2	3	4	5	6	7	8	9	0
Number of Characters:	12	24	36	48	60	72	84	96	108	120
Average Time:	17.65	18.30	18.95	19.60	20.25	20.90	21.55	22.20	22.85	23.50
Maximum Time:	34.65	35.30	39.95	36.60	37.25	37.90	38.55	39.20	39.85	40.50

The data in any search control location may be read in the same manner as a unit record, with the exception that the 6-digit address (DS CH '0) is placed in the GSAR before Read Unit Record is initiated.

The time required to read the contents of a search control location into the GSB (assuming that GSAR is already loaded) is:

Average time (in milliseconds): 17.65
Maximum time (in milliseconds): 34.65

Note: Six characters of the dead space are read into GSB in addition to the 6 characters stored in the search control location, since a minimum of 12 characters is processed in this general storage operation.

Write Unit Record (W1 - W4)

Write Unit Record (1) locates the unit record or search control data held in the GSB and (2) stores that record in its proper location in the general storage system. For this operation two conditions are necessary:

(1) The general storage address of the specific unit record or the search control location must be placed in the GSAR.

(2) The letter "M" must be placed in the special character "S/C" position of the instruction word, or the "WRITE UR" hub on the plugboard must be signaled. The example given above would apply here, except that the letter "L" would be replaced by an "M" which indicates the Write Unit Record Operation.

Plugboard wiring of "Write Unit Record" is as follows:

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
58								W1

WRITE URA (W)		
NO	IN	OUT
1	FROM ST. 58	TO ST. 59
2		
3		
4		

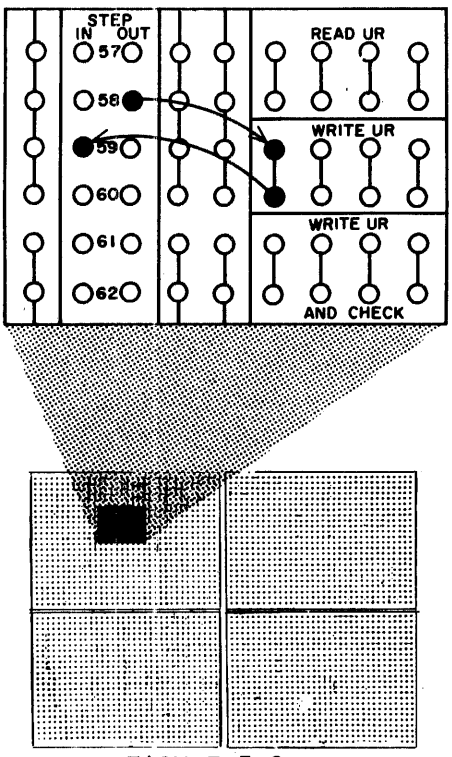


FIGURE 5-8
WRITE UNIT RECORD

Access to the general storage address register and general storage buffer is interlocked or prevented during this operation, as it is during "Read Unit Record".

The time required to execute a Write Unit Record operation (assuming that GSB and GSAR are already loaded) is the same as that required for a Read Unit Record operation of a comparable number of characters. (See the table included with Read Unit Record.)

The Write Unit Record operation is also used in the loading of overflow addresses into search control locations, as described under Search Control locations in this chapter.

The time required to write the 6-character overflow address or stop code into a search control location (assuming that GSB and GSAR are already loaded) is:

Average time (in milliseconds): 17.65
Maximum time (in milliseconds): 34.65

Note: Characters 7 through 12 of GSB are written into the dead space of the channel in addition to characters 1 through 6 which are written into the search control location, since a minimum of 12 characters is processed in this general storage operation.

Write Unit Record and Check (W/CI - W/C 4)

This operation is similar to Write Unit Record in two ways: (1) it locates the unit record or search control data held in the GSB, and (2) it stores that record in its proper location in the general storage system. It differs from Write Unit Record in one respect: after the record has been written on the drum, it is re-read to check the accuracy of the writing operation.

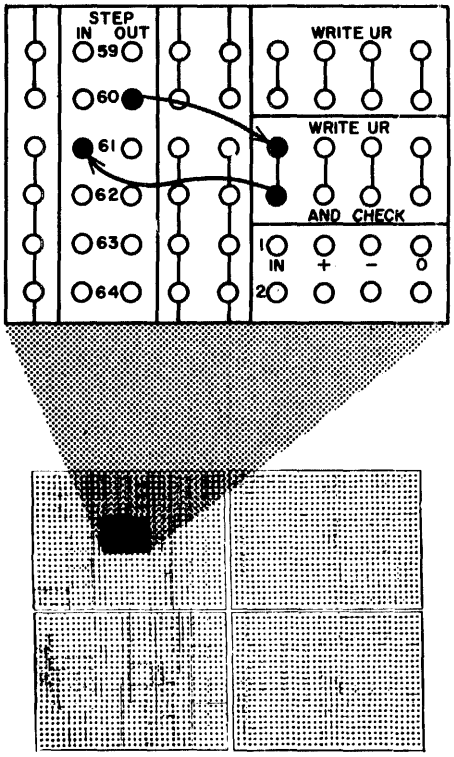
Two conditions must be met in order for this operation to be initiated:

(1) The general storage address of the specific unit record or search control location must be placed in the GSAR.

(2) The letter "P" is placed in the special character "S/C" position of the instruction word, or the WRITE UR AND CHECK hub is signaled.

Plugboard wiring is as follows:

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
60								W/C 1



WRITE & CHECK URA (W/C)		
NO	IN	OUT
1	FROM ST. 60	TO ST. 61
2		
3		
4		

FIGURE 5-9

WRITE UNIT RECORD AND CHECK

Access to the general storage address register and general storage buffer is interlocked or prevented during this operation, as in the preceding two.

The time required to execute a Write Unit Record and Check operation (assuming that GSB and GSAR are already loaded) is indicated in the following table:

Length Code:	1	2	3	4	5	6	7	8	9	0
Number of Characters:	12	24	36	48	60	72	84	96	108	120
Average Time:	51.65	52.30	52.95	53.60	54.25	54.90	55.55	56.20	56.85	57.50
Maximum Time:	68.65	69.30	69.95	70.60	71.25	71.90	72.55	73.20	73.85	74.50

The Write Unit Record and Check operation also may be used in the loading of overflow addresses into search control locations, as described under Search Control locations in this chapter.

The time required to write and check the 6-character overflow address or stop code into a search control location (assuming that GSB and GSAR are already loaded) is:

Average time (in milliseconds): 51.650
Maximum time (in milliseconds): 68.325

Note: Characters 7 through 12 of GSB are written into the dead space of this channel in addition to characters 1 through 6 which are written into the search control location, since a minimum of 12 characters is processed in this general storage operation.

Clear General Storage Buffer to Ignores (CLGSB)

This operation clears the GSB to ignores; in other words, each of the 120 characters in the buffer is replaced with an ignore code (i). It is initiated either by a "K" in the special character position of an instruction word, or by wiring of the CLEAR GSB TO IGNORES hub on the plugboard.

An important use of this operation occurs in relation to channel search operations discussed immediately below. It is essential that before initiating a channel search operation, the GSB is first cleared to ignores.

Programming to accomplish this operation is as follows:

Instruction Location	INSTRUCTION WORD				
	U	V	W	PR	S/C
					K

Plugboard:

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
54								CLGSB

CLEAR GEN'L STORAGE BUFFER (CLGSB)		
NO	IN	OUT
1	FROM ST. 54	TO ST. 55
2		
3		
4		

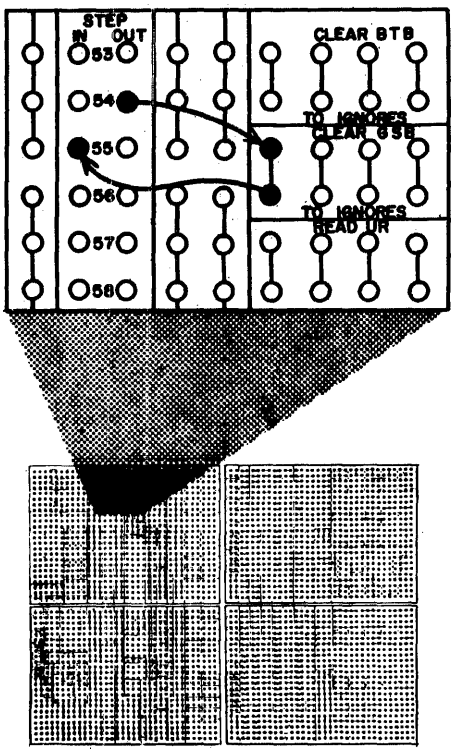


FIGURE 5-10

CLEAR GENERAL STORAGE BUFFER TO IGNORES

Access to the general storage buffer only, is interlocked during this process. Time required to execute this operation, either as a subinstruction or substep, is 5 milliseconds.

CHANNEL SEARCH OPERATIONS

Introduction

In some applications, the exact drum address for a particular unit record area is readily available or may be calculated from the source data. However, in instances where it is impossible to obtain a complete address or where it is desired to identify all records falling within a desired category, a unique feature of the Univac File-Computer, Channel Search, is extensively useful.

Channel search scans the unit records, starting with the one at the address contained in GSAR, in order to locate those that either agree or disagree, in certain character positions, with an identifier contained in the general storage buffer (GSB).

An identifier is a group of characters which can be used to isolate (1) a unique item in storage, such as an individual employee or account number, or (2) a group of individuals or accounts which satisfies a given condition, such as all employees whose labor has been charged to a certain account.

Identifiers stored in the GSB may be any length, from one to the entire 120 character length of the buffer. They may also be split, that is, combinations of characters from various areas of the buffer may form the complete identifier. This is made possible by the fact that comparison is made through the entire length of the unit record size specified.

The channel search operation is time-shared with program control storage operations of the central computer, and as such may run simultaneously with the arithmetic unit. The channel search operation establishes a condition in channel search storage which the programmer may call upon at a later time. This provides an efficient method of time-sharing, since the program may continue after the search has been initiated and a periodic check may be made to see if the channel search is completed. If the channel search is still active, the program can continue to process additional data.

In channel searching, the identifier upon which the search is conducted is contained in the general storage buffer. All other character positions in the GSB must be ignored, since the comparison is made on the entire unit record area. Initially, therefore, the GSB must be cleared to ignores (i) and the desired identifier written in the exact field on the GSB.

The role of the overflow address located in a special memory location, called a "Search Control Location", on each channel of the general storage drums, has been discussed previously in this chapter. It is important to note, however, that through proper loading of overflow addresses into the search control locations, the programmer may effectively control the channels to be searched with no restrictions as to the sequence of channels. The length code of unit records that are being searched cannot be changed while a search is in progress, but searches in various areas of the drums may be conducted on different lengths of unit records if each new search initiated contains the proper length code.

The blank area associated with each search control location is so coordinated with the drum revolutions that searching on the new channel, designated by the overflow address, can begin at the first character of the new channel. Thus, a new channel can be searched on each drum revolution; this provides an important time-saving factor during channel search operations.

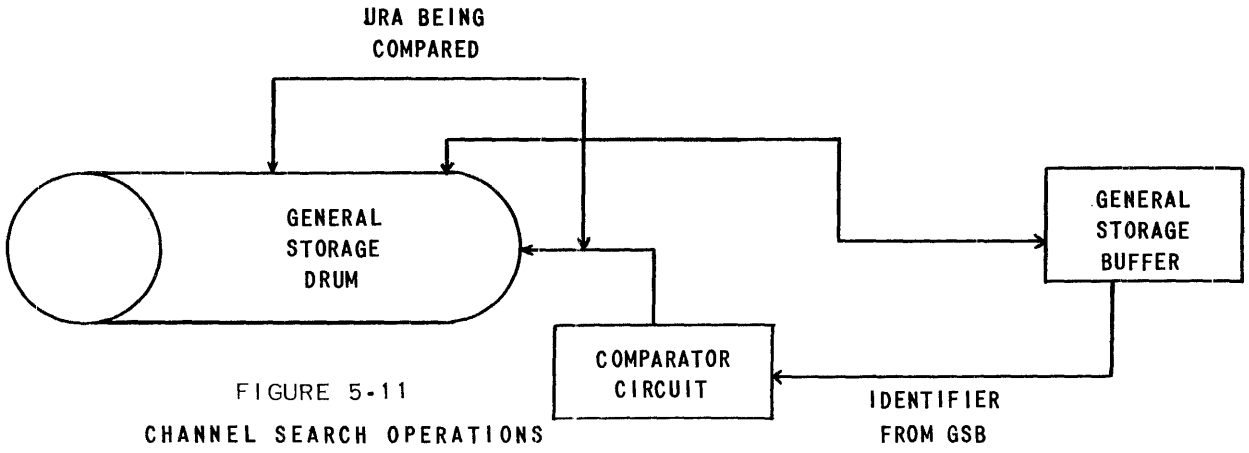


FIGURE 5-11
CHANNEL SEARCH OPERATIONS

Channel Search Equal (ECS)

Channel Search Equal is an effective means of locating a desired unit record when the identifying characteristic is known but its specific storage location on the general storage drum is unknown. For example, an employee record, inventory item, or account receivable may be identified by a badge number, part number, or account number respectively, even though the drum storage location is unknown. (This is only one of many data processing techniques where Channel Search Equal might be utilized.)

Channel Search Equal is initiated when: (1) the identifier to which each unit record is to be compared is loaded into GSB, (2) the general storage address of the starting point of the search is loaded into GSAR, and (3) the letter "N" appears in the "S/C" position of the instruction word, or the "CS=" hub on the plugboard is signaled.

Instruction Location		INSTRUCTION WORD				
		U	V	W	PR	S/C
						N

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
75								ECS-1

CHANNEL SEARCH EQUAL (ECS)		
NO	IN	OUT
1	FROM ST. 75	TO ST. 76
2		
3		
4		

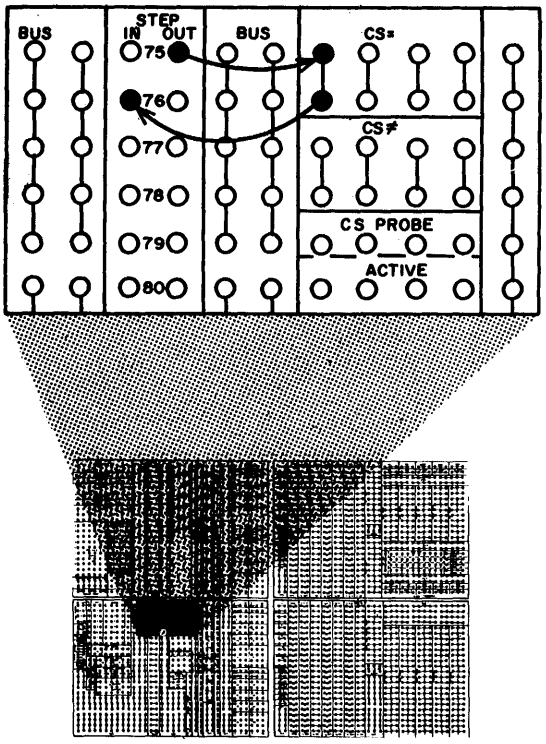


FIGURE 5-12

CHANNEL SEARCH =

Beginning at the address given in GSAR, the contents of each unit record on the specified channel are compared with the contents of the general storage buffer. Ignore codes in the GSB suppress comparison in the position in which they occur.

If an equal comparison occurs, (1) channel search storage is set to the plus (+) state, indicating that the desired record has been "found" (2) channel search terminates, (3) the unit record on which the equal comparison occurs is transferred to the GSB, and (4) its address is transferred to GSAR.

If ignore codes (i) appear in the unit record at each position for which a character other than an ignore code appears in the buffer, (1) channel search storage is set to the 0 state, indicating an "ignore found" condition, (2) channel search terminates, (3) the unit record is transferred from general storage to the general storage buffer, and (4) its address is transferred to the general storage address register.

If neither of the above conditions occurs before the six characters of the search control location are examined, these six characters are transferred to GSAR and the search is continued from the new address, if valid.

If the six characters in the search control location, however, contain an "end of file" (XXXX'0) code, channel search is terminated and channel search storage is set to the minus state, indicating a "not found" condition.

Channel Search Unequal (UCS)

Channel Search Unequal is a method of finding exceptions to a general rule, such as all direct labor employees who charge time to a certain direct labor account, or all inventory items in stock that have had withdrawals during the past month. During the loading of information on the general storage drums, Channel Search Unequal may be used to locate the first "open" unit record on which to write the next item. (Many other applications of Channel Search Unequal can be developed by an ingenious programmer.)

Channel Search Unequal is initiated when: (1) the identifier, to which each unit record is to be compared, is loaded into GSB, (2) the general storage address of the starting point of the search is loaded into GSAR, and (3) the letter "O" appears in the "S/C" position of the instruction word, or the "CS≠" hub on the plugboard is signaled.

Beginning at the address given in GSAR, the contents of each unit record on the specified track is compared with the contents of the general storage buffer.

Instruction Location	INSTRUCTION WORD				
	U	V	W	PR	S/C
					O

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
75								UCS-1

CHANNEL SEARCH UNEQUAL (UCS)		
NO	IN	OUT
1	FROM ST. 75	TO ST. 76
2		
3		
4		

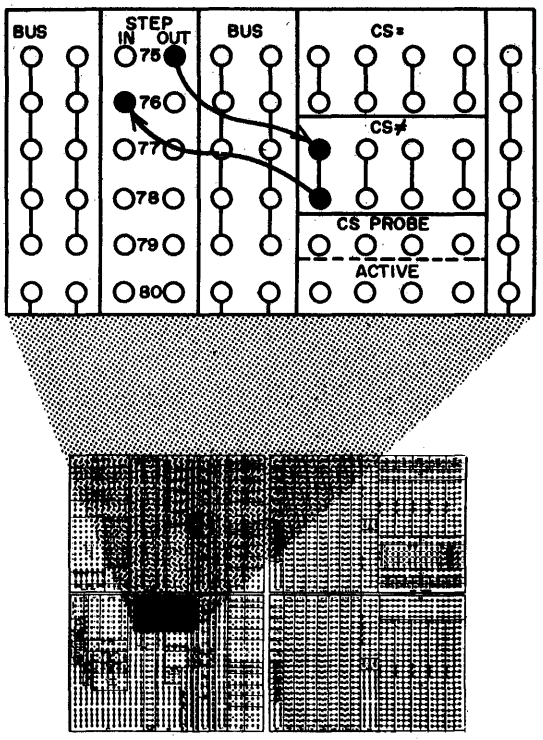


FIGURE 5-13

CHANNEL SEARCH ≠

Ignore codes in the buffer suppress comparison in the positions in which they occur.

If no unequal comparison occurs before the six characters in the search control location are examined, these six characters are transferred to GSAR and the search is continued from the new address, if valid.

If the six characters in the search control location, however, contain an "end of file" (XXXX'0) code, channel search is terminated and channel search storage is set to the minus state, indicating a "not found" condition.

Note that an "ignore found" condition is not possible on channel search unequal.

Access to the general storage address register and general storage buffer is interlocked or prevented during this operation.

Channel Search Probe (CSP)

Channel search operations described above may proceed as time-shared operations while program control proceeds to execute other instruction words or program steps. Periodically, the main program may interrogate channel search storage to determine the status of the search previously initiated. This interrogation is performed either by a channel search probe instruction (SP) in the process section of an instruction word as below, or by plugboard wiring as later discussed.

Instruction Location	INSTRUCTION WORD											
	U			V			W			PR		S/C
	2	0	1	3	1	0	4	0	0	5	P	A

Assuming that Channel Search is in progress and the instruction word Search Probe (SP) is to be executed, the following conditions and results would be in force:

(1) If channel search (equal or unequal) is not yet completed, the next instruction is taken from the U address of the instruction word.

(2) If search is completed unsuccessfully (not found), channel search storage is set to the minus (-) condition, and the next instruction word is taken from the V address of the instruction word.

(3) If search is completed and "ignore found" has occurred, thereby setting the channel search storage to the zero (0) condition, the next instruction word is taken from the W address of the instruction word.

(4) If search is completed successfully and the "found" has occurred, thereby setting channel search storage to the plus (+) condition, the next instruction is taken from the PAK.

Where a series of subroutines are to be processed during the time a channel search is in progress, a series of Channel Search Probe instructions provide the means of initiating these subroutines. At the end of each subroutine, a jump instruction should be used to return program control to the main program.

If channel search storage is to be interrogated on the plugboard, the CS PROBE hub can be signaled to initiate a substep which tests whether or not the channel search has been completed.

(1) If channel search is not completed, a pulse will be emitted from the ACTIVE hub; if channel search is completed, a pulse will be emitted from the +, -, or 0 hub, depending on whether channel search storage has been set to +, -, or 0 as a result of the channel search.

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
76								CS P

CHANNEL SEARCH PROBE (CS P)					
NO	IN FROM	ACTIVE	+	−	0
1	ST. 76	ST. 77	ST. 80	ST. 83	ST. 85
2					
3					
4					

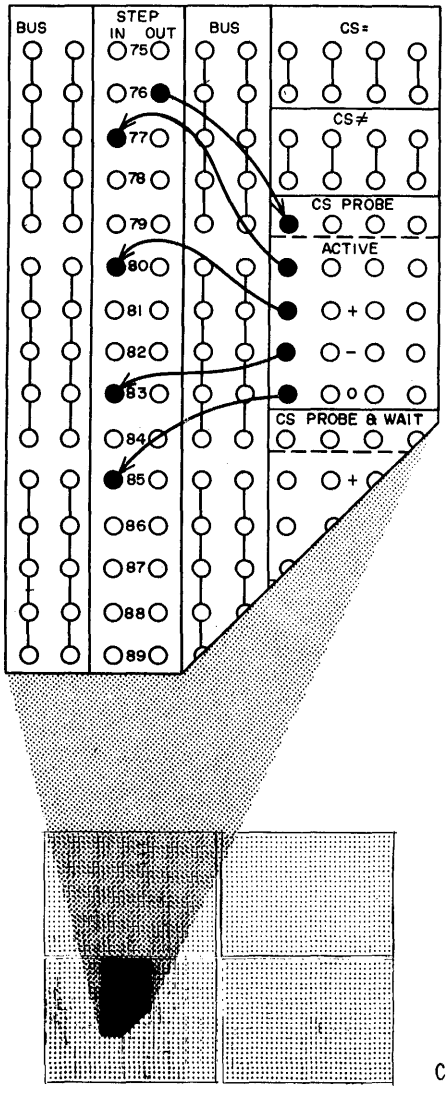


FIGURE 5-14
CHANNEL SEARCH PROBE

Where it is desirable to proceed with subroutines during the time a channel search is in progress, the signalling of the ACTIVE hub in each channel search probe operation may initiate a subroutine. It is advisable, however, to provide a means of keeping track of which subroutines have been completed. For example, picking up a selector could prevent repetition of a subroutine once it has been performed in a channel search probe sequence.

The CS PROBE & WAIT hub can also be signaled subsequent to the initiation of a channel search equal or unequal substep. In this case, however, the substep initiated causes program control to wait until the channel search is completed. When the channel search is completed, a signal is emitted from the appropriate +, -, or 0 hub as in the channel search probe substep, and the program continues.

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
82								CS P/W

CHANNEL SEARCH PROBE & WAIT (CS P/W)

NO	IN FROM	+	−	0
1	St. 82	St. 85	St. 88	St. 89
2				
3				
4				

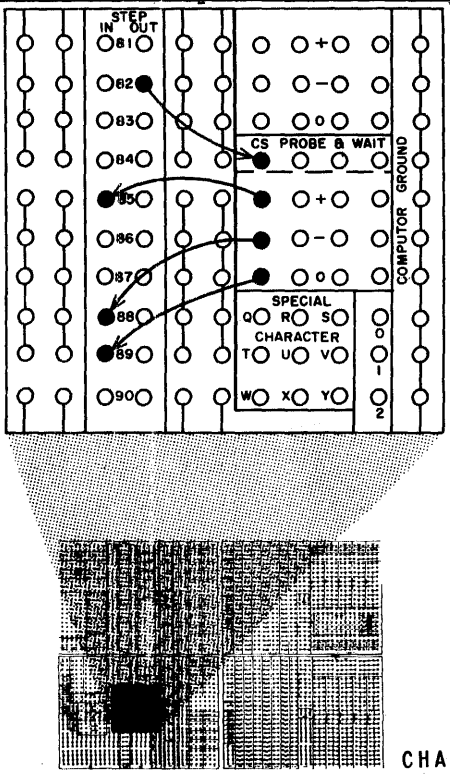


FIGURE 5-15
CHANNEL SEARCH PROBE AND WAIT

Access to the general storage buffer is prevented during this operation. Channel search probe is not considered a general storage operation, but is included here because of its logical relation to general storage operations involving channel search.

Example of a Channel Search

The following example indicates a possible sequence of events in a channel search operation. Comparable internal and plugboard programs which accomplish the same purpose are shown:

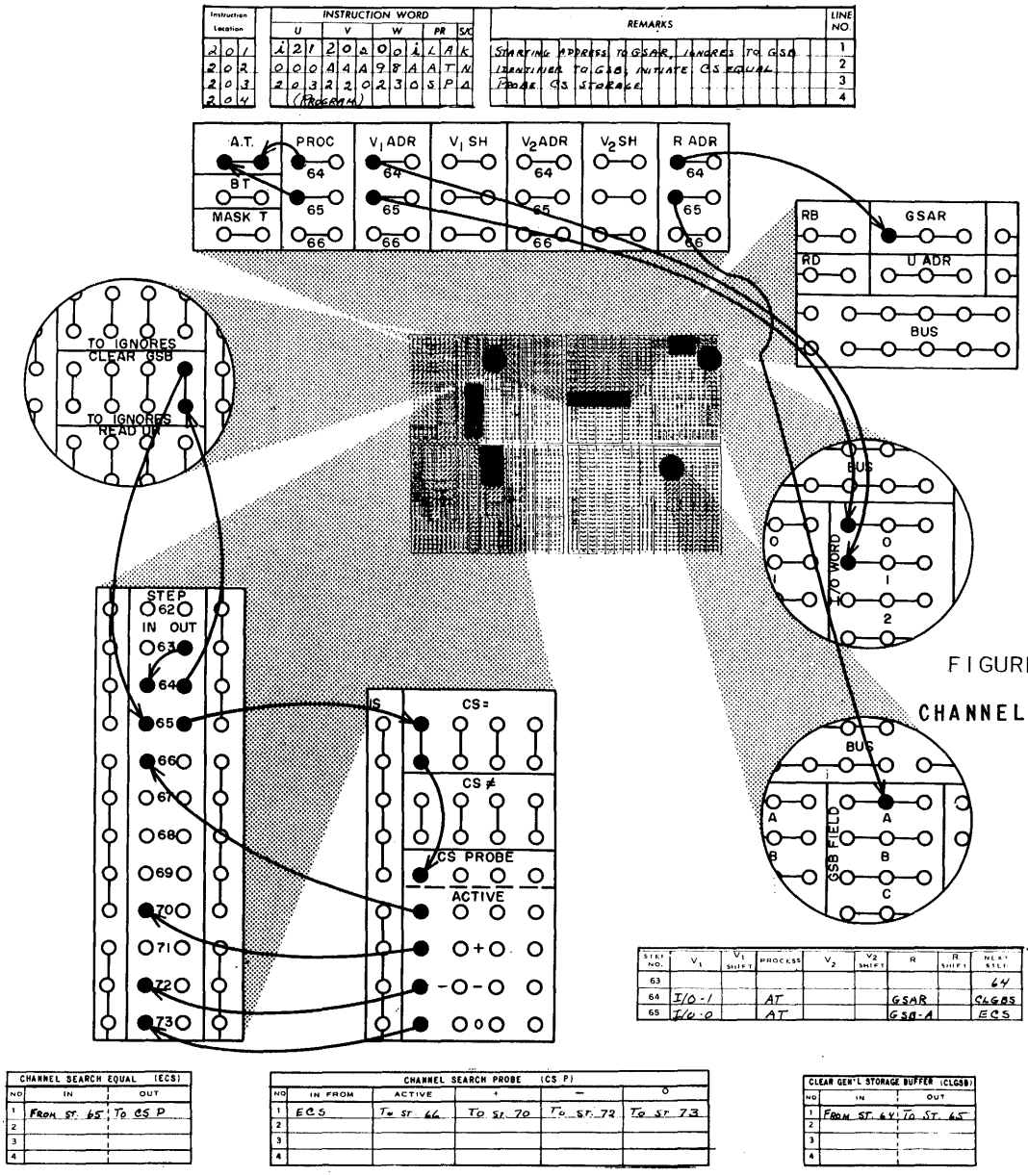


FIGURE 5-16

CHANNEL SEARCH

CHANNEL SEARCH EQUAL (ECS)		
NO	IN	OUT
1	FROM ST. 65	TO 65 P
2		
3		
4		

CHANNEL SEARCH PROBE (CS P)					
NO	IN FROM	ACTIVE	+	−	0
1	ECS	To ST 66	To ST 70	To ST 72	To ST 73
2					
3					
4					

CLEAR GEN'L STORAGE BUFFER (CLGSB)		
NO	IN	OUT
1	FROM ST. 64	TO ST. 65
2		
3		
4		

Assumptions:

The general storage drum contains inventory items, each one 24 characters in length, with data arranged as follows:

Price (5)	On Order (3)	On Hand (4)	Part Number (12)

The item to be found by channel search equal contains the following information and may be addressed by the indicated fields when this information is held in the general storage buffer.

0 1 2 3 5	2 0 8	0 9 4 7	A 4 7 3 4 7 2 9 0 0 D F
Field D	Field C	Field B	Field A

Instruction Word 201
or
Program Step 64

Load starting address of search into General Storage Address Register:

GSAR	2	1	2	0	2	0	0
	L	D	S	C	H	A	A

Sub-instruction K or substep "CLEAR GSB TO IGNORES" accomplishes that purpose. General storage buffer would appear as follows:

i	i i	i i i i i	i i i	i i i i	i i i i i i i i i i i i
Field V	Field E	Field D	Field C	Field B	Field A

Instruction Word 202
or
Program Step 65

Identifier transferred from Input/ Output track to Field A of General Storage Buffer:

i	i i	i i i i i	i i i	i i i i	A 4 7 3 4 7 2 9 0 0 D F
Field V	Field E	Field D	Field C	Field B	Field A

Sub-instruction "N" or substep "CS=" initiates channel search equal.

Each unit record of channel 02 and subsequent overflow channels is compared to the identifier held in the general storage buffer. The general storage address register advances to the address of each unit record as that unit record is compared to the identifier.

Instruction Word 203
or
CS PROBE hub

Probe channel search storage:

If Search is still in progress:

Internal:
Program continues at instruction word 203 (channel search storage will be probed again until the search is completed).

External:
Plugboard hub ACTIVE emits a pulse and the program continues at program step 66.

If the search is unsuccessful:

Internal:
Channel search storage is set to a minus (-) condition, and the internal program continues at instruction word 220.

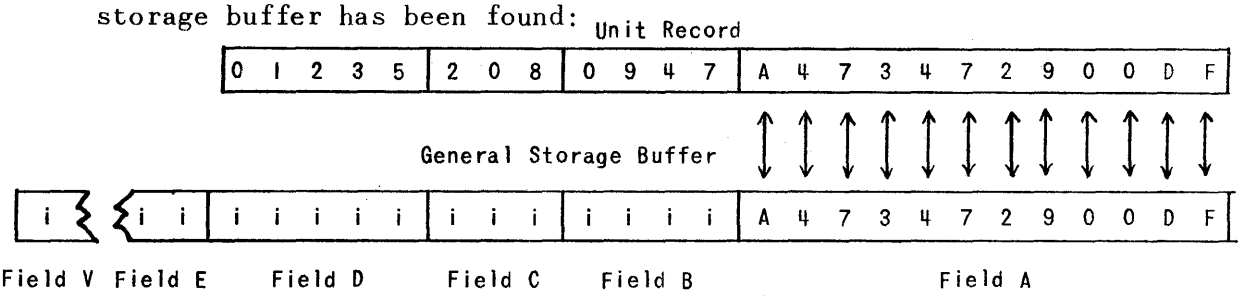
External:
Plugboard hub "-" emits a pulse and the program continues at program step 72.

If the search is successful:

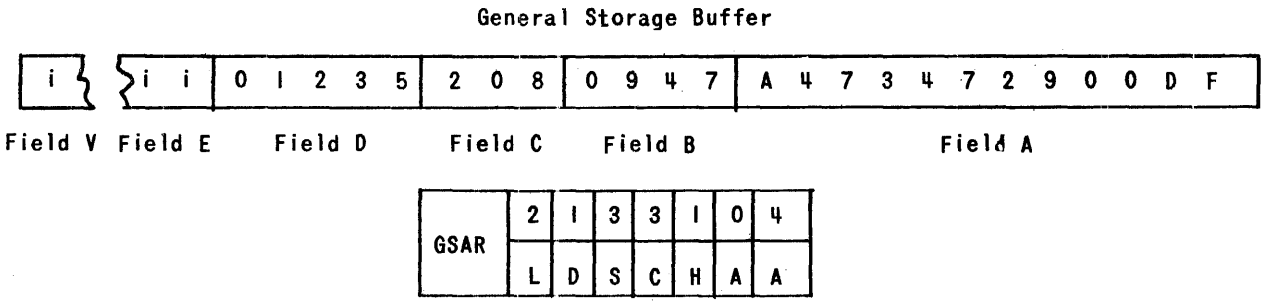
Internal:
Channel search storage is set to a plus (+) condition, the unit record is automatically read into the general storage buffer, and the internal program continues at instruction word 204.

External:
Plugboard hub "+" emits a pulse, the unit record is read into the general storage buffer, and the program continues at program step 70.

A unit record containing the same identifier as Field A of the general storage buffer has been found:



The found unit record is transferred into the lower order character positions of the general storage buffer and the address of the unit record is held in the general storage address register.



This assumes that the record was found in drum section 13, channel 31, unit record 04.

If the search ends in an "ignore found" (possible only on channel search equal):

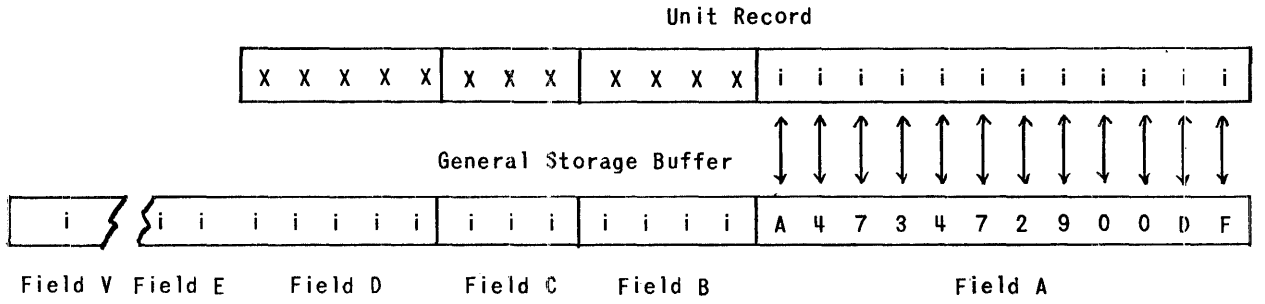
Internal:

Channel search storage is set to a zero (0) condition, the unit record at which the ignores were found is automatically read into the general storage buffer, and the address of that unit record is held in the general storage address register. The internal program continues at instruction word 230.

External:

Plugboard hub "0" emits a pulse, the unit record is read into the general storage buffer, and the program continues at program step 73.

A unit record containing ignore codes (i) in all of the identifier character positions is found:



The found unit record is transferred into the lower order character positions of the general storage buffer and the address of that unit is held in the general storage address register as indicated in the successful search above.

For further study of some of the methods of locating data in the general storage drums, turn to Appendix C, General Storage Drum Addressing Subroutines.

chapter 6

INPUT-OUTPUT SYSTEM

INTRODUCTION

The components of the UFC-I Input/Output System which are of special interest to the programmer include the demand stations, input/output tracks of the high speed drum and their track switching circuitry, input/output control lines, and the various input/output devices which may be included in any UFC-I system. These devices are the

Console System
Inquiry Typewriter
90 Column Card System (with post-read checking)
80 Column Card System (Bull)
UFC High Speed Printer
High Speed Paper Tape System
UFC Magnetic Tape Unit
UFC Sort-Collate System
Airline Reservation System

Several time-sharing and time-saving features contribute to the ability of the UFC-I input/output system to transfer large volumes of data into and out of the central computer. To gain maximum efficiency in accomplishing a given data processing task, the programmer should always attempt to exploit to the fullest extent the following time-sharing features provided in this system:

Each I/O unit is capable of performing its functions independently, without being subject to control by the central computer, except during the intervals when the computer and the I/O device are exchanging control information.

Track switching circuits enable the computer and each I/O device to share the use of pairs of I/O tracks on the high speed drum, providing for the simultaneous reading or writing of data on one track while the computer processes data from the other track.

Without interruption of its activities, each I/O unit may be tested to determine its ready or not ready condition. If an I/O unit is busy (not ready) at the time it is tested, the computer may proceed with its program or may communicate with other I/O units. Thus, several I/O units may be tested in sequence to determine which one is currently ready for further instructions.

COMPONENTS OF INPUT/OUTPUT CONTROL

Demand Station

A demand station is a group of circuits that serves as the communication center between the central computer and the I/O units employed in a UFC-I system. Because all demand stations are logically identical, an I/O unit may be plugged into any one of the ten demand station connectors on the program control cabinet of the computer.

The principal function of a demand station is to facilitate the exchange of information between the central computer and the I/O units. Through the demand station, the *computer* is able to

test the status of an I/O unit to determine whether it is ready or not ready, and

put a unit "on demand" so that a) control information may be given to the specific I/O unit, b) control information may be received from that I/O unit, and c) access may be provided to the I/O tracks associated with that I/O unit through the plugboard addressing system.

Through the demand station, the *I/O unit* is able to

notify the computer regarding the I/O unit's ready or not ready status,

receive from the computer the instruction the I/O unit is to perform,

and notify the computer of conditions existing in the I/O unit so that the computer may modify its program accordingly.

Other functions of the demand station include track switching on command of the central computer, and the routing of data to the specific I/O unit on demand.

When an I/O unit is placed "on demand", either by internal or external programming, that unit remains "on demand" until it receives an I/O instruction from the central computer, or until some other I/O unit is placed "on demand". Most UFC-I system I/O units automatically take themselves "off demand" when an I/O instruction is received. Thus they are "not ready" for further instructions until the operation initiated by the current instruction is completed.

Input/Output Tracks

When the demand station of an I/O unit is plugged into a demand station connector, the pair of I/O tracks associated with that connector on the high speed drum is automatically assigned to that I/O unit. The central computer and I/O unit may then share the use of this pair of tracks, the computer communicating with one track while the I/O unit communicates with the other. (See Chapter 2.) When a "track switch" operation is performed, the computer communicates with the I/O track previously used by the I/O unit and vice versa. If less than ten I/O units are employed, the I/O tracks not associated with a unit are available for use as additional storage locations. No track switch can be performed, however, on I/O tracks to which an I/O unit is not connected.

Only the pair of I/O tracks associated with the I/O unit currently on demand is available through the plugboard addressing system. Internally, however, information on any I/O track is directly addressable regardless of the demand status of the associated I/O unit.

Track Switch

One of the outstanding time-sharing features of UFC-I is the circuitry which allows the computer to have access to data from one of a pair of I/O tracks while the I/O unit is in communication with the other. For example, the com-

puter may be processing data from the left track of I/O 5 while the I/O unit is loading new information onto the right track of I/O 5. Then a "track switch" is performed and the newly loaded data on the right track becomes available to the computer while the left track is loaded with new information.

Track switch is initiated by the wiring of a pulse source to TRACK SWITCH (0-9) hubs on the plugboard or by the insertion of a 1 in the proper positions of Test Incoming Control and Demand In instruction words.

When a track switch command is initiated, it is performed immediately whether or not the I/O unit is on demand and/or ready. Therefore it is advisable to program a demand test in sequence to determine whether an I/O unit is ready before initiating a track switch of the corresponding I/O tracks.

Computer - Input/Output Lines (A-J)

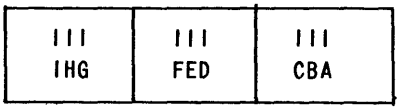
Computer - I/O control lines provide the means of sending instructions from the central computer to the I/O unit. Only the I/O unit which is on demand and ready receives the information sent via these lines.

The instruction which is executed by an I/O unit when it receives a signal over one or more of the computer - I/O control lines depends on the characteristics of the I/O unit. The magnetic tape unit, for example, has been assigned a complete set of instructions and corresponding computer - I/O control lines. In other I/O devices, however, all or most of the computer - I/O instructions and the control lines are coordinated through proper wiring.

Externally, the choice of the I/O - computer control lines is made by wiring a pulse source, usually a DEMAND OUT hub or a SPECIAL OUT hub to the computer - I/O CONTROL LINES desired.

Internally, the computer -- I/O control lines are set up by a pattern of bits in the V-section of the "Demand In" instruction word. In this usage, the computer interprets only the three low-order bits of each character. In each bit position where a "1" bit is found, the corresponding computer - I/O control line is activated.

V section of Instruction Word



Although other computer characters containing the same low-order bits (see "Univac Code" in Chapter 1) may be used to activate these lines, a pattern composed of the digits 0-9 is used in most programs. An example of a method of coding V to activate each of the computer - I/O lines is given below:

Computer - I/O Line Activated	Coding of V-section Digits	Bit Pattern
A	556	000 000 001
B	557	000 000 010
C	559	000 000 100
D	565	000 001 000
E	575	000 010 000
F	595	000 100 000
G	655	001 000 000
H	755	010 000 000
I	955	100 000 000

The following table is a further example of the method used in programming control instructions via the computer - I/O lines. The pattern in this example illustrates the instructions performed by the UFC-I magnetic tape unit. Only the four lower order bits of the 9 bit pattern are used in this instance because combinations of lines A, B, C and D *only* are used by the magnetic tape unit.

Instruction	Instruction No.	External OCL used	Internal V-section Digits	Bit Pattern
Read Forward	1	---A(0001)	556	000 000 001
Search Forward =	2	--B-(0010)	557	000 000 010
Search Forward = or >	3	--BA(0011)	558	000 000 011
Wind Forward	4	-C--(0100)	559	000 000 100
Write	6	-CB-(0110)	553	000 000 110
Write & Check	7	-CBA(0111)	554	000 000 111
Tr. Tape Unit Buffer contents to I/O Track	8	D---(1000)	565	000 001 000
Read Backward	9	D--A(1001)	566	000 001 001
Search Backward =	10	D-B-(1010)	567	000 001 010
Search Backward = or <	11	D-BA(1011)	568	000 001 011
Rewind	12	DC--(1100)	569	000 001 100
Rewind with Interlock	13	DC-A(1101)	562	000 001 101
Tr. Contents of I/O Track to Tape Unit Buffer	15	DCBA(1111)	564	000 001 111

I/O - Computer Control Lines (a-l)

The I/O - computer control lines (a-l) are low speed lines which may be energized when a particular I/O unit goes on demand. These lines may be caused to emit a B+ current by signals originating in the I/O unit during the time an I/O unit is "on demand", and are normally used to activate selectors on the main program control plugboard.

Externally the control information received via a low speed I/O - computer control line is put into effect by wiring the I/O - COMPUTER CONTROL LINE (a-l) hubs to a SELECTOR PICKUP hub or other hubs capable of accepting a B+ current.

Internally the control information received via an I/O - computer control line is put into effect by programming a transcop or breakpoint sequence so that the I/O - COMPUTER CONTROL LINE (a-l) hubs may control the program as described in the preceding paragraph. In other words, program modification via I/O - computer control lines (a-l) can only be effected by plugboard wiring.

High Speed I/O - Computer Control Lines (W, X, Y, Z)

The high speed I/O - computer control lines W, X, Y, and Z provide a means by which the I/O unit notifies the central computer of conditions which may exist in the I/O unit. For example, a UFC-I magnetic tape unit may have detected an "End of Tape" signal in the tape; this information must be relayed to the central computer immediately so that appropriate action may be taken.

Whenever pulsing of a DEMAND IN hub results in a signal being emitted by a SPECIAL OUT hub on the plugboard, or a Demand In instruction results in a special out jump to the W address, this indicates that a condition (such as "End of Tape") exists in the I/O unit and some programmed action is required before the program continues. High Speed I/O - Computer Control Line Storage "remembers" which condition is found. HS I/O - computer control line storage must then be tested by plugboard wiring as in Diagram 6-1, or by the execution of a series of Test Incoming Control instruction words to determine which condition or combination of conditions, W, X, Y, or Z, is being remembered by High Speed I/O - computer control line storage.

As many as sixteen different conditions may be indicated by combinations of W, X, Y and Z. Therefore, when a special out condition arises, it may be necessary to probe the W, X, Y and Z hubs successively on the plugboard, or to initiate a series of Test Incoming Control instructions internally in order to determine the exact status of the I/O unit which indicated the condition.

Demand Test In Hubs

The plugboard hubs labeled DEMAND TEST IN (0-9), READY (0-9) and NOT READY (0-9) and the corresponding functions performed internally, are discussed in a following section, in their relationship to I/O instructions.

Demand In Hubs

The plugboard hubs labeled DEMAND IN (0-9), DEMAND OUT (0-9) and SPECIAL OUT (0-9), and the corresponding functions performed internally are discussed below in their relationships to I/O instructions.

INPUT - OUTPUT INSTRUCTIONS

Demand Test In

The Demand Test In sequence is used to determine the "ready" or "not ready" status of a particular I/O unit. A unit is considered "ready" when a) it

is not engaged in the execution of a previously initiated operation *and* where b) it has no abnormal or erroneous condition present which would prevent the I/O unit from functioning properly.

Internally, a Demand Test In sequence is performed by the execution of a Test Demand In, TD, instruction word. If the I/O unit tested is ready, the next instruction is taken from the V address of the TD instruction word. If the unit is not ready, the next instruction is taken from the W address of the TD instruction word. (See Test Demand In, in the repertory of instructions, Chapter 4.)

Externally, the "demand test in" substep is accomplished by the wiring of a pulse source to the DEMAND TEST IN hub associated with the I/O unit the status of which is to be tested, and the wiring of the corresponding READY and NOT READY hubs to continue the program.

In the plugboard sequence shown in diagram 6-1, the START hub has been wired to DEMAND TEST IN of I/O unit 0.

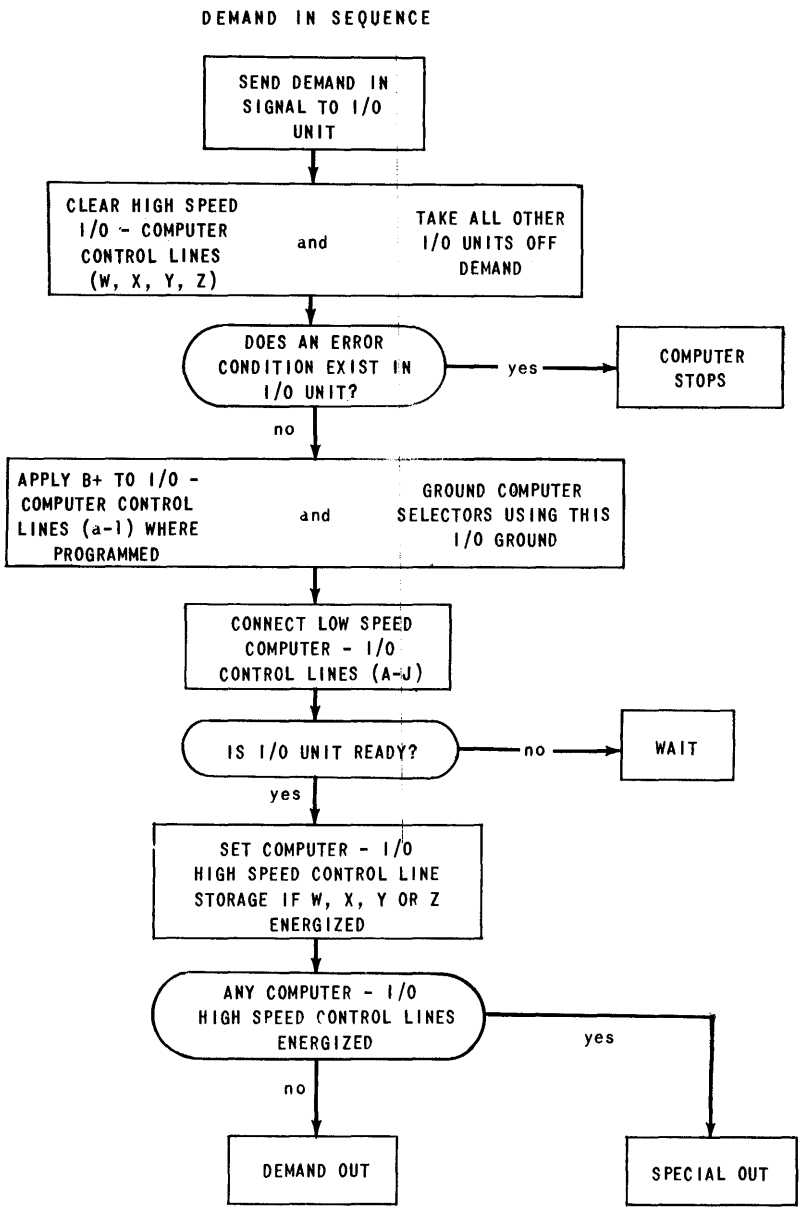
If 0 is ready, a pulse will be received from the READY hub and I/O unit 0 will be demanded by pulsing the DEMAND IN hub. If I/O unit 0 is not ready, a pulse will be emitted by the NOT READY hub which has been wired to the DEMAND TEST IN hub associated with I/O unit 1.

If I/O unit 1 is ready, a pulse travels from the READY hub to the DEMAND IN hub of I/O unit 1. If I/O unit 1 is not ready at this time, the pulse emitted by its NOT READY hub is directed to DEMAND TEST IN of I/O unit 0 to determine whether it is ready or not ready.

In the above manner, several I/O units may be tested in sequence in order to determine which I/O unit is currently ready to supply information to, or receive information from the computer.

Demand In

The Demand In sequence is usually used to place a particular I/O unit "on demand", so that control information may be exchanged via the computer - I/O control lines (A-J), low speed I/O computer control lines (a-l) and the high speed I/O - computer control lines W, X, Y and Z. It is also used to place an I/O unit on demand so that storages on the corresponding I/O track may be referred to on the plugboard (the reference must occur during the time the I/O unit is on demand). Also, during the time the I/O unit is on demand, any selectors wired to the DEMAND GROUND (0-9) hub corresponding to the I/O unit demanded, are grounded and may be activated.



Internally, a demand in sequence is performed by the execution of a Demand In instruction word, as described in detail in the repertory of instructions in Chapter 4.

Externally, the demand in substep is defined by the wiring of a pulse source to the DEMAND IN (0-9) hub corresponding to the I/O unit being placed on demand, and the wiring of the DEMAND OUT (0-9) and SPECIAL OUT (0-9) hubs associated with that DEMAND IN to hubs which define the operations which will continue the program.

Diagram 6-1 includes a possible Demand In sequence on the plugboard:

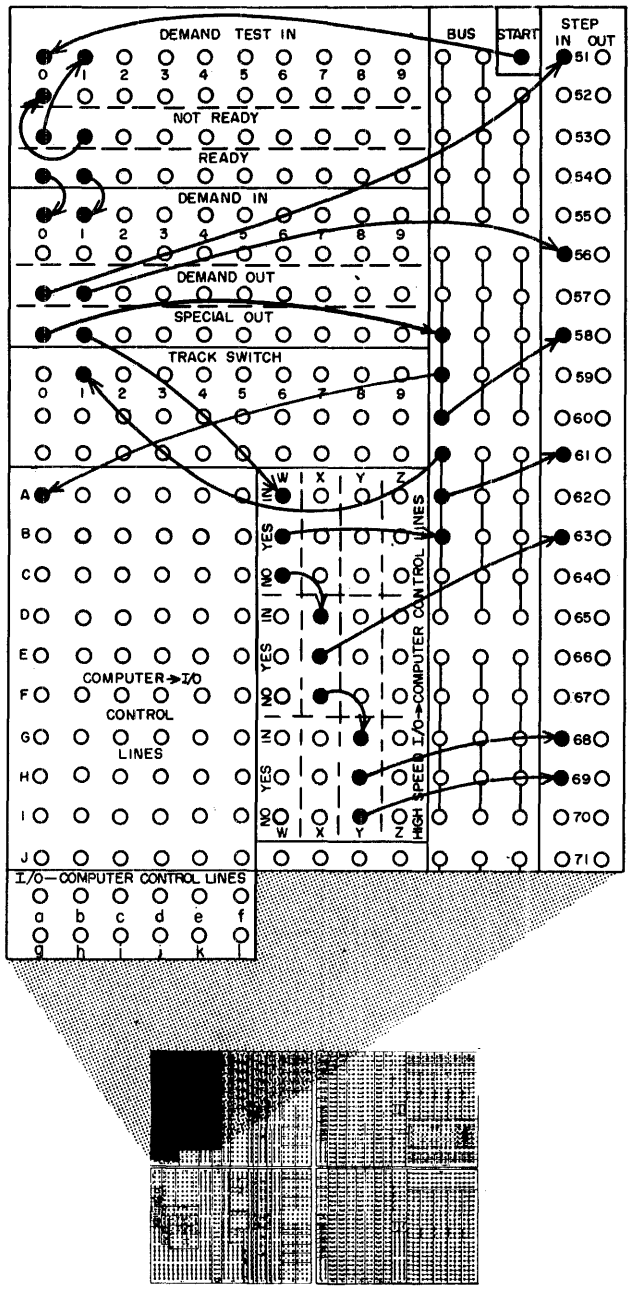


FIGURE 6-1

INPUT-OUTPUT HUBS

START TO:
TEST IN - DMD 0

HI-SPEED CONTROL LINES (HCL)			
NO	IN	YES	NO
W	SPEC. OUT - DMD 1	ST. 61 TRACK SWITCH-1	IN - HCL X
X	NO. HCL W	ST. 63	IN - HCL Y
Y	NO. HCL X	ST. 68	ST. 69
Z			

OUTPUT CONTROL LINES (OCL)			
NO	FROM	NO	FROM
A	SPEC. OUT - DMD-0	F	
B		G	
C		H	
D		I	
E		J	

DEMAND UNITS (DMD)							
UTS	TEST IN	NOT READY	READY	DEMAND IN	DEMAND OUT	SPECIAL OUT	TRACK SWITCH
0	FROM: START NR, DMD-1	TEST IN-DMD 1	DEMAND IN-0	READY-0	ST. 51	OCL A; ST. 58	
1	NR-DMD-0	TEST IN-DMD 0	DEMAND IN-1	READY-1	ST. 56	HCL W	YES, HCL W

When DEMAND IN associated with I/O unit 0 is pulsed, a DEMAND OUT directs the program to STEP IN of step 51. A SPECIAL OUT, however, pulses COMPUTER - I/O CONTROL LINE A, and directs the program to STEP IN of step 58.

When DEMAND IN 1 is pulsed, a DEMAND OUT directs the program to STEP IN of step 56. A SPECIAL OUT, in this instance, is directed to test the HIGH SPEED I/O - COMPUTER CONTROL LINES W, X, Y, Z to determine the condition which caused the SPECIAL OUT and to take appropriate action by pulsing STEP IN of one of the steps numbered 61, 63, 68 or 69.

Test Incoming Control

When a Demand In sequence results in the emission of a pulse from a Special Out hub on the plugboard or a jump to the W address during the execution of a Demand In instruction, a series of test Incoming Control signals must be programmed to determine the exact condition or conditions present in the I/O unit which caused the Special Out situation.

Internally, the function of testing the incoming control signals received via the high speed I/O - computer control lines W, X, Y, Z, is performed by the execution of a series of Test Incoming Control (TI) instructions. This instruction is described in detail in the Repertory of Instructions, Chapter 4.

Externally, Test Incoming Control is usually initiated by a pulse emitted by a SPECIAL OUT hub. This pulse is directed by plugboard wiring to test the W, X, Y, Z conditions remembered by high speed I/O - computer control line storage in order to determine which incoming condition or conditions caused the SPECIAL OUT hub to emit. (See Diagram 6-1.)

PERIPHERAL EQUIPMENT OF THE UFC-I SYSTEM

Console System

The UFC-I Console system consists of a console control panel, a UFC Inquiry Typewriter, a typewriter control panel and a desk which contains the control circuitry.

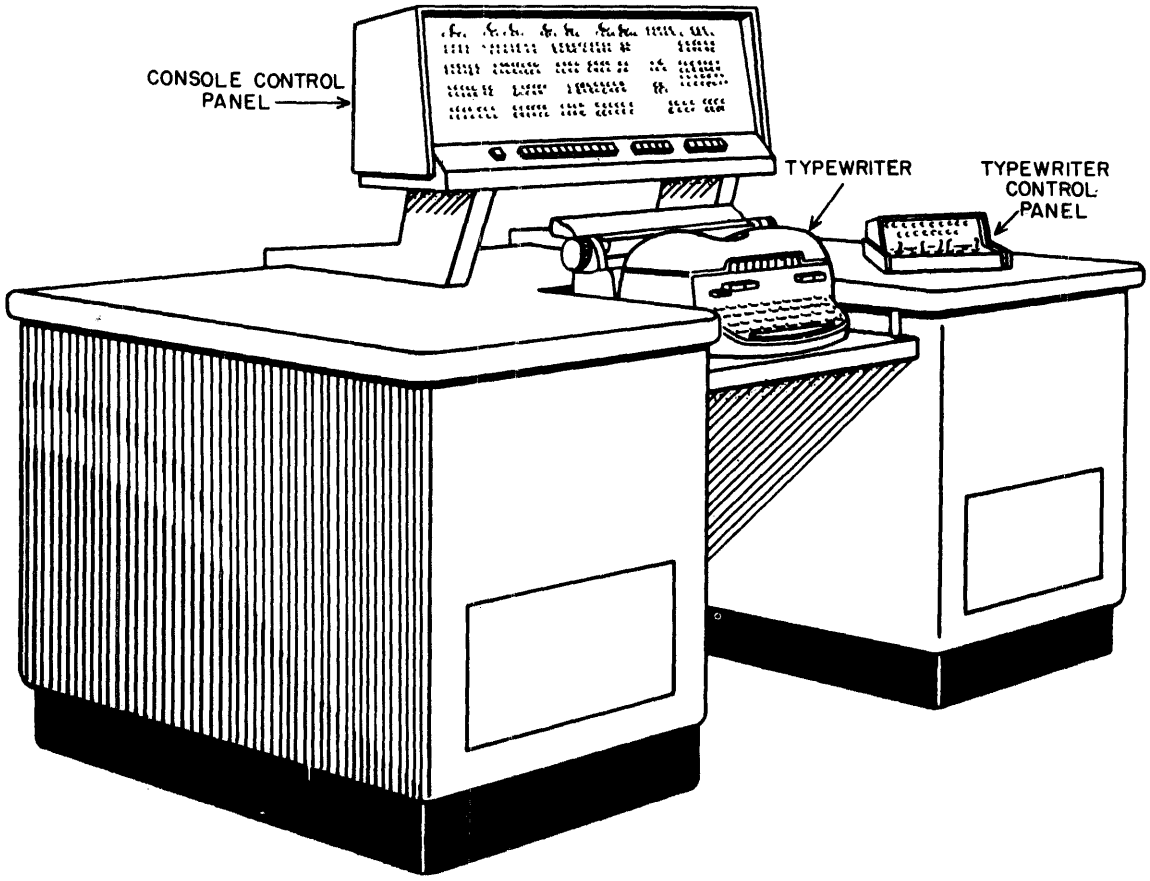


FIGURE 6-2

CONSOLE SYSTEM

The console control panel provides the operator with a visual display of lights which indicate the progress of the computer program. The panel also provides controls which enable the operator to alter the program when necessary.

Demand station "0" is assigned to the console system when the console includes the inquiry typewriter. The modes of operation of the typewriter in its relationship to the console system are discussed under the section "Inquiry Typewriter".

Inquiry Typewriter

The UFC Inquiry Typewriter is a Remington Rand Encoding/Decoding Typewriter with the additional manual controls and electronic circuitry necessary for its use in on-line, two-way communication with the UFC-I system. Operation of the typewriter is essentially manual; however, an automatic output mode is provided which allows repetitive output from one selected word address in the central computer.

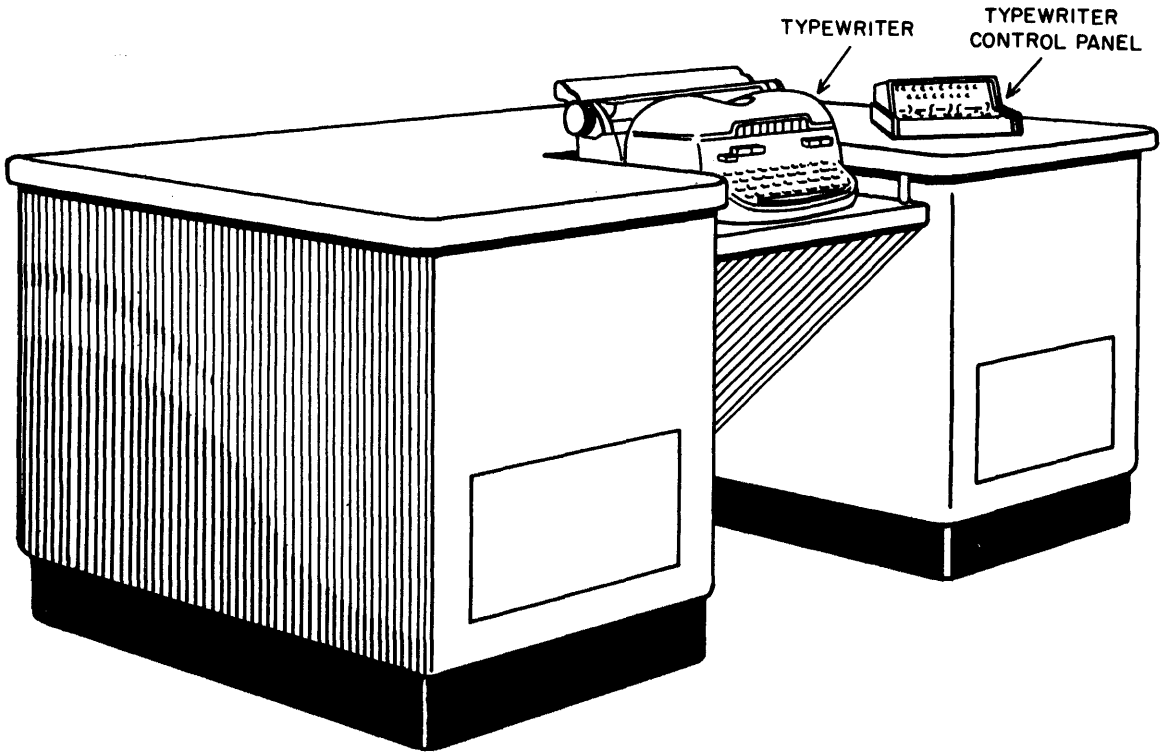


FIGURE 6-3

INQUIRY TYPEWRITER

In its *input mode,* the inquiry typewriter may execute either a NORMAL or CONSOLE type of operation. The NORMAL operation provides a means of entering information manually into the computer for use in a manner determined by the computer program. The CONSOLE operation provides a means by which an operator may intervene at any time during a program run for the purpose of changing or modifying the program.

In its *output mode,* the inquiry typewriter may execute either a MANUAL or AUTOMATIC type of operation. Under MANUAL operation, each computer word address is selected manually by depressing the appropriate WORD ADDRESS button. As each word address is selected, the contents of that location are automatically typed out. Under AUTOMATIC operation, the computer program selects the successive words to be typed out and transfers them into the word location indicated by the WORD ADDRESS button. The typewriter automatically types out the contents of the word address as the computer places each new word in that address.

Computer - I/O Instructions

Computer - I/O control lines A, B, and C are used to send to the typewriter signals which exercise control over typewriter operation.

Control line A takes the typewriter "off demand", and places it in a "not ready" condition. The typewriter becomes "ready" when the "COMPUTE" button on the typewriter is depressed.

Control line B is used in the output mode to initiate the typing of the word designated by the particular WORD ADDRESS button that is depressed on the typewriter control panel.

Control line C is used in conjunction with control line B in the automatic output mode of typewriter operation. Control line C sets the typewriter to the "ready" condition as soon as the word initiated by control line B is typed.

I/O - Computer Instructions

Signals sent to the computer over low speed I/O - computer control lines a, b, c and d may be used to modify the computer program. As each of four switches, labeled a, b, c and d on the typewriter control panel, are activated by the operator, a B+ current flows from the corresponding hub on the program plugboard. This current is normally used to activate selectors which modify the computer program.

Demand Station

Whenever the inquiry typewriter is used in conjunction with the UFC-I console or when the typewriter, by itself, is operated in the "console mode", demand station 0 is assigned to the typewriter.

90-COLUMN CARD SYSTEM (WITH POST-READ CHECKING)

The UFC 90-Column Card System (with Post-Read Checking) may be used as a UFC tabulating card input unit, as a card output unit, or as a combined input-output unit. The "post-read" checking consists of an automatic re-reading of the processed card and a checking of it against comparison data stored in the system.

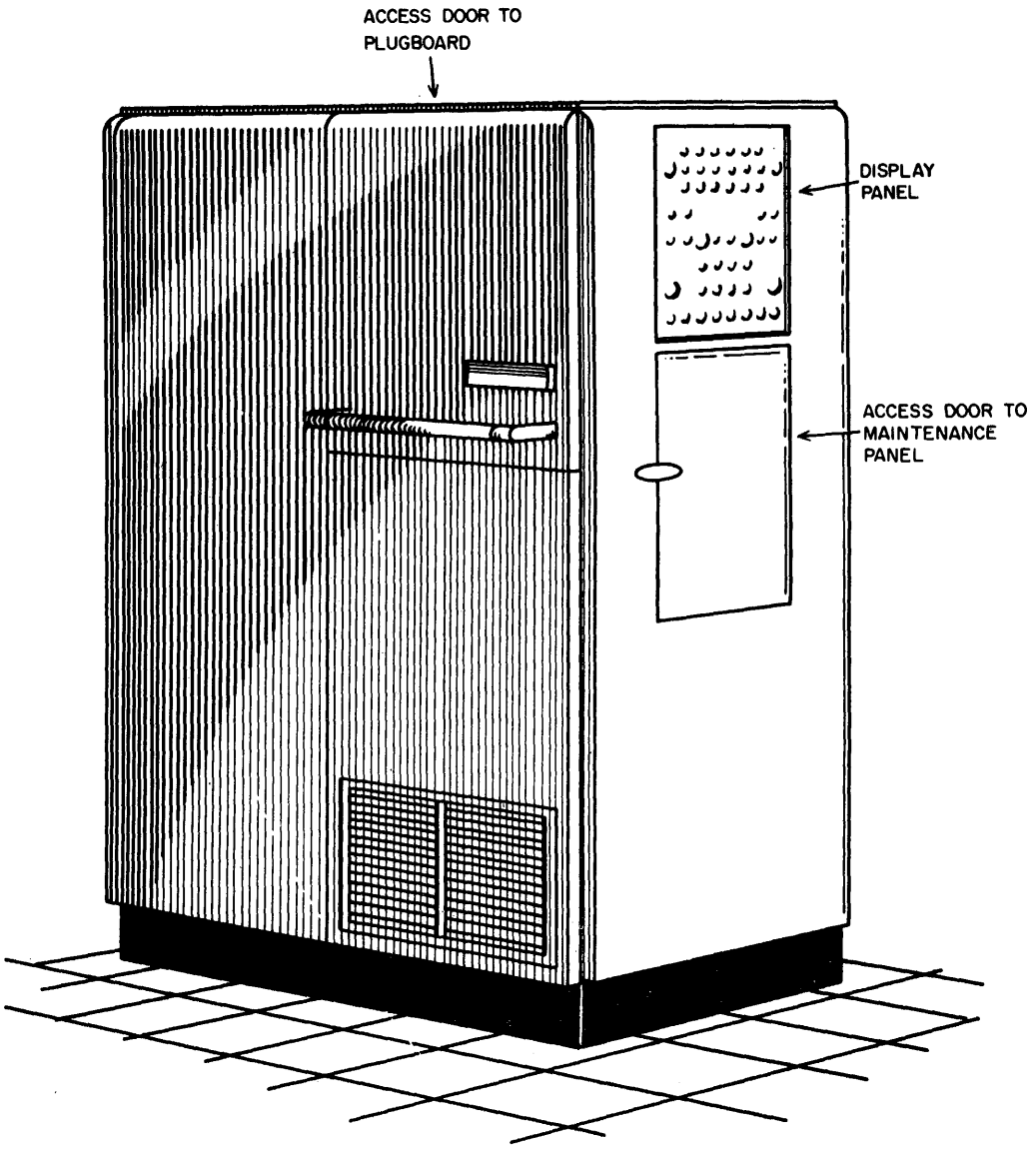


FIGURE 6-4

CONTROL CABINET

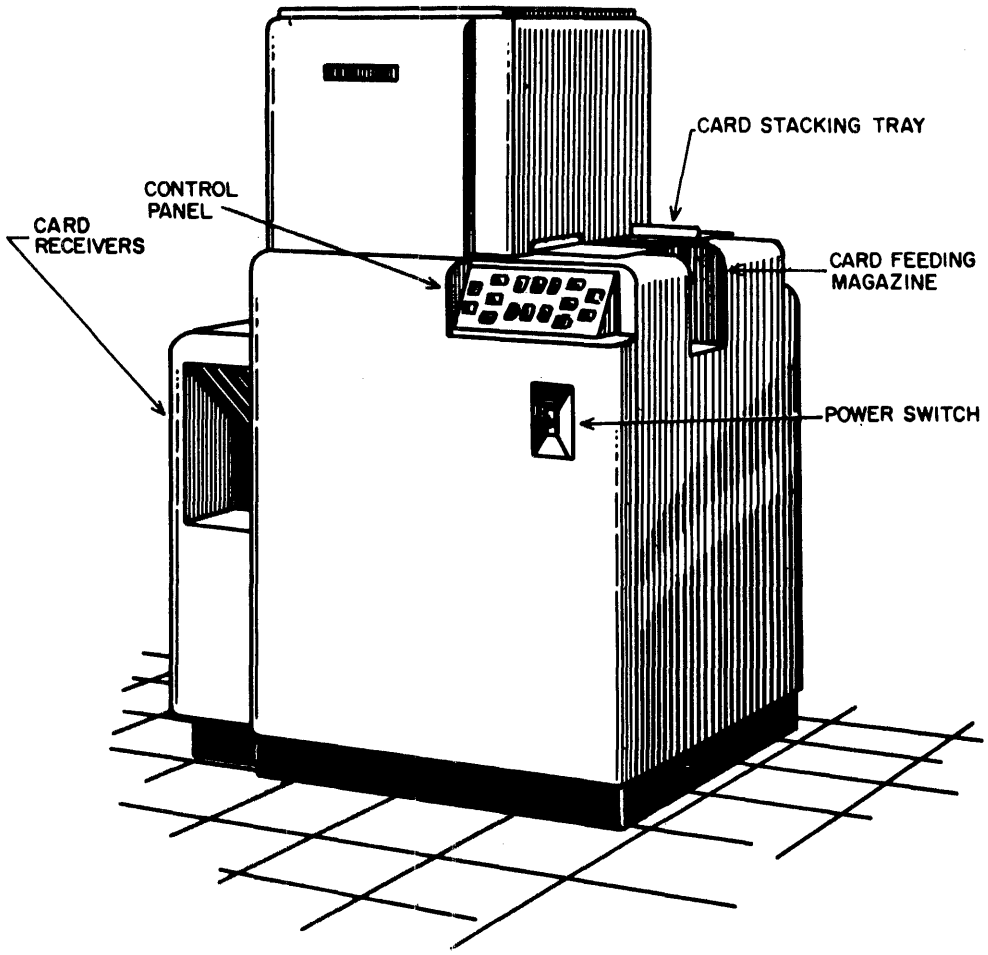


FIGURE 6-5

CARD PROCESSOR

Used as an input device, the 90-column card system reads information from cards and transmits it to the computer. Used as an output device, it accepts output information from the computer and punches it into cards. Used as a combined input-output device, it performs both functions together. Except during manual run-in and run-out operations, the system is controlled entirely by the computer commands received via the computer - I/O control lines. Conversely, the system may send program-altering signals to the computer by way of the I/O - Computer control lines.

Computer - I/O Instructions

The 90-column card system is capable of executing any of the following instructions when they are received via the computer - I/O control lines:

A *Trip* instruction initiates a card cycle during which one new (input) card is taken in and one new (output) card is produced.

A *Skip* instruction prevents the punching operation.

A *No Check* instruction inhibits the checking operation.

A *Sort* instruction initiates the isolation of a selected card.

The trip instruction is assigned to computer - I/O control line A; assignment of other lines is optional.

I/O - Computer Instructions

High speed I/O - computer control lines may be used to modify the computer program. As many as sixteen codes may be used via combinations of lines W, X, Y and Z.

Whenever a card jam, an error in the post-read check, or a malfunction exists in the card unit, a "not ready" condition is indicated within the computer.

80-COLUMN CARD SYSTEM (BULL)

The UFC 80-Column Card system (Bull) is used as a UFC-I tabulating card input unit, as a card output unit, or as a combined input-output unit. It can operate simultaneously on two stacks of cards, punching output data into one stack and reading input data from either or both stacks.

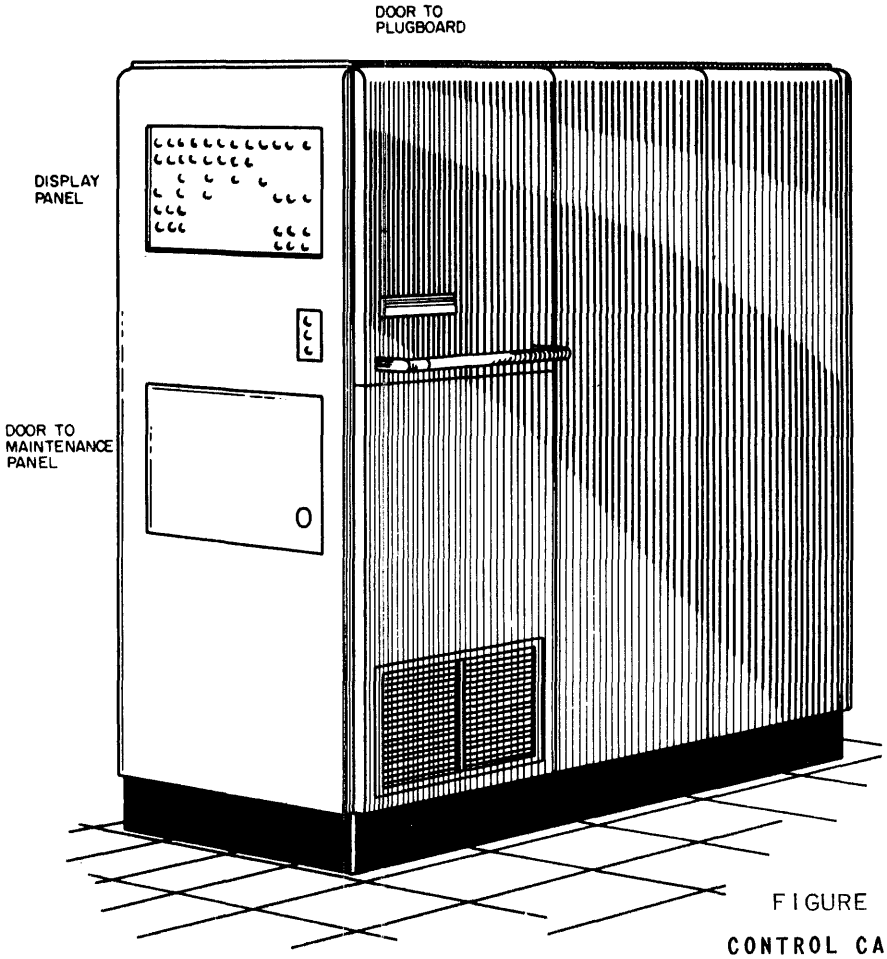


FIGURE 6-6
CONTROL CABINET

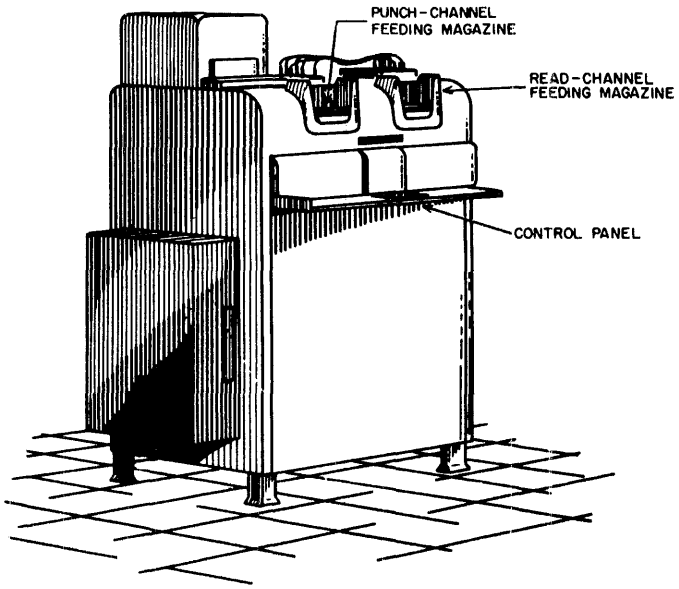


FIGURE 6-7
CARD PROCESSOR

The system is provided with two card feeding magazines, each of which feeds cards through a separate channel to a corresponding receiver. The channels are called the "punch" and "read" channels, respectively.

The two channels work simultaneously, each performing three, parallel, sequential operations. The punch channel picks the bottom card from the punch magazine, reads it, punches it, checks it, and deposits it in the punch receiver.

The read channel picks the bottom card from the read magazine, reads it, checks it, re-reads it, and deposits it in the read receiver. Operation of the two channels is synchronized so that the processing of corresponding cards from the two stacks may be co-ordinated.

Computer - I/O Instructions

The 80-Column punched card unit performs any of the following five commands received from the central computer via the computer - I/O control lines assigned by the programmer:

Program Complete. Each Program Complete command advances the cards in both channels to the next station. Five Program Complete commands are required to carry one card through the card processor from feed magazine to receiver.

Trip Read Feed prepares the first station in the read channel to execute a Program Complete command.

Skip prevents the punching operation.

Punch Non-Check suppresses the checking operation in the punch channel.

Stop causes the card processor to stop at the end of the card cycle under way when the signal is received.

I/O - Computer Instructions

The I/O - computer control lines are used to transmit to the computer any control signals that are read from punched cards in either channel. Control signals indicating the beginning and end of the card file in both channels are also carried by these lines.

HIGH SPEED PRINTER

When used as an on-line output device the UFC High Speed Printer consists of a printer unit and a control cabinet. During off-line operations, a magnetic tape unit is added as the source of data to be printed.

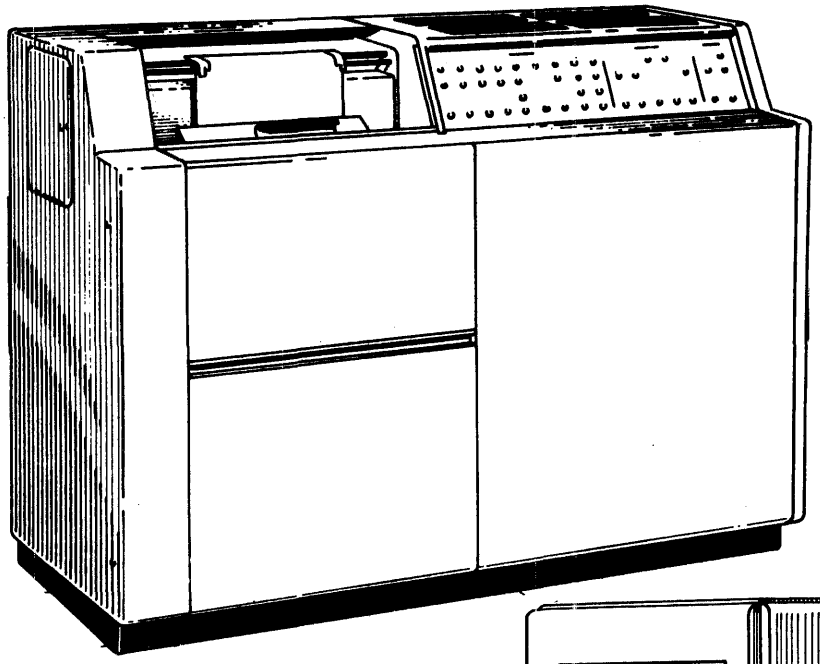


FIGURE 6-8
PRINTER UNIT

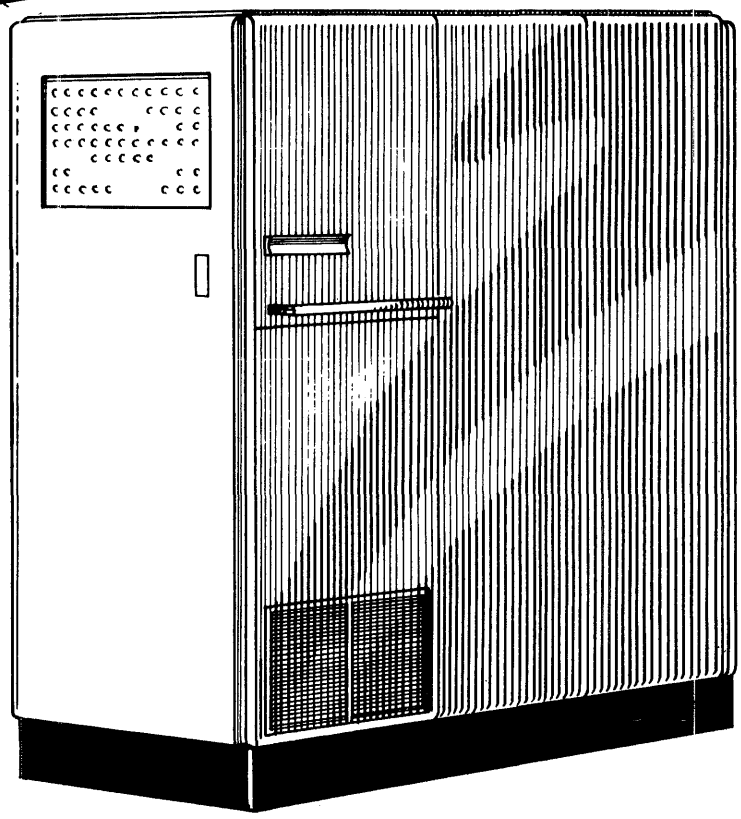


FIGURE 6-9
PRINTER CONTROL CABINET

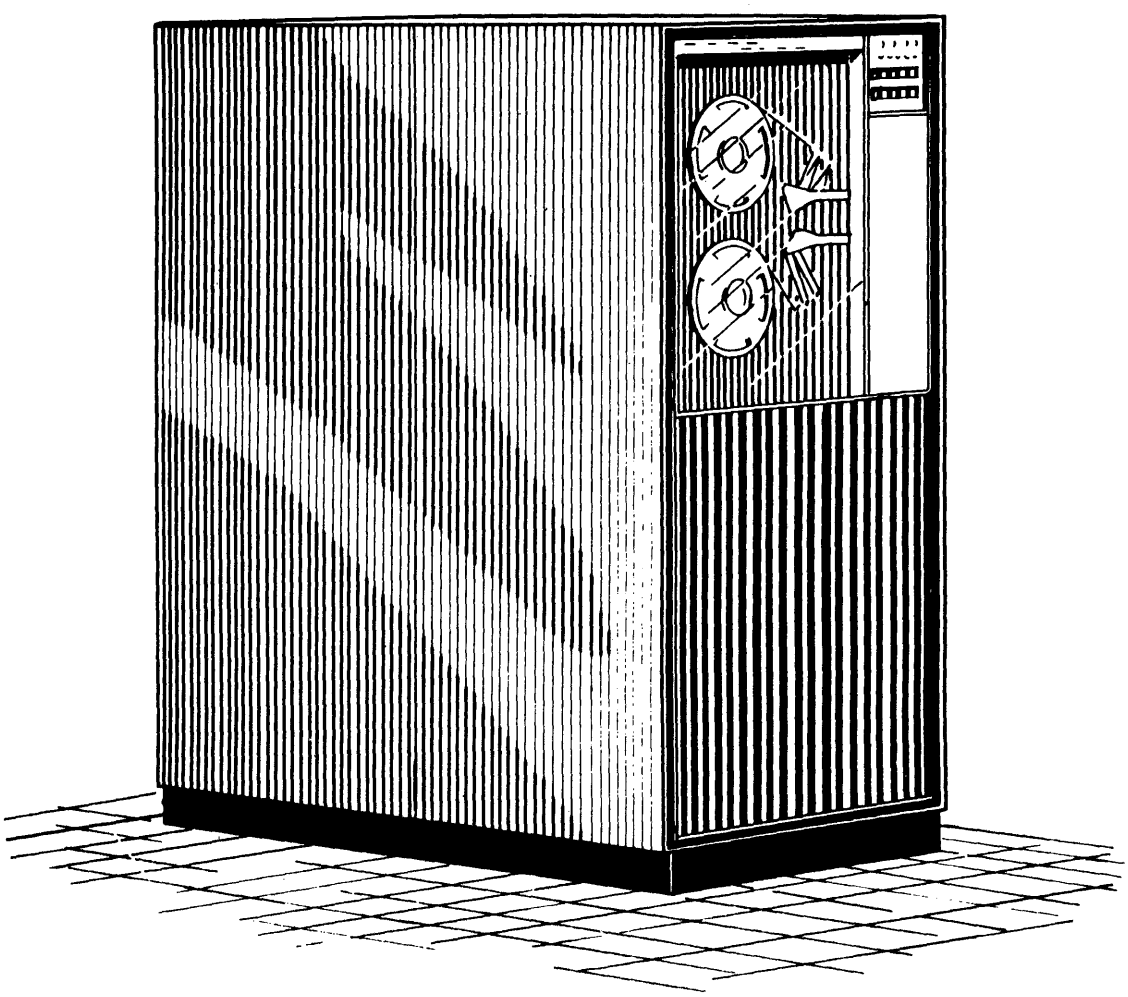


FIGURE 6-10

MAGNETIC TAPE UNIT

The high speed printer is capable of printing alpha-numeric data in 130 columns at a rate of 600 lines per minute off-line or 400 lines per minute on-line. Information fed into the printer must be expressed in 120-character blockettes, with 1.0" interblockette spacing, and a 2.4" spacing every sixth blockette.

On-Line Operations

During on-line operation, the printer is controlled from the computer, and any special control characters contained in the blockettes are ignored. The following control signals transmitted over computer-to-I/O control lines affect printer operation:

(1) Print
(2) Fast Feed 1, Fast Feed 2, Fast Feed 3, Fast Feed 4
(3) Multiline
(4) Double Space
(5) Triple Space

The form sensing signals from the paper feed control loop are sent over I/O-to-computer control lines, and may be used to modify the computer program.

The choice of which computer-to-I/O and I/O-to-computer control lines should be used for specific purposes is optional. However, care must be exercised to insure the proper coordination of commands between the high speed printer and the UFC central computer.

Off-Line Operations

During off-line operation, printer operation may be modified by any of the following special control characters stored in the blockettes:

(1) Fast Feed 1, Fast Feed 2, Fast Feed 3, Fast Feed 4
(2) Multiline
(3) Breakpoint
(4) Stop
(5) Suppress Printing

HIGH SPEED PAPER TAPE SYSTEM

The high speed paper tape system is an input/output device used for processing 5 level, 6 level, or 7 level punched paper tape. The data contained in the tape as a 5, 6, or 7 level code is translated into the 7-level Univac code as it is entered into or received from the I/O tracks of the UFC central computer.

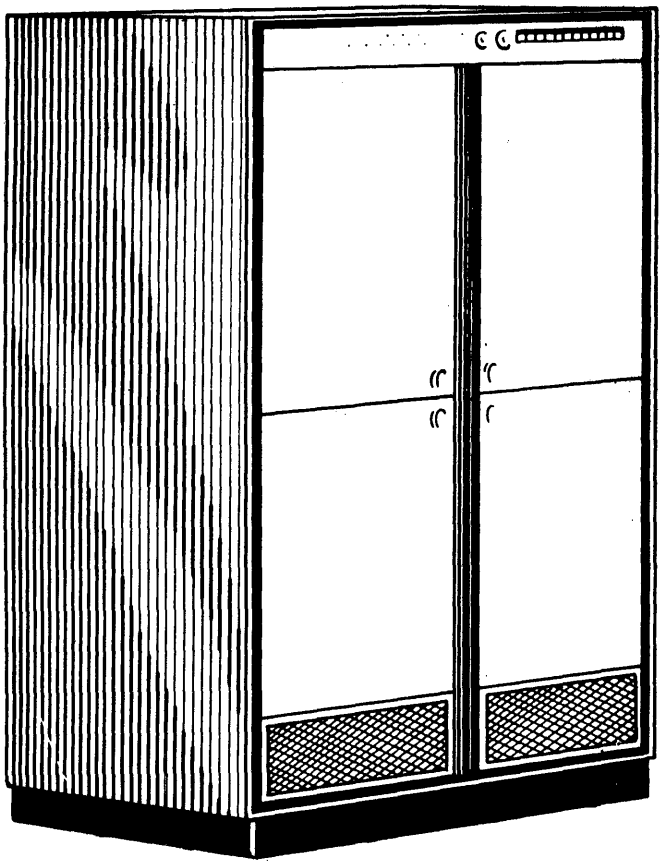


FIGURE 6-11
READ/PUNCH CABINET

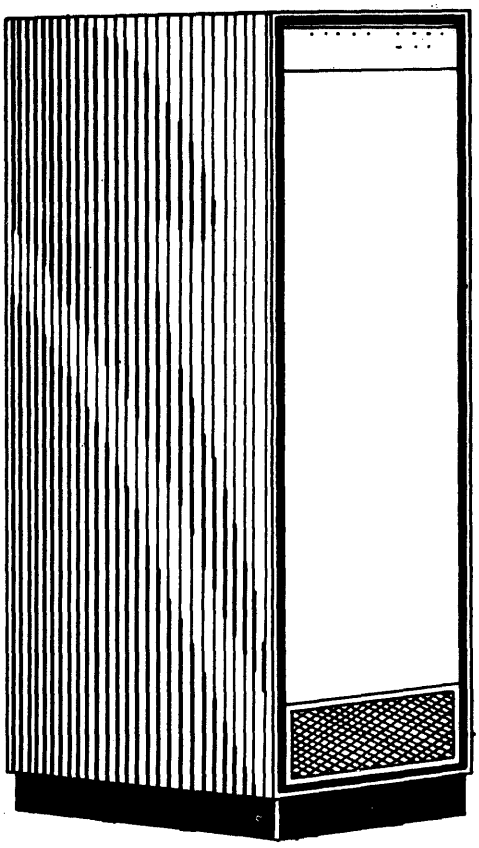


FIGURE 6-12
HSPTS CONTROL CABINET

The HSPTS is strictly an on-line device; that is, it operates under control of a computer program. Its function is to supply input data to or receive output data from the computer. Accordingly, two modes of operation, Input or Output, can be specified by the computer. External wiring on the HSPTS plugboard is employed to define the details of each input or output operation.

In either mode, operation is initiated when the required manual preparations are completed and the computer program sends the appropriate I/O Instruction (Input or Output) to the HSPTS. Stop control can be achieved by an I/O Instruction (Stop) or by the HSPTS's own plugboard-defined program.

Computer - I/O control lines, high speed I/O - computer control lines, and low speed I/O - computer control lines may be used to control the operation of the HSPTS system. Assignment of the control lines to be used and the function each will perform are at the option of the programmer.

MAGNETIC TAPE UNIT

As an I/O unit of the UFC-I system, the magnetic tape unit is an on-line device; i.e., its operations are controlled by the central computer program. Except during intervals when control information is exchanged, however, the central computer and the magnetic tape unit operate independently on a time-shared basis.

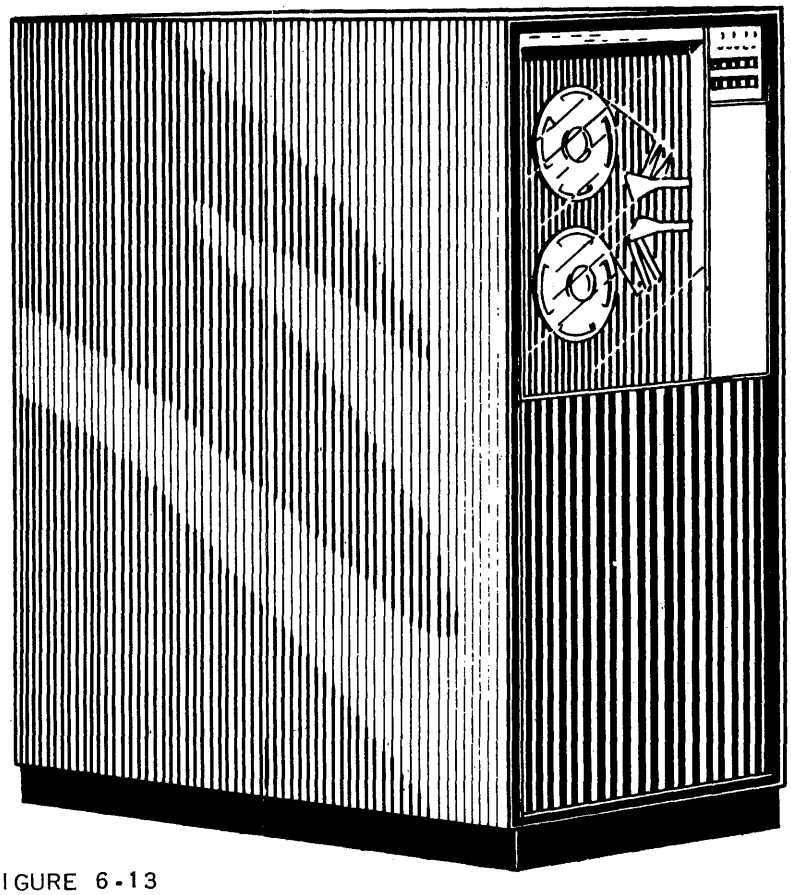


FIGURE 6-13
MAGNETIC TAPE UNIT

The magnetic tape unit is also used as a component of the Sort Collate System and as a data source for the High Speed Printer during off-line activities.

Data are recorded on mylar magnetic tapes at a density of 139 characters per inch, with blockettes of 120 characters separated by blank spaces of either 1.0" or 0.5". Standard reels of tape are 2400' in length and have a capacity of 20,000 blockettes at 0.5" spacing or 14,100 blockettes at 1.0" spacing. When 1.0" interblockette spacing is used, a 2.4" space occurs every sixth blockette.

Computer - I/O Instructions

The thirteen magnetic tape unit instructions which are received via the computer-I/O control lines have been given above in the discussion of the computer-I/O control lines.

I/O - Computer Instructions

The following seven signals may be sent to the computer during "Demand In" sequence via the high speed I/O - computer control lines:

Search Find =	End of Blockette Count
Search Find > or <	Beginning of Tape
End of File	End of Tape
End of Data	

Each of the seven types of control information is detected automatically by circuitry in the Magnetic Tape Unit. "End of File" and "End of Data" are code words which must be stored on the tape together with the actual data.

An *End of File* code consists of at least one complete word of "Z's".
An *End of Data* code consists of at least one complete word of "%'s".

UFC SORT-COLLATE SYSTEM

The UFC Sort-Collate System is a special-purpose magnetic tape file-processing device which performs a wide variety of collating operations, including a sort-by-collation and a sequence-checking operation. Although designed primarily as an independent, off-line device for use in Univac File Computer installations, the UFC Sort-Collate System also has a (psuedo) on-line mode of operation (computer alert) in Univac File Computer systems.

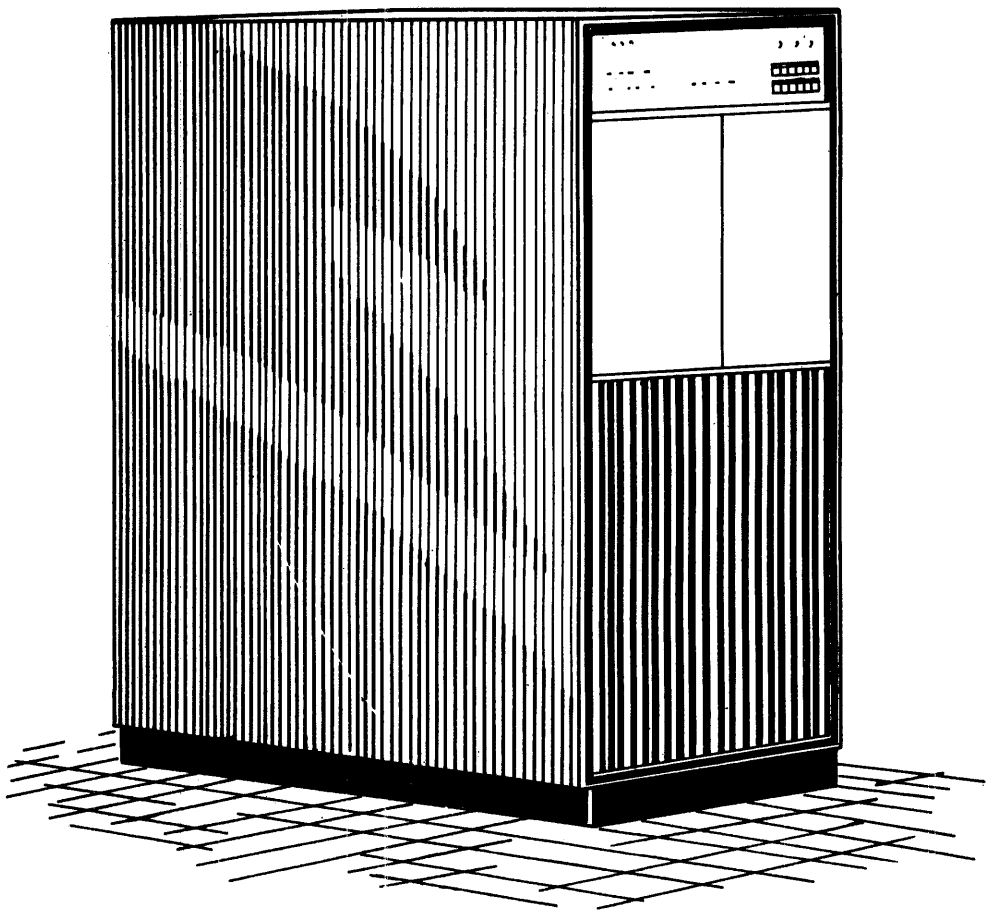


FIGURE 6-14

SORT-COLLATE UNIT

Four UFC magnetic tape units and a sort-collate unit compose the system. Each tape unit is either an input tape unit, functioning as a source to supply tape data to the system, or an output unit, functioning as a destination at which the system rewrites sorted or collated data. The sort-collate unit controls the operation of each tape unit and executes the programs required to place the tape data in a desired order.

Three major types of data manipulation, extraction, merging, and sorting-by-collation, are performed by the sort-collate system. Extraction and merging programs are defined by plugboard wiring; sorting-by-collation is defined by internal wiring within the system and is therefore fully automatic.

Computer Alert Operations

The computer alert operation may be utilized to permit more efficient use of tape units in the updating of magnetic tape files. At some point or points in the plugboard-defined program of the sort collate system, a COMPUTER ALERT hub on the sort collate plugboard receives a pulse. When this occurs, collating stops, the information held in each tape unit's buffer is written on the I/O track associated with that tape unit, control of the four tape units is turned over to the computer, and each tape unit becomes READY to accept computer commands.

After the updating routine has been completed, the computer returns control of the tape units to the sort-collate unit, collation operations resume, and the computer continues its own program.

The I/O - computer control lines which direct magnetic tape units during computer alert operations are the same lines, and perform the same functions, as are discussed under UFC Magnetic Tape Unit.

AIRLINE RESERVATION SYSTEM

A Univac Airlines Reservation System includes a Model-I Univac File Computer together with special-purpose devices that adapt it to rapid scheduling of flight space. It acts as a central information file that is shared by ticket agents at widely separated point-of-sale offices. The central computer stores a current record of flight space. Each ticket agent in the system can obtain information from the file and can alter the file to record reservations and cancellations.

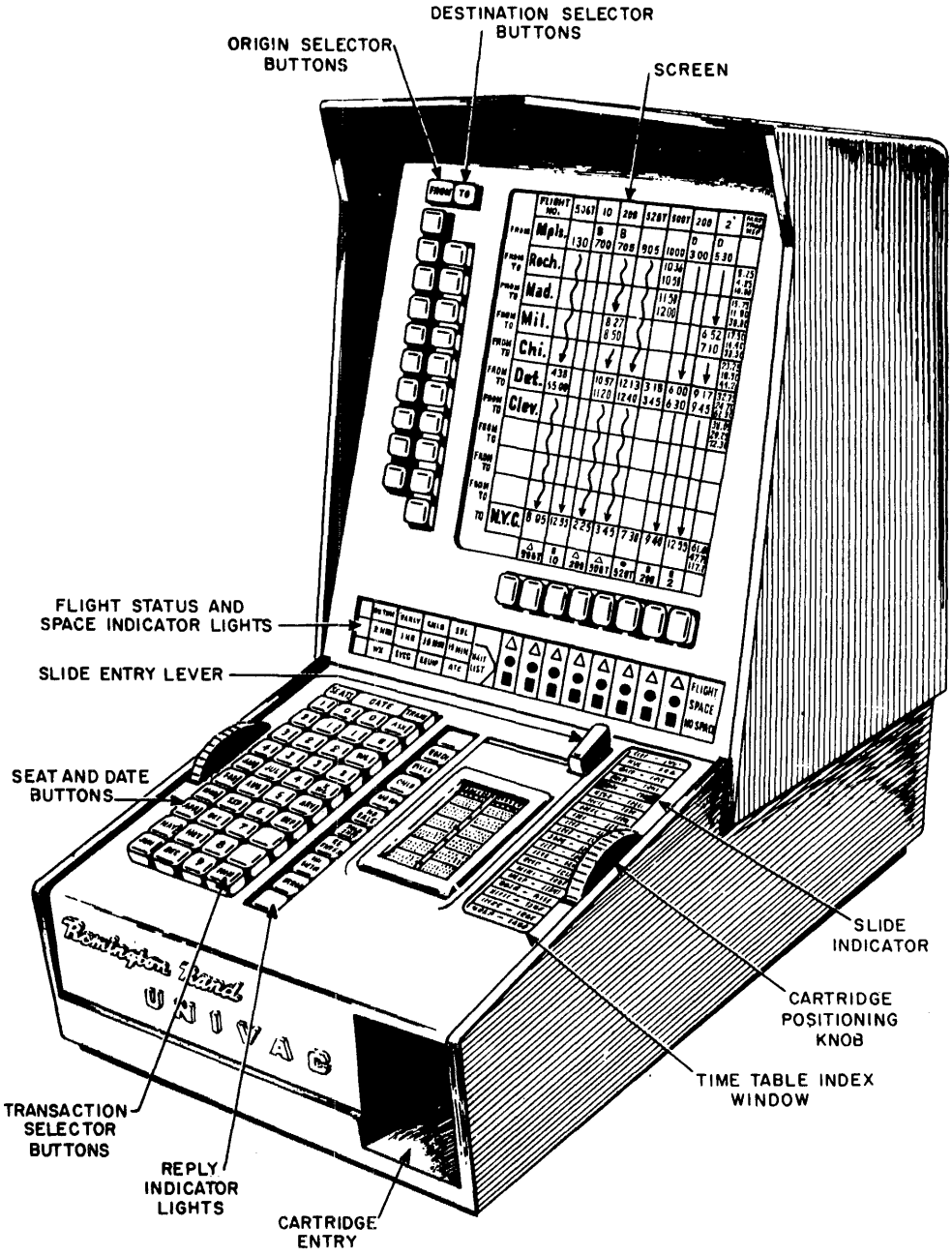


FIGURE 6-15

TICKET AGENT SET

A typical Univac Airlines Reservation System includes a Univac File Computer, and local and remote ticket agent sets, an updating set, connective circuitry linking the agent and updating sets to the central computer, an inquiry typewriter system, and a program loading system.

chapter 7

ARITHMETIC SECTION

INTRODUCTION

All of the arithmetic processes: Add, Subtract, Multiply and Divide, and most of the logical processes, such as Compare, Normalize, and Mask Transfer, are performed in the arithmetic section of the computer.

The values to be processed are temporarily stored, during the above operations, in arithmetic registers called Register A (RA), Register B (RB), Register C (RC), and Register D (RD). Each arithmetic register is a 12-character shift register, the lower order position of which is reserved for the algebraic sign.

During the execution of arithmetic processes, the arithmetic registers receive the following information:

RA receives the first operand from program control storage.

RB receives the second operand, the quantity used to operate on the first operand.

RC accumulates the result in add and subtract operations so that checking may be performed by a reverse process; in divide operations, it receives the remainder, and in multiplication, it receives the most significant digits of the product.

RD accumulates the result in add and subtract operations; in divide operations, it stores the quotient, and in multiplication, it stores the least significant product digits.

ARITHMETIC OPERATIONS

The following processes are carried out in the Arithmetic Section of the computer:

Add

The contents of RA and RB are added and the sum is stored in both RC and RD.

Add and Check

The process is the same as add, except that after the sum is stored, the contents of RB are subtracted from the contents of RC and the remainder is stored in RC; then the contents of RA are subtracted from RC and that remainder is tested to determine whether or not it equals zero. If it equals zero, the process terminates and the result is ready for storage. If it is not zero, an Arithmetic Error pulse is emitted on the program plugboard.

Subtract

The smaller of the contents of RA and RB is subtracted from the larger and the difference is stored in both RC and RD. The sign of the difference is the sign of the larger quantity in RA and RB.

Subtract and Check

The operation is the same as subtract, except that after the difference is stored, the smaller of the contents of RA and RB is added to the contents of RC and the sum is stored in RC. Then the contents of the larger of RA and RB is subtracted from RC and the remainder is zero tested. If the remainder is zero, the process is terminated, and the result is ready for storage. If the remainder is not zero, an Arithmetic Error pulse is emitted on the program plugboard.

Multiply

The contents of RA are multiplied by the contents of RB and the high and low order product digits are accumulated in RC and RD respectively.

Multiply and Check

The operation is the same as multiply except that after the product digits are transferred from RC or RD to memory, the contents of RA are subtracted from the contents of RC and RD the number of times specified by the contents of RB, and the remainder is stored in RC. The remainder is tested for zero, and if zero, the process is terminated. If the remainder is not zero, an Arithmetic Error pulse is emitted on the program plugboard.

Divide

The contents of RA are divided by the contents of RB, the quotient digits are stored in RD, and the remainder is stored in RC. If the quotient is more than 11 significant digits, the Divide Overflow hub on the program plugboard emits a pulse. If the quotient is equal to or less than zero, the quotient is zero and the dividend is stored as the remainder.

Divide and Check

The same as Divide, except that after the quotient and remainder are stored, the remainder in RC is added to the product of the contents of RB and RD and subtracted from the contents of RA. The remainder is zero tested and if zero, the process is terminated. If it is not zero, an Arithmetic Error pulse is emitted on the program plugboard.

Compare

The contents of RA and RB are compared for relative magnitude. If the contents of RA are greater than the contents of RB, branch storage is set to +; if the contents of the registers are equal, branch storage is set to 0; if the contents of RB are less than the contents of RA, branch storage is set to -.

Arithmetic Transfer

This process uses only RD as the information goes from one storage location to another.

Mask Transfer

The contents of RA are masked by the contents of RB and transferred to RD.

Suppress Left Zeros

All zeros to the left of the most significant digit in RA are replaced by space codes and a count of the number of zeros suppressed is placed in RB.

Normalize

The contents of RA are shifted left until the most significant digit is in the high order position. The normalizing count is placed in RB and the unused positions of RB are filled with space codes. If the contents of RA is zero, the program is interrupted, the Normalize Overflow hub on the program plugboard emits a pulse, and RB is set to a count of one.

RULES FOR ARITHMETIC OPERATIONS

In Add, Subtract, Multiply, and Divide operations, space codes are converted to zeros when the result is formed in RC and RD.

In Add and Subtract, the following rules for signs are observed:

(1) A zero, space code, or plus sign in the RB sign position is interpreted as a plus sign.

(2) A minus sign in the RA sign position is interpreted as a negative sign.

(3) All other characters in the RA sign position except an ignore code are treated as if they were alpha characters.

(4) In the RB sign position, all characters except a minus sign or an ignore code are treated as plus signs.

(5) An alpha character in the RA sign position is transferred to the result in either an add or subtract operation. In all other cases the algebraic sign is produced.

(6) An ignore code in the sign position of either register will cause the sign character to be transferred from the other register to the result. (Exception: a space code (Δ) and an ignore code (i) will result in a 0.)

ADD - SUBTRACT

RESULT SIGN - UFC - I

V_2 Sign \ V_1 Sign	Δ, +, 0	-	1-9 A-Z Special Characters	i
Δ, +, 0	Δ (plus)	Algebraic Sign Δ = Plus - = Minus	V_1 Sign Character	V_2 Sign Character *
-	Algebraic Sign Δ = Plus - = Minus	- (Minus)	V_1 Sign Character	V_2 Sign Character
1-9 A-Z Special Characters	Δ (Plus)	Algebraic Sign Δ = Plus - = Minus	V_1 Sign Character	V_2 Sign Character
i	V_1 Sign Character *	V_1 Sign Character	V_1 Sign Character	i

* See exception in Sign Rule 6.

In Add and Subtract operations the following rules are observed for the character positions:

(1) If the corresponding characters in RA and RB are numeric, the quantities are added (subtracted).

(2) If, in corresponding character positions, either RA and RB or both contain alphabetic characters, the character from RA is transferred unchanged to the corresponding position in the result.

(3) Ignore codes appearing in either register will cause the corresponding character in the other register to be transferred to the result.

(4) Carries (borrows) generated by preceding characters in (2) and (3) above are destroyed.

ADD - SUBTRACT

RESULT CHARACTERS - UFC-I

V_2 \ V_1	0 - 9	Alpha Special Characters +, -	i
0 - 9	Sum	V_1 Character *	V_2 Character *
Alpha Special Characters +, -	V_1 Character *	V_1 Character *	V_2 Character *
i	V_1 Character *	V_1 Character *	i *

* Carry or borrow from adjacent digit position is destroyed.

In Add and Subtract operations the following rules are observed for the character positions:

(1) If the corresponding characters in RA and RB are numeric, the quantities are added (subtracted).

(2) If, in corresponding character positions, either RA and RB or both contain alphabetic characters, the character from RA is transferred unchanged to the corresponding position in the result.

(3) Ignore codes appearing in either register will cause the corresponding character in the other register to be transferred to the result.

(4) Carries (borrows) generated by preceding characters in (2) and (3) above are destroyed.

ADD - SUBTRACT

RESULT CHARACTERS - UFC-I

V_2 \ V_1	0 - 9	Alpha Special Characters +, -	i
0 - 9	Sum	V_1 Character *	V_2 Character *
Alpha Special Characters +, -	V_1 Character *	V_1 Character *	V_2 Character *
i	V_1 Character *	V_1 Character *	i *

* Carry or borrow from adjacent digit position is destroyed.

In Multiplication and Division, any character in the character positions of RA and RB other than 0-9 will be treated as numeric.

In Multiplication and Division, any character except a minus sign in the RA or RB sign position is treated as a plus sign and the sign of the result is determined in the usual algebraic manner.

In Division the sign of the remainder corresponds to the sign of the dividend.

Branch Storage Settings

The following statements further detail and supplement the discussion of branch storage found in Chapter 3. The conditions peculiar to each type of arithmetic operation are given.

Numeric Add or Subtract:

V_1 Sign is Δ, 0, +, or -
V_2 Sign is not an Ignore Code
V_1 and V_2 are numbers

If:	*Sign of Result*	*Setting of Branch Storage*
(R ≠ 0)	Δ	+
(R ≠ 0)	-	-
(R = 0)	Δ or -	0

Alpha Add or Subtract:

V_1 Sign is Δ, 0, ≠, or -
V_2 Sign is not an Ignore code
V_1 and/or V_2 (apart from sign character) are alphanumeric.

Sign of Result	*Setting of Branch Storage*
Δ	+
-	-

Note: The conditions R = 0 and R ≠ 0 are not detected when the computer performs an Alpha Add or Subtract. Branch Storage is therefore always given either a + or - setting.

In Multiplication and Division, any character in the character positions of RA and RB other than 0-9 will be treated as numeric.

In Multiplication and Division, any character except a minus sign in the RA or RB sign position is treated as a plus sign and the sign of the result is determined in the usual algebraic manner.

In Division the sign of the remainder corresponds to the sign of the dividend.

Branch Storage Settings

The following statements further detail and supplement the discussion of branch storage found in Chapter 3. The conditions peculiar to each type of arithmetic operation are given.

Numeric Add or Subtract:

V_1 Sign is Δ, 0, +, or -
V_2 Sign is not an Ignore Code
V_1 and V_2 are numbers

If:	*Sign of Result*	*Setting of Branch Storage*
(R ≠ 0)	Δ	+
(R ≠ 0)	-	-
(R = 0)	Δ or -	0

Alpha Add or Subtract:

V_1 Sign is Δ, 0, ≠, or -
V_2 Sign is not an Ignore code
V_1 and/or V_2 (apart from sign character) are alphanumeric.

Sign of Result	*Setting of Branch Storage*
Δ	+
-	-

Note: The conditions R = 0 and R ≠ 0 are not detected when the computer performs an Alpha Add or Subtract. Branch Storage is therefore always given either a + or - setting.

Alpha Sign or Subtract:

V_1 Sign is not Δ, 0, +, or -
and/or
V_2 Sign is an Ignore code

Branch Storage has a 0 setting after all Alpha Sign Add or Subtract operations, *regardless of the sign of the result.*

Multiply:

If:	*Sign of Result*	*Setting of Branch Storage*
(R ≠ 0)	Δ	+
(R ≠ 0)	-	-
(R = 0)	Δ or -	0

Divide:

Sign of Result	*Setting of Branch Storage*
Δ	+
-	-

The conditions Quotient = 0 and Quotient ≠ 0 are not detected in connection with the setting of Branch Storage. Branch Storage is always set to ≠ or - after a Divide instruction.

Alpha Sign or Subtract:

V_1 Sign is not Δ, 0, +, or -
and/or
V_2 Sign is an Ignore code

Branch Storage has a 0 setting after all Alpha Sign Add or Subtract operations, *regardless of the sign of the result.*

Multiply:

If:	*Sign of Result*	*Setting of Branch Storage*
(R ≠ 0)	Δ	+
(R ≠ 0)	-	-
(R = 0)	Δ or -	0

Divide:

Sign of Result	*Setting of Branch Storage*
Δ	+
-	-

The conditions Quotient = 0 and Quotient ≠ 0 are not detected in connection with the setting of Branch Storage. Branch Storage is always set to ≠ or - after a Divide instruction.

chapter 8

TIMING

DESCRIPTION OF TIMING FACTORS

To determine the approximate interval of time required by UFC-I to execute a complete program step or instruction word, four factors derived from the following tables and text should be used. All of the figures quoted below are based on average computer time.

Memory Reference Times

In the table labeled "Memory Reference Times" will be found the time required to locate the first operand and load it into RA, the time required to locate the second operand and load it into RB, and the time required to store the result from RC or RD.

chapter 8

TIMING

DESCRIPTION OF TIMING FACTORS

To determine the approximate interval of time required by UFC-I to execute a complete program step or instruction word, four factors derived from the following tables and text should be used. All of the figures quoted below are based on average computer time.

Memory Reference Times

In the table labeled "Memory Reference Times" will be found the time required to locate the first operand and load it into RA, the time required to locate the second operand and load it into RB, and the time required to store the result from RC or RD.

Process Times

The time required for the performance of the process in an instruction may be determined from the table headed "Process Times". However, it must be recognized that during the execution of one instruction word, the computer locates the next instruction during the process time of the current instruction. Where the process time is found to be less than 3.1 milliseconds, the 3.1 value should be used as the process time.

Constant

An instruction word requires a slightly longer time to execute than a program step performing the same operation. This additional time is included in Table A as a constant.

Programmed Shifts

Table A includes time factors necessary to determine the time of execution of a computer instruction except the time required to execute programmed shifts of the contents of the arithmetic registers. When a shift of V_1, V_2 or R (U, V, or W) is programmed, the time required to execute the shift must be added to the total of the other factors in Table A. Each operand shifted requires the following shift time:

.042 (n + 1) milliseconds
where n = the number of programmed shifts

MEMORY REFERENCE TIMES

The time required to refer to any of the following locations as V_1, V_2 or R, is 0.9 milliseconds:

Register A	GSAR
Register B	PAK
Register C	CDR
Register D	SRV
IRV	

Process Times

The time required for the performance of the process in an instruction may be determined from the table headed "Process Times". However, it must be recognized that during the execution of one instruction word, the computer locates the next instruction during the process time of the current instruction. Where the process time is found to be less than 3.1 milliseconds, the 3.1 value should be used as the process time.

Constant

An instruction word requires a slightly longer time to execute than a program step performing the same operation. This additional time is included in Table A as a constant.

Programmed Shifts

Table A includes time factors necessary to determine the time of execution of a computer instruction except the time required to execute programmed shifts of the contents of the arithmetic registers. When a shift of V_1, V_2 or R (U, V, or W) is programmed, the time required to execute the shift must be added to the total of the other factors in Table A. Each operand shifted requires the following shift time:

.042 (n + 1) milliseconds
where n = the number of programmed shifts

MEMORY REFERENCE TIMES

The time required to refer to any of the following locations as V_1, V_2 or R, is 0.9 milliseconds:

Register A	GSAR
Register B	PAK
Register C	CDR
Register D	SRV
IRV	

The time required to locate and transfer the contents of the following locations to an arithmetic register is:

From	*Milliseconds*	
BTB or GSB Word	0.9	
High Speed Drum Word	3.1	
BTB or GSB Fields:		(x = the number of the character
Field A	0.9	(position of the first character
Field B - V	(.021x + 0.9)	(of the field.
High Speed Drum Field	3.1	

The time required to transfer R from an arithmetic register to the following addresses is:

To	*Milliseconds*	
BTB or GSB Word	0.9	
High Speed Drum Word	3.1	(x = the number of the
BTB or GSB Fields:		(character position of the
>12 characters		(first character of the
Field A	(.042z + 0.9)	(field.
Field B - V	(.021x + .042z + 0.9)	(z = number of characters (in the field minus 12.)
<12 characters		
Field A	0.9	
Field B - V	(.021x + 0.9)	
High Speed Drum Fields		
>12 characters	(.042z + 3.1)	
<12 characters	3.1	

The time required to locate and place V_1 in the BTB is:

From	*Milliseconds*	
GSB Word	0.9	
High Speed Drum Word	3.1	(x = the number of the (character position of (the first character of (the field. (y = the number of char- (acters in the field.)
GSB Fields:		
Field A	(.042y + 0.4)	
Field B	(.021x + .042y + 0.3)	
High Speed Drum Field	(.042y + 2.6)	

The time required to locate and transfer the contents of the following locations to an arithmetic register is:

From	*Milliseconds*	
BTB or GSB Word	0.9	
High Speed Drum Word	3.1	
BTB or GSB Fields:		(x = the number of the character
Field A	0.9	(position of the first character
Field B - V	(.021x + 0.9)	(of the field.
High Speed Drum Field	3.1	

The time required to transfer R from an arithmetic register to the following addresses is:

To	*Milliseconds*	
BTB or GSB Word	0.9	
High Speed Drum Word	3.1	(x = the number of the
BTB or GSB Fields:		(character position of the
>12 characters		(first character of the
Field A	(.042z + 0.9)	(field.
Field B - V	(.021x + .042z + 0.9)	(z = number of characters (in the field minus 12.)
<12 characters		
Field A	0.9	
Field B - V	(.021x + 0.9)	
High Speed Drum Fields		
>12 characters	(.042z + 3.1)	
<12 characters	3.1	

The time required to locate and place V_1 in the BTB is:

From	*Milliseconds*	
GSB Word	0.9	
High Speed Drum Word	3.1	(x = the number of the (character position of (the first character of (the field. (y = the number of char- (acters in the field.)
GSB Fields:		
Field A	(.042y + 0.4)	
Field B	(.021x + .042y + 0.3)	
High Speed Drum Field	(.042y + 2.6)	

GSB (120 characters)	5.4
High Speed Drum (120 characters)	7.7

The time required to transfer R from the BTB to the following locations is:

To	*Milliseconds*	
GSB Word	0.9	
High Speed Drum Word	3.2	(x = the number of the (character position of (the first character of (the field. (y = the number of char- (acters in the field.)
GSB Fields:		
Field A	(.042y + 2.6)	
Field B - V'	(.021x + .042y + 2.6)	
High Speed Drum Fields	(.042y + 2.7)	
BTP	5.1	

PROCESS TIMES

Below are the process times for Add, Subtract, Multiply and Divide. All times are in milliseconds:

	Normal	V_1 V_2	Alpha
Add (AD) and Subtract (SB)	1.2	1.2	0.7
Check	0.7	1.3	
TOTAL	1.9	2.5	0.7

Multiply (MU or ML)		1.60 + .588 ($m_{11} + m_{10} + \ldots m_1$)
Check		1.00 + ($m_{11} + m_{10} + \ldots m_1$)
	where:	($m_{11} + m_{10} + \ldots m_1$) is the sum of the multiplier digits.
Normal Division		.042 (26 - u - v) + 1.2 (u - v + n + 1) + .462 - .042v + .588 (sum of the quotient digits)
Check		1.6 + .042 (11 - v - n) + .588 (sum of the quotient digits)
	where:	u = number of V_1 digits v = number of V_2 digits n = number of programmed V_1 shifts

GSB (120 characters)	5.4
High Speed Drum (120 characters)	7.7

The time required to transfer R from the BTB to the following locations is:

To	*Milliseconds*	
GSB Word	0.9	
High Speed Drum Word	3.2	(x = the number of the (character position of (the first character of (the field. (y = the number of char- (acters in the field.)
GSB Fields:		
Field A	(.042y + 2.6)	
Field B - V'	(.021x + .042y + 2.6)	
High Speed Drum Fields	(.042y + 2.7)	
BTP	5.1	

PROCESS TIMES

Below are the process times for Add, Subtract, Multiply and Divide. All times are in milliseconds:

	Normal	V_1 V_2	Alpha
Add (AD) and Subtract (SB)	1.2	1.2	0.7
Check	0.7	1.3	
TOTAL	1.9	2.5	0.7

Multiply (MU or ML)		1.60 + .588 ($m_{11} + m_{10} + \ldots m_1$)
Check		1.00 + ($m_{11} + m_{10} + \ldots m_1$)
	where:	($m_{11} + m_{10} + \ldots m_1$) is the sum of the multiplier digits.
Normal Division		.042 (26 - u - v) + 1.2 (u - v + n + 1) + .462 - .042v + .588 (sum of the quotient digits)
Check		1.6 + .042 (11 - v - n) + .588 (sum of the quotient digits)
	where:	u = number of V_1 digits v = number of V_2 digits n = number of programmed V_1 shifts

Special Cases

Division (V_2) = 0 : 1.1
Dividend (V_1) = 0 : 1.596 - .042 (v)
Divide Overflow: .042(26 - u - v)

TIMING CHART - UFC-I

(milliseconds)

	AD, SB	AD, SB, ML, MU, DQ, DR	AT	MT, *SU, *SV *SW	BT	* JZ, JP JN, UJ	CC	* LS LA	* SZ LN	CP	* TD	* DE	* TI	* SP
V_1 address	X	X	X	X	X	---	---	---	X	X	---	---	---	---
V_2 address	X	X	---	X	---	---	---	---	---	X	---	---	---	---
Process	P	P	0.0	0.6	0.0	---	8.0@	---	1.0	0.6	---	---		---
R address	X	X	X	X	X	---	---	---	X	---	---	---	---	---
Program Step Totals						---	8.0	---			---	---		---
U address	X	X	X	X	X	0.8	---	---	X	X	---	---	---	---
V address	X	X	---	X	---	---	---	---	---	X	---	---	---	---
Process	3.1	P	3.1	3.1	3.1	3.1	3.1@	3.1	3.1	3.1	3.1	3.1	3.1	3.1
W address	X	X	X	X	X	X	---	0.8	X	---	---	---	---	---
Constant	1.8	1.8	4.1	1.8	1.5	2.3	8.7	1.4	3.7	1.4	1.5	2.0	1.5	1.1
Instruction Word Totals							11.8	5.3			4.6	5.1	4.6	4.1

* These are internal program instructions that do not appear on the plugboard.
P See Process Times
X See Memory Reference Times
@ The Channel Clear process does not occur simultaneously with the finding of the next instruction.

TABLE A

Special Cases

Division (V_2) = 0 : 1.1
Dividend (V_1) = 0 : 1.596 - .042 (v)
Divide Overflow: .042(26 - u - v)

TIMING CHART - UFC-I

(milliseconds)

	AD, SB	AD, SB, ML, MU, DQ, DR	AT	MT, *SU, *SV *SW	BT	* JZ, JP JN, UJ	CC	* LS LA	* SZ LN	CP	* TD	* DE	* TI	* SP
V_1 address	X	X	X	X	X	---	---	---	X	X	---	---	---	---
V_2 address	X	X	---	X	---	---	---	---	---	X	---	---	---	---
Process	P	P	0.0	0.6	0.0	---	8.0@	---	1.0	0.6	---	---		---
R address	X	X	X	X	X	---	---	---	X	---	---	---	---	---
Program Step Totals						---	8.0	---			---	---		---
U address	X	X	X	X	X	0.8	---	---	X	X	---	---	---	---
V address	X	X	---	X	---	---	---	---	---	X	---	---	---	---
Process	3.1	P	3.1	3.1	3.1	3.1	3.1@	3.1	3.1	3.1	3.1	3.1	3.1	3.1
W address	X	X	X	X	X	X	---	0.8	X	---	---	---	---	---
Constant	1.8	1.8	4.1	1.8	1.5	2.3	8.7	1.4	3.7	1.4	1.5	2.0	1.5	1.1
Instruction Word Totals							11.8	5.3			4.6	5.1	4.6	4.1

* These are internal program instructions that do not appear on the plugboard.
P See Process Times
X See Memory Reference Times
@ The Channel Clear process does not occur simultaneously with the finding of the next instruction.

TABLE A

APPENDIX A

GLOSSARY

Term	*Explanation*
access time	The interval between the time that program control initiates a storage reference and the time that storage reference is completed.
address	A label, name or number, identifying a memory location at which information can be stored, and from which information can be obtained.
alpha-numeric	A symbol containing letters and/or other non-numeric characters in addition to numbers.
angular address	A term used in connection with magnetic drums. It refers to a distinct and separate location around the periphery of a track or channel. (A track number and an angular address define a unique location on a magnetic drum.)
arithmetic operation	An operation involving addition, subtraction, multiplication or division.
B+ current	A steady direct current used by the programmer to pick up selectors and to light indicators (any other use of this current will result in damage to the computer.)
binary-coded decimal notation	One of many systems of writing numbers in which each decimal digit of the number is expressed by a different code written in binary (two-state) digits. (See "excess-three code".)
binary digit	A representation of a two-state condition of a unit of storage in a digital computer: 0 or 1, on or off, yes or not.
bit	An abbreviation of "binary digit".
block	A group or ensemble of characters.

APPENDIX A

GLOSSARY

Term	*Explanation*
access time	The interval between the time that program control initiates a storage reference and the time that storage reference is completed.
address	A label, name or number, identifying a memory location at which information can be stored, and from which information can be obtained.
alpha-numeric	A symbol containing letters and/or other non-numeric characters in addition to numbers.
angular address	A term used in connection with magnetic drums. It refers to a distinct and separate location around the periphery of a track or channel. (A track number and an angular address define a unique location on a magnetic drum.)
arithmetic operation	An operation involving addition, subtraction, multiplication or division.
B+ current	A steady direct current used by the programmer to pick up selectors and to light indicators (any other use of this current will result in damage to the computer.)
binary-coded decimal notation	One of many systems of writing numbers in which each decimal digit of the number is expressed by a different code written in binary (two-state) digits. (See "excess-three code".)
binary digit	A representation of a two-state condition of a unit of storage in a digital computer: 0 or 1, on or off, yes or not.
bit	An abbreviation of "binary digit".
block	A group or ensemble of characters.

blockette	A group of 120 adjacent characters. The blockette is the UFC System's fundamental unit of input/output format (e.g., card unit and magnetic tape unit data transmissions to and from the computer are 120 character or blockette transmissions. In these devices the blockette is the principal operating unit). The blockette is also used to specify the entire contents of a 120-character buffer or a track on the high speed drum.
borrow (noun)	The digit to be taken from the next higher digit position (and subtracted from that digit position) when the subtrahend digit of one digit position exceeds the minuend digit of that digit position.
bus	A line, or trunk, over which data transmissions occur from any of several sources to any of several destinations.
buffer	A temporary storage for data; an isolating device, generally used to transfer data between two storage elements that are not synchronized.
carry (noun)	The digit to be taken to the next-higher order digit position (and there added) when the sum of the digits in one digit position equals or exceeds the number base.
channel	A narrow band on the periphery of a general storage drum; the area which passes beneath a read/write head as the general storage drum revolves; the equivalent of a track of the high speed drum. Also a level on punched paper tape or magnetic tape.
clear (verb)	To replace information in a register by (Univac coded) zeros, ignore codes, or space codes.
code (noun or verb)	(Noun) A system of symbols for representing information in a computer and the rules for associating them. (See Univac code.)

blockette	A group of 120 adjacent characters. The blockette is the UFC System's fundamental unit of input/output format (e.g., card unit and magnetic tape unit data transmissions to and from the computer are 120 character or blockette transmissions. In these devices the blockette is the principal operating unit). The blockette is also used to specify the entire contents of a 120-character buffer or a track on the high speed drum.
borrow (noun)	The digit to be taken from the next higher digit position (and subtracted from that digit position) when the subtrahend digit of one digit position exceeds the minuend digit of that digit position.
bus	A line, or trunk, over which data transmissions occur from any of several sources to any of several destinations.
buffer	A temporary storage for data; an isolating device, generally used to transfer data between two storage elements that are not synchronized.
carry (noun)	The digit to be taken to the next-higher order digit position (and there added) when the sum of the digits in one digit position equals or exceeds the number base.
channel	A narrow band on the periphery of a general storage drum; the area which passes beneath a read/write head as the general storage drum revolves; the equivalent of a track of the high speed drum. Also a level on punched paper tape or magnetic tape.
clear (verb)	To replace information in a register by (Univac coded) zeros, ignore codes, or space codes.
code (noun or verb)	(Noun) A system of symbols for representing information in a computer and the rules for associating them. (See Univac code.)

	(Verb) To program.
command (noun)	A pulse, signal, or set of signals initiating one step in the execution of a computer instruction, sub-instruction, or sub-step.
comparator	A group of circuits which compare two quantities and present an appropriate indication of the result of the comparison.
computer character	The basic unit of data for the UFC. A legal character is any letter, number, or symbol that can be expressed in Univac code.
computer ground	A computer current recognized by the programmer as the plugboard hubs labelled COMPUTER GROUND, which complete the coil circuit of a selector allowing that selector to be picked up at any time during the computer program (see demand ground).
computer instruction	A completely defined operation for the computer; the principal unit of a computer program. Two types of computer instructions are employed: Instruction Words for internally-stored programs. Program Steps for plugboard-defined programs.
computer language	Univac coded characters.
contents of ( )	The information stored within.
d-c enable	A computer-controlled direct current the plugboard wiring of which allows the computer to gain access to storages, processes and shifts.
debug	To isolate and remove a computer malfunction or the program mistakes in a computer program.
demand ground	A computer current recognized by the programmer as the plugboard hubs labelled DEMAND GROUND

	(Verb) To program.
command (noun)	A pulse, signal, or set of signals initiating one step in the execution of a computer instruction, sub-instruction, or sub-step.
comparator	A group of circuits which compare two quantities and present an appropriate indication of the result of the comparison.
computer character	The basic unit of data for the UFC. A legal character is any letter, number, or symbol that can be expressed in Univac code.
computer ground	A computer current recognized by the programmer as the plugboard hubs labelled COMPUTER GROUND, which complete the coil circuit of a selector allowing that selector to be picked up at any time during the computer program (see demand ground).
computer instruction	A completely defined operation for the computer; the principal unit of a computer program. Two types of computer instructions are employed: Instruction Words for internally-stored programs. Program Steps for plugboard-defined programs.
computer language	Univac coded characters.
contents of ( )	The information stored within.
d-c enable	A computer-controlled direct current the plugboard wiring of which allows the computer to gain access to storages, processes and shifts.
debug	To isolate and remove a computer malfunction or the program mistakes in a computer program.
demand ground	A computer current recognized by the programmer as the plugboard hubs labelled DEMAND GROUND

	0-9, which complete the coil circuit of a selector, allowing that selector to be picked up only during the time the correspondingly numbered input/output unit is on demand.
destination address	An address that specifies a particular location (of definite capacity) at which information held in some intermediate location is to be stored.
digit	A term for the computer characters 0-9.
End of Data code	A term generally used in connection with input operations. An End of Data code is control information that is included along with the actual data to indicate that no further data is available. On magnetic tapes, End of Data is designated by at least 12 characters of "%" codes.
End of File code	A special code stored along with actual data; used to separate files. In general storage, prime zero ('0) is used; on magnetic tape 12 "Z's" in at least one word are employed.
error	A detected computer malfunction.
erase	A term usually applied to magnetic drums and magnetic tapes to mean "record binary 0's" on the drum or tape. In this connection it effectively "clears" the area on which the 0's are recorded. Since other methods of "clearing" are employed in UFC (e.g., space codes and ignore codes are used by certain devices to clear drum areas) the term "erase" is used in UFC only in connection with revolvers and magnetic tapes, as: "the area ahead of the read/write head is erased prior to recording".
excess-three code	A binary coded decimal notation for decimal numbers. This code represents each decimal digit by a binary number (four bits in length) whose value is three greater than the value of the decimal digit that is coded.

	0-9, which complete the coil circuit of a selector, allowing that selector to be picked up only during the time the correspondingly numbered input/output unit is on demand.
destination address	An address that specifies a particular location (of definite capacity) at which information held in some intermediate location is to be stored.
digit	A term for the computer characters 0-9.
End of Data code	A term generally used in connection with input operations. An End of Data code is control information that is included along with the actual data to indicate that no further data is available. On magnetic tapes, End of Data is designated by at least 12 characters of "%" codes.
End of File code	A special code stored along with actual data; used to separate files. In general storage, prime zero ('0) is used; on magnetic tape 12 "Z's" in at least one word are employed.
error	A detected computer malfunction.
erase	A term usually applied to magnetic drums and magnetic tapes to mean "record binary 0's" on the drum or tape. In this connection it effectively "clears" the area on which the 0's are recorded. Since other methods of "clearing" are employed in UFC (e.g., space codes and ignore codes are used by certain devices to clear drum areas) the term "erase" is used in UFC only in connection with revolvers and magnetic tapes, as: "the area ahead of the read/write head is erased prior to recording".
excess-three code	A binary coded decimal notation for decimal numbers. This code represents each decimal digit by a binary number (four bits in length) whose value is three greater than the value of the decimal digit that is coded.

external memory

Any memory device in the system which can permanently store (and subsequently supply) data but which is not directly accessible to program control storage.

fault

A condition indicating a computer malfunction or a mistake in programming; an error.

field

A unit of data on the High Speed Drum, in the General Storage Buffer, or in the Block Transfer Buffer, which is a collection of adjacent characters; the number of characters vary from 1 up to 119 as defined by a field selection pattern.

file

The programming unit used for overall organization of data. On magnetic tape, a collection of adjacent items each of which consists of a fixed master portion comprising subjects common to all, or practically all, items, and to which are added or appended a variable number of other "trailer" subjects which may range in number from none to several hundred. In general storage a similar concept applies except that:

(a) Unit records are involved, and

(b) The general storage address for a unit record can be used to specify a portion of the information that must be coded in the master portion of items on magnetic tape.

flow-chart

A graphical representation of a sequence of programming operations using symbols to represent operations such as compute, substitute, compare, jump, etc.

flow diagram

A schematic-type of representation of a sequence of sub-routines designed to solve a problem. It is less detailed and less symbolic that a flow-chart and frequently includes descriptive text.

external memory

Any memory device in the system which can permanently store (and subsequently supply) data but which is not directly accessible to program control storage.

fault

A condition indicating a computer malfunction or a mistake in programming; an error.

field

A unit of data on the High Speed Drum, in the General Storage Buffer, or in the Block Transfer Buffer, which is a collection of adjacent characters; the number of characters vary from 1 up to 119 as defined by a field selection pattern.

file

The programming unit used for overall organization of data. On magnetic tape, a collection of adjacent items each of which consists of a fixed master portion comprising subjects common to all, or practically all, items, and to which are added or appended a variable number of other "trailer" subjects which may range in number from none to several hundred. In general storage a similar concept applies except that:

(a) Unit records are involved, and

(b) The general storage address for a unit record can be used to specify a portion of the information that must be coded in the master portion of items on magnetic tape.

flow-chart

A graphical representation of a sequence of programming operations using symbols to represent operations such as compute, substitute, compare, jump, etc.

flow diagram

A schematic-type of representation of a sequence of sub-routines designed to solve a problem. It is less detailed and less symbolic that a flow-chart and frequently includes descriptive text.

higher-order	The significance given data in the computer which is the same as that associated with the reading or hand-writing of data; the left-most character is the most significant or highest order digit; the right-most character is the least significant or lowest-order digit. (Exception: the sign digit is always the lowest-order digit in the computer.)
hub	A receptacle for wiring on a plugboard; the breakoff point between plugboard wiring (variable) and internal wiring (fixed).
information	A general term for both actual data and for control data.
ignore code	A computer character (1000000 in Univac code) used primarily to suppress comparisons.
instruction word	A 12-character computer word that defines a computer instruction; is stored in the operation memory of the computer, usually in sequence with other instruction words on the high speed drum.
intermediate storage (buffer storage)	A temporary storage location into which data is automatically transmitted from a programmed source or from which data is automatically obtained for storage in a programmed destination.
logical operations	The operations of masking, comparing, normalizing, etc., where, in essence, characters or bits constitute the elements being operated upon. In contrast to arithmetic operations wherein the elements of the operation are numerical values.
lower-order	Pertaining to the right-most digit of a typewritten or handwritten number or message. (See higher-order.) In UFC words, the sign is the lowest-order character.

higher-order	The significance given data in the computer which is the same as that associated with the reading or hand-writing of data; the left-most character is the most significant or highest order digit; the right-most character is the least significant or lowest-order digit. (Exception: the sign digit is always the lowest-order digit in the computer.)
hub	A receptacle for wiring on a plugboard; the breakoff point between plugboard wiring (variable) and internal wiring (fixed).
information	A general term for both actual data and for control data.
ignore code	A computer character (1000000 in Univac code) used primarily to suppress comparisons.
instruction word	A 12-character computer word that defines a computer instruction; is stored in the operation memory of the computer, usually in sequence with other instruction words on the high speed drum.
intermediate storage (buffer storage)	A temporary storage location into which data is automatically transmitted from a programmed source or from which data is automatically obtained for storage in a programmed destination.
logical operations	The operations of masking, comparing, normalizing, etc., where, in essence, characters or bits constitute the elements being operated upon. In contrast to arithmetic operations wherein the elements of the operation are numerical values.
lower-order	Pertaining to the right-most digit of a typewritten or handwritten number or message. (See higher-order.) In UFC words, the sign is the lowest-order character.

magnetic core	A form of rapid-access storage wherein information is represented as the polarization of a wire-wound core.
magnetic drum	A rapidly rotating cylinder, the surface of which is coated with a magnetic material on which information can be stored as small polarized spots.
magnetic tape	Tape consisting of a metal or a plastic base that is coated with a magnetic material on which polarized spots representing information can be recorded.
malfunction	Equipment failure.
memory	Information storage; any device into which information can be introduced and from which it can be extracted at a later time.
millisecond	One thousandth of a second.
mnemonic code	The (two) letters that suggest the name of the instruction and have the same lower-order four bits in Univac code as the numbers they represent. The process codes of instruction words have a mnemonic as well as numeric listing.
microsecond	One millionth of a second.
normalize	An operation in the UFC arithmetic section wherein an operand is shifted left until its most significant character is in a register's most significant character position.
"on demand"	The status of an input/output unit after a "Demand In" sequence is executed. In general, an I/O unit stays "on demand" until it receives an I/O instruction or until another I/O unit is placed "on demand".
"off demand"	The status of an input/output unit after it receives an I/O instruction or after another I/O unit is placed "on demand".
operand	The contents of a computer location used in arithmetic and logical operations.

magnetic core	A form of rapid-access storage wherein information is represented as the polarization of a wire-wound core.
magnetic drum	A rapidly rotating cylinder, the surface of which is coated with a magnetic material on which information can be stored as small polarized spots.
magnetic tape	Tape consisting of a metal or a plastic base that is coated with a magnetic material on which polarized spots representing information can be recorded.
malfunction	Equipment failure.
memory	Information storage; any device into which information can be introduced and from which it can be extracted at a later time.
millisecond	One thousandth of a second.
mnemonic code	The (two) letters that suggest the name of the instruction and have the same lower-order four bits in Univac code as the numbers they represent. The process codes of instruction words have a mnemonic as well as numeric listing.
microsecond	One millionth of a second.
normalize	An operation in the UFC arithmetic section wherein an operand is shifted left until its most significant character is in a register's most significant character position.
"on demand"	The status of an input/output unit after a "Demand In" sequence is executed. In general, an I/O unit stays "on demand" until it receives an I/O instruction or until another I/O unit is placed "on demand".
"off demand"	The status of an input/output unit after it receives an I/O instruction or after another I/O unit is placed "on demand".
operand	The contents of a computer location used in arithmetic and logical operations.

operation code	The right-most (or lowest-order) *three* characters of an instruction word.
overflow	A number which is beyond the capacity of a counter or register.
packing	Combining several different brief fields of information into one or more machine units of storage (as word locations on a track) for the purpose of conserving memory space.
pad	The filling out of a word or field with (Univac coded) zeros, space codes, or ignore codes.
parity bit	A redundant bit stored with each 6-bit Univac coded character. If the number of "1's" in the 6 bits of the Univac code is even, the parity bit is a "1"; if the number of "1's" in the 6 bits of the Univac code is odd, the parity bit is "0". An odd-even check on each character can thus be made during data transmissions. The parity bit is also used in defining the field selection patterns used in field addressing.
parity check	A check made on each character, generally during data transmissions, to determine if the number of "1's" in each 7-bit computer character is odd or even. If *odd,* operation continues; if *even,* a parity error is indicated.
plugboard	A removable connection panel whereon external wiring can be employed to define a program. The central computer's plugboard is called the "program control plugboard". Several I/O units also have plugboards.
probe	To interrogate; both pulse and d.c. probes are used.
process	The basic operation to be performed during a computer instruction.

operation code	The right-most (or lowest-order) *three* characters of an instruction word.
overflow	A number which is beyond the capacity of a counter or register.
packing	Combining several different brief fields of information into one or more machine units of storage (as word locations on a track) for the purpose of conserving memory space.
pad	The filling out of a word or field with (Univac coded) zeros, space codes, or ignore codes.
parity bit	A redundant bit stored with each 6-bit Univac coded character. If the number of "1's" in the 6 bits of the Univac code is even, the parity bit is a "1"; if the number of "1's" in the 6 bits of the Univac code is odd, the parity bit is "0". An odd-even check on each character can thus be made during data transmissions. The parity bit is also used in defining the field selection patterns used in field addressing.
parity check	A check made on each character, generally during data transmissions, to determine if the number of "1's" in each 7-bit computer character is odd or even. If *odd,* operation continues; if *even,* a parity error is indicated.
plugboard	A removable connection panel whereon external wiring can be employed to define a program. The central computer's plugboard is called the "program control plugboard". Several I/O units also have plugboards.
probe	To interrogate; both pulse and d.c. probes are used.
process	The basic operation to be performed during a computer instruction.

program control plugboard	The central computer's plugboard.
pulse	A short burst of electrical energy which is wired by the programmer to route the plugboard program from step to step and to initiate substeps.
random access	Access to storage locations in a nonsequential order.
read	The action whereby stored data is sensed and transmitted elsewhere, or is shifted out of a location and sent elsewhere.
read/write head	A dual-purpose device used to read and record data on a magnetic medium.
register (noun)	A device capable of storing a computer word or computer address.
result	The sole (or, in some cases, the principal) quantity produced in an arithmetic or logical process.
reverse process	The interchange of the relative function of operands in an arithmetic sequence to check a previous (normal) arithmetic operation.
revolver	A device which makes special use of a track on a magnetic drum to store and recirculate a fixed number of characters for the purpose of making the recirculated data readily accessible to the computer program.
rewind	To reposition tape by moving the tape in a backward direction.
serial (adjective)	Handling one character (or bit) after another using the same device, as opposed to parallel, the handling of all characters (or bits) simultaneously, each having a separate device.

program control plugboard	The central computer's plugboard.
pulse	A short burst of electrical energy which is wired by the programmer to route the plugboard program from step to step and to initiate substeps.
random access	Access to storage locations in a nonsequential order.
read	The action whereby stored data is sensed and transmitted elsewhere, or is shifted out of a location and sent elsewhere.
read/write head	A dual-purpose device used to read and record data on a magnetic medium.
register (noun)	A device capable of storing a computer word or computer address.
result	The sole (or, in some cases, the principal) quantity produced in an arithmetic or logical process.
reverse process	The interchange of the relative function of operands in an arithmetic sequence to check a previous (normal) arithmetic operation.
revolver	A device which makes special use of a track on a magnetic drum to store and recirculate a fixed number of characters for the purpose of making the recirculated data readily accessible to the computer program.
rewind	To reposition tape by moving the tape in a backward direction.
serial (adjective)	Handling one character (or bit) after another using the same device, as opposed to parallel, the handling of all characters (or bits) simultaneously, each having a separate device.

sign position	The low-order character position in any arithmetic register or in any location specified by a word address.
source address	An address specifying the location from which data can be obtained.
space code Δ	A computer character (0000001) in Univac code) used to represent a space or blank area.
storage reference	Usually, the action whereby a memory location specified by a computer address is found, and data is placed in or received from that location. Each storage reference actually involves two locations: in general, one is programmed, and the other (the intermediate or buffer location) is automatically determined by the instruction being executed.
stored program or (internally-stored program)	A sequence of instruction words which defines a computer program.
sub-instruction	An operation specified by one of the legal values of the special character "S/C" in an instruction word.
sub-step	An addition to or extension of a program step. The plugboard counterpart of sub-instruction.
temporary storage	Intermediate or buffer storage.
three-address logic	A type of computer instruction which includes: (a) The address (or storage location) from which the first operand is obtained, (b) The address from which the second operand is obtained, (c) A third address where the result is to be stored.
track	A narrow band on the periphery of the high-speed drum; the area which passes beneath a

sign position	The low-order character position in any arithmetic register or in any location specified by a word address.
source address	An address specifying the location from which data can be obtained.
space code Δ	A computer character (0000001) in Univac code) used to represent a space or blank area.
storage reference	Usually, the action whereby a memory location specified by a computer address is found, and data is placed in or received from that location. Each storage reference actually involves two locations: in general, one is programmed, and the other (the intermediate or buffer location) is automatically determined by the instruction being executed.
stored program or (internally-stored program)	A sequence of instruction words which defines a computer program.
sub-instruction	An operation specified by one of the legal values of the special character "S/C" in an instruction word.
sub-step	An addition to or extension of a program step. The plugboard counterpart of sub-instruction.
temporary storage	Intermediate or buffer storage.
three-address logic	A type of computer instruction which includes: (a) The address (or storage location) from which the first operand is obtained, (b) The address from which the second operand is obtained, (c) A third address where the result is to be stored.
track	A narrow band on the periphery of the high-speed drum; the area which passes beneath a

	read/write head as the high-speed drum revolves; the equivalent of a channel on a general storage drum.
transfer	To move a copy of data from one location to another.
translate	To change information from one code to another without affecting the value. In the UFC, all I/O units are equipped with translators so that Univac coded characters are delivered to and can be received from the central computer without any translation or conversion time being required in the central computer.
Univac	Universal Automatic Computer.
Univac code	A system of notation in which (commonly) 63 letters, numbers, and symbols are given values in a binary scale of notation. The notation is based on the (four-bit) excess-three code employed for the digits 0-9. In this code, the decimal digit "0" is represented as 0011 (binary "three"); the decimal digit "1" is represented as 0010 (binary "four"); and in general each of the decimal digits 0-9 is represented by a binary code whose value is three greater than the actual value of the digit. To the four-bits required to represent the decimal digits 0-9 in excess-three binary code, two higher-order bits, called zone bits, were added to formulate the rest of the Univac code. That is, the digit-portion of the code was expanded, and letters and symbols assigned to different values of the six bit combinations.
unpacking	Separating packed fields of information.
wiring	Manually-plugged wiring used on a plugboard or pinboard to connect various circuits within a unit for the purpose of defining what operations the unit is to perform.
word	12 characters; the operational (or machine) unit of data in arithmetic and logical operations of UFC.
write	To record; to store on a magnetic medium.

	read/write head as the high-speed drum revolves; the equivalent of a channel on a general storage drum.
transfer	To move a copy of data from one location to another.
translate	To change information from one code to another without affecting the value. In the UFC, all I/O units are equipped with translators so that Univac coded characters are delivered to and can be received from the central computer without any translation or conversion time being required in the central computer.
Univac	Universal Automatic Computer.
Univac code	A system of notation in which (commonly) 63 letters, numbers, and symbols are given values in a binary scale of notation. The notation is based on the (four-bit) excess-three code employed for the digits 0-9. In this code, the decimal digit "0" is represented as 0011 (binary "three"); the decimal digit "1" is represented as 0010 (binary "four"); and in general each of the decimal digits 0-9 is represented by a binary code whose value is three greater than the actual value of the digit. To the four-bits required to represent the decimal digits 0-9 in excess-three binary code, two higher-order bits, called zone bits, were added to formulate the rest of the Univac code. That is, the digit-portion of the code was expanded, and letters and symbols assigned to different values of the six bit combinations.
unpacking	Separating packed fields of information.
wiring	Manually-plugged wiring used on a plugboard or pinboard to connect various circuits within a unit for the purpose of defining what operations the unit is to perform.
word	12 characters; the operational (or machine) unit of data in arithmetic and logical operations of UFC.
write	To record; to store on a magnetic medium.

APPENDIX B

EXAMPLE OF INTERNAL ⇌ EXTERNAL PROGRAMMING

The UFC-I, with its combination of internal and external programming, provides a wide range of programming possibilities. Several of the ways in which control may be transferred from the internally-stored program to the plugboard and from the plugboard to the internal program are included in the accompanying program chart.

In example #1 control is transferred via a transcop instruction (instruction word 150) to program step #72. The result of step #72 is wired to the W ADR hub. This allows access to word location 162 from the plugboard. Wiring of the STEP OUT hub to NI returns control to the internal program.

Instruction Location	INSTRUCTION WORD											
	U			V			W			PR		S/C
150	A	A	A	A	A	A	1	6	2	7	2	A
151												

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
72	I/O-1		+ C	FS#1-8		W ADR		NI

Example #2 shows how the shift revolver may be loaded internally just prior to the transcop instruction, making the shifts defined in the shift revolver available to the external program. The shifts specified in instruction word 200 and the storages specified in instruction word 201 are available to external control until activation of the next instruction NI hub returns control to the internal program.

Instruction Location	INSTRUCTION WORD											
	U			V			W			PR		S/C
200	A	A	A	3	0	3	A	A	A	L	S	A
201	A	A	A	0	2	A	A	A	A	6	0	A
202												

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
60	GSB-A	IL	+ C	V ADR	V	GSB-A		NI

APPENDIX B

EXAMPLE OF INTERNAL ⇌ EXTERNAL PROGRAMMING

The UFC-I, with its combination of internal and external programming, provides a wide range of programming possibilities. Several of the ways in which control may be transferred from the internally-stored program to the plugboard and from the plugboard to the internal program are included in the accompanying program chart.

In example #1 control is transferred via a transcop instruction (instruction word 150) to program step #72. The result of step #72 is wired to the W ADR hub. This allows access to word location 162 from the plugboard. Wiring of the STEP OUT hub to NI returns control to the internal program.

Instruction Location	INSTRUCTION WORD											
	U			V			W			PR		S/C
150	A	A	A	A	A	A	1	6	2	7	2	A
151												

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
72	I/O-1		+ C	FS#1-8		W ADR		NI

Example #2 shows how the shift revolver may be loaded internally just prior to the transcop instruction, making the shifts defined in the shift revolver available to the external program. The shifts specified in instruction word 200 and the storages specified in instruction word 201 are available to external control until activation of the next instruction NI hub returns control to the internal program.

Instruction Location	INSTRUCTION WORD											
	U			V			W			PR		S/C
200	A	A	A	3	0	3	A	A	A	L	S	A
201	A	A	A	0	2	A	A	A	A	6	0	A
202												

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
60	GSB-A	IL	+ C	V ADR	V	GSB-A		NI

In Example #3, a breakpoint sequence is initiated by the special character 6. Assuming that BREAKPOINT #1 is depressed on the computer control panel, the BREAKPOINT #1 hub on the plugboard emits a pulse. This pulse is wired to CONDITION COMPARE and then to the IN of step #80, demonstrating that a breakpoint sequence may be used to transfer control to the plugboard, where both program steps and substeps may be executed.

Instruction Location	INSTRUCTION WORD U	V	W	PR	S/C	REMARKS	LINE NO.
300	98D04A98DAD	6				BKPT 1 - COND. COMP - 80	1
301							2

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
80	GSB-D		COMP	FS #1-A				BR 12
81	GSB-D		-C	FS #2-B		GSB-D		STOP

BRANCHING (BR)				
NO	IN FROM	+	−	0
12	STEP 80	STEP 82	STEP 84	STEP 81

Wiring of the STOP hub out of step #81 stops the computer program. However, when STOP is wired in this manner, the program will resume *internally* with instruction word 301 when the START button on the control panel is pressed. This illustrates a method of transferring control from an external to internal program.

In Example #3, a breakpoint sequence is initiated by the special character 6. Assuming that BREAKPOINT #1 is depressed on the computer control panel, the BREAKPOINT #1 hub on the plugboard emits a pulse. This pulse is wired to CONDITION COMPARE and then to the IN of step #80, demonstrating that a breakpoint sequence may be used to transfer control to the plugboard, where both program steps and substeps may be executed.

Instruction Location	INSTRUCTION WORD U	V	W	PR	S/C	REMARKS	LINE NO.
300	98D04A98DAD	6				BKPT 1 - COND. COMP - 80	1
301							2

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP
80	GSB-D		COMP	FS #1-A				BR 12
81	GSB-D		-C	FS #2-B		GSB-D		STOP

BRANCHING (BR)				
NO	IN FROM	+	−	0
12	STEP 80	STEP 82	STEP 84	STEP 81

Wiring of the STOP hub out of step #81 stops the computer program. However, when STOP is wired in this manner, the program will resume *internally* with instruction word 301 when the START button on the control panel is pressed. This illustrates a method of transferring control from an external to internal program.

APPENDIX C

EXAMPLES OF GENERAL STORAGE DRUM ADDRESSING SUBROUTINES

EXAMPLE 1: A CALCULATED GENERAL STORAGE DRUM ADDRESSING SUBROUTINE

In many cases, a system of part numbers, account numbers, employee badge numbers or inventory item numbers already in use by a company can be utilized as the basis for calculation of the drum addresses at which this information will be stored in the general storage drum system.

The following example demonstrates a method of arriving at a general storage drum address, using an account number carried on each labor ticket (punched card) as the basis for calculation.

Assumptions:

(1) A labor cost accounting system uses 6000 accounts, with account numbers ranging from 1001 to 7000.

(2) Accounts will be drum stored, 120 characters allotted to each account. Total requirements: four general storage drums.

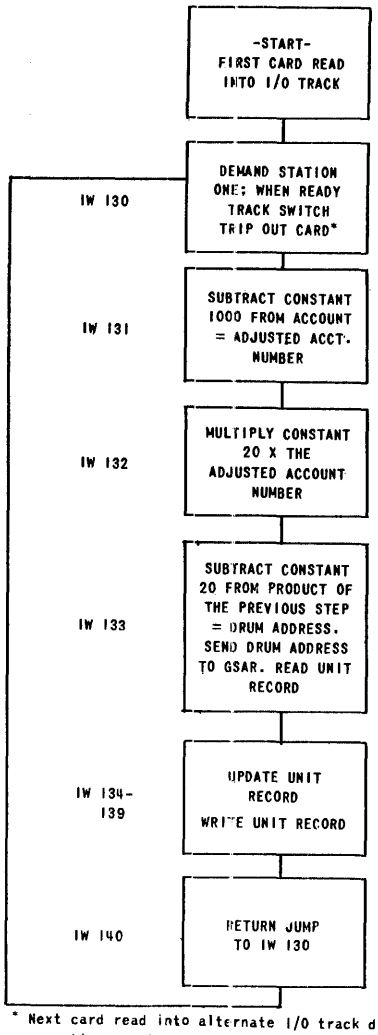
General Storage Drum	*Range of Addresses* From DS	CH	AA	To DS	CH	AA	Number of Unit Records Available
1st drum	00	00	00	02	99	80	1500
2nd drum	03	00	00	05	99	80	1500
3rd drum	06	00	00	08	99	80	1500
4th drum	09	00	00	11	99	80	1500
							Total.....6000

(3) Each incoming labor ticket (punched card) will carry an account number, which will be read into field A of the I/O track.

(4) Constant 1000 is stored in word address 111. Constant 20 is stored in word address 112.

(5) Unit record selector switch is set at position 0.

FLOW CHART - EXAMPLE I



* Next card read into alternate I/O track during the execution of the main program.

Instruction Location	INSTRUCTION WORD U	V	W	PR	SC	REMARKS	LINE NO.
						LABOR TICKET DATA READ ONTO I/O TRACK 09 FROM I/O DEVICE	1
							2
130	111 556	150	DE	Q		DEMAND STATION ONE, WHEN READY TRACK SWITCH, TRIP CARD	3
							4
131	01A 111	120	SB	A		SUBTRACT 1000 FROM ACCOUNT NO.	5
132	120 112	120	ML	A		MULTIPLY BY CONSTANT 20	6
133	120 112	299	SB	L		SUBTRACT CONSTANT 20, RESULT TO GSAR, PUR	7
134							8
135							9
136	MAIN PROGRAM						10
137							11
138							12
139							13
140	AAA 992	130	UJ	A		RETURN JUMP TO IW 130	14

Following is a series of typical account numbers and their condition at various points during the program as the address is being calculated.

I.W.	*Typical Account Numbers*			
130	1001	4127	6999	7000
	↓	↓	↓	↓
131	0001	3127	5999	6000
	↓	↓	↓	↓
132	0020	62540	119980	120000
	↓	↓	↓	↓
133	000000	062520	119960	119980

EXAMPLE 2: GENERAL STORAGE DRUM ADDRESS CALCULATED FROM CODES

The following example describes a method of using a combination of two codes to determine the exact drum location to which each transaction will be posted.

Assumptions:

(1) The ABC Freight Company desires a summarization of (1) the number of shipments, (2) total weight, and (3) total revenue, accumulated by a combination of commodity codes and destination codes. These summaries show the distribution, by commodity, of all shipments directed to each of 25 destinations.

(2) Input to the computer is obtained from a punched card containing data taken from a common carrier freight billing. The following information will be punched into the card:

Commodity code (range 01-190) (read into field 00A)
Destination code (range 01-25) (read into field 00B)
Amount of billing (read into field 00C)
Weight of shipment (read into field 00D)

(3) Each summary unit record will contain 24 characters, which will be addressed as follows when held in the general storage buffer:

Field A - Number of shipments (4 characters)
Field B - Total weight (10 characters)
Field C - Total revenue (10 characters)

A total of 4750 Unit Records (190 x 25) will be required.

Range of Addresses

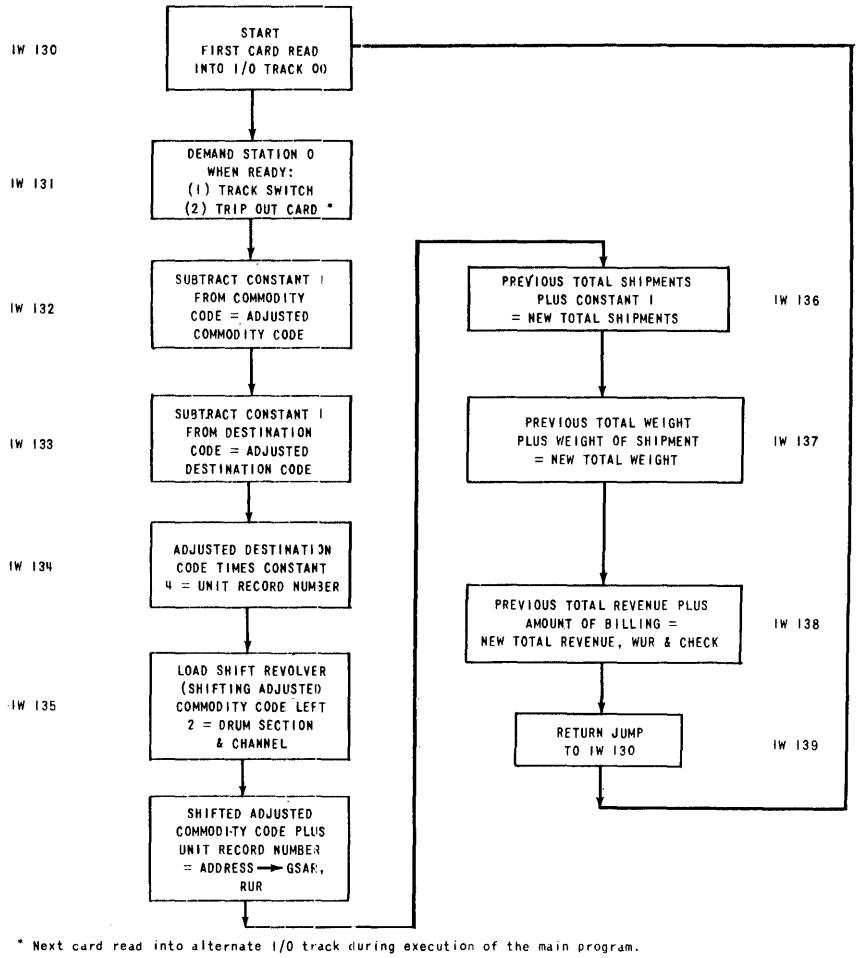
	From			To			Number of Unit Records Available
	DS	CH	AA	DS	CH	AA	
Drum Section 00:	00	00	00	00	99	96	2500
Drum Section 01:	01	00	00	01	89	96	2250
						Total.....	4750

(4) Channel number is determined from commodity code. Unit record number is determined from destination code.

(5) Constant 1 is stored in word 111.
Constant 4 is stored in word 112.

(6) Unit Record Selector Switch set at position 2.

FLOW CHART - EXAMPLE 2



* Next card read into alternate I/O track during execution of the main program.

Instruction Location	INSTRUCTION WORD U	V	W	PR	S/C	REMARKS	LINE NO.
						READ DATA ONTO I/O TRACK 00 FROM I/O UNIT	1
130	100556	156	0	E	Q	DEMAND STATION ZERO, WHEN READY TRACK SWITCH, TRIP CARD	2 3
131	00A111	100	S	B	Δ	SUBTRACT CONSTANT 1 FROM COMMODITY CODE	4
132	00B111	101	S	B	Δ	SUBTRACT CONSTANT 1 FROM DESTINATION CODE	5
133	101112	101	M	L	Δ	ADJUSTED DESTINATION CODE TIMES CONSTANT 4	6
134	202ΔΔΔ	ΔΔΔ	L	S	Δ	LOAD SHIFT REVOLVER (4 SHIFT 2 LEFT)	7
135	100101	995	A	D	L	ASSEMBLE ADDRESS → GSAR, RUN	8
136	98A111	98A	A	D	Δ	PREVIOUS NO. OF SHIPMENTS + CONSTANT 1 = UPDATED SHIPMENTS	9 10
137	98B00D	98B	A	D	Δ	PREVIOUS WEIGHT TO DATE + NEW WEIGHT = NEW TOTAL WEIGHT	11 12
138	98C00C	98C	A	D	P	PREVIOUS REVENUE TO DATE + BILLING = NEW TOTAL REVENUE. WUR & CHECK.	13 14
139	ΔΔΔ992	130	U	J	Δ	RETURN JUMP TO 130.	15

EXAMPLE 3: PARTIAL ADDRESS CALCULATION WITH CHANNEL SEARCH AND OVERFLOW

The following example describes one of many methods which may be used to store, and subsequently locate, items in general storage when the item identifier is long, and is complicated by mixed alpha-numeric characters.

Problem:

(1) The XYZ company maintains inventory records for 10,000 items within the general storage system of UFC-I. Each item contains 60 characters.. 1000 channels of general storage are required.

(2) Input to the computer consists of random sequenced punched cards representing various types of inventory activity.

(3) Stock numbers vary in length up to a maximum of nineteen alpha-numeric characters, and a direct addressing scheme or calculated addressing scheme is not feasible.

Solution:

(1) Add the low-order 10 characters of each nineteen character stock number (identifier) to the 9 high-order characters of the same identifier to obtain a randomized identifier.

(2) Choose a prime number (defined as: a number, not zero, divisible only by itself and ± 1) nearly equivalent to the number of channels required for storage of the inventory. This example uses 997 as the prime number.

Identifier: 8RQ394269932SA69378

Randomize: 8RQ394269
+ 932SA69378
101RQ363647

(3) Divide the sum of the previous addition by the prime number selected (997), developing a remainder of 797. The high-order digit of this remainder provides the drum section (07) and the low-order two digits provide the channel (97) at which a channel search is initiated.

$$997 \overline{)\ 101RQ363647}\ 10229050$$

797 or DS 07
CH 97

(4) During loading of data onto the drums, channel search locates the first open unit record area on this channel, and the item is stored at that location.

During the processing of data, channel search is initiated at the drum section and channel address calculated as above, and the item is read into the general storage buffer when found.

(5) An overflow address is loaded into the search control location of each channel. This overflow address will direct the search to an overflow area when the track where the search begins has been filled.

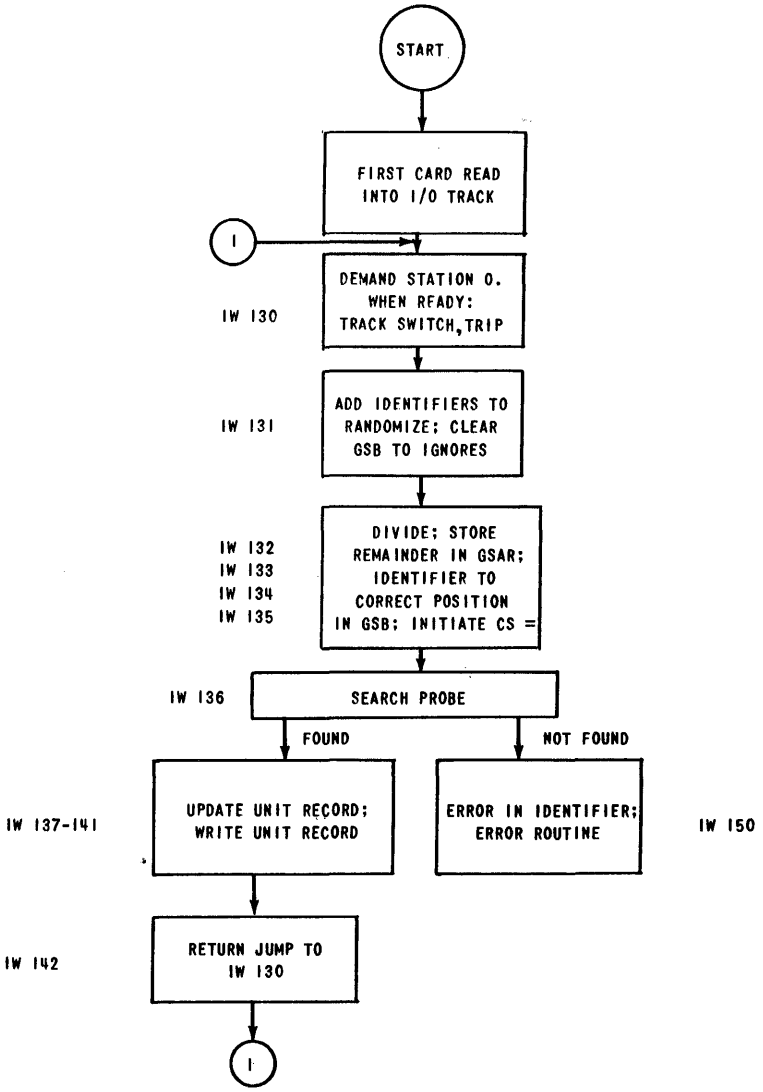
Comment:

In summary, the objective of the overflow method of storing and locating data in general storage is to distribute the items to be stored as evenly as possi-

ble over the available drum area by developing reproducible random numbers from the file identifiers to be used as track addresses. Thus, instead of directly addressing a record, its approximate location is ascertained and a channel search locates the exact record desired.

If the criterion for a successful overflow system is met, by far the greater percentage of the file items are located by search of a single channel, and the remainder of the items are located in a search of less than two channels.

OVER FLOW METHOD DRUM ADDRESSING



Instruction Location	INSTRUCTION WORD U	V	W	PR	S/C	REMARKS	LINE NO.
130	111	556	260	DE	A	DEMAND STATION 1, TRIP, TRACK SWITCH	1
131	01A	01B	111	AD	K	ADD PORTIONS OF IDENTIFIERS; CLEAR GSB	2
132	000	000	202	LS	A	SET UP SHIFT	3
133	111	110	995	DR	E	IDENTIFIER ÷ CONSTANT 997 → GSAR; LEFT 2	4
134	01A	AAA	98A	AT	A	FIRST HALF OF IDENTIFIER → GSB	5
135	01B	AAA	98B	AT	N	SECOND HALF OF IDENTIFIER → GSB; CS =	6
136	136	150	150	SP	A	SEARCH PROBE	7
137							8
138							9
139	UPDATING ROUTINE					PROCESS ITEM	10
140							11
141					M	FINAL STEP OF PROCESSING; WUR	12
142	992	993	130	UJ	A	RETURN JUMP TO IW 130	13

APPENDIX D

PROGRAMMING FORMS

The following group of programming forms are suggested for use by programmers and analysts in coding programs for UFC-I. Segments of these forms are used throughout the manual to illustrate coding techniques for both internal and external programming.

Figure 1 is designed for use in internal programming.

Figure 2 is designed for use in external programming.

Figures 3, 4, and 5 provide an aid to the programmer in keeping track of the many computer functions available to him on the program control plugboard.

Figure 6 serves a similar purpose in programming for the use of selectors.

Model 1 UNIVAC® File-Computer Program Chart

Customer	Programmed by Date	Revisions by Date	Revisions by Date	Program No. Page ____ of ____
Application	Checked by Date	Revisions by Date	Revisions by Date	Date Installed by

Instruction Location	INSTRUCTION WORD U	V	W	PR	S/C	REMARKS	LINE NO.
							1
							2
							3
							4
							5
							6
							7
							8
							9
							10
							11
							12
							13
							14
							15
							16
							17
							18
							19
							20
							21
							22
							23
							24
							25
							26
							27
							28
							29
							30
							31
							32
							33
							34
							35
							36
							37
							38
							39
							40

FIGURE 1

MODEL 1 UNIVAC FILE-COMPUTER EXTERNAL PROGRAM CHART

CUSTOMER:	APPLICATION:	PROGRAMMED BY: DATE	PROGRAM NO.

STEP NO.	V_1	V_1 SHIFT	PROCESS	V_2	V_2 SHIFT	R	R SHIFT	NEXT STEP	REMARKS
51									
52									
53									
54									
55									
56									
57									
58									
59									
60									
61									
62									
63									
64									
65									
66									
67									
68									
69									
70									
71									
72									
73									
74									
75									
76									
77									
78									
79									
80									
81									
82									
83									
84									
85									
86									
87									
88									
89									
90									
91									
92									
93									
94									
95									
96									
97									
98									

FIGURE 2

MODEL 1 UNIVAC® FILE COMPUTER CONTROL CHART #1

CUSTOMER:	APPLICATION:	PROGRAMMED BY: DATE	PROGRAM NO.

DEMAND UNITS (DMD)

UTS	TEST IN	NOT READY	READY	DEMAND IN	DEMAND OUT	SPECIAL OUT	TRACK SWITCH
0							
1							
2							
3							
4							
5							
6							
7							
8							
9							

UNIBUSES (U/B)

NO	IN	IN	IN	IN	OUT
1					
2					
3					
4					
5					
6					
7					
8					

START TO:

ERROR SIGNALS

TYPE	TO
PARITY	
÷ O'FLOW	
± O'FLOW	
NO'FLOW	
ARITH.	
GS PROG	

HI-SPEED CONTROL LINES (HCL)

NO	IN	YES	NO
W			
X			
Y			
Z			

STEP CLEAR (SC)

NO	IN	OUT
1		
2		
3		
4		

STEP REPEAT (SR)

NO	FROM
1	
2	
3	
4	

FUNCTION SEQUENCE (FS)

NO	SET	PROBE	OUT
1			
2			
3			
4			

OUT EXPANDERS (OE)

NO	IN	IN	OUT
1			
2			
3			
4			
5			
6			
7			
8			

FUNCTION DELAY (FD)

NO	IN_1	IN_2	OUT
A			
B			
C			
D			

OUTPUT CONTROL LINES (OCL)

NO	FROM	NO	FROM
A		F	
B		G	
C		H	
D		I	
E		J	

REMARKS:

U 1356.4

FIGURE 3

MODEL 1 UNIVAC ® FILE-COMPUTER CONTROL CHART #2

CUSTOMER:	APPLICATION:	PROGRAMMED BY: DATE	PROGRAM

READ URA (R)

NO	IN	OUT
1		
2		
3		
4		

WRITE URA (W)

NO	IN	OUT
1		
2		
3		
4		

WRITE & CHECK URA (W/C)

NO	IN	OUT
1		
2		
3		
4		

CHANNEL SEARCH EQUAL (ECS)

NO	IN	OUT
1		
2		
3		
4		

CHANNEL SEARCH UNEQUAL (UCS)

NO	IN	OUT
1		
2		
3		
4		

CONDITION COMPARE (C/C)

NO	IN	OUT
1		
2		
3		
4		

CLEAR BLOCK TRANSFER BUFFER (CLBTB)

NO	IN	OUT
1		
2		
3		
4		

CLEAR GEN'L STORAGE BUFFER (CLGSB)

NO	IN	OUT
1		
2		
3		
4		

BRANCHING (BR)

NO	IN FROM	+	−	O
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				

CHANNEL SEARCH PROBE & WAIT (CS P/W)

NO	IN FROM	+	−	O
1				
2				
3				
4				

CHANNEL SEARCH PROBE (CS P)

NO	IN FROM	ACTIVE	+	−	O
1					
2					
3					
4					

CODE DISTRIBUTOR (CD)

ALPHA/NUMERIC IN:

GROUP 1 IN		GROUP 2 IN		GROUP 3 IN		GROUP 4 IN	
NO	OUT	NO	OUT	NO	OUT	NO	OUT
0		;		,		+	
1		A		J		/	
2		B		K		S	
3		C		L		T	
4		D		M		U	
5		E		N		V	
6		F		O		W	
7		G		P		X	
8		H		Q		Y	
9		I		R		Z	

CDR PULSE IN:

NO	OUT	NO	OUT	NO	OUT	NO	OUT
0		3		6		9	
1		4		7			
2		5		8			

U 1356.5

FIGURE 4

MODEL 1 UNIVAC® FILE-COMPUTER CONTROL CHART #3

CUSTOMER:	APPLICATION:	PROGRAMMED BY: DATE	PROGRAM NO.

INPUT CONTROL LINES (ICL)

NO	TO
a	
b	
c	
d	
e	
f	
g	
h	
i	
j	
k	
l	

INDICATORS (IND)

NO	FROM
1	
2	
3	
4	
5	
6	
IND. SWITCH IN:	
IND. SWITCH OUT:	

BREAKPOINTS (B/P)

1	
2	
3	

PROGRAM SELECTS (PS)

NO	IN	DELAY OUT	DROP OUT	B+
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
CLEAR PROGRAM SELECTS:				

SPECIAL CHARACTER OUTS (SCO)

NO	TO	NO	TO	NO	TO
Q		T		W	
R		U		X	
S		V		Y	

ALTERNATE SWITCHES (ALT.SW.)

NO	SELECT	COMMON	NON-SELECT
1			
2			
3			
4			
5			
6			

REMARKS:

U 1356.6

FIGURE 5

MODEL I UNIVAC FILE-COMPUTER SELECTOR CHART

CUSTOMER:	APPLICATION:	PROGRAMMED BY: DATE	PROGRAM NO.

T #	PICK-UP FROM	GROUND	SELECT	COMMON	NON-SELECT

U 1356.7

FIGURE 6

APPENDIX E

PROGRAM CONTROL PLUGBOARD

COMPUTER CURRENTS AND PULSES

As the programmer wires the program control plugboard of the UFC-I, he should bear in mind that the internal wiring behind the plugboard provides several kinds of electrical energy designed for specific programming purposes:

Pulse:	A short burst of electrical energy which is wired by the programmer to route the plugboard program from step to step, and to initiate substeps.
D-C Enable:	A computer-controlled direct current, the plugboard wiring of which allows the computer to gain access to storages, processes and shifts.
B+ Current:	A steady direct current used by the programmer to pick up selectors and to light indicators (any other use of this current may result in damage to the computer).
Computer Ground:	A level of zero potential which completes the coil circuit of a selector allowing that selector to be picked up at any time during the computer program.
Demand Ground:	A level of zero potential which completes the coil circuit of a selector allowing that selector to be picked up only during the time the correspondingly-numbered I/O unit is on demand.

FUNDAMENTAL PLUGBOARD PROGRAMMING RULES

1. Pulse outs (red) may be wired only to pulse ins (green). This can be done directly or via buses, selectors or alternate switches.

2. Enable outs (blue) may be wired only to enable ins (yellow). This can be done directly or via buses, selectors or alternate switches.

3. B+ outs (black) may be wired only to B+ ins (purple). This can be done directly or via buses, selectors or alternate switches.

4. Selector ground (orange) may be wired to either computer ground (orange) or demand ground (orange).

5. Multipurpose hubs (not color coded) may be wired to route pulses, enables, ground or B+.

6. The combination green-red hubs are actually bused pulse in hubs; either hub can be wired from a pulse source, and the other wired to define the next operation.

7. The combination blue-yellow hubs are actually bused enable out hubs; they are shown in combination color code because these hubs may be "chain wired".

8. The combination purple-black hubs are connected by a switch-controlled bus; either can function as an in and the other as an out for B+.

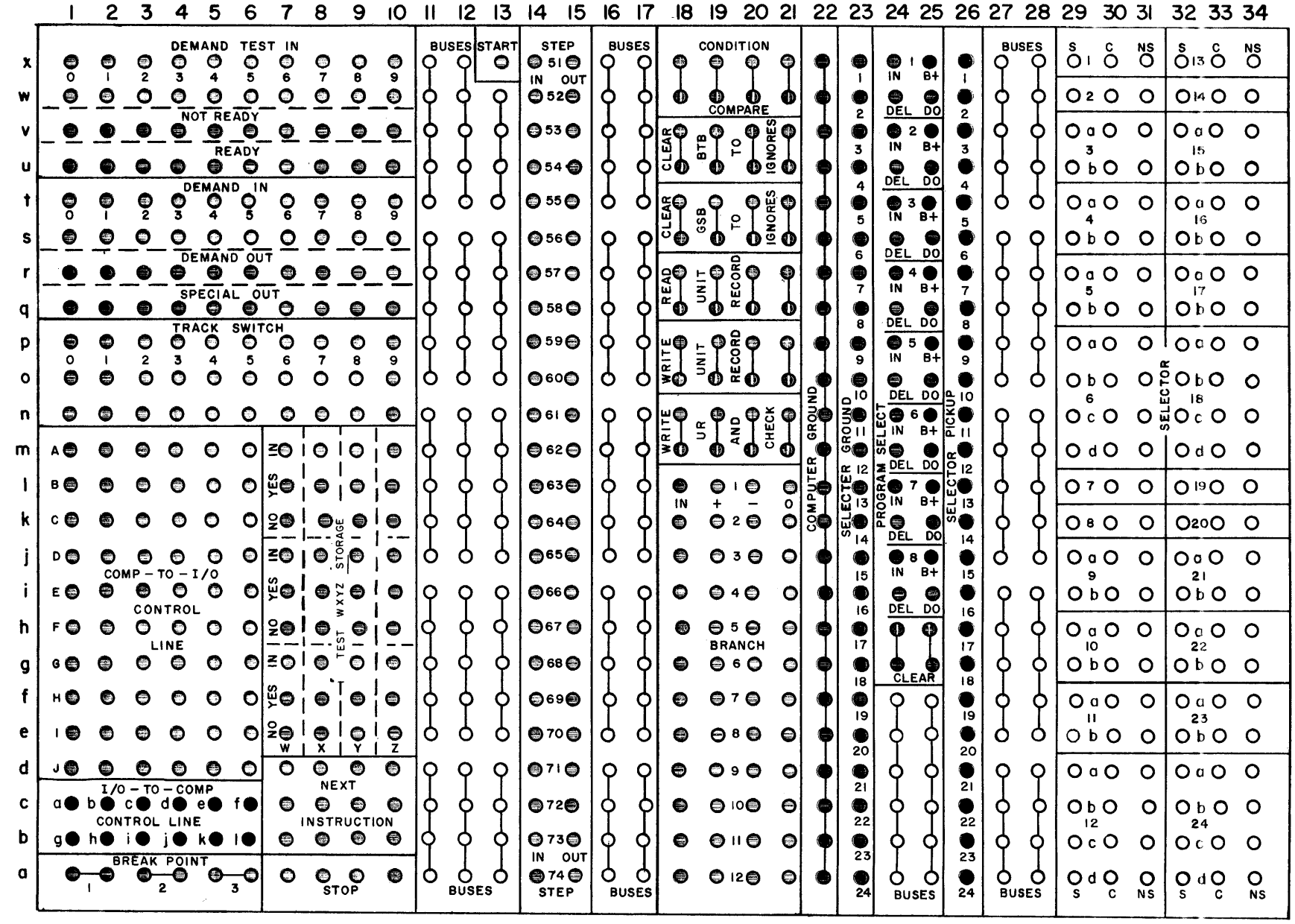
CHAIN WIRING

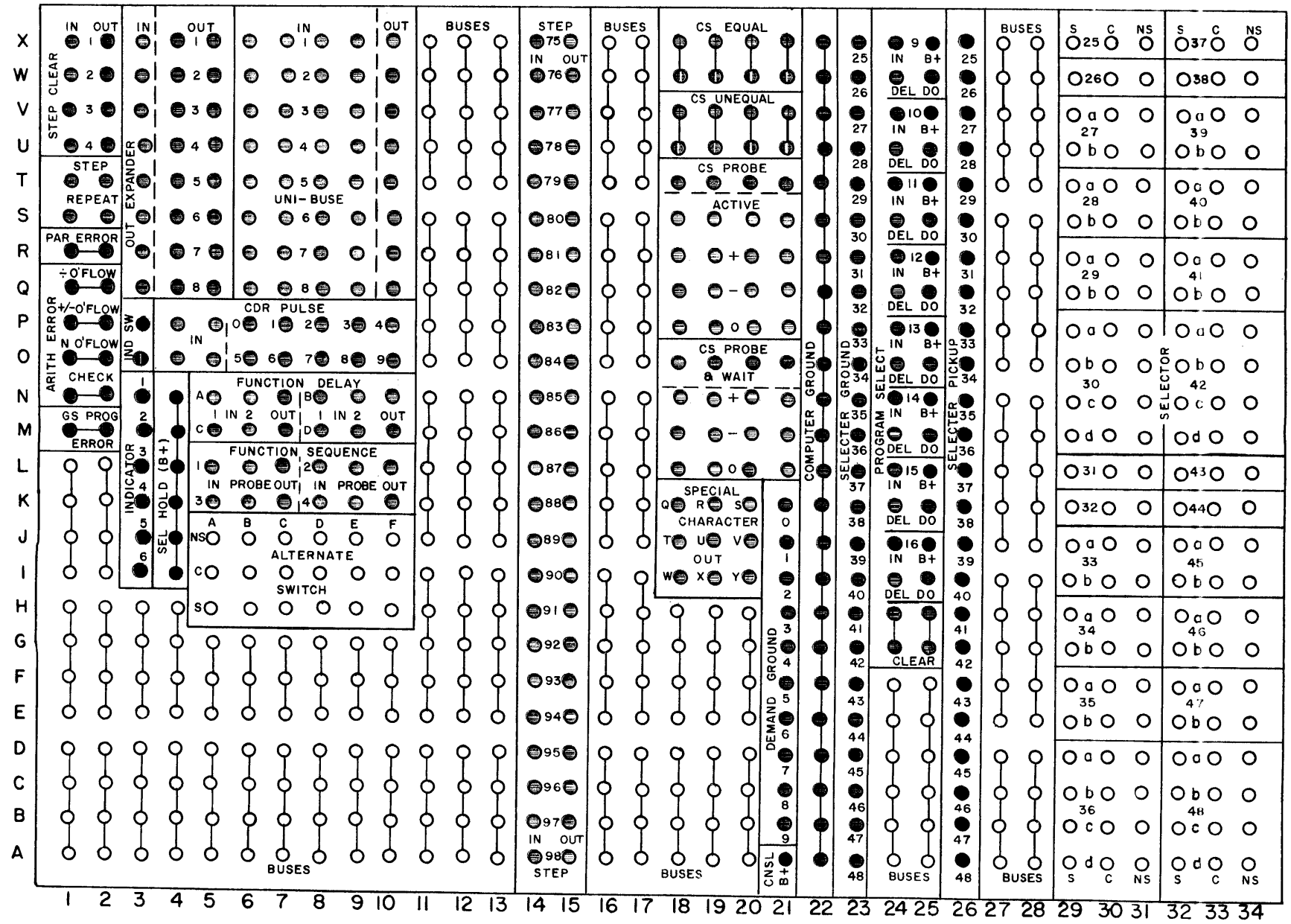
Chain wiring is a technique which allows the programmer to wire several enable out (yellow-blue) hubs to a common enable in (yellow) hub without the use of a bus. For example: assuming that the process for steps 51, 52, 55 and 60 is arithmetic transfer, the process hubs for these steps may be chain wired together and the chain tied to arithmetic transfer AT by a single wire. The PROC, V_1ADR, V_2ADR, R ADR, V'_1SH, V'_2SH and R SH may be chain wired to any of the enable in (yellow) hubs which specify processes, storage locations and shifts.

Another convenient wiring technique which incorporates chain wiring is facilitated by the three plugboard hubs provided for each storage location and shift on the plugboard. Whenever a V_1ADR or V_1SH is wired either singly or in a chain, the receiving hub at the left of the storage or shift enable in may be used. Similarly a V_2ADR or V'_2SH may use the middle enable in hub, and the R ADR and R SH may use the right enable in hub.

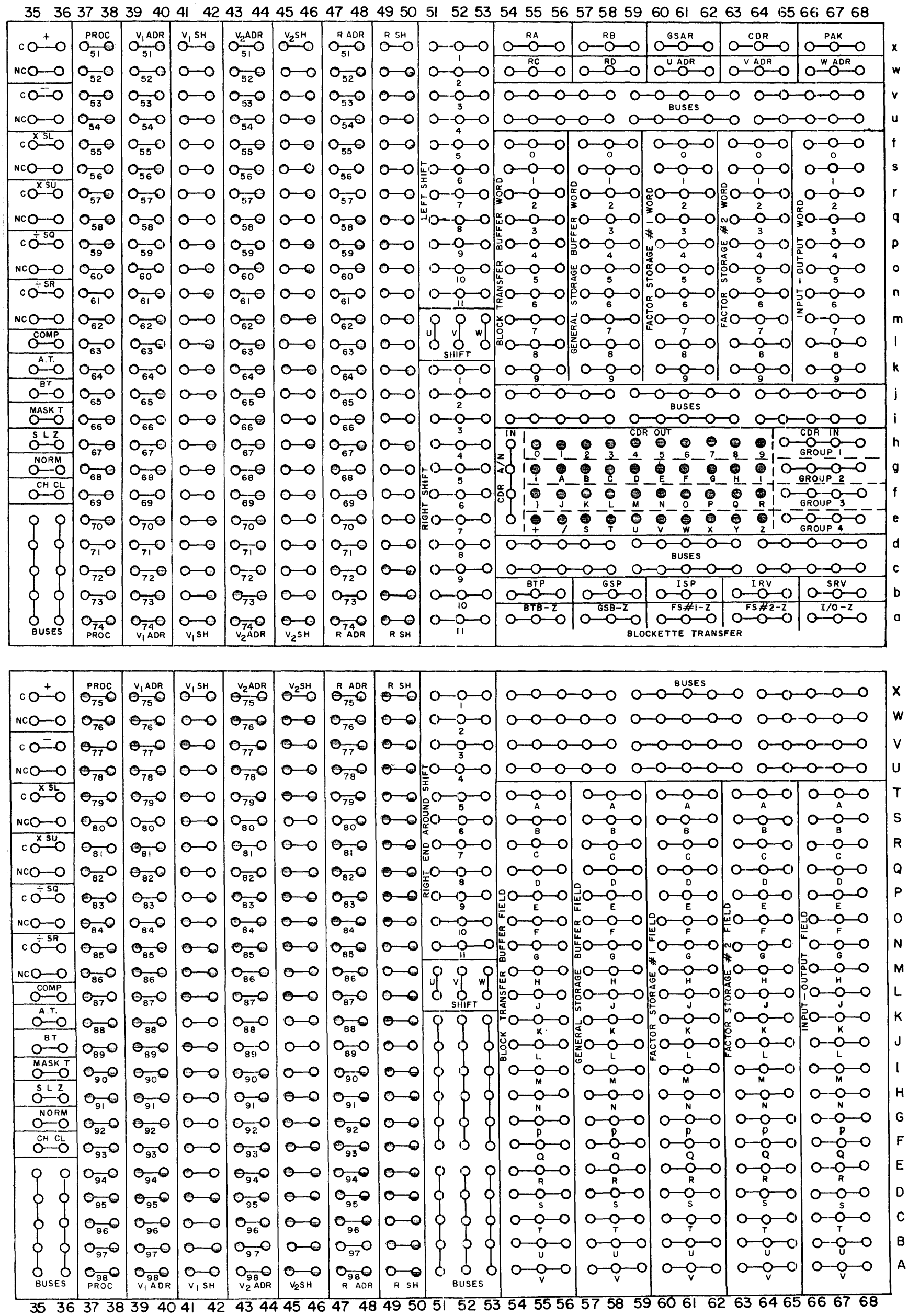
Through the use of the above techniques, buses are made available for other purposes, plugboard wiring is simplified, and debugging procedures made easier.

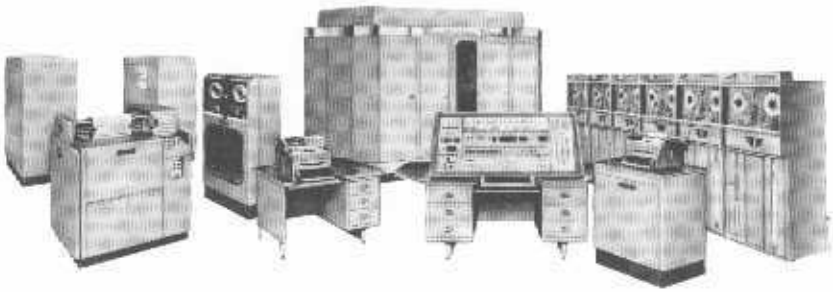
PULSE IN PULSE OUT B+ IN B+ OUT GROUND POTENTIAL





○ ENABLE IN ◉ ENABLE OUT





Univac II Systems • For data-automation which involves large volumes of input and output.

THE UNIVAC® FAMILY

Univac File-Computer • For instantaneous random access to large-scale internal storage—plus computation.

Univac 60 & 120 Computers • For speeding and simplifying the procedures of punched-card systems.

OF ELECTRONIC COMPUTERS

Univac Scientific Systems • For complex and intricate computations of engineering and research.

UNIVAC®—The FIRST Name in Electronic Computing Systems

U-1474