

*Reference Manual*

*ALTRAN*  
*Translator*  
*for the*  
*Bendix G-15 Computer*

ALTRAN TRANSLATOR REFERENCE MANUAL

FOR THE

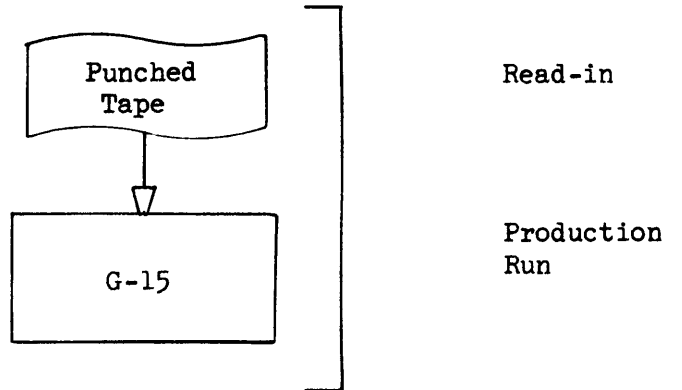
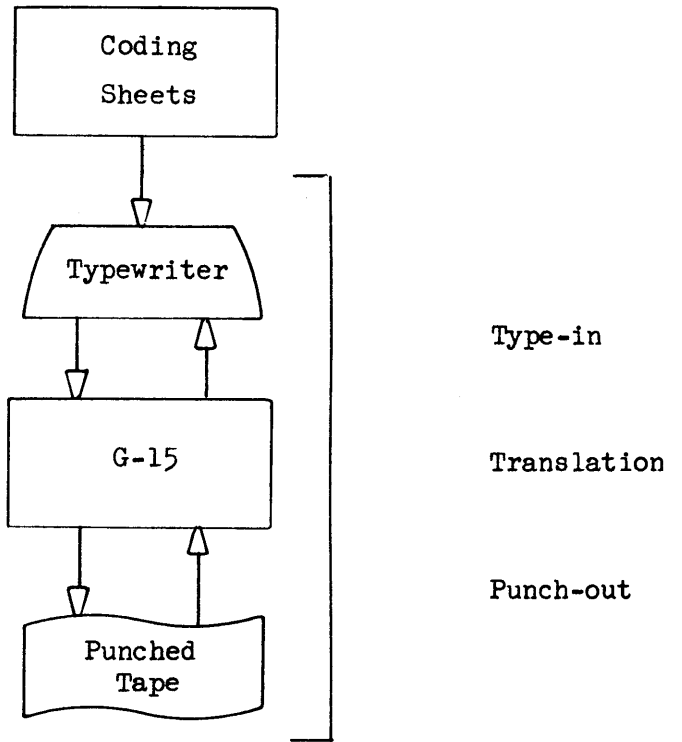
BENDIX G-15 COMPUTER

TABLE OF CONTENTS

		<u>Page</u>
	Introduction	iii
Chapter 1	Introduction to ALTRAN Coding, Formats, Examples	1
Chapter 2	Basic Hardware Information, Commands, Numbers, Scaling	6
Chapter 3	Basic ALTRAN Information	13
Chapter 4	Instructions	23
Chapter 5	Commands	46

TABLE OF CONTENTS

		<u>Page</u>
	Introduction	<u>iii</u>
Chapter 1	Introduction to ALTRAN Coding, Formats, Examples	1
Chapter 2	Basic Hardware Information, Commands, Numbers, Scaling	6
Chapter 3	Basic ALTRAN Information	13
Chapter 4	Instructions	23
Chapter 5	Commands	46



## INTRODUCTION

This manual is intended to instruct the programmer in the use of ALTRAN and to serve as a reference when coding in ALTRAN. It assumes little previous knowledge of the G-15 hardware or programming in machine language. Only the following concepts are necessary in order to use ALTRAN:

1. Basic knowledge of programming, including branches, loops, and subroutines.
2. Binary and decimal representation.
3. Logic operations---extract, unite and complement.

### THE G-15 COMPUTER

The Bendix G-15 is a powerful, compact, internally programmed, digital computer. The basic unit provides a complete, general purpose, computing system in a single cabinet. A photo-electric tape reader and a tape punch are built into the machine. A special typewriter is provided for control and can be used for input and output. The system is expandable by means of numerous accessories.

Information can be expressed in decimal notation during input and output. Internally during computation, information exists in serial, binary form.

The memory of the G-15 consists of a rotating magnetic drum on which information is stored serially in binary form. A "word" of information contains 29 binary digits (bits) and may represent either a command or a number. If the word represents a number, the least significant bit is used for the sign of the number and the remaining 28 bits represent the number's absolute value.

Most of the information stored on the drum is in 20 channels or "long lines", numbered 00 through 19. Each long line can contain 108 words numbered 00 through 99 and u0 through u7 (100 thru 107).

Four rapid access or "short lines" are provided on the drum, numbered 20 through 23. Each short line can contain 4 words numbered 00 through 03. Average access time for the short lines is 1/27 of the access time for a long line. The average access time for a short line is .54 milliseconds.

Three 2-word registers are also provided for arithmetic and storage as well as a one-word accumulator. The details of the operation of these registers and the accumulator are discussed in Chapter 2.

### CAPABILITIES OF ALTRAN

ALTRAN is a programming system which will translate simplified commands with alphabetic operation codes into optimized G-15 machine language commands. The translator operates on a "one for one" basis, producing one machine language command for each simplified command entered. The translation takes place immediately and the translated command may be typed out for the operator's use, at the option of the operator.

ALTRAN occupies lines 02, 05, 07-19, 20-23, and all the registers and the accumulator during the translation operation. The remaining long lines can be used by the programmer for the storage and debugging of the translated machine language program. Therefore, the operator can, if he desires, use lines 00, 01, 03, 04, and 06 for his program while retaining all the debugging facilities of ALTRAN. If his program cannot be made to fit within these lines, the programmer has the option of breaking the program into pieces small enough to debug within these lines or translating his program a line at a time and punching the object program onto paper tape for later debugging through standard procedures such as the use of "Short PPR" (a debugging routine available to all G-15 users). It should be emphasized that the object program can be used in any command line and make use of the full capabilities of the G-15 memory. The restrictions in memory mentioned above apply only during the translation of the simplified commands to machine language commands, not during the execution of the object program.

In translating a program to machine language, the programmer would proceed as outlined below.

1. Load the ALTRAN programming system magazine.
2. Type in the simplified commands and constants used by his program.
3. Punch a tape of his program.
4. Debug and repunch program, if necessary.
5. Run the translated program.

## CHAPTER 1

### INTRODUCTION TO ALTRAN CODING

This chapter discusses some of the basic aspects of the ALTRAN language and demonstrates some of the concepts involved in programming with ALTRAN. Detailed explanations of the instructions and commands used in this chapter are presented in Chapters Four and Five and are used here merely as examples of ALTRAN coding.



## LOADING ALTRAN

ALTRAN is loaded into the computer by placing the rewound paper tape magazine on the photo-reader, typing P with the Enable switch ON, waiting for the Ready light to go ON, and placing the Compute switch to GO. When the typewriter returns the carriage, ALTRAN is loaded and ready for input from the operator. During the loading process, lines 00, 01, 03, 04, and 06 are not changed.

## INPUT FORMAT

Instructions to the ALTRAN system and Commands to be translated are typed into the computer in the following format:

### Instructions

1-letter instruction code, "space", address, "tab"

Example:           Z 00ZZ           (clears line 00)

### Commands

3-letter command code, "space", address, "tab"

Example:           ADD 2202           (add the contents of 2202)

Notice that "space" and "tab" mean that these keys are typed and not the letters s p a c e and t a b.

In the case where the "address" field is zero (0000), the Instruction or Command code should be followed by "tab" to avoid unnecessary typing.

Notice that all Instruction and Command code letters must be entered in lower case. If upper case letters are entered they will be rejected by the system as invalid. (Upper case letters are shown in this manual only because this corresponds more closely to the lower case Elite Gothic type of the alphanumeric typewriter.)

## EXAMPLES

The following pages represent some very simple examples of programs translated and run using ALTRAN. They were typed into the computer as shown and the comments were added after the fact. The reader may wish to follow these examples to get a "feel" for the operation of the system. Each example includes all of the type-in used including the loading operation.

Information enclosed in a box is typed by the operator in all of the following examples.

ADDITION OF TWO NUMBERS

It is required to write a program which will accept the input of two 7-digit decimal numbers, add the numbers together, and type the sum (in decimal) on the typewriter.

P		Load ALTRAN
Z 00ZZ	Z .00ZZ	Clear or "Zero" Line 00
X 18u2	X .18u2	Suppress Machine Language Command Type-back
X 18u4	X .18u4	Suppress Verification of Input
C 0000		Permit ALTRAN Command Entry Starting at 0000.

.0000	TII	Type-in Integer
.0002	SAR 2200	Store AR in 2200
.0005	TII	Type-in Integer
.0007	SAR 2201	Store AR in 2201
.0010	CLA 2200	Clear and Add Contents of 2200
.0013	ADD 2201	Add Contents of 2201
.0018	SSD 2800	Store Sum or Difference in 2800 (AR)
.0020	PIC	Print Integer, Carriage Return.
.0022	TRA 0000	Transfer to 0000
.0000	w.02.81.2.21.31	<span style="border: 1px solid black; padding: 2px;">S 0000</span> Start Computing at 0000

1111111	Ⓢ 2222222	Ⓢ .3333333	} Program in Operation Showing Input of the Two Numbers And Sum Typed by the Computer.
1234567	Ⓢ 1234567	Ⓢ .2469134	
7777777	Ⓢ 2222222	Ⓢ .9999999	
1010101	Ⓢ 0101010	Ⓢ .1111111	
1	Ⓢ 2	Ⓢ .0000003	
222	Ⓢ 333	Ⓢ .0000555	
Ⓢ c5f		Regain ALTRAN Control.	
P 0023		Punch 0000 Through 0023-1	
T 0204		Transfer Line 02 to Line 04.	
C 0444		Permit ALTRAN Command Type-in Starting at 0444	

.0444	u.46.45.5.20.31	<span style="border: 1px solid black; padding: 2px;">RMP 00</span>	Return to Marked Place, Line 00
-------	-----------------	--	---------------------------------

<span style="border: 1px solid black; padding: 2px;">1</span>	1 ?	<span style="border: 1px solid black; padding: 2px;">P 0400</span>	Punch Line 04
---	-----	--	---------------

← Error. Number 1 Entered Instead of P. Note Rejection of Input Followed By ?

MULTIPLICATION OF TWO FRACTIONS

It is required to write a program which will accept the input of two 7-digit decimal fractions, find their product, and type out the product as a decimal fraction.

P			Load ALTRAN
Z 00ZZ	Z	.00ZZ	Clear (Zero) Line 00
C			Permit ALTRAN Command Entry Starting at 0000.

.0000	TIF	TIF	w.02.56.2.21.31	Type in Fraction
.0002	SAR	2200	SAR .2200 w.04.05.0.28.22	Store AR in 2200
.0005	TIF	TIF	w.07.56.2.21.31	Type in Fraction
.0007	SAR	2201	SAR .2201 w.09.10.0.28.22	Store AR in 2201
.0010	CLR	CLR	u.13.13.0.23.31	Clear 2-word Registers
.0013	LID	2200	LID .2200 w.16.19.6.22.25	Load ID from 2200
.0019	LMQ	2201	LMQ .2201 w.21.23.0.22.24	Load MQ from 2201
.0023	MPY	MPY	u.56.80.0.24.31	Multiply (Single Precision).
.0080	TPR	TPR	u.82.82.0.26.28	Transfer Product to AR
.0082	PFC	PFC	w.84.14.2.21.31	Print Fraction, Carriage Return.
.0084	TRA	0000	TRA .0000 u.00.00.0.21.31	Transfer to 0000.
.0000		w.02.56.2.21.31	S 0000	S .0000 Start Computing at 0000.

.500000	S .500000	S .0250000
.500000	S .500000	S .2500000
.1000000	S .1234567	S .0123457
.1000000	S .1234565	S .0123456
.1000000	S .1234566	S .0123457
.0010000	S .0010000	S .0000010

Actual Input and Output  
During Program Execution.



## CHAPTER 2

### SOME BASIC HARDWARE INFORMATION

This chapter explains some of the basic information about the G-15 computer which is necessary for the ALTRAN programmer. The subjects covered include machine language command structure, number forms, and scaling of numbers.

## MACHINE LANGUAGE COMMAND STRUCTURE

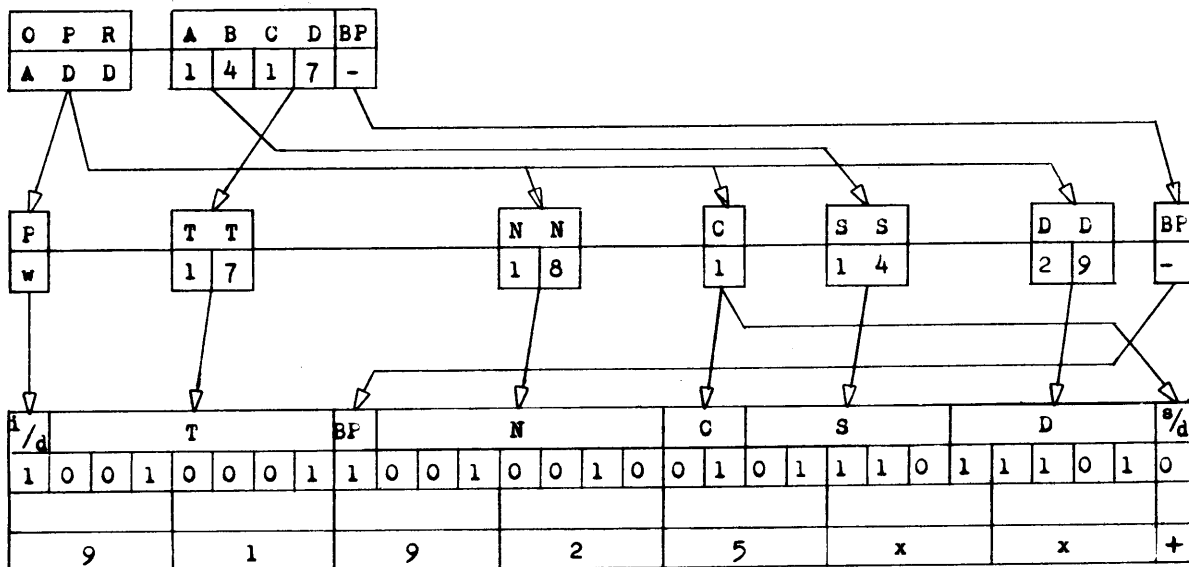
The command generated by the ALTRAN system is called a machine language command. This command is one which can be understood, without further interpretation or translation, by the basic circuits of the computer. The computer looks at this command as being in binary. The programmer will normally use the decimal equivalent of the binary command in communicating with the computer in its language. That is to say, the programmer or operator will normally enter the decimal equivalent, it will be converted to binary, and the computer can then understand it.

It is important, since the ALTRAN system produces machine language commands, that the programmer be familiar with the various parts of the command. This information will enable the programmer to modify commands to give the desired operation in the case where it is not available in the ALTRAN system and to understand what each command generated by ALTRAN will accomplish.

The various formats which a command may have in the ALTRAN system are described below. An explanation of the various components of a command follows the format diagram.

### COMMAND FORMATS

The following illustration shows the various forms which a command may take in the ALTRAN system. The first is the standard ALTRAN command format. It is followed by the decimal machine language command which would be generated by ALTRAN. The binary representation of this command is then shown, followed by the hexadecimal equivalent. It is assumed that the command is to be stored in location 0000 and that line 00 is clear or empty when the command is translated by the ALTRAN system.



Notice that P, N, C, and D are made up by the ALTRAN system operating on the ADD command.

In the preceding example of formats the path taken by the various components between forms is shown. The ALTRAN command which was entered was:

ADD 1417-

meaning that the contents of location 1417 (line 14, word 17) are to be added to the contents of the accumulator. The minus (-) sign attached to the address field meant that a Breakpoint was to be included in the translated command. The system would translate this to the following machine language command:

P TT NN C SS DD  
w.17.18.1.14.29-

P Since it was stated that the command was to be stored in location 0000 the operation of adding the contents of word 17 must be deferred until word 17 is read. (Otherwise words 01, 02, 03, 04, etc. would be added to the accumulator.) Any operation in the computer is delayed until the word time specified by TT if the prefix (P) is made equal to w. The prefix (P) of w is equivalent to a 1-bit in the most significant bit position of the command. Thus ALTRAN would generate a command which would be "deferred" until word time 17. At word time 17 the operation would take place.

TT

Under conditions where the operation specified can begin immediately, ALTRAN would make up a prefix of u signaling the computer that the operation must start immediately after the command is read and continue until time TT. A command with a prefix of u is often called an "immediate" or "block" command.

NN

In the example shown, the operation of adding the contents of location 1417 to the accumulator takes place at word time 17. Since line 00 was clear before entry of this command, location 18 is available for storage of the next command. Therefore, NN is assigned as location 18. This is the best choice of location for the next command from the information given in the example. (By "best" we mean that the program appears to be optimum in that no time is wasted waiting for the next command to be read after the execution of the current command.)

SS

The source of information is line 14; therefore, in the SS position of the command, ALTRAN will place the number 14. It may now be seen that in this case (and many others) the address of the operand is specified by the combination of SS (the source line) and TT (the word time at which information will be used from the source line).

DD

In the G-15, information is usually transferred from one line to another. The DD portion of the command is the line number to which information is transferred. (There are some exceptions to this rule in the special and input/output commands.) The accumulator is normally considered to be line 28. If the programmer wishes to add to the accumulator he specifies the destination as 29 rather than 28 to signify that the information is to be added to AR (accumulator) and does not replace the information in AR. In the example, then, the system recognizes that the operation is an "add to AR" and makes the DD part of the command equal to 29.

C

ALTRAN recognizes that in the addition process the number to be added to the accumulator may be negative or positive. In the case where a negative number is to be added to the accumulator that number must be complemented. It therefore supplies the C portion of the command with the characteristic of 1 which will cause negative numbers to be complemented before the addition process.

BP

The addition of a breakpoint (BP) was signaled by the addition of a minus (-) sign on the ALTRAN command entered. The system will include a bit in the proper place if the command is meant to halt on a breakpoint. A command with a breakpoint will cause the computer to halt immediately after execution of this command if the Compute switch is in the BP position.

s/d

Below the binary representation of the command is the number 91925xx. This is the hexadecimal equivalent of the binary command. If the contents of a location holding a command are typed out by the computer as a hexadecimal number they would appear as a 7-digit hexadecimal number and sign. The sign position holds a bit which tells the computer whether the operation to be performed is a single or double-precision function. In general, a single-precision operation involves a single word whereas a double-precision operation usually involves two words which are processed as one long word.

Most of the operation in ALTRAN will normally be single-precision. Double-precision operations may be performed, if desired.

A more detailed explanation of the components of the command may be found in Coding Manual for the Bendix G-15 (T 3-3) or Programming for the G-15. Both manuals are available from Bendix Computer Division.



## NUMBER FORMATS

In the G-15 computer numbers are normally held in binary as either fractions or integers. Most engineering computations are done with binary fractions while data processing and counting operations are usually computed using the binary integer representation.

### Fractional Notation

Binary numbers which are considered to be fractional are usually stored so that the most significant bit position of the number represents  $1/2$ , the next position  $1/4$ , the next  $1/8$  and so forth. The chief disadvantage of this system is that many decimal fractions cannot be represented exactly in this notation. That is to say, the number can be represented very closely with the 28 bits available but an infinite number of bits would be required to represent the fraction exactly.

Some fractions such as  $1/2$ ,  $1/4$ , etc. can be represented exactly using this notation. This notation is often used for engineering work because the numbers used do not have to be more precise than approximately 7 decimal digits. (Single-precision operations)

Consider the problem where an engineer has computed the length of each leg of a course enclosing a piece of land. Suppose he now wants to add up the length of all the legs to find the perimeter of the land. If there were no more than ten legs and each leg was no longer than 900.000 feet then he might enter the lengths into the computer as follows:

<u>Length (ft.)</u>	<u>Length Entered into Computer (ft.)</u>
900.000	.0900000
545.750	.0545750
89.125	.0089125
125.000	.0125000
600.500	.0600500

Notice that the actual values he computed were multiplied by  $10^{-4}$  before they were entered into the computer. This was necessary so they would be fractional. They were multiplied by  $10^{-4}$  rather than  $10^{-3}$  so that the sum of all the lengths would not exceed the capacity of the word. (.9999999) Notice that with the numbers multiplied by  $10^{-4}$  before entry, the sum will never exceed .0900000 times 10 lengths or .9000000.

Because all of the legs were multiplied by  $10^{-4}$  before entry the answer will have to be multiplied by  $10^4$  to get the correct answer. The computer would come up with an answer of .2260375 as the sum of the number entered into the problem. The answer desired by the engineer is 2260.375. He can get this either by mentally multiplying his answer by  $10^4$  or by making up a "format" which will cause the typewriter to type the decimal point in the desired place. By moving the decimal point from the left end of the number right four places the programmer can "multiply" his answer by  $10^4$ .

The entire process described above is normally called scaling. Scaling is a necessary part of programming any fixed point computer. Bendix supplies programming systems which will automatically take care of the scaling for the programmer but they do so at some cost in operating speed. If the problem being programmed can be scaled easily (the ranges of intermediate and final results are well known) and the program is to be run often it will probably pay to program in ALTRAN rather than a system such as INTERCOM 1000. The final program will normally run considerably faster in machine language (ALTRAN) than in an interpretive system.

The object of "scaling" is to keep as much significance in the number as possible and yet not cause an overflow. Thus, numbers should normally be held as far to the left (more significant end) as possible. Notice that in the example shown above the input number could have been scaled or multiplied by  $10^{-3}$  on entry but that an overflow would have resulted when the addition took place, causing an erroneous answer.

It is important to realize that when numbers are held in fractional form they may not represent the number exactly. For example if the fraction .0000001 is entered into the computer and repeatedly added to the accumulator, it will be found that after 1000 additions to the accumulator the answer will not be .0001000 as you might expect. Again, this is because the fraction .0000001 cannot be represented exactly as a binary fraction.

Multiplication in the G-15 is normally considered to operate on fractions. For instance, if the multiplicand is loaded into register ID.1 (LID command) and the multiplier is loaded into MQ.1 (IMQ command) and both numbers are the binary fractions equivalent to the decimal fraction .5000000, then the product of .2500000 will be taken from the product register PN.1 (using a TPR command).

The two-word registers mentioned above are described more fully in the manual, Programming for the G-15.

## Integer Notation

In some applications, such as data processing, the fact that a binary fraction cannot represent a decimal fraction exactly is a decided disadvantage. Perhaps the best example of this is in the case where many sales of one or two dollars each are to be added together to get a Sales Total. If the individual items are entered as fractions such as:

<u>Amount</u>	<u>Amount Entered into the Computer</u>
\$ 1.23	.0000123
2.45	.0000245
1.98	.0000198

eventually the total of all the amounts will be "off" by a few cents or dollars. Normally this cannot be tolerated in a data processing application so the amounts are held internally as integers.

A binary integer is normally held in the computer so that the least significant bit in the word represents one unit. The next most significant bit represents two and so forth. Notice that the amount \$1.23 can then be entered as 123 units and thus represented exactly in the computer. When the final total is typed out by the computer the "format" controlling the type-out will supply the decimal point where necessary.

Normally amounts of money, counts of items, etc. are processed as binary integers. Counters used in the program for such things as command modification and timing are normally held as integers. ALTRAN has a command (DAR) which is most useful for decrementing a binary integer in the AR by one in the least significant place. Thus, counters can be modified easily if they are kept as integers.

Multiplication of the number of units times unit cost can be accomplished in the computer using the following multiplication scheme. The registers are cleared. The register ID.1 is loaded with the cost. The ID register is shifted right one bit position. The register MQ.1 is loaded with the number of units. The multiply command is given followed by a command to take the product from PN.0. The ALTRAN sequence would be:

```
CLR
LID cost      (where cost represents the location holding
               the value of the cost as an integer.)

SFT 02
LMQ units    (where units represents a location.)
MPY
LAR 2600     (Load AR from PN.0)
```

## CHAPTER 3

### BASIC ALTRAN INFORMATION

This chapter deals with the basic methods used in the ALTRAN system for command translation and other operations. It is meant to give the reader some familiarity with the way in which the system functions. The subjects covered include command and instruction input, translation, information flow, reservation of storage with rules for type-out and the check-summing feature in ALTRAN.

The Opcode and Skeleton command tables are included at the end of this chapter.

## COMMAND AND INSTRUCTION INPUT

When the 3-letter command code or 1-letter instruction code is typed in by the operator and either a "space" or "tab" follows, the system immediately begins to search through a "Valid Opcode Table". This table is located in line 16 of the memory and holds an ordered list of all the valid command or instruction codes. (The search is called a binary search and takes place normally while the operator is entering the address portion of the command or instruction.) When the searching process finds a "match" between the code typed in and a code in the Valid Opcode List, the word position in which the matching code was found is noted. (The Valid Opcode List and the corresponding skeleton commands appear at the end of this chapter.)

This word number is used to pick up the corresponding word from line 17. Line 17 holds a "Skeleton Command Table". There is one skeleton command stored in line 17 for each valid opcode in line 16. The skeleton command contains information necessary for the translation (for commands) or execution (for instructions) of the code typed into the computer.

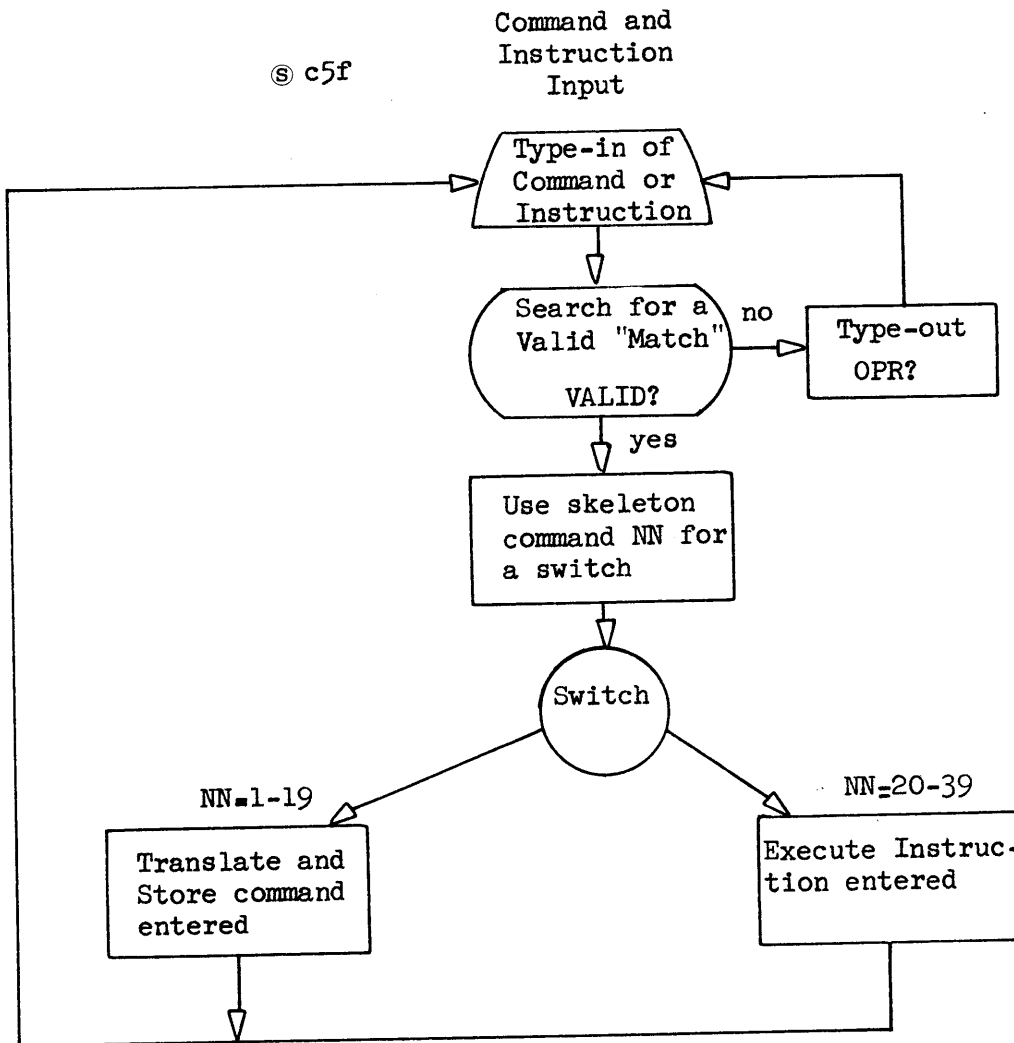
Both the Valid Opcode Table and the Skeleton Command Table are 108 words in length permitting a maximum of 108 possible codes. (Notice that not all of the available locations are presently used by the system. It is expected that users will add codes which are of particular use to themselves.)

## TRANSLATION

Translation and decoding begins immediately after the skeleton command has been located for the code typed in and the address information is known. The NN portion of the skeleton command is extracted from it and used to make a transfer of control command. This transfer of control command is executed and the system branches to the required place. In the case of a command code the translation of the command is accomplished by 16 possible routines. The NN of the skeleton command controls which of the 16 routines will be used for the translation of this particular command code. Thus for command translation, 16 "classes" of commands exist.

Each of the translation routines is designed to use the other information in the skeleton command and the address field entered to make up the correct machine language command. It is, in some cases, a simple matter to add codes to the valid opcode table to increase the usefulness of the system. If the skeleton command can be made to conform to the rules for one of the existing classes, no additional programming is necessary.

INFORMATION FLOW IN ALTRAN



The flow diagram represents, in general, the manner in which both Commands and Instructions are entered, decoded and either executed or translated and stored. Notice that the operator can always return to Command and Instruction Input by typing © c5f with the Compute switch OFF and the Enable switch ON. If the operator then moves the Compute switch to GO the program will return to Command and Instruction Input.

## RESERVATION OF STORAGE

In ALTRAN, locations which are reserved are marked either by the operator or by ALTRAN itself by storing the hexadecimal number xxxxxxxx in that location. If the system finds that the next available location for a command is location 0018, it will store 18 in NN of the command being translated and then mark location 0018 with the number xxxxxxxx. (Some commands such as mark place and transfer would store the next location number 18 in the TT portion of the command.)

Sometimes ALTRAN will reserve several locations in the course of translating one command. In the case of a Test command it will mark locations NN and NN plus 1 so both sides of the test will be reserved. In the case where storage locations are left to ALTRAN for assignment, they will also be marked with xxxxxxxx. (For example, the command LAR 00zz will find the next free location after the command in line 00 and reserve it.)

It is not practical to type out the xxxxxxxx as a command each time ALTRAN finds a reserved location. To avoid having xxxxxxxx converted to the equivalent machine language command (w.93.93.3.14.29 or SUB 1493 in some cases), the system tests for the presence of the hexadecimal number xxxxxxxx and behaves in the following manner:

<u>Instruction</u>	<u>xxxxxxx Found</u>	<u>000000 Found</u>
M ABCD	No Type-out	No type-out
M ABCD-	Go to Command and Instruction Input	Type as Machine Language Command
D ABCD	No type-out	No type-out
D ABCD-	Type Address Only	No type-out
C ABCD	No type-out	No type-out

In the case where locations are typed with instructions H, F, or I, the contents of the location will be treated as a number regardless of whether it is xxxxxxxx. Zero locations will not be typed out using these instructions.

Refer to Chapter 4 on Instructions for a full explanation of the operation of the codes mentioned above.

## CHECK-SUM FEATURE

The system has been programmed so that the operator can type Enable@c5f to gain control and go to Command and Instruction Input. Word 00 of any line normally executed from line 05 as a part of the ALTRAN system will transfer the correct starting line to line 05 to give the operator control of the program.

When the starting line is in line 05, one of the first operations performed is the check-summing of all the fixed storage associated with the ALTRAN system program. (Variable information such as switches is stored in a block in the high order end of line 18 and of course is not check-summed.) This feature is quite useful in that it provides the operator with a means for easily checking the validity of the entire system program without unnecessarily reloading the program from paper tape. If the system does not check-sum the program will ring the bell 5 times and then halt. If the Compute switch is moved to the center position and then back to GO, the system will again be check-summed. If the check-sum is incorrect the operator should reload the ALTRAN program from paper tape.



OPCODE & SKELETON COMMAND TABLE

ADDR	OP	HEX	ADDR	SKEL. CMD.
.1600	ADD	.x1x4x40	.1700	u.00.01.1.00.29
.1601	ADM	.x1x4y40	.1701	u.00.01.2.00.29
.1602	APN	.x1y7y50	.1702	u.w0.01.0.00.30
.1603	ARL	.x1y9y30	.1703	u.00.04.2.28.29
.1604	BDF	.x2x4x60	.1704	w.35.07.2.21.31
.1605	BDI	.x2x4x90	.1705	w.37.07.2.21.31
.1606	BEL	.x2x5y30	.1706	u.01.06.0.17.31
.1607	BLA	.x2y3x10	.1707	u.00.16.1.00.29
.1608	BPT	.x2y7z30	.1708	u.02.06.0.06.31
.1609	CAM	.x3x1y40	.1709	u.00.01.2.00.28
.1610	CAR	.x3x1y90	.1710	w.50.07.2.21.31
.1611	CLA	.x3y3x10	.1711	u.00.01.1.00.28
.1612	CLR	.x3y3y90	.1712	u.03.06.0.23.31
.1613	CLS	.x3y3z20	.1713	u.00.01.3.00.28
.1614	DAR	.x4x1y90	.1714	u.00.03.7.28.28
.1615	DBF	.x4x2x60	.1715	w.22.07.2.21.31
.1616	DBI	.x4x2x90	.1716	w.23.07.2.21.31
.1617	DEL	.x4x5y30	.1717	u.01.06.0.01.01
.1618	DIV	.x4x9z50	.1718	u.57.05.1.25.31
.1619	HLT	.x8y3z30	.1719	u.02.06.0.16.31
.1620	LAR	.y3x1y90	.1720	u.00.01.0.00.28
.1621	LID	.y3x9x40	.1721	u.w0.01.0.00.25
.1622	LMQ	.y3y4y80	.1722	u.w0.01.0.00.24
.1623	LPN	.y3y7y50	.1723	u.w0.01.0.00.26



OPCODE & SKELETON COMMAND TABLE

ADDR	OP	HEX	ADDR	SKEL. CMD.
.1624	LTL	.Y3z3Y30	.1724	u.00.02.0.00.00
.1625	MPT	.Y4Y7z30	.1725	w.00.12.0.21.31
.1626	MPY	.Y4Y7z80	.1726	u.56.05.0.24.31
.1627	NOP	.Y5Y6Y70	.1727	u.00.15.0.01.01
.1628	NOR	.Y5Y6Y90	.1728	u.54.05.0.27.31
.1629	OAR	.Y6x1Y90	.1729	u.02.11.0.31.31
.1630	PAA	.Y7x1x10	.1730	u.05.06.4.08.31
.1631	PAN	.Y7x1Y50	.1731	u.05.06.4.09.31
.1632	PFC	.Y7x6x30	.1732	w.14.07.2.21.31
.1633	PFR	.Y7x6Y90	.1733	w.34.07.2.21.31
.1634	PFT	.Y7x6z30	.1734	w.76.07.2.21.31
.1635	PHC	.Y7x8x30	.1735	w.13.07.2.21.31
.1636	PHE	.Y7x8x50	.1736	w.68.07.2.21.31
.1637	PHT	.Y7x8z30	.1737	w.41.07.2.21.31
.1638	PIC	.Y7x9x30	.1738	w.79.07.2.21.31
.1639	PIN	.Y7x9Y50	.1739	w.24.07.2.21.31
.1640	PIT	.Y7x9z30	.1740	w.64.07.2.21.31
.1641	PNA	.Y7Y5x10	.1741	u.02.06.0.08.31
.1642	PNN	.Y7Y5Y50	.1742	u.02.06.0.09.31
.1643	PRE	.Y7Y9x50	.1743	w.u3.07.2.21.31
.1644	PUN	.Y7z4Y50	.1744	w.15.07.2.21.31
.1645	RMP	.Y9Y4Y70	.1745	u.00.13.0.20.31
.1646	RMT	.Y9Y4z30	.1746	u.02.06.0.13.31
.1647	RPT	.Y9Y7z30	.1747	u.02.06.0.15.31

OPCODE & SKELETON COMMAND TABLE

ADDR	OP	HEX	ADDR	SKEL. CMD.
.1648	SAR	.z2x1y90	.1748	u.00.01.0.28.00
.1649	SFO	.z2x6y60	.1749	u.16.06.0.05.31
.1650	SFT	.z2x6z30	.1750	u.08.05.1.26.31
.1651	SID	.z2x9x40	.1751	u.00.01.0.25.00
.1652	SMA	.z2y4x10	.1752	u.00.01.2.28.00
.1653	SMQ	.z2y4y80	.1753	u.00.01.0.24.00
.1654	SNZ	.z2y5z90	.1754	u.00.01.3.29.00
.1655	SPN	.z2y7y50	.1755	u.00.01.0.26.00
.1656	SRE	.z2y9x50	.1756	u.16.06.0.04.31
.1657	SRY	.z2y9z80	.1757	u.02.06.0.00.31
.1658	SSD	.z2z2x40	.1758	u.00.01.1.28.00
.1659	SUB	.z2z4x20	.1759	u.00.01.3.00.29
.1660	SZE	.z2z9x50	.1760	u.00.01.0.29.00
.1661	TAB	.z3x1x20	.1761	w.30.07.2.21.31
.1662	TIA	.z3x9x10	.1762	u.05.06.4.12.31
.1663	TIF	.z3x9x60	.1763	w.56.07.2.21.31
.1664	TIH	.z3x9x80	.1764	w.82.07.2.21.31
.1665	TII	.z3x9x90	.1765	w.81.07.2.21.31
.1666	TIN	.z3x9y50	.1766	u.02.06.0.12.31
.1667	TNE	.z3y5x50	.1767	u.02.09.0.22.31
.1668	TNZ	.z3y5z90	.1768	u.w1.01.0.00.27
.1669	TOF	.z3y6x60	.1769	u.02.09.0.29.31
.1670	TPR	.z3y7y90	.1770	u.01.08.0.26.28
.1671	TPS	.z3y7z20	.1771	u.02.09.1.17.31

OPCODE & SKELETON COMMAND TABLE

ADDR	OP	HEX	ADDR	SKELETON CMD.	
.1672	TQU	.z3y8z40	.1772	u.00.08.0.24.28	
.1673	TRA	.z3y9x10	.1773	u.00.10.0.21.31	
.1674	TRY	.z3y9z80	.1774	u.02.09.0.28.31	
.1675	TVA	.z3z5x10	.1775	u.00.14.2.00.00	
.1676	WFC	.z6x6x30	.1776	u.05.06.0.30.31	
.1677	WMT	.z6y4z30	.1777	w.00.06.0.01.31	
.1678	WRY	.z6y9z80	.1778	w.31.07.2.21.31	
.1679	ZAA	.z9x1x10	 1679 ---- 1693 and 1779 ---- 1793 are used only to fill table.		
.1680	ZAB	.z9x1x20			
.1681	ZAC	.z9x1x30			
.1682	ZAD	.z9x1x40			
.1683	ZAE	.z9x1x50			
.1684	ZAF	.z9x1x60			
.1685	ZAG	.z9x1x70			
.1686	ZAH	.z9x1x80			
.1687	ZAI	.z9x1x90			
.1688	ZAJ	.z9x1y10			
.1689	ZAK	.z9x1y20			
.1690	ZAL	.z9x1y30			
.1691	ZAM	.z9x1y40			
.1692	ZAN	.z9x1y50			
.1693	ZAO	.z9x1y60			
.1694	B	.zxzxx20		.1794	u.00.31.0.00.00
.1695	C	.zxzxx30		.1795	u.00.22.0.00.00

OPCODE & SKELETON COMMAND TABLE

ADDR	OP	HEX	ADDR	SKEL. CMD.
.1696	D	.zxzxx40	.1796	u.00.33.0.00.00
.1697	F	.zxzxx60	.1797	u.00.24.0.00.00
.1698	H	.zxzxx80	.1798	u.00.23.0.00.00
.1699	I	.zxzxx90	.1799	u.00.25.0.00.00
.16u0	L	.zxzxy30	.17u0	u.00.32.0.00.00
.16u1	M	.zxzxy40	.17u1	u.00.30.0.00.00
.16u2	P	.zxzxy70	.17u2	u.00.26.0.00.00
.16u3	R	.zxzxy90	.17u3	u.00.29.0.00.00
.16u4	S	.zxzxx20	.17u4	u.00.28.0.00.00
.16u5	T	.zxzxx30	.17u5	u.00.27.0.00.00
.16u6	X	.zxzxx70	.17u6	u.00.21.0.00.00
.16u7	Z	.zxzxx90	.17u7	u.00.20.0.00.00

## CHAPTER 4

### INSTRUCTIONS

This chapter presents a detailed description of the Instructions which the operator or programmer uses to control ALTRAN.

A summary of the Instructions appears on page 26 of this chapter.

## INSTRUCTIONS

Instructions are used by the operator to send a message to the ALTRAN system. They cause ALTRAN to proceed through certain steps such as clearing a line of memory, permitting the operator to enter a series of numbers or documenting a program already in memory. In a number of cases ALTRAN will return after the execution of the instruction to permit the operator to enter still other instructions.

The reader will find that ALTRAN instructions are in many ways similar to instructions given in the PPR system. That is, provision is made to assist the operator in the preparation, documentation, debugging and recording of his program.

Instructions can only be entered into the computer when the ALTRAN system is at a point in its program called Command and Instruction Input. ALTRAN will be at this point after any one of the following conditions has taken place.

1. The ALTRAN system has just been read into the computer from paper tape.
2. The operator has moved the Compute switch to center, typed sc5f and returned the Compute switch to GO. (The underlined sc5f means that the Enable switch was ON when sc5f was typed.)
3. The operator has been entering ALTRAN commands. After the execution of the instruction to permit entry of ALTRAN commands, the system returns to Command and Instruction Input. After each command is entered the system returns to this same location.
4. The operator has been entering machine language commands and has typed a "hollow point".
5. Instruction C, Z, X, P, T, R, L or B has been executed.

In the examples describing the use of instructions, information which is typed by the operator will be shown enclosed in a box. Information not shown enclosed may be assumed to have been typed by the computer.

Notice that all instructions are entered using the same format. The examples do not show "spaces" and "tabs" typed by the operator. The format follows:

Instruction letter, "space", address, "tab"

Errors in instruction entry can be corrected by typing a "hollow point" if the error is discovered before typing the "tab". Control is transferred to Command and Instruction Input and the correct instruction may then be typed. If the "tab" key was typed before the error was discovered the instruction may be executed and the operator must take appropriate corrective action.

If the error consisted of an incorrect address or the substitution of one instruction code for another, the instruction will be executed. If the instruction code is illegal (not a letter in the list of valid instructions) the illegal code will be typed out followed by a "?" signifying to the operator that the code was questioned and that the instruction was not executed. The operator would then enter the correct instruction since control is returned to Command and Instruction Input after typing out the "?".

It is important to note that all of the opcodes for instructions and commands must be typed as lower case letters. Failure to type opcodes as lower case will result in the rejection of that particular code by the system.



## ALTRAN INSTRUCTIONS

Z ABCD	ZERO (CLEAR) LOCATION ABCD. IF CD MADE EQUAL TO ZZ, THEN ZERO ENTIRE LINE AB.
X ABCD	FILL LOCATION ABCD WITH XXXXXXXX. IF CD MADE EQUAL TO ZZ, THEN FILL ENTIRE LINE AB.
C ABCD	ENTER ALTRAN COMMANDS STARTING WITH LOCATION ABCD.
H ABCD H ABCD-	ENTER HEXADECIMAL NUMBERS STARTING WITH LOCATION ABCD. LIST HEXADECIMAL NUMBERS STARTING WITH LOCATION ABCD.
F ABCD F ABCD-	ENTER FRACTIONS STARTING WITH LOCATION ABCD. LIST FRACTIONS STARTING WITH LOCATION ABCD.
I ABCD I ABCD-	ENTER INTEGERS STARTING WITH LOCATION ABCD. LIST INTEGERS STARTING WITH LOCATION ABCD.
P ABCD	PUNCH PAPER TAPE FROM LOCATIONS ABOO THROUGH ABCD-1.
T ABCD	TRANSFER ALL OF LINE AB TO LINE CD.
S ABCD	START COMPUTING AT LOCATION ABCD.
R ABCD	READ PAPER TAPE INTO LOCATIONS ABOO THROUGH ABCD-1.
M ABCD M ABCD-	ENTER MACHINE LANGUAGE COMMANDS STARTING WITH ABCD. LIST MACHINE LANGUAGE COMMANDS STARTING WITH ABCD.
D ABCD D ABCD-	ENTER MACHINE LANGUAGE COMMANDS SEQUENTIALLY STARTING AT LOCATION ABCD. DOCUMENT MACHINE LANGUAGE COMMANDS SEQUENTIALLY STARTING AT LOCATION ABCD.
L ABCD	LIST ALL EMPTY (CLEAR) LOCATIONS IN LINE AB STARTING WITH LOCATION ABCD.
B ABCD B ABCD-	ADD A BREAKPOINT TO THE COMMAND IN LOCATION ABCD. REMOVE A BREAKPOINT FROM THE COMMAND IN LOCATION ABCD.

## INSTRUCTIONS

Z ABCD        Zero location ABCD. If CD=ZZ, zero line AB.

The location ABCD will be cleared to zero where ABCD is one of the long line locations available to the operator while ALTRAN is in the computer. If the CD portion of ABCD is made equal to ZZ, the entire long line AB will be cleared to zero. Since ALTRAN searches for clear or zero locations during data or command assignment, it is necessary to clear those locations which the operator wishes to release to the system for assignment by the system.

Example: Assume that all of line 00 is to be made available to ALTRAN for command or data assignment. This can be accomplished by the instruction:

Z 00ZZ

which clears or zeros all of line 00 (0000-00u7).

Example: To clear location 0625 type the instruction:

Z 0625

X ABCD      Fill location ABCD with XXXXXXXX. If CD=ZZ, fill line AB.

The location ABCD will be filled with the hexadecimal number XXXXXXXX where AB is one of the long lines available to the operator while ALTRAN is in the computer. If the CD portion of ABCD is made equal to ZZ, the entire line AB will be filled with XXXXXXXX. Since ALTRAN searches for clear or zero locations during data or command assignment, it is possible to reserve location, making them unavailable to the ALTRAN system, by use of the instruction X. (Any non-zero information would serve the same purpose as the number XXXXXXXX but since this can be easily recognized by the operator as a reserved location signal it is used in ALTRAN.)

Example: Assume that locations 0199, 01u0 and 0406 are to be reserved for later use. The following instructions would be used:

X 0199
X 01u0
X 0406

Locations may be reserved in any order desired.

C ABCD

Enter Commands starting with location ABCD.

Permit entry of ALTRAN Commands starting with location ABCD. The computer will respond by causing the type-writer to carriage return twice and print; .ABCD (where ABCD is some long line location available to the operator while ALTRAN is in the computer). The computer then waits for the operator to type in the command. If location ABCD held a non-zero value, other than a hexadecimal XXXXXX, it will be typed out in decimal machine language form prior to command entry.

Example: Assume that commands are to be stored beginning with location 0033 and that this location presently holds a zero (000000). The operator types:

C 0033

The computer responds with:

.0033

Example: Assume that commands are to be stored beginning with location 0000 and that location 0000 presently holds the machine language command w.35.36.0.06.28. The operator would type:

C 0000

The computer responds with:

.0000 w.35.36.0.06.28

Even after the operator has typed in, as in the previous examples, he may change his mind and then enter any other instruction rather than the command normally entered at this time.

Example: If, in the previous example, the operator decided that he wanted to start entering commands at 0050 rather than 0000, the sequence would have been:

C 0000

.0000 w.35.36.0.06.28

C 0050

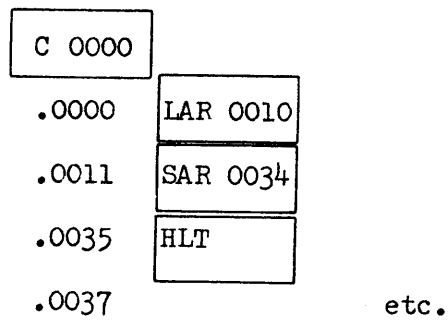
.0050

C ABCD (continued)

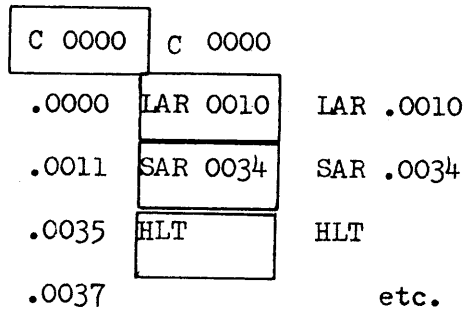
Since the input section is common for both commands and instructions it is possible to enter a command when no location has been specified. If this is done, the command will be stored in the location held in the location counter L. L is set to 0000 when ALTRAN is loaded. L is kept in location 1896 by the ALTRAN system.

The following examples demonstrate the use of the Verify and Command Type-out switches.

Example: Assume that line 00 is clear and the Verify and Command Type-Out switches are both off (non-zero). The following sequence illustrates that no verification or command type-out occurs.



Example: The same example as above with the Verify switch on (zero) would be:



C ABCD (continued)

Example: The same example as above with both the Verify and Command Type-out switches on (zero) would be:

C 0000	C 0000		
.0000	LAR 0010	LAR 0010	w.10.11.0.00.28
.0011	SAR 0034	SAR 0034	w.34.35.0.28.00
.0035	HLT	HLT	u.37.37.0.16.31
.0037			etc.

Either one or the other or both of the Verify or Command Type-out switches may be on (zero) or off (non-zero).

H ABCD     Enter Hexadecimal numbers starting with location ABCD.

H ABCD-    List Hexadecimal numbers starting with location ABCD.

If no sign is attached to the ABCD field, it is assumed to be positive which is interpreted to mean that Hexadecimal numbers are to be entered. These numbers are right justified so that leading zeros need not be typed.

Example: Assume that line 04 is clear and the following numbers are to be stored:

0415    0000002

0416    1234567-

0417    Z0Z0Z0Z-

The operator would type the following information to enter these numbers:

```
H 0415
.0415 0000002 ⑤ (or 2 ⑤)
.0416 1234567- ⑤
.0417 Z0Z0Z0Z- ⑤
.0418
```

An error in input can be corrected as shown below for the same example.

Example: Same as previous example only with errors and corrections.

```
H 0415
.0415 3 0000002 ⑤ (hit "tab" then type correct 7 digits)
.0416 124 1234567- ⑤
.0417 Z0Z0Z0- Z0Z0Z0Z- ⑤
```

H ABCD

H ABCD- (continued)

Example: Same numbers desired as in above example. This example demonstrates error correction after the "tab" Ⓢ has been typed.

```
H 0415
.0415 2 Ⓢ
.0416 1234456- Ⓢ
.0417 Ⓢc5f (Take Compute switch OFF, put Enable switch ON
H 0416 while typingⓈc5f. Compute switch to GO.)
.0416 1234456 1234567 Ⓢ (note previous contents typed.)
.0417 ZOZOZOZ- Ⓢ
.0418
```

The previous example showed that the contents of a location will be typed if it is non-zero. (Location 0416 held the hexadecimal number 1234456 after the error was made. The number was typed out when the operator requested hexadecimal input to 0416.)

When a negative sign is attached to the ABCD field the instruction H is useful for listing the contents of a series of sequential locations.

Example: Assume that the following locations hold the numbers:

```
0415 0000002
0416 1234567-
0417 ZOZOZOZ-
0418 0000000
0419 ZZZ0000
```



H ABCD

H ABCD (continued)

Confirmation of the previous contents can be obtained by the use of:

H 0415-
---------

.0415 .0000002

.0416 -.1234567

.0417 -.Z0Z0Z0Z

.0419 .ZZZ0000

Listing of non-zero locations will continue to u7 of the current line unless stopped by the operator. Listing is terminated by moving the Compute switch to center, typing Ⓢc5f, and then, Compute to GO.

F ABCD Enter Fractions starting with location ABCD.

F ABCD- List Fractions starting with location ABCD.

This instruction is operationally the same as instruction H with the exception that it will convert as follows:

Input--Decimal fractions converted to binary.

Output--Binary fractions converted to decimal.

Example: Store the following decimal fractions as binary fractions in memory:

0078 .0000100

0079 .5000000

0080 -.9999900

The operator would type the following (assume line 00 clear):

F 0078

.0078 0000100 Ⓢ (or simply 100 Ⓢ )

.0079 5000000 Ⓢ

.0080 9999900- Ⓢ

.0081

When a negative sign is attached to the ABCD field, the contents of locations starting with ABCD are converted to decimal fractions and typed out.

Example: List the contents of locations starting with 0078 after execution of the previous example.

F 0078-

.0078 .0000100

.0079 .5000000

.0080 -.9999900

.0081 etc.

See instruction H for details of the operation of error correction, stopping listing and so forth.

- I ABCD     Enter Integers starting with location ABCD.
- I ABCD-    List Integers starting with location ABCD.

This instruction is operationally the same as instruction H with the exception that it will convert as follows:

Input--Decimal integers to binary.

Output--Binary integers to decimal.

Example: Store the following decimal integers as binary integers in memory:

```
0445    0001234
0446    0000001
0447    0000128
0448    0000044
```

The operator would type the following (assume that line 04 is clear):

```
I 0445
.0445  0001234  S  (or simply 1234  S )
.0446  0000001  S  ( or simply 1    S )
.0447  128      S
.0448  44       S
```

When a negative sign is attached to the ABCD field, the contents of locations starting with ABCD are converted to decimal integers and typed out.

Example: List the contents of locations starting with 0445 after execution of the previous example.

```
I 0445-
.0445    .0001234
.0446    .0000001
.0447    .0000128
.0448    .0000044
```

See instruction H for details of the operation of error correction, stopping listing and so forth.

P ABCD      Punch paper tape from locations ABOO thru ABCD-1.

Line 19 is cleared and information transferred to it from line AB words 00 through CD-1. If words 00 through 03 are clear, a minus zero (0000000-) is stored in location 1900. (This ensures that a line punched out with the P instruction can be read back into the original locations.) A standard line 19 format is stored in locations 0200 through 0203 and line 19 is precessed toward word u7. When non-zero information is stored in locations 19u4 through 19u7, line 19 is punched onto paper tape.

If the entire line AB is to be punched, then the CD portion of the instruction is made equal to 00.

Example: Punch all of line 00 onto paper tape.

P 0000
--------

Example: Punch locations 1800 through 1899 onto paper tape.

P 18u0
--------

Do not attempt to punch line 19 onto tape. It is cleared by this instruction.

T ABCD          Transfer all of line AB to line CD.

The contents of locations ABOO through ABu7 are transferred to line CD, replacing the previous contents of line CD. The contents of line AB remain unchanged by the execution of this instruction.

Example: Transfer the contents of line 06 to line 00.

T 0600
--------

Line CD of course should be memory which is available to the operator while ALTRAN is stored in the computer.

This instruction will normally be used to position data and commands prepared using the ALTRAN system.

S ABCD     Start computing at location ABCD.

The ALTRAN line 02 input/output subroutine will have a "return to marked place in line 05" command stored in it during the operation of ALTRAN. When the operator desires to begin computing automatically using his own program, this instruction is usually given. It stores a "return to marked place in line 00" command in line 02 so the input/output subroutine will function normally for his program. This assumes that his program will use the input/output commands which require subroutines and that these commands are located in line 00. If the input/output subroutines are entered from other lines (other than line 00) the operator will have to make provision for storing his own "return" command in 0244 before entry is made.

Example: Assume that the operator wishes to start computing at location 0000. He would type:

S 0000
--------

The operator must of course choose a "command line" from which to execute commands. Lines 00, 01, 03, 04 in ALTRAN Commands may not be executed from line 06.

Notice that "overflow" is reset before command is transferred to the location specified.

R ABCD Read paper tape and store in locations ABOO through ABCD-1

Lines 23 and 19 are cleared and the information is read from paper tape through line 23 to line 19. When "Ready" is set (stop code sensed on tape), line 19 is transferred to location ABOO through ABCD-1. If all of line 19 is to be transferred to line AB, then CD is made equal to 00.

Example: Read punched paper tape into line 19 and transfer all of line 19 to line 04.

R 0400

Example: Read punched paper tape into line 19 and transfer words 00 through 99 into line 06.

R 06u0

The line into which the information is transferred should be one which is available to the operator while ALTRAN is in the computer memory. It should not be a line occupied by ALTRAN.

M ABCD Enter Machine language commands starting with location ABCD.

M ABCD- List Machine language commands starting with location ABCD.

This instruction is used when the operator wishes to enter or list machine language commands.

The commands entered or listed have the form:

P.TT.NN.C.SS.DD (no breakpoint)

or

P.TT.NN.C.SS.DD- (breakpoint)

The various functions of the parts of a machine language command are given in Chapter 2 of this manual, in "Coding Manual for the G-15" (T 3-2), and in the "Programming for the G-15" (Jan. 1960) both by Bendix Computer.

ALTRAN will not change the timing number (TT) typed in by the operator to some other value. Each command entered should have a prefix (P) attached to it. If no prefix is entered, ALTRAN will consider the command as if a prefix of "u" had been typed with the command.

When listing commands using the M ABCD- instruction, commands will be listed in their NN number order. In most cases this will be normal order of execution.

Example: Assume that line 00 is clear and the operator wishes to enter the following machine language command into location 0040:

w.44.60.0.06.28

he would type:

M 0040

.0040

w.44.60.0.06.28

.0060 etc.



M ABCD

M ABCD- (continued)

Example: If the operator now wanted to modify the command in location 0040 to be:

w.46.60.1.06.28

he would type:

M 0040

.0040 w.44.60.0.06.28

w.46.60.1.06.28

.0060 etc.

Since location 0040 held the previously entered command, it was typed out by the computer.

When listing machine language commands using the M ABCD-instruction listing will continue unless the operator moves the Compute switch to center and types Enable @c5f and returns the Compute switch to GO. This action can always be used to return to Command and Instruction input in ALTRAN.

D ABCD Enter machine language command sequentially starting at ABCD.

D ABCD- Document machine language commands sequentially starting at location ABCD.

This instruction is very similar to instruction M. It differs only in the choice of the next location to be filled or typed by the computer. Entry or listing of machine language commands will always proceed in sequence without regard to the NN number in the command entered or listed.

L ABCD List all empty locations in line AB starting at word CD.

This instruction is normally used to present the operator with a statement of how many and which locations in a line are as yet unused. If a location is equal to zero its number will be typed by the computer.

Example: Assume that line 04 is full with the exception of locations 10, 45, 60, and 99. If the operator was interested in knowing which locations above 0450 were empty (zero) he would type:

L 0450
--------

60 99

Locations reserved by ALTRAN (filled with the hexadecimal number XXXXXXXX) will not be typed out by the L instruction. Therefore locations which are reserved may still be useful for command storage and not be typed out by the L instruction.

B ABCD Add a Breakpoint to the command in location ABCD.

B ABCD- Remove a Breakpoint from the command in location ABCD.

This instruction is used normally during the debugging phase of programming. The addition of a Breakpoint to a command allows the computer to compute at full speed up until it executes the command with the breakpoint. At that point, if the "Compute" switch is in the "BP" (breakpoint) position, computation will stop. Computation may be resumed by placing the "Compute" switch to "GO" or "BP" after moving it to the center position. Computation can also be resumed one command at a time by striking the "i" key on the typewriter keyboard with the "Enable" switch "on".

Thus portions of a program already "debugged" may be executed quite rapidly up until the uncertain part is reached. The programmer may then proceed with caution by "i keying" through the unchecked part of his program. The operator will normally place a breakpoint at the end of portions of his program as they are checked-out, removing the previously added breakpoint.

Example: Add a breakpoint to the command in location 0467.

B 0467
--------

Example: Remove breakpoints from the commands in locations 0054, 0199 and 04u6. The operator would type:

B 0054-
---------

B 0199-
---------

B 04u6-
---------

## CHAPTER 5

### COMMANDS

The detailed description of the Command functions is presented in this chapter.

A summary of the Commands appears on page 51 of this chapter.

## COMMANDS

Commands are typed into the computer by the operator, translated by ALTRAN and then become part of the object or final program desired by the operator. There are two types of commands in ALTRAN:

1. Commands such as ADD, SUB, MPY and DIV which are translated into an optimized machine language command which performs one basic operation.
2. Commands such as TIF which are translated into a transfer and mark command. When the command TIF is executed it will cause the program to mark place and transfer to an optimized subroutine stored in line 02. The subroutine in line 02 is concerned with input/output functions and conversion routines. It contains all of the commonly used input/output functions and is normally used during the operation of ALTRAN and of the object program. If the operator must use line 02 for another function in his object program he should avoid the use of all commands of this type. Commands of this type are marked with an asterisk (\*) as they are discussed.

Commands can only be entered into the computer when the system is at a point in its program called Command and Instruction Input. Normally an instruction C ABCD will be used to set the location counter (L) to the number ABCD. Control is then returned to Command and Instruction Input and the next command entered and translated will be stored in location ABCD. The location counter is adjusted by the ALTRAN System and need not be changed during command entry unless the operator wishes to break a sequence. (He will want to do this to follow both branches of a test, for example.)

In the examples describing the use of commands, information which is typed by the operator will be shown enclosed in a box. Information not shown enclosed may be assumed to have been typed by the computer.

The format for command entry follow:

3 command letters, "space", address, "tab"

LAR "space" 0400 "tab"

3 command letters, "tab" (where no address is required)

TIF "tab"

Errors in command entry can be corrected by typing a "hollow point" if the error is discovered before typing the "tab". Control is transferred to Command and Instruction Input, the location counter is unchanged, and the correct entry may then be typed. If the error is not discovered until the "tab" has been entered the operator will have to take corrective measures to remove the incorrect command, clear any locations reserved by the translator, reset the location counter by using the instruction C ABCD, and then enter the correct command. Examples are shown of this type of error correction.

In the following examples the comments describe the operation performed by the command at the time the object program is run, not at the time it is typed-in.

Command opcodes (such as ADD, SUB, etc.) must be typed in lower case or they will be rejected as invalid. If they are rejected as invalid the operator may then type the correct command opcode following the "?" typed by the system.

When the ALTRAN system is read into the computer from paper tape, 2 switching locations are clear or zero. These switches are the Verify Switch (location 18u4) and the Command Type-out Switch (location 18u2). The Verify Switch, when set to zero, will cause all commands and instructions entered by the operator to be typed back out by the computer. The Command Type-out Switch, when set to zero, will cause the machine language command resulting from the translation to be typed out for the operator's use.

These switches may be set to zero or non-zero by the operator. A convenient method of changing the setting of a switch is by using the Z ABCD (to set the switch to zero) or the X ABCD instruction (to set a switch to non-zero). For instance, the Verify and Command Type-out functions can be stopped by setting both switches to non-zero:

X 18u2        Set the Command Type-out switch to non-zero.

X 18u4        Set the Verify switch to non-zero.

Although not shown on the individual command descriptions, each command entered may have a "Breakpoint" attached if the programmer desires. A Breakpoint allows the operator, at the time the program is run, to place the Compute Switch in the Breakpoint position and run at "full speed" up to the command which has the Breakpoint attached. After reaching this point the operator or programmer can proceed with caution by executing one command at a time in a process called "single-cycling". (Each time the "i" key is typed with the Enable switch ON, the computer executes one command.)

The programmer specifies commands to be Breakpointed by adding a minus (-) sign to the address field of the commands entered.

Notice that the Instruction B can be used to add or remove Breakpoints at the time a program is debugged.

Example: Attach a Breakpoint to the first command in the program. The first command is IAR 0625.

```
.0000    IAR 0625-
```

With ALTRAN, automatic optimization of storage locations is permitted within the long lines available to the programmer while ALTRAN is in memory. (Lines 00, 01, 03, 04, and 06) The optimum locations are marked by hexadecimal number XXXXXXXX. The locations reserved at the request of the operator are assigned and those locations are typed out by the computer beside the ALTRAN command entered.

Example: Assume that the programmer has planned to use line 06 for the storage of constants and some infrequently used temporary storage locations. The programmer at one point in the program wishes to pick up a constant which is to be stored in line 06 but does not



know the best available word position to choose. ALTRAN will pick the word location as the next available (empty or zero) location after the command using that location is read. The command using this location is LAR and is to be entered into location 0160. Line 06 is non-zero from locations 0661 to 0695.

.0160	LAR 06zz	.0696	ALTRAN would pick 0696
.0197			as shown at the left.

This technique of assigning locations may not be used with commands which are assigned 5 digit addresses. (LID KABCD, SID KABCD, etc.) It would be permissible with LID 06ZZ, ADD 00ZZ, etc.

## ALTRAN COMMANDS

### INFORMATION TRANSFER COMMANDS

LAR ABCD	LOAD AR
ARL AB	SHIFT AR LEFT
DAR	DECREMENT AR
TVA ABCD	TRANSFER VIA AR
SZE ABCD	STORE ZERO
SNZ ABCD	STORE NON-ZERO
SMA ABCD	STORE MAGNITUDE
SAR ABCD	STORE AR
LID KABCD	LOAD ID
LMQ KABCD	LOAD MQ
LPN KABCD	LOAD PN
SID KABCD	STORE ID
SMQ KABCD	STORE MQ
SPN KABCD	STORE PN
LTL ABCD	LINE TO LINE

### ADDITION AND SUBTRACTION COMMANDS

CIA ABCD	CLEAR AND ADD
CAM ABCD	CLEAR AND ADD MAGNITUDE
CIS ABCD	CLEAR AND SUBTRACT
ADD ABCD	ADD
ADM ABCD	ADD MAGNITUDE
SUB ABCD	SUBTRACT
SSD ABCD	STORE SUM OR DIFFERENCE
APN KABCD	ADD TO PN
BLA AB	BLOCK ADD

### MULTIPLY DIVIDE SHIFT NORMALIZE COMMANDS

CLR	CLEAR TWO WORD REGISTERS
LID ABCD	LOAD ID.1
LMQ ABCD	LOAD MQ.1
LPN ABCD	LOAD PN.1
MPY AB	MULTIPLY
DIV AB	DIVIDE
SFT AB	SHIFT
NOR AB	NORMALIZE
TPR	TRANSFER PRODUCT
TQU	TRANSFER QUOTIENT

TEST COMMANDS

TNZ ABCD	TEST NON-ZERO
TNE	TEST NEGATIVE
TOF	TEST OVERFLOW
TRY	TEST READY
TPS	TEST PUNCH SWITCH

TRANSFER OF CONTROL COMMANDS

TRA ABCD	TRANSFER CONTROL
NOP AB	NO OPERATION
OAR AB	OBEY AR
MPT ABCD	MARK PLACE AND TRANSFER
RMP AB	RETURN TO MARKED PLACE

INPUT OUTPUT AND SPECIAL COMMANDS

TIA	TYPE IN ALPHANUMERIC
TIN	TYPE IN NUMERIC
PAA	PRINT ALPHANUMERIC AR
PAN	PRINT ALPHANUMERIC NINETEEN
PNA	PRINT NUMERIC AR
PNN	PRINT NUMERIC NINETEEN
RPT	READ PAPER TAPE
BPT	BACK PAPER TAPE
RMT A	READ MAGNETIC TAPE
WMT A	WRITE MAGNETIC TAPE
WFC A	WRITE FILE CODE
SFO A	SEARCH FORWARD
SRE A	SEARCH REVERSE
DEL	DELAY
SRV	SET READY
* WRY	WAIT READY
BEL	BELL
HLT	HALT

### CONVERSION COMMANDS

* DBF	DECIMAL TO BINARY FRACTION
* DBI	DECIMAL TO BINARY INTEGER
* BDF	BINARY TO DECIMAL FRACTION
* BDI	BINARY TO DECIMAL INTEGER

### INPUT OUTPUT COMMANDS

* TIH	TYPE IN HEXADECIMAL
* TIF	TYPE IN FRACTION
* TII	TYPE IN INTEGER
* PHE	PRINT HEXADECIMAL
* PFR	PRINT FRACTION
* PIN	PRINT INTEGER
* PHT	PRINT HEXADECIMAL TAB
* PFT	PRINT FRACTION TAB
* PIT	PRINT INTEGER TAB
* PHC	PRINT HEXADECIMAL CARRIAGE RETURN
* PFC	PRINT FRACTION CARRIAGE RETURN
* PIC	PRINT INTEGER CARRIAGE RETURN
* TAB	TAB
* CAR	CARRIAGE RETURN
* PRE	PRECESS
* PUN	PUNCH

\* Line 02 subroutine

## COMMANDS

### INFORMATION TRANSFER COMMANDS

LAR ABCD      Load AR

The contents of location ABCD are transferred to the accumulator (AR) replacing the previous contents of AR. The contents of location ABCD are left unchanged. Information transferred in this manner is not changed (complemented) during transfer.

Normally the address ABCD will be a long line or a four word line. With experience, however, the operator can address the two-word registers as well as sources 27, 29, 30 and 31. Sources 27, 30 and 31 are used for extracting while source 29 may be used as a source of zeros if no "special input register" is tied to the G-15. Some of the following examples illustrate the use of this command.

Example: Assume that location 0665 holds the hexadecimal number -1234567 and it is desired to move this information to the accumulator (AR). The operator would type the following: (Assume that location 0000 has been chosen for the command.)

.0000      LAR 0665      AR will hold -1234567

Example: Assume that location 2201 holds the number 0000050 in hexadecimal and the number is to be transferred to the accumulator. The operator would type:

.0000      LAR 2201      AR will hold 0000050

Example: Assume that no "special input register" is attached to the G-15 and it is desired that the accumulator (AR) be cleared to zero. As above, the command location has already been chosen to be 0000. The operator types:

.0000      LAR 2900      AR will hold 0000000

The extract sources 27, 30 and 31 involve the four word lines 20 and 21. The mask or extractor is placed in line 20 while the number to be operated on is put in the corresponding word in line 21. The functions are outlined below:

<u>Source</u>	<u>Function</u>
27	$20 \cdot 21 + \overline{20} \cdot AR \rightarrow AR$
30	$\overline{20} \cdot 21 \rightarrow AR$
31	$20 \cdot 21 \rightarrow AR.$

Example: Source 31. Assume that location 2102 holds the number 1201234 and it is required that the most significant three digits be extracted or placed in AR. An extractor containing 1-bits in the positions to be extracted must be placed in the corresponding word of line 20 (2002). Such an extractor in hexadecimal would be the number ZZZ0000. The operator, after loading the extractor would type the command as follows:

.0000 IAR 3102 AR will hold the number 1200000

Example: Source 30. Assume that the number 1201234 is in location 2102 as above and that the extractor ZZZ0000 is stored in location 2002. It is required to place the four least significant digits of the number in 2102 into AR. The operator would type:

.0000 IAR 3002 AR will hold the number 0001234

LAR ABCD (continued)

Notice that with only one extractor in line 20, the number being operated upon can be broken into two "pieces".

Example: Source 27. Assume that the number 1234567 is in location 2100 and that the number UVWXYZO is stored in AR in hexadecimal. It is required to combine the four most significant digits of the number in 2100 with the three least significant digits in AR and to send the result of the combination to AR. An extractor of ZZZZ000 would be loaded into location 2000 and the operator would type:

.0000 LAR 2700 AR will hold the number 1234YZO

The two-word registers can be addressed as:

<u>Register</u>	<u>Address</u>
MQ.0	2400
MQ.1	2401
ID.0	2500
ID.1	2501
PN.0	2600
PN.1	2601

In each case where a two-word register is addressed using an LAR command, the sign is taken from the sign flip-flop (IP) associated with the two word registers and placed in AR.

Machine language commands which are to be modified in the accumulator should always be loaded into the accumulator using the LAR command. This is true even in the case where addition or subtraction is to take place. Since double precision commands are treated as negative numbers they may be complemented on their way to the accumulator, if the LAR command is not used. Modification using addition or subtraction in most cases requires that the amount added or subtracted apply to the absolute value of the command and yet the sign of the command must be preserved. The use of the LAR command will preserve the sign of the command but will not complement the command in the case where it is negative (double precision).

ARL AB      Shift AR Left AB bits

The contents of the accumulator (AR) are shifted left AB "bits". The original sign of AR is retained and is not shifted with the numerical portion of AR.

The shifting is accomplished by adding the absolute value of the contents of AR to AR. Since the shifting is an addition process, it is important that digits not be shifted out of AR since this is equivalent to an overflow. If the shifting produces an overflow, the sign of AR may change from its original value although the shifting continues normally. If the operator is not concerned about the sign of the number in AR then, of course, it is permissible to shift digits out of AR causing overflows.

Example: Assume that the accumulator holds the number 0000005 and it is required that it be shifted left 20 bit positions. The operator would type:

.0000    ARL 20      AR will hold 0500000

The shift AR command is very useful for "multiplying" by a power of 2 since each left shift of one bit doubles the value in AR.

Example: Assume that the number in AR is 0000001- and that this number is to be "multiplied" by 16 or  $2^4$ . The operator would type:

.0000    ARL 4    or ARL 04

The accumulator then holds the hexadecimal number 0000010- which is equivalent to the decimal integer 0000016-.

The command ARL is also quite useful for moving counters into the various positions of a machine language command when the operator is modifying commands. It is then possible to keep all counters as binary integers and move them to the appropriate positions when required.



DAR            Decrement AR

The contents of AR, if positive, will be decremented by  $1 \cdot 2^{-28}$ .

Example: Assume that a counter equal to the number 0000006 has been loaded into the accumulator. It is required that the number in the accumulator be decremented by 1 in the least significant position. The operator would type:

.0000

DAR

AR will hold the number 0000005

The use of this command makes it convenient to store all counters as binary integers scaled at  $1 \cdot 2^{28}$ . The standard input/output ALTRAN subroutine provides for converting decimal to binary integers so scaled.

If the programmer attempts to decrement a number until it becomes negative, the result will be incorrect. Normally this command will be useful in connection with a test for non-zero since positive numbers in AR can be successfully decremented to zero.

TVA ABCD      Transfer Via AR

Transfer the contents of line AB to line CD via the AR. The operation begins at the command location plus 1 and continues through word u7.

This operation is useful for inserting and removing information from a recirculating list. Information is inserted into the list from AR, the list is "pushed down", and the value pushed out of the list is left in AR.

Example: A list or table of information is stored in line 10 words 50 through u7. Each time the list is consulted the value in word u7 is required by the program, the list must be advanced 1 position (50 into 51, 51 into 52, etc.) and the original contents of AR placed in location 50. The TVA command must be located in word 49 so the action can begin at time or word 50. AR will receive the contents of 10u7.

.0049

TVA 1010
----------

It is not necessary that AB and CD in the address of the TVA command be the same. One line of information can be transferred to another via AR if desired.

SZE ABCD

Store Zero

TO THE READER (SZE)

Store a positive zero word into location ABCD. This command assumes that the object program will be run on a computer which does not have a "special input register". Most G-15 computers have provision for attaching an external input register but no register is required. The command generated calls for input from the special register and if no register is plugged into the back of the computer, zeros will be sent to location ABCD.

This command is very useful for setting locations to zero, such as a switching location which is later tested for non-zero.

Example: Location 0699 is to be cleared to zero. The operator would type:

.0000

SZE 0699

SNZ ABCD

Store Non-Zero

Non-zero information will be sent to location ABCD by the execution of this command. The information will consist of the hexadecimal number 0000000-

This command is very useful for setting a switching location to non-zero for later testing. Notice that the sign position of the information sent to ABCD will hold a 1-bit (minus) and that this bit is the information which is non-zero.

The same restrictions apply to this command as to SZE in that the object program should not use this command if it is to be run on a computer with a "special input register".

Example: Location 0699 is to be set to non-zero. That is to say the word 0699 must contain one or more 1-bits.

.0000

SNZ 0699

SMA ABCD            Store Magnitude

The magnitude of the contents of AR are sent to location ABCD. The sign of the information stored in AR is ignored when the magnitude is transferred to ABCD.

Example: AR holds the hexadecimal number -1234567 and the magnitude (1234567) is to be stored in location 2203.

.0000    

SMA 2203
----------

After execution of this command location 2203 will hold the number 1234567.

SAR ABCD      Store AR

The contents of AR are stored in location ABCD. The contents of the accumulator are not changed. (Unless AB is made 29.)

Normally the address will be one of the long lines or a four word line.

Example: The AR holds the hexadecimal number -1234567 and the contents of AR are to be stored in location 2203. The operator would type:

```
.0000 SAR 2203            Location 2203 will hold -1234567  
                                 AR is unchanged
```

Commands which are modified in the accumulator (AR), even by addition or subtraction, should be stored using the SAR command. This is true because double-precision commands have a negative sign attached to them and yet must not be complemented during the storage process. The SAR command will not complement negative numbers (or commands) during its execution.

The information transfer commands:

```
LID KABCD  
LMQ KABCD  
LPN KABCD  
SID KABCD  
SMQ KABCD  
SPN KABCD
```

are included here for use by programmers familiar with the operation of the two-word registers and are not intended to be used by persons programming the G-15 for the first time. They do provide the experienced programmer with a more flexible command structure but require more of the programmer.

It is suggested that they be avoided by persons writing their first ALTRAN programs.

LID KABCD

Load ID using a characteristic of K

The ID register will be loaded from location ABCD (and ABCD plus 1 in some cases). The transfer command will be translated and supplied with the characteristic designated by the K in the address field. Refer to Chapter 2 for a discussion of the use of characteristics with the two-word registers.

Example: Location 0420 and 0421 holds a double-precision number which is to be transferred to register ID. The sign of the double-precision number is to be transferred to the sign flip-flop associated with the two-word registers (IP).

.0000 LID 40420

The command will be supplied with a characteristic of 4.

Example: Location 2202 holds a single-precision number which is to be transferred to ID.0. The sign of the number in 2202 is to be placed in IP.

.0000 LID 02202

The command will be supplied with a characteristic of 0.

It is important to notice that the 0 must be typed if a characteristic of 0 is desired, it will not be supplied by ALTRAN.

The command LID operates in a different manner if the address field is only 4 digits long. This case is discussed under Multiply Divide Shift and Normalize commands.

The LID command is discussed here because of its possible use for storing and transferring information. It is included so that persons quite familiar with the G-15 can retain the flexibility associated with the two-word registers.

It is suggested that the programmer review the operation of the two-word registers carefully if he is not quite familiar with them, and that persons using the G-15 for the first time not use the two-word register commands having a 5 digit address field.

LMQ KABCD

Load MQ using a characteristic of K

The MQ register will be loaded from location ABCD (and ABCD plus 1 in some cases). The command will be translated and supplied with the characteristic designated by the K in the address field.

Example: Location 2101 holds the number 1234567- in complemented form. The normal form of the number is desired in the odd side of MQ. The operator would type:

.0000 LMQ 12101

The normal form of the number and the negative sign would be loaded into MQ.1.



LPN KABCD            Load PN using a characteristic of K

The PN register will be loaded from location ABCD (and ABCD plus 1 in some cases). The command will be translated and supplied with the characteristic designated by the K in the address field.

Example: Locations 0366 and 0367 hold a double-precision number in normal form. This number is to be transferred to PN in normal form and its sign is to be added to the sign in the IP flip-flop.

.0000    LPN 40366

Example: Two double-precision numbers are to be added in PN. The first number is held in 2202 and 2203 and is in normal form (absolute value and sign). It is transferred to PN and complemented if negative by the following command:

.0000    LPN 52202

The number in PN is now in a form ready for addition or subtraction.

SID KABCD

Store ID

The contents of ID will be stored in location ABCD (and ABCD plus 1 in some cases). The store command will be translated and contain the characteristic designated by the K in the address field of the command.

Example: A number shifted right in ID.1 is to be stored in location 2003. The sign of the number is presently held in the IP flip-flop.

.0000

SID 02003

Location 2003 will hold  
the number and its sign.

Example: A double-precision number has been transferred to ID and the sign of the number is presently stored in IP. Transfer the number and its sign to locations 0080 and 0081.

.0000

SID 40080

SMQ KABCD

Store MQ

The contents of MQ will be stored in location ABCD (and ABCD plus 1 in some cases). The store command will be translated and contain the characteristic designated by the K in the address field.

Example: A number has been shifted left in MQ.1 and is to be stored in AR. The sign of the number is presently held in the IP flip-flop.

.0000

SMQ 12801

AR will hold the number  
and its sign.

SPN KABCD      Store PN

The contents of PN will be stored in location ABCD (and ABCD plus 1 in some cases). The store command will be translated and contain the characteristic designated by the K in the address field.

Example: A double-precision addition has just been made to PN and the answer is to be stored in location 2100 and 2101. The number in PN may be in complemented form after the addition and must be recomplemented if negative in order to be in normal form. It will be complemented if necessary and stored in 2100 and 2101 by the following command:

.0000

SPN 52100

The number in 2100 and 2101 will be in normal form.

LTL ABCD          Line To Line

The contents of all of line AB are transferred to line CD while the contents of line AB remain unchanged. Lines AB and CD will normally be long lines and 108 words of information will be transferred.

Example: Assume that line 06 holds commands of a program which is to operate from line 01. The operator can transfer the commands from line 06 to line 01, for execution, by the command:

.0000    LTL 0601

The information in line 06 will not be changed.

Some other uses for the command include the transfer of information from the accumulator to a long line.

Example: Assume that the accumulator holds the number ZZZZZZZ and it is required to fill all of line 06 with the contents of the accumulator. The operator types:

.0000    LTL 2806

Example: Assume that the G-15 computer being used has no "special input register" attached to it. It is required to clear all of line 07 during the execution of a program. The operator would type:

.0000    LTL 2907          Line 07 will be cleared

With no "special input register" attached, the source 29 supplies "zeros" to the destination line.

The command is operative for one drum cycle and therefore is not efficient for the transfer of information between four word lines or two-word registers.

## ADDITION AND SUBTRACTION COMMANDS

CLA ABCD      Clear and Add

The number in location ABCD is transferred to the accumulator, replacing the previous contents of the accumulator, in preparation for an addition or subtraction. If the number transferred is negative, it will be complemented so that addition or subtraction can take place. The number which is then stored in AR is not in absolute value and sign form.

Example: Assume that location 0067 holds the number 000005 and that this number is to be transferred to the accumulator in preparation for an addition or subtraction. The operator types:

.0000

CLA 0067

AR will hold the number 000005

Normally the address ABCD will refer to a long line or a four word line in memory. The two-word registers should not be addressed unless the operator is quite familiar with the logic involving the two-word registers. (Refer to pages 37 and 38 of "Programming for the G-15".) Addresses such as 27, 30, and 31 may be used. Refer to the LAR command.

In the execution of the CLA command, the previous contents of location ABCD will remain unchanged unless the accumulator itself was addressed. If this is done, the contents of the accumulator will be complemented if the number was negative.

Example: Assume the accumulator holds the number 000001- and the command CLA is executed. The operator types:

.0000

CLA 2800

AR will hold ZZZZZZ-

(The number ZZZZZZ- is the complement of 000001-)

CAM ABCD

Clear and Add Magnitude

The magnitude or absolute value of the contents of location ABCD is transferred to the accumulator, replacing the previous contents of the accumulator. The previous contents of location ABCD are left unchanged. (Provided ABCD is not equal to 2800.)

Example: Assume that location 2203 holds the number 0001234- and it is required to transfer only the magnitude of this number to the accumulator. The operator types:

.0000

CAM 2203

AR will hold the number  
0001234

Refer to CIA for address limitations.

CLS ABCD

Clear and Subtract

The contents of location ABCD will be subtracted from zero and transferred to the accumulator, replacing the previous contents of the accumulator. The previous contents of location ABCD are left unchanged. (Provided ABCD is not equal to 2800.)

Example: Assume that the number 1234567- is stored in location 0099 and the command CLS is to be executed. The operator types:

.0000

CLS 0099

AR will hold the number  
1234567

Refer to CIA for address limitations.

This command will normally be followed by another arithmetic operation. The result of the CLS command may not be in absolute value and sign form.



ADD ABCD

ADD

The contents of location ABCD will be added to the contents of the accumulator and the result stored in the accumulator. The contents of location ABCD are not disturbed. (Unless ABCD was equal to 2800.)

If the resulting sum in the accumulator is positive, it will be in absolute value and sign form; if negative, it will be in complementary form. As a general rule, this command should be followed by a SSD (Store Sum or Difference) command which will automatically put the stored sum in the form of absolute value and sign.

Example: Assume that location 2100 holds the number 0000002 and the accumulator holds the number 0000006. It is required to add the contents of location 2100 to the accumulator. The operator types:

.0000

ADD 2100

AR will hold the sum  
0000008

Refer to CLA for address restrictions.

ADM ABCD

Add Magnitude

The magnitude (absolute value) of the contents of location ABCD will be added to the contents of the accumulator and the result stored in the accumulator. The contents of location ABCD are not disturbed. (Unless ABCD was equal to 2800.)

If the resulting sum in the accumulator is positive, it will be in absolute value and sign form; if negative, it will be in complementary form. As a general rule, this command should be followed by a SSD (Store Sum or Difference) command which will automatically put the stored sum in the form of absolute value and sign.

Example: Assume that location 2100 holds the number 0000002- and the accumulator holds the number 0000006. It is required to add the magnitude of the contents of location 2100 to the accumulator. The operator types:

.0000

ADM 2100

AR will hold the sum  
0000008

Refer to CIA for address restrictions.

SUB ABCD            Subtract

The contents of location ABCD will be subtracted from the contents of the accumulator and the difference stored in the accumulator. The contents of location ABCD are not disturbed. (Unless ABCD was equal to 2800)

If the resulting difference in the accumulator is positive, it will be in absolute value and sign form; if negative, it will be complemented. As a general rule, this command should be followed by a SSD (Store Sum or Difference) command which will automatically put the stored difference in the form of absolute value and sign.

Example: Assume that location 0410 holds the number 0000003 and the accumulator holds the number 0000009. It is required to subtract the contents of location 0410 from the contents of the accumulator. The operator would type:

.0000

SUB 0410

AR will hold the difference  
0000006

For addressing restrictions, refer to CLA.

SSD ABCD

Store Sum or Difference

The contents of the accumulator (the result of an addition or a subtraction) will be stored in location ABCD in the absolute value and sign form. This command will normally be used after an addition or a subtraction. Numbers which are held in the accumulator in complementary form after an addition or a subtraction will be recomplemented as necessary by this command. The contents of the accumulator are undisturbed, unless ABCD was made equal to 2800 or 2900.

Example: Assume that the accumulator holds the complement of 0000001- as the result of a subtraction. (The complement of 0000001- is ZZZZZZZZ-.) It is desired to store the number held in the accumulator in its absolute value and sign form in location 2202. The operator types:

.0000

SSD 2202

AR will still hold  
the complement ZZZZZZZZ-.  
Location 2202 will hold  
0000001-.

Refer to CIA for address restrictions.

APN KABCD

Add to PN

The contents of ABCD (and ABCD plus 1 in some cases) will be added to the double-precision accumulator PN. The command which is translated will contain a characteristic designated by K in the address field.

Example: The PN register has been loaded with a double-precision number in preparation for an addition. Locations 0340 and 0341 hold the number to be added to PN. The operator would type the following command:

.0000 APN 50340

The sum of PN and the contents of 0340 and 0341 will be in PN.

Example: Same as example above only the number in 0340 and 0341 is to be subtracted from the contents of PN.

.0000 APN 70340

The difference between PN and the number in 0340 and 0341 will be in PN.

BLA AB

Block Add

The contents of line AB are added to the AR. The operation takes place for one drum cycle (108 word-times) and is often used for check-summing information, or adding up various parts of an answer.

Example: Line 06 is clear with the exception of word 50 which holds 0000008 and word 60 which holds the number 0000003-. The accumulator is clear and the programmer wishes to sum the contents of line 06.

.0000 

BLA 06
--------

 The AR will hold the number 0000005

MULTIPLY, DIVIDE, SHIFT AND NORMALIZE COMMANDS

CLR            Clear two-word registers.

The contents of the two-word registers MQ, ID and PN are cleared to zero. The sign associated with the two-word registers (IP flip-flop) is also set to zero or plus. This command is normally given prior to a multiply, divide, shift or normalize command.

Example: To clear the two-word registers, the operator would type in the command:

.0000

CLR

The registers will be cleared to zero.

LID ABCD

Load ID.1

The contents of location ABCD will be transferred to the odd half of the two-word register called ID.1. The information transferred to ID.1 will normally serve as the multiplicand in multiplication, as the denominator in division, or as the value to be shifted right in ID.1. The contents of the accumulator will be changed if ABCD is even.

As a general rule, ABCD should be the address of a location in either a long line or a four word line. (Refer to pages 37 and 38 of "Programming for the G-15" before using other addresses.)

This command must be given before the LMQ command in a multiply sequence and before a LPN command in a divide sequence.

Example: Assume that the contents of location 0050 are to be loaded into ID.1. Location 0050 holds the number 1234567. The operator types:

.0000

LID 0050

ID.1 will hold 1234567.

The contents of 0050 remain unchanged.  
The contents of AR will change.

The ALTRAN system will assign the next command location as an odd location so that a shift command following the LID command will be located in an odd word position. (This is required by the shift command.)



LMQ ABCD

Load MQ.1

The contents of location ABCD will be transferred to the odd half of the two-word register called MQ.1. The information transferred will normally serve as the multiplier in multiplication or as the value to be shifted left in MQ.1. The contents of the accumulator will be changed if ABCD is even.

As a general rule, ABCD should be the address of a location in either a long line or a four word line. (Refer to pages 37 and 38 of "Programming for the G-15" before using other addresses.)

The contents of ABCD remain unchanged after the execution of this command.

Example: Assume that location 0409 holds the number 0000111 and it is required to load this number into the odd half of MQ (MQ.1). The operator would type:

.0000

LMQ 0409

MQ.1 will hold the number 0000111

ALTRAN will assign the next command location as an odd word position for multiply or shift commands.

LPN ABCD

Load PN.1

The contents of location ABCD will be transferred to the odd half of the two-word register called PN.1. The information transferred will normally serve as the numerator for division. The contents of the accumulator will be changed if ABCD is even.

As a general rule, ABCD should be the address of a location in either a long line or a four word line. (Refer to pages 37 and 38 of "Programming for the G-15" before using other addresses.)

The contents of location ABCD remain unchanged after the execution of this command.

Example: Assume that location 0666 holds the number 7654321 and it is required to load this number into the odd half of PN (PN.1). The operator would type:

.0000

LPN 0666

PN.1 will hold  
7654321

ALTRAN will assign the next command location as an odd word position for a divide command following.

MPY AB

## Multiply

The number in register ID is multiplied by the number in register MQ and the product is stored in register PN. This command must be located in an odd word position, a detail taken care of by the operation of the LMQ command.

For single-precision multiplication, AB need not be entered. If the operator wishes to multiply for a number of word times different than the standard 56 used for single-precision, he can enter a value for AB corresponding to the length of the multiplication. One bit of the multiplier is processed for every two word times of multiplication.

Example: Assume that the operator wishes to carry on a single-precision multiplication. He would type:

.0001

MPY

The operation will be a single-precision multiplication.

DIV AB      Divide

The number in the PN register is divided by the number in the ID register and the quotient is placed in MQ. This command must be located in an odd word position, a detail taken care of by the LPN command which should precede it.

For single-precision division the number of word times for the division need not be specified (AB is not typed). If the operator wishes to divide for a number of word times other than the standard 57 used for single-precision, he should enter a value for AB corresponding to the length of the division. One bit of the quotient is shifted into MQ.0 for each 2 word times allowed for operation.

It is important to remember that division should not be attempted if the quotient is equal or greater than 1. The quotient should be a fraction and the operator must scale his numbers so this occurs. Overflow will be turned on if the quotient is not a fraction and the result in MQ will be incorrect.

Example: Assume that the operator wishes to carry on a single-precision division. He would type:

.0001

DIV

The operation will be a single-precision division.

SFT AB

Shift

The contents of ID will be shifted right and the contents of MQ will be shifted left 1-bit position for each 2 word times of execution. The command should be located in an odd word position, a detail taken care of by the LID or LMQ command which should precede it.

If the operator wishes to shift these registers other than 4-bit positions he should type in an AB equal to twice the number of bits. If no AB is typed ALTRAN will assign an AB of 08 which will cause a shift of 4-bit positions.

Example: Assume that the number 1234567 has been loaded into ID.1 and is to be shifted right 16 bit positions so that the 123 remain in the numeric portion of ID.1. The operator would enter:

.0001

SFT 32

The number 0000123 will be in the numeric portion of ID.1

Notice that during the shift of ID, MQ also shifts.

Notice that the contents of the AR are not changed by the execution of this command (characteristic is made equal to 1).

NOR AB

## Normalize

The contents of MQ will be normalized. A left shift of 1-bit will occur in the attempt to normalize the contents of MQ for each 2 word times of operation. The command should be located in an odd word position, a detail taken care of by the LMQ command which should precede it.

If no AB is entered the translated command will be assigned a timing number of 54 permitting the normalization of a single-precision number. If the programmer knows that the number which he is trying to normalize will never have to be shifted more than say 10 bit positions he would enter an AB equal to 20.

The AR is incremented by  $1 \cdot 2^{-28}$  for each bit shift. Thus, a tally of the number of shifts required to normalize the number is easily kept in AR. This facilitates operating on floating point numbers.

Example: The programmer has loaded the odd side of MQ with the hexadecimal number 0090000. He has cleared AR. To normalize this number he would type:

.0001

NOR

MQ.1 will hold 9000000 and  
AR will contain 0000008.

TPR

## Transfer Product

The contents of PN.1 are transferred to AR along with the sign in IP. This command is normally used after a single-precision multiply command.

Example: The odd half of PN (PN.1) holds the result of a single-precision multiplication. The operator wishes to transfer the product to AR for further processing.

.0000

TPR

AR will hold the product transferred from PN.1.

TQU

## Transfer Quotient

The contents of MQ.0 are transferred to AR along with the sign in IP. This command is normally used after a single-precision division command.

Example: The even half of MQ (MQ.0) holds the quotient after a single-precision division command has been executed. The operator wishes to transfer the quotient to AR for further processing.

.0000

TQU
-----

AR will hold the quotient transferred from MQ.0.



## TEST COMMANDS

TNZ ABCD

Test Non-Zero

The contents of location ABCD are tested for non-zero. ALTRAN will look for the next available pair of commands following location ABCD and reserve them with XXXXXXXX.

As with all of the test commands, if the answer to the test is "NO", control is sent to the next command; if "YES" control is sent to the next command plus 1. ALTRAN will assign the pair (reserve them) and then type out the lower of the two location numbers so the operator may pursue the "NO" branch of the test. The operator will, at a later time, break this sequence and begin to enter commands in the "YES" branch of the test.

Example: Location 0602 holds a switch which may be either zero or non-zero. The programmer wishes to test the "position" of this switch to determine which of two possible branches of his program to follow. He would type:

.0000

TNZ 0602

If the program line 00 is clear ALTRAN will assign 0003 and 0004 as the "NO" and "YES" branches, respectively.

TNE

## Test Negative

The contents of AR are tested for negative. If the contents of AR are not negative, control is transferred to the next command. If the contents of AR are negative, control is transferred to the next command plus 1.

Example: Test the contents of AR for negative (sign minus).

.0000

TNE

If line 00 is clear  
ALTRAN will assign  
0002 and 0003 as the  
"NO" and "YES" branches  
of this test command.

TOF            Test Overflow

If an overflow has occurred as the result of an addition or division the answer to this test will be "YES"; if not the answer will be "NO".

Example: The programmer has reached a point in his program where he must test for the overflow condition.

.0000    

TOF
-----

If line 00 is clear, ALTRAN will assign 0002 and 0003 as the "NO" and "YES" branches of this test command.

Notice that the testing of overflow resets the overflow flip-flop or turns it off. It will be turned on again by an overflow resulting from an addition or a division.

When the instruction S ABCD is used to start computing, ALTRAN will turn overflow off prior to transferring control to location ABCD.

TRY

TEST READY

The computer's input-output circuitry is tested by this command. If the answer to this test is "YES" then the computer is "READY" to perform another input-output task. If the answer to this test is "NO" then the input-output circuits of the computer are busy working. This command is normally given before an input-output command to make certain that one input-output operation is complete before another is started.

There are some qualifications to the answer "YES". For instance, if the operator has been typing a number into the computer and then hit the Ⓢ key, Ready will test "YES" and yet another type-in command should not be given for at least 3 drum cycles to allow the Ⓢ function to reset.

Magnetic tape operations require some delays after READY has tested "YES". These timing restrictions are found in the Technical Bulletin describing the magnetic tape unit MTA-2.

Example: The programmer has initiated an alphanumeric type-in and wishes to know when the operator has hit the Ⓢ key.

.0000

TRY

If line 00 is clear,  
ALTRAN will assign  
0002 and 0003.

TPS

Test Punch Switch

The Punch Switch on the typewriter is tested for "ON". If the switch is "ON" the answer to this test will be "YES". If the switch is in the center position the answer to the test will be "NO".

Example: The programmer wishes to interrupt the computation of a problem whenever the Punch Switch is turned "ON". He would test its position by giving the commands:

.0000

TPS

If line 00 is clear, ALTRAN will assign 0002 and 0003 as the "NO" and "YES" branches of the test.

## TRANSFER CONTROL COMMANDS

TRA ABCD

Transfer Control

Control is transferred to command line AB word CD. Any previously marked place is destroyed.

This command is normally used to transfer control from one command line to another. It is an unconditional transfer and does not depend on a previous mark.

Example: The programmer wishes to transfer control from line 00 to word 40 of line 03. He would type:

.0000

TRA 0340

.0340

ALTRAN will not reserve location 0340 but will expect the programmer to enter a command in 0340.

The AB portion of the address may refer to any valid command line. It may be: 00, 01, 02, 03, 04, 05, 19 or 23.

Notice that no "next command" location will be reserved by the command TRA. The programmer may reserve a location using the X ABCD instruction if he desires.

NOP AB      No Operation

The NOP command is used when control is to be transferred to some command in the present command line and the programmer either does not need to, or does not want to change some "Mark Place". It is most useful for transferring control within a subroutine to which the programmer has "Marked and Transferred". In this case a TRA command would remove the "Mark" and "Return" from the subroutine would be improbable.

Example: The programmer wishes to transfer control to word 0010 in a subroutine located in line 00.

.0000

NOP 10
--------

Control will be transferred to location 0010.

ALTRAN will not reserve the location to which control is being transferred. If the programmer wishes to reserve this location he may do so by using the instruction X ABCD.

OAR AB

Obey AR

Control is transferred to the command held in AR. After the execution of the command in AR, control is automatically returned to the original command line at the word specified by the command executed from AR.

Example: Command is to be transferred to the command in AR so that the command in AR behaves as though it were stored in word 60 of some command line. The operator types:

.0000

OAR 60
--------

If AB is not entered the command in AR will be executed as though it were stored in word position 00.

Notice that ALTRAN will not assign any next command location for the command OAR.



MPT ABCD

Mark Place and Transfer

The next available location in the present command line will be marked and control will be transferred to location ABCD. The next available location is reserved by ALTRAN and the operator would normally continue entering commands in the present command line.

Example: The programmer is about to enter a MPT command into location 0000. The next available location in line 00 is at word 50. He wishes to transfer control to 0330.

.0000 MPT 0330

.0050

The AB portion of the address may be 00, 01, 02, 03, 04, 05, 19 or 23.

In the above example it should be noticed that location 0050 will be marked with XXXXXXXX to show its reservation by ALTRAN.

RMP AB

Return to Marked Place

A command will be translated which will be a "return to marked place" command. This command normally would come at the end of a subroutine which is to be entered from several places in one command line.

Example: Assume that the programmer has marked place in line 00 and transferred to location 0330. A sequence of commands in line 03 ended with the return to marked place command. To return to line 00 at the word marked the programmer would type:

.03--

RMP 00

Upon execution of this command, control will be returned to the marked place in line 00.

Address AB may be 00, 01, 02, 03, 04, 05, 19 or 23.

No next command location will be assigned or reserved by ALTRAN for this command.

## INPUT-OUTPUT AND SPECIAL COMMANDS

TIA                    Type in Alphanumeric

This command will cause the computer to permit alphanumeric information to enter lines 23 and 19. Information is automatically transferred to line 19 when line 23 is full to permit more information to enter line 23. When the Ⓢ key is typed, line 23 will normalize to the left and transfer the last information to line 19. The information may then be transferred to another line for storage.

Example: The programmer wishes to permit the entry of alphanumeric information into the computer. He would type:

.0000

TIA

Alphanumeric type-in will be permitted after the execution of this command.

Under normal circumstances, no Input or Output command should be given unless "READY" is set. That is, no input-output operation should be started while one is already in progress.

Lines 19 and 23 should not be used for other program purposes during the type-in operation.

### Timing Restrictions

The TIA command should not be given until a delay of 3 drum cycles has elapsed since Ready was set in the case where Ready was set by Ⓢ. It takes 3 drum revolutions for the Ⓢ function to reset or return to its normal position. The same timing restriction holds for the command TIN.

TIN           Type In Numeric

Numeric information may be entered into the computer through line 23 and line 19 after the execution of this command. Each numeric key typed by the operator will cause a 4-bit binary-coded-decimal or hexadecimal number to enter line 23. Information is shifted into line 23 from the least significant end of location 2300. Line 23 may be transferred to line 19 by use of the slash (/) key. Input is terminated by typing @.

Example: The programmer wishes to permit type-in of a 12 digit number. The number will be picked up from words 2300 and 2301 after type-in is complete. He would give the command:

.0000    TIN                   Numeric type-in will be permitted.

Lines 19 and 23 may not be used during the type-in operation for other program purposes such as storage of information.

Timing Restriction

See command TIA for the timing restriction placed on this command.

PAA            Print Alphanumeric AR

The contents of AR will be typed as alphanumeric characters. Information in AR must be left justified or normalized. The alphanumeric characters will be taken from AR 8 bits at a time.

This command will be of some use in typing short (3 character) labels.

Example: The programmer has stored 3 alphanumeric characters in AR and would like to type them out on the typewriter.

.0000

PAA

Alphanumeric type-out will  
be initiated.

AR may not be used during the type-out operation for computation or information transfer.

PAN            Print Alphanumeric Nineteen

The execution of this command will cause the alphanumeric characters in line 19 to be typed. They must be left justified or normalized so that the most significant character is in the most significant position of location 19u7.

Example: Assume that alphanumeric information is stored in line 19 and that it is normalized or justified toward the most significant position of 19u7. To initiate the type-out the operator would type the command:

.0000    

PAN
-----

            The output will be initiated.

Notice that information typed into the computer in the alphanumeric mode is left justified within any 4 word group but that the four word groups may not reach words 19u4 through 19u7. The PRE(Precess) command described later will permit the programmer to move the information until the most significant group of four words is positioned in 19u4 through 19u7.

Line 19 may not be used during the type-out operation.

PNA                    Print Numeric AR

The contents of AR will be typed out in numeric mode under the control of a format in line 03 words 00 through 03.

Example: The programmer has tested for Ready, loaded a format into 03,00-03, loaded information into AR and wishes to initiate a type-out.

.0000

PNA

Numeric type-out of AR  
will begin

The operation and preparation of formats is described in detail beginning on page 134 of "Programming for the G-15".

It is important to realize that once the type-out of AR has been initiated, the programmer cannot use AR for computation or information transfer during the duration of the output. The operator will have to "Wait for Ready".

PNN

Print Numeric Nineteen

The contents of line 19 will be typed out in numeric mode under the control of a format in line 02 words 00 through 03.

Example: The programmer has tested for Ready, loaded a format into 02.00-03, loaded information into line 19 and wishes to initiate a numeric type-out of line 19.

.0000

PNN

Numeric type-out of 19  
will begin

Computation may be resumed after the execution of this command so long as line 19 is not required during the type-out.



RPT            Read Paper Tape

This command will initiate paper tape reading. Information will be read into line 23 and transferred to line 19 whenever a "reload" or "stop" code is read. The reading of a "stop" code will terminate read-in.

Example: The programmer has tested for Ready, cleared lines 23 and 19 if necessary and wishes to read-in paper tape.

.0000

RPT

The photo-electric reader will start reading paper tape.

Lines 19 and 23 are "busy" during the entire read-in process and should not be used by the program until reading stops.

BPT            Back Paper Tape

The execution of this command causes the photo-reader to reverse the paper tape magazine 1 block. The contents of lines 23 and 19 may be destroyed by this operation and so should not be used during the BPT operation.

Example: The operator has tested for Ready and wishes to reverse the paper tape magazine 1 block.

.0000

BPT

The reader will reverse and Ready will be set when the operation is complete.

RMT A

## Read Magnetic Tape

This command causes magnetic tape unit "A" to read until a "stop" code is sensed. The reading operation uses lines 23 and 19 so they must not be used for other purposes at that time.

Example: The programmer has tested for Ready, cleared line 19 and wishes to initiate a magnetic tape read-in from tape unit 3.

.0000

RMT 3
-------

Magnetic tape unit 3 will begin to read information.

### Timing Restrictions

A RMT command must be delayed by the following amounts of time after Ready is set following these commands:

WMT	4 drum cycles
WFC	4 drum cycles
RMT	15 word times
SFO	16 drum cycles *
SRE	16 drum cycles *

\* This delay after a search on any tape unit, not just a search on the tape unit being read with the RMT command.

The WMT command causes the contents of line 19 to be written onto magnetic tape unit "A". Lines 19 and 23 are involved in this operation and so should not be used during this time. Information is written beginning with word u7 and proceeding until line 19 is empty. If words 00 through 03 are empty, a "short" block of tape will be written. To avoid this the programmer should store information in words 00 through 03. (One "bit" anywhere in the four words is sufficient to cause a "full" 108 word block to be written.)

If the programmer wishes to write a "short" block of information he should remember that information is read (and shifted) into line 19 beginning with word 00 and so information may be displaced from its original position.

Example: The programmer has waited for Ready, transferred a line of information into line 19, checked to be sure that some non-zero information is in locations 00 through 03 and wishes to write the block onto magnetic tape unit 2.

.0000

WMT 2
-------

The information will start to be written onto magnetic tape from line 19.

### Timing Restrictions

A WMT command must be delayed by the following amounts of time after Ready is set following these commands:

WMT	15 word times
WFC	0
RMT	15 word times
SFO	16 drum cycles *
SRE	16 drum cycles *

\* This delay after a search on any tape unit, not just a search on the tape unit addressed with the WMT command.

WFC A

## Write File Code on magnetic tape

The execution of this command will cause a file code to be written on magnetic tape unit "A".

A file code is a mark placed in the sixth channel on the magnetic tape. The file code on magnetic tape cannot be read in the normal sense. It does not contain a block number. It does however, permit the execution of "search" operations on magnetic tape. A file code will stop the magnetic tape transport, if it is searching forward or backward. Once the unit is stopped, the programmer can then initiate a read operation which could be used to read a short block containing a block identification number.

Example: The programmer has waited for Ready after positioning the magnetic tape to the desired place, and wishes to write a file code on magnetic tape unit 0 (could also be called unit 4).

.0000 



 or

### Timing Restrictions

The only timing restriction associated with WFC is that it must be given only when Ready is set.

The tape unit specified by "A" will begin to search forward after the execution of the command SFO. Searching will continue until a File Code is sensed by the magnetic tape unit.

The search operation proceeds at 45 inches per second and permits the programmer to quickly move from one group of information to another. As an example, the programmer might write a file code every 5 or 10 blocks and thus be able to search at high speed to the desired section of magnetic tape before initiating a read operation. The programmer might also count the number of file codes sensed until he has searched the desired number of file codes. (He would have to initiate a search operation after each file code is sensed.)

Example: The programmer has waited for Ready and wishes to search magnetic tape in the forward direction until a file code is sensed on unit 3.

.0000

SFO 3

Searching will begin in the forward direction.

#### Timing Restrictions

A SFO or SRE command must be delayed by the following amounts of time after Ready is set following these commands:

WMT	4 drum cycles
WFC	4 drum cycles
RMT	15 word times
SFO	15 word times
SRE	15 word times

SRE A

## Search Reverse

This command operates in the same fashion as the SFO command with the exception that searching takes place in the reverse direction.

Example: The programmer wishes to reverse magnetic tape unit number 1 until a file code is sensed.

.0000

SRE 1

Tape unit 1 will begin to search in the reverse direction.

### Timing Restrictions

The timing restrictions which apply to the SFO command also apply to the SRE command. See SFO for this information.

DEL            Delay

The programmer may wish to delay the next operation for a number of drum cycles in certain input-output situations. This command will provide a delay of at least 1 drum cycle.

The next command will be executed as soon after the one drum cycle delay as possible. In the case where the command line is almost full, the next free command location may not be found for as much as one drum cycle. In this case, the DEL command will provide a total delay of 2 drum cycles. (One for the delay and one waiting to read the next command.)

Example: Assume that the programmer has waited for Ready and stored the input from a TIN (type-in numeric) command. Because the © function takes 3 drum cycles to reset, the programmer should not give another input-output operation without waiting or delaying for 3 drum cycles. The delay sequence might look like this:

.0000	<input type="checkbox"/> DEL	Delay 1 drum cycle
.0001	<input type="checkbox"/> DEL	Delay 1 drum cycle
.0002	<input type="checkbox"/> DEL	Delay 1 drum cycle
.0003	<input type="checkbox"/> TIA	Type in Alphanumeric
.0005		



SRY

Set Ready

The execution of this command causes Ready to be set and stops any input-output operation which may be in process.

The SRY command might be used, for example, in the case where the programmer has called for a RPT (read paper tape) and yet no information has entered the computer after X seconds. The programmer might then want to stop the photo-reader and type a signal to the operator saying that the photo-reader is "out" of paper tape information. He would stop the photo-reader by giving a SRY command.

Example: Assume that the magnetic tape unit assigned to a problem has been searching forward for 30 seconds and yet has not encountered a file code. The programmer knows if the tape unit is in the area of his information, it will find at least one file code every 10 seconds. Since the unit has searched forward for 30 seconds without finding a file code the programmer wishes to stop the tape unit and then search in the other direction for his information. The command to stop the tape unit would be:

.0000

SRY

The tape unit will stop searching.

\* WRY

Wait for Ready

The command WRY is included as a convenience to the programmer, to provide him with means of waiting for an input-output operation without using the normal TRY (test ready) command. In the majority of cases the programmer will come to a point in his program where he must wait for an input-output operation to end. This can be done using a TRY and TRA pair arranged so the "NO" side of the Test Ready command transfers back to the Test Ready command. In this fashion the program goes through the TRY, TRA loop until Ready is set. Control is then transferred to the "YES" branch of the TRY command.

In the case where the programmer has the line 02 input-output subroutine in memory at the time of execution of his object program, he may use the WRY command. ALTRAN will generate a mark place and transfer command which will transfer control to a section of the input-output subroutine which has a TRY, NOP combination of commands. Control will remain in line 02 until Ready is set, at which time the subroutine will return to the marked place. (The return to marked place command is stored in location 0244 of the input-output subroutine and may be changed if desired.)

- \* Requires that the line 02 input-output subroutine be in memory during the execution of the object program.

Example: Assume that the programmer has given a PAN (print alphanumeric nineteen) command and would like to read in a block of paper tape. He must wait until the input-output circuitry is Ready before giving the RPT command. He would therefore give a WRY command before the RPT command.

.0000

WRY

Wait for Ready

.0002

BEL

## Ring Bell

The bell in the computer will ring if the command BEL is executed. The command is active for one drum cycle in order to give the control circuits time to respond.

Three drum cycles should elapse before this command is again executed to allow time for the bell solenoid to return to its rest position.

Example: The programmer wants to signal the operator at one point in his program. The signal will mean that the program is waiting for the operator to type in some data. He would give the command:

.0000

BEL

The bell will ring once.

HLT

### Halt computation

The execution of this command will cause computation to halt immediately after it is processed. Computation may be resumed only by moving the compute switch to the center position and then returning it to BP (breakpoint) or GO. Computation will be resumed at the location assigned by ALTRAN after the HLT command.

Example: Assume that the programmer has written a data processing program which requires that paper tape magazines be changed at intervals during its execution. The programmer could have written his program so that the operators attention would be called to this task by ringing the bell and then halting computation. The operator would change the paper tape magazine and then move the compute switch to resume computation.

.0000	BEL	
.0001	HLT	Computation will halt
.0003		Computation resumes here

Notice that the HLT command will reserve the next location with ~~XXXXXX~~ (location 0003) as would a LAR or ADD command.

## CONVERSION SUBROUTINES

The conversion subroutines discussed in this section are included in the ALTRAN line 02 subroutine for input-output. It is necessary that this subroutine be in memory at the time of execution of the object program if these conversion commands are to be used.

The conversion subroutines included in the ALTRAN input-output subroutine use the common "return to marked place" command stored in location 0244. When the S ABCD instruction is given, this return command (then a return to line 05 command) is replaced with a return to marked place in line 00 command. Thus ALTRAN assumes that the operator will use the subroutines stored in line 02 with commands stored or executed in line 00. If any of these subroutines are entered from a line other than line 00 the programmer should first store a return command in 0244 which will return control to the line from which his program is being executed. If he has changed the return command to other than return to line 00 he must of course restore the return to line 00 command when he wants to execute the commands from line 00.

In practice, it may be simpler to execute all of these functions from line 00 bringing new program into line 00 as needed.

\* DBF

### Decimal to Binary Fraction

The decimal (binary-coded-decimal) number in AR is converted to a binary fraction and stored in AR.

Example: The AR holds the decimal fraction .500000 and the programmer wants to convert this to a binary fraction.

.0000

DBF

After execution, AR will hold .800000

\* DBI

### Decimal to Binary Integer

The decimal (binary-coded-decimal) number in AR is converted to a binary integer and stored in AR.

Example: The AR holds the decimal integer 000015 and the programmer wants to convert this to a binary integer.

.0000

DBI

After execution, AR will hold 00000Z

\* BDF

### Binary to Decimal Fraction

The binary number in AR is converted to a decimal fraction and stored in AR.

Example: The AR holds the binary fraction .8000000 and the programmer wants to convert this to a decimal fraction.

.0000

BDF

After execution, AR will hold .5000000

\* BDI

### Binary to Decimal Integer

The binary integer in AR is converted to a decimal integer and stored in AR.

Example: The AR holds the binary integer 000000Z and the programmer wishes to convert this to a decimal integer.

.0000

BDI

After execution, AR will hold 0000015

## INPUT-OUTPUT SUBROUTINES

The input-output functions described in this section are included in the ALTRAN line 02 subroutine for input-output. If these commands are to be used, this subroutine must be in memory at the time the object program is executed. See the Conversion Subroutines preceeding this section for details regarding the execution of these commands from command lines other than line 00.

These subroutines have been programmed to provide most of the typewriter input-output, punch-out, and conversion functions normally associated with single-precision programs. In some cases the formats associated with type-outs will want to be changed to suit the programmer's task more closely.

The format associated with PHT, PFT, and PIT is stored in location 0290 and 0291 of the input-output subroutine.

The format associated with PHC, PFC, and PIC is stored in locations 0292, and 0293 of the input-output subroutine.

The common return to marked place command is stored in 0244.

It is important to remember that lines 19 and 23 as well as the two-word registers and AR are used by some of the line 02 subroutines.

In the following commands it was necessary to store the format in 0202 and 0203 before testing Ready in the input-output subroutine. For most single-precision work this will be of no disadvantage since formats for single-word output operations seldom "reload" from 0202 and 0203 during the type-out process. The format for one operation will have been picked up from line 02 before another can possibly be stored in line 02 using these subroutines. These commands are:

PHT, PFT, PIT

PHC, PFC, PIC

TAB, CAR



There is one area where caution should be exercised and that is in the case where one of the single-word output commands follows the command PUN. Since the PUN (punch-out) command usually is concerned with the punch-out of an entire line of information, it is common for the format to reload itself from line 02. The use of another command such as PIC following PUN will cause the format in line 02 to be changed to the PIC format. This will cause the information being punched to be in error. The insertion of a WRY command between PUN and another type-out command will prevent this from happening.

Test Ready commands are not used in the subroutine following an output operation so the normal buffering can take place to provide simultaneous output and computation.

\* TIH

## Type in Hexadecimal

This command provides the programmer with an easy method of entering hexadecimal information into the running program. The following operations take place:

Wait for Ready

Clear 2300

Gate Numeric Type-in

Wait for Ready

Store input number in AR

Return to Marked Place

Example: The programmer wants the operator to enter some data into the program at "run-time". The information is to be entered as a 3 digit hexadecimal number. (Notice that up to 7 digits plus sign could have been entered and stored in AR.) For the purposes of this example let us assume that the operator types in the hexadecimal number XYZ ©.

.0000

TIH

The operations described above take place and AR will hold the hexadecimal number XYZ after the operator's type-in.

\* Line 02 subroutine

TIF

## Type In Fraction

The execution of the TIF command allows the operator to enter decimal information, have it converted to a binary fraction and stored in AR during the execution of a running program. The following operations take place:

Wait for Ready

Clear 2300

Gate Numeric Type-in

Wait for Ready

Convert the decimal input to a Binary Fraction

Store the Binary Fraction in AR

Return to Marked Place

Example: The programmer wants the operator to enter some decimal information into the program at "run-time". The information to be entered in this case is the decimal number .5000000. The number must be converted to a binary fraction so that it may enter the computations.

.0000

TIF

The operations described above take place and AR will hold the converted binary fraction .8000000 after the operator's type-in.

\* Line 02 subroutine

\* TII

Type In Integer

The execution of the TII command allows the operator to enter decimal information, have it converted to a binary integer and stored in AR during the execution of a running program. The following operations take place:

Wait for Ready

Clear 2300

Gate Numeric Type-in

Wait for Ready

Convert the decimal input to a Binary Integer

Store the Binary Integer in AR

Return to Marked Place

Example: The programmer wants the operator to enter some decimal information into the program at "run-time". The information to be entered in this case is the decimal integer 15. The number must be converted to a binary integer so it may enter the computations.

.0000

TII

The operations described above will take place and AR will hold the converted binary integer 000000Z after the operator's type-in.

\* Line 02 Subroutine

\* PHE

### Print Hexadecimal

The hexadecimal number stored in the AR will be printed on the typewriter from line 19. The format previously stored by the programmer in 0200 through 0203 will be used to control the print-out.

The following operations take place:

Wait for Ready

Clear line 19

Store the contents of AR in 19u7

Print Numeric Nineteen

Return to Marked Place

Example: The AR holds the hexadecimal number 0123XYZ and a format is already stored for its print-out in locations 0202 and 0203. Print the contents of AR from line 19 under control of the stored format in line 02.

.0000

PHE

The contents of AR will be printed.

\* Line 02 Subroutine

\* PFR

### Print Fraction

This command is similar to PHE with the exception that the number in AR is converted from binary to a decimal fraction prior to print-out. As in PHE, the format used is the format previously stored in 0200 through 0203 by the programmer.

\* PIN

### Print Integer

This command is similar to PHE with the exception that the number in AR is converted from binary to a decimal integer prior to print-out. As in PHE, the format used is the format previously stored in 0200 through 0203 by the programmer.

\* Line 02 Subroutine

\* PHT

## Print Hexadecimal and Tab

The hexadecimal number stored in the AR will be printed on the typewriter from line 19. The format stored in 0290 and 0291 is stored in 0202 and 0203 to control the print-out.

The following operations take place:

Store SP7DTE format from 0290, 0291 to 0202, 0203

Wait for Ready

Clear line 19

Store the contents of AR in 19u7

Print Numeric Nineteen (Line 19)

Return to Marked Place

Example: The AR holds the hexadecimal number OZ0Z0Z0- which is to be printed out on the typewriter followed by a "tab".

.0000 PHT

The contents of AR will be printed as -.OZ0Z0Z0 (tab) on the typewriter.

\* Line 02 Subroutine

\* PFT                    Print Fraction and Tab

This command is similar to the PHT command with the exception that the number in AR will be converted from binary to a decimal fraction before print-out.

\* PIT                    Print Integer and Tab

This command is similar to the PHT command with the exception that the number in AR will be converted from binary to a decimal integer before print-out.

\*    Line 02 Subroutine



\* PHC

### Print Hexadecimal and Carriage return

The hexadecimal number stored in AR will be printed on the typewriter from line 19. The format stored in 0292 and 0293 is stored in 0202 and 0203 to control the print-out.

The following operations take place:

Store SP7DCE format from 0292, 0293 to 0202, 0203

Wait for Ready

Clear line 19

Store the contents of AR in 19u7

Print Numeric Nineteen (line 19)

Return to Marked Place

Example: The AR holds the hexadecimal number 123UVWX which is to be printed on the typewriter followed by a "carriage return".

.0000

PHC

The contents of AR will be printed as .123UVWX (carriage return) on the typewriter.

\* Line 02 Subroutine



\* TAB

Tabulate

The execution of this command causes the typewriter carriage to move to the next "TAB" stop. The following operations take place:

Store TE format in 0202 and 0203

Clear AR

Wait for Ready

Clear line 19

Store AR (clear) in 19u7

Print Numeric Nineteen

Return to Marked Place

\* CAR

Carriage Return

The execution of this command causes the typewriter carriage to advance one line and "return" to the left side of the page. The operations are similar to those of TAB except that a CE format is stored in 0202 and 0203.

\* Line 02 Subroutine

\* PRE

### Precess line 19

This command causes line 19 to be precessed or moved 4 words at a time toward word u7. Precession will not occur when there is non-zero information in words 19u4 through 19u7. If started, precession will cease when there is non-zero information in 19u4 through 19u7.

This command is most useful when punching paper tape. It is much faster to precess information toward word u7 than to punch zero information. Normally then, information is precessed in line 19 before punching. If this is done it is important to remember that the tape punched may not have 108 words in it. During input therefore it may be necessary to clear line 19 before reading the punched tape into it. This will place zero information in the positions originally holding zeros.

It is important that line 19 have at least one 1-bit in it when it is precessed. If not the precession will be attempted anyway and will have to be stopped by manual intervention of the operator.

At the end of the precession control will return to the marked place.

\* Line 02 Subroutine

\* PRE

Precess (continued)

Example: Assume that line 19 is clear except for location 1900 which holds the hexadecimal number 0000009. The programmer wishes to punch this information onto paper tape. He wants to precess line 19 so he can avoid punching unnecessary zeros.

.0000

PRE

After the precession line 19 will be clear except for location 19u4 which holds 0000009

It is important to notice that no test for Ready precedes the precession operations in the subroutine. It is assumed that the operator waited for Ready before transferring his information to line 19 for precession.

This operation must proceed at full computer speed. That is the programmer must not attempt to "single-cycle" through the PRE command or the precession will not operate correctly. The programmer should add a "Breakpoint" to the command following PRE if he anticipates single-cycle operation through his program. He can then proceed through PRE to his "Breakpoint" at full speed.

\* Line 02 Subroutine

\* PUN

Punch

This command causes the contents of line 19 to be punched out preceded by about 8 inches of "leader". It assumes that the programmer has tested for Ready, loaded a format into line 02 words 00 through 03, loaded the output information into line 19, and precessed line 19 is desired.

It is important to notice that the format loaded into line 02 must contain a "sign" format character as its first character. If the first character of the format is not a sign format character the "leader" will not be punched correctly, and useful output information may be lost.

The PUN command, like the PRE command, must be operated at full computer speed for correct operation. The PUN command (subroutine in line 02) should not be executed one command at a time by means of the "i" key on the typewriter.

The subroutine exits without testing for Ready so that the normal output may proceed while computation is going on.

\* Line 02 Subroutine

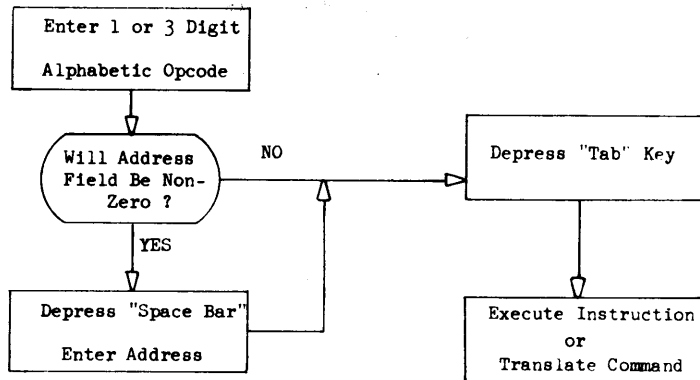
## I. LOADING ALTRAN

1. Altran magazine, rewind, on photo-reader. Compute switch center, punch switch off, type\* P and wait for the photo-reader light to turn off.
2. Compute switch GO and wait for the typewriter carriage to return. (Computer will be ready to accept ALTRAN instructions.)

## II. TO PERMIT ENTRY OF ALTRAN INSTRUCTIONS

1. Compute switch center, punch switch off, type sc5f and return the compute switch to GO, or
2. Type a "hollow point" if entering Instructions or Commands.

## III. INSTRUCTION AND COMMAND INPUT FORMAT



## IV. ALTRAN TYPE-BACK FEATURES

INSTRUCTION	FUNCTION
x 18u2	Suppress machine command type-back
z 18u2	Permit machine command type-back
x 18u4	Suppress verification
z 18u4	Permit verification

## V. TO PUNCH LINE 02 SUBROUTINE

1. Move line 02 to a free line (0, 1, 3, 4, 6)
2. Make location 44 a RMP 0 command.
3. Punch the new information.
4. Example:

```

T 0200
C 0044
.0044 RMP
P
  
```

## VI. PROGRAM EXAMPLE

**PROBLEM:** Write a program to allow the input of two 7-digit decimal fractions, sum the numbers and print the 7-digit decimal sum.

**SOLUTION:**

z 00zz	Zero line 00.
c	Enter command in 0000.
.0000 TIF	Type in fraction; Convert.
.0002 SAR 2200	Store binary fraction.
.0005 TIF	Type in fraction; Convert.
.0007 SAR 2201	Store binary fraction.
.0010 CLA 2200	Clear and Add.
.0013 ADD 2201	Add.
.0018 SSD 2800	Store sum or difference.
.0020 PFC	Convert, Print fraction; C.R.
.0022 HLT	Halt.
.0024 S	Start computing at 0000.

.5000000 s 1234567 s .6234567

\* Underlined characters signify that the Compute switch is OFF and the "Enable" switch is ON while those characters are typed.

ALTRAN COMMANDSINFORMATION TRANSFER COMMANDS

LAR ABCD LOAD AR  
 ARL AB SHIFT AR LEFT  
 DAR DECREMENT AR  
 TVA ABCD TRANSFER VIA AR  
 SZE ABCD STORE ZERO  
 SNZ ABCD STORE NON-ZERO  
 SMA ABCD STORE MAGNITUDE  
 SAR ABCD STORE AR

LID KABCD LOAD ID  
 LMQ KABCD LOAD MQ  
 LPN KABCD LOAD PN  
 SID KABCD STORE ID  
 SMQ KABCD STORE MQ  
 SPN KABCD STORE PN

LTL ABCD LINE TO LINE

ADDITION AND SUBTRACTION COMMANDS

CLA ABCD CLEAR AND ADD  
 CAM ABCD CLEAR AND ADD MAGNITUDE  
 CLS ABCD CLEAR AND SUBTRACT  
 ADD ABCD ADD  
 ADM ABCD ADD MAGNITUDE  
 SUB ABCD SUBTRACT  
 SSD ABCD STORE SUM OR DIFFERENCE  
 APN KABCD ADD TO PN  
 BIA AB BLOCK ADD

MULTIPLY DIVIDE SHIFT NORMALIZE COMMANDS

CLR CLEAR TWO WORD REGISTERS  
 LID ABCD LOAD ID.1  
 LMQ ABCD LOAD MQ.1  
 LPN ABCD LOAD PN.1  
 MPY AB MULTIPLY  
 DIV AB DIVIDE  
 SFT AB SHIFT  
 NOR AB NORMALIZE  
 TPR TRANSFER PRODUCT  
 TQU TRANSFER QUOTIENT

TEST COMMANDS

TNZ ABCD TEST NON-ZERO  
 TNE TEST NEGATIVE  
 TOF TEST OVERFLOW  
 TRY TEST READY  
 TPS TEST PUNCH SWITCH

TRANSFER OF CONTROL COMMANDS

TRA ABCD TRANSFER CONTROL  
 NOP AB NO OPERATION  
 OAR AB OBEY AR  
 MPT ABCD MARK PLACE AND TRANSFER  
 RMP AB RETURN TO MARKED PLACE

INPUT OUTPUT AND SPECIAL COMMANDS

TIA TYPE IN ALPHANUMERIC  
 TIN TYPE IN NUMERIC  
 PAA PRINT ALPHANUMERIC AR  
 PAN PRINT ALPHANUMERIC NINETEEN \* LINE 02 SUBROUTINE  
 PNA PRINT NUMERIC AR  
 PNN PRINT NUMERIC NINETEEN  
 RPT READ PAPER TAPE  
 BPT BACK PAPER TAPE  
 RMT A READ MAGNETIC TAPE  
 WMT A WRITE MAGNETIC TAPE  
 WFC A WRITE FILE CODE  
 SFO A SEARCH FORWARD  
 SRE A SEARCH REVERSE  
 DEL DELAY  
 SRY SET READY  
 \* WRY WAIT READY  
 BEL BELL  
 HLT HALT

ALTRAN INSTRUCTIONS

Z ABCD ZERO (CLEAR) LOCATION ABCD.  
 IF CD MADE EQUAL TO ZZ, THEN ZERO ENTIRE LINE AB

X ABCD FILL LOCATION ABCD WITH XXXXXX.  
 IF CD MADE EQUAL TO ZZ, THEN FILL ENTIRE LINE AB..

C ABCD ENTER ALTRAN COMMANDS STARTING WITH LOCATION ABCD.

H ABCD ENTER HEXADECIMAL NUMBERS STARTING WITH LOCATION ABCD.  
 H ABCD- LIST HEXADECIMAL NUMBERS STARTING WITH LOCATION ABCD.

F ABCD ENTER FRACTIONS STARTING WITH LOCATION ABCD.  
 F ABCD- LIST FRACTION STARTING WITH LOCATION ABCD.

I ABCD ENTER INTEGERS STARTING WITH LOCATION ABCD.  
 I ABCD- LIST INTEGERS STARTING WITH LOCATION ABCD.

P ABCD PUNCH PAPER TAPE FROM LOCATIONS ABOO THROUGH ABCD-1.

T ABCD TRANSFER ALL OF LINE AB TO LINE CD.

S ABCD START COMPUTING AT LOCATION ABCD.

R ABCD READ PAPER TAPE INTO LOCATIONS ABOO THROUGH ABCD-1.

M ABCD ENTER MACHINE LANGUAGE COMMANDS STARTING WITH ABCD.  
 M ABCD- LIST MACHINE LANGUAGE COMMANDS STARTING WITH ABCD.

D ABCD ENTER MACHINE LANGUAGE COMMANDS SEQUENTIALLY STARTING  
 AT LOCATION ABCD.

D ABCD- DOCUMENT MACHINE LANGUAGE COMMANDS SEQUENTIALLY  
 STARTING AT LOCATION ABCD.

L ABCD LIST ALL EMPTY (CLEAR) LOCATIONS IN LINE AB STARTING  
 WITH LOCATION ABCD.

B ABCD ADD A BREAKPOINT TO THE COMMAND IN LOCATION ABCD.  
 B ABCD- REMOVE A BREAKPOINT FROM THE COMMAND IN LOCATION  
 ABCD.

CONVERSION COMMANDS

\* DBF DECIMAL TO BINARY FRACTION  
 \* DBI DECIMAL TO BINARY INTEGER  
 \* BDF BINARY TO DECIMAL FRACTION  
 \* BDI BINARY TO DECIMAL INTEGER

INPUT OUTPUT COMMANDS

\* TIH TYPE IN HEXADECIMAL  
 \* TIF TYPE IN FRACTION  
 \* TII TYPE IN INTEGER  
 \* PHE PRINT HEXADECIMAL  
 \* PFR PRINT FRACTION  
 \* PIN PRINT INTEGER  
 \* PHT PRINT HEXADECIMAL TAB  
 \* PFT PRINT FRACTION TAB  
 \* PIT PRINT INTEGER TAB  
 \* PHC PRINT HEXADECIMAL CARRIAGE RETURN  
 \* PFC PRINT FRACTION CARRIAGE RETURN  
 \* PIC PRINT INTEGER CARRIAGE RETURN  
 \* TAB TAB  
 \* CAR CARRIAGE RETURN  
 \* PRE PRECESS  
 \* PUN PUNCH



LO2  
ALTRAN

27 51.33 GAS FOR CARS

31 1.95 BOOKS

36 114.85 BANK LOANS

33 4.04 CLOTHES CLEAN

43 39.05 LICENSE PLATES

44 12.35 NEWSPAPER

50 4.16 WARD.

33.34 CHECK 9

9 3.90 LIQ

SELECT

-1122334 445566.7 778899

-UUUVVWX XXYYZZ.0 2345

CAR CAR W.36.50.2.21.31

.0036 PRE PRE W.38.U3.2.21.31

.0038 PUN PUN W.40.15.2.21.31

.0040 ©c5F

©c5F

M 01-31 M- .0131

.0131 U.31.31.0.28.31

.0131 U.31.31.0.28.31

.0131 U.31.31.0.28.31 ©c5F

M -0132 M- .0132

.0132 U.44.44.0.00.00

.0144 U.46.45.5.20.31

.0145 W.00.05.0.10.31

.0105 U.08.12.7.28.28 ©c5F

©px00 ©p07x08 ©

0131 2

.02 .40

.31 .31.31.0.28.31

~~.02 .40~~  
~~.31 .31.31.0.28.31~~  
~~.32 u.44.44.0.00.00~~  
~~.44 .46.45.5.20.31~~

22 .40.42.0.29.02  
~~.42 .46.46.0.23.31~~  
~~.46 .47.48.0.02.24~~  
~~.48 .50.51.0.28.26~~  
~~.51 .53.53.3.23.31~~  
~~.53 .06.60.0.24.31~~  
~~.60 w.62.63.3.23.31~~  
~~.63 .12.78.0.24.31~~  
~~.78 w.80.83.3.23.31~~  
~~.83 .24.04.0.24.31~~  
~~.04 .06.10.0.26.28~~  
~~.10 .40.43.0.02.27~~

~~.43 .49.49.0.23.31~~  
~~.49 .51.58.0.28.26~~  
~~.58 .59.61.1.02.25~~  
~~.61 .57.11.1.25.31~~  
~~.11 .12.44.0.24.28~~

~~23 .40.42.3.29.02~~

~~37 .40.54.0.29.02~~  
~~.54 .57.57.0.23.31~~  
~~.57 .59.98.0.02.25~~  
~~.98 .99.u1.0.28.26~~  
~~.u1 .57.55.1.25.31~~  
~~.55 .64.65.0.24.28~~  
~~.65 .70.70.0.23.31~~  
~~.70 .71.72.0.02.29~~  
~~.72 .73.74.0.23.25~~  
~~.74 .75.77.0.02.24~~  
~~.77 .06.84.0.24.31~~  
~~.84 .87.87.3.23.31~~  
~~.87 .06.94.0.24.31~~  
~~.94 .97.97.3.23.31~~  
~~.97 .06.u4.0.24.31~~  
~~.u4 .u7.u7.3.23.31~~  
~~.u7 .06.26.0.24.31~~  
~~.26 .29.29.3.23.31~~  
~~.29 .06.36.0.24.31~~  
~~.36 .37.38.0.26.28~~  
~~.38 .40.67.0.02.27~~

~~.67 u.44.44.0.00.00~~

~~.68 .68.68.0.23.31~~  
~~.69 u.70.86.0.29.19~~  
~~.86 .u7.25.0.28.19~~  
~~.25 .27.44.0.09.31~~

82 .67.20.0.02.28  
~~.20 w.20.20.0.28.31~~  
~~.21 .40.39.0.29.23~~  
~~.39 w.39.88.0.12.31~~  
~~.88 w.68.88.0.28.31~~  
~~.89 .40.40.2.23.02~~  
u.00.00.0.08.16

81 .23.20.0.02.28

68

~~34 .40.65.3.29.02~~

~~24 .40.54.3.29.02~~

~~41 .90.33.4.02.25~~

~~.33 .02.68.4.25.02~~

~~76 .90.u2.4.02.25~~

~~.u2 .02.34.4.25.02~~

~~64 .90.u0.4.02.25~~

~~.u0 .02.24.4.25.02~~

~~13 .92.33.4.02.25~~

~~14 .92.u2.4.02.25~~

~~79 .92.u0.4.02.25~~

~~30 .88.96.6.02.25~~

~~.96 .w0.33.0.29.23~~

~~50 .68.96.6.02.25~~

~~u3 u.00.66.0.19.27~~

~~.66 .68.73.0.08.31~~

~~.73 .u0.99.0.00.31~~

~~.99 u.u4.66.0.19.27~~

~~15 .17.45.2.02.28~~

~~.45 w.00.05.0.10.31~~

~~.05 .06.12.7.28.28~~

~~.12 .13.44.0.28.27~~

-

~~.01 u.00.00.0.06.16 .00000x0~~

~~.03 u.96.01.2.04.00 .6001880~~

~~.06 .09.09.3.23.31 .0909YZZ~~

~~.07 .00.00.7.31.31 -.0000zzz~~

~~.08 w.w7.w7.3.31.31- .zzzzzzz~~

~~.09 .06.16.0.24.31 .061031z~~

~~.16 .19.19.3.23.31 .1313YZZ~~

~~.17 .00.00.4.07.31 -.00000zz~~

~~.18 w.w7.w7.3.31.31- .zzzzzzz~~

~~.19 .06.26.0.24.31 .061u31z~~

~~.27 u.00.00.4.00.15 -.000000z~~

~~.28 w.w7.w7.3.31.31- .zzzzzzz~~

~~.35 .40.65.0.29.02 .u8413u2~~

~~.47 .44.19.2.04.00- .uw93380~~

~~.52 u.15.15.0.07.16 .0z0z0z0~~

~~.56 .22.20.0.02.28 .961405w~~

~~.59 u.09.09.1.20.00- .0939630~~

~~.62 .v2.15.3.24.00 .z00zz00~~

~~.71 u.00.00.0.00.13 .000000x~~

~~.75 .54.91.1.20.00- .v6xv680~~

~~.80 .w7.v2.0.00.00- .zzz0000~~

~~.95 .15.w7.7.31.31- -.0zzzzzz~~

~~.96 u.16.00.0.00.00 .1000000~~

~~.91 .12.00.4.00.01 -.3w00001~~

~~.92 u.16.00.0.00.00 .1000000~~

~~.93 .12.00.4.00.00 -.3w00000~~

~~.95 .00.w7.7.31.31- -.00zzzzz~~

~~.u5 .00.15.7.31.31 -.000zzzz~~

~~.u6 w.w7.7.31.31- -.000zzzz~~



.52	u.15.15.0.07.16	.0z0z0z0
.59	u.09.09.1.20.00-	.0989680
.62	.v2.15.3.24.00	.z00zz00
.64	.90.u0.4.02.25	-.xu64059
.71	u.00.00.0.00.13	.000000x
.75	.54.91.1.20.00-	.v6xv680
.76	.90.u2.4.02.25	-.xu66059
.80	.w7.v2.0.00.00-	.zzz0000
.85	.15.w7.7.31.31-	-.0zzzzzz
.90	u.16.00.0.00.00	.1000000
.91	.12.00.4.00.01	-.8w00001
.92	u.16.00.0.00.00	.1000000
.93	.12.00.4.00.00	-.8w00000
.95	.00.w7.7.31.31-	-.00zzzzz
.u5	.00.15.7.31.31	-.000zzzz
.u6	w.w7.w7.3.31.31-	.zzzzzzz

P

0222 2

.40

.22 .40.42.0.29.02

Entry for DBF

.42 .46.46.0.23.31

.46 .47.48.0.02.24

(u w 93880)

.48 .50.51.0.28.26

.51 .53.53.3.23.31

(o z o z o z o)

.53 .06.60.0.24.31

.60 w.62.63.3.23.31

(z o o z z o o)

.63 .12.78.0.24.31

.78 w.80.83.3.23.31

(z z z o o o o)

.83 .24.04.0.24.31

.04 .06.10.0.26.28

.10 .40.43.0.02.27

.43 .49.49.0.23.31

DBF (cont)

.49 .51.58.0.28.26

.58 .59.61.1.02.25

10<sup>7</sup>.2<sup>-28</sup>

.61 .57.11.1.25.31

.11 .12.44.0.24.28

.44 .46.45.5.20.31

.23 .40.42.3.29.02

Entry for DBI

.35 .40.65.0.29.02

Entry for BDF

.65 .70.70.0.23.31

.70 .71.72.0.02.29

Add R, 0.

.72 .73.74.0.28.25

.50 .68.96.6.02.25 Entry for CAR

.03 u.00.66.0.19.27 Entry for PRE

.66 .68.73.0.08.31

.73 .u0.99.0.00.31

.99 u.u4.66.0.19.27

.15 .17.45.2.02.28 Entry for PUN

.45 w.00.05.0.10.31

.05 .06.12.7.28.28

.12 .13.44.0.28.27

u.00.00.0.08.16 .0000110

.01 u.00.00.0.06.16 .00000x0

.02 u.16.00.0.00.00 .1000000

.03 u.97.00.7.00.09- -.6180w09

.06 .09.09.3.23.31 .0909yzz

*all but D<sub>1</sub> D<sub>2</sub> D<sub>3</sub> D<sub>4</sub>*

.07 .00.00.7.31.31 -.0000zzz

.08 w.w7.w7.3.31.31- .zzzzzzz

.09 .06.16.0.24.31 .061031z

*All but D<sub>1</sub> → D<sub>5</sub>*

.16 .19.19.3.23.31 .1313yzz

.17 .00.00.4.07.31 -.00000zz

.18 w.w7.w7.3.31.31- .zzzzzzz

.19 .06.26.0.24.31 .061u31z

.27 u.00.00.4.00.15 -.000000z

.28 w.w7.w7.3.31.31- .zzzzzzz

.41 .90.33.4.02.25 -.xu21059

.47 .44.19.2.04.00 .w93880



.82 .67.20.0.02.28 Entry for T I H  
.20 w.20.20.0.28.31  
.21 .40.39.0.29.23  
.39 w.39.88.0.12.31  
.88 w.68.88.0.28.31  
.89 .40.40.2.23.02  
.40 M.44.44.0.00.00 for T I H

.56 .22.20.0.02.28 Entry for T I F  
.40 .40.42.0.29.02 for T I F

.81 .23.20.0.02.28  
.40 .40.42.3.29.02 for T I I

.68  
.34 .40.65.3.29.02 Entry for P F R

.24 .40.54.3.29.02 Entry for P I N

.13 .92.33.4.02.25 Entry for P H C  
.33 .02.68.4.25.02

.14 .92.02.4.02.25 Entry for P F C  
.02 .02.34.4.25.02

.79 .92.00.4.02.25 Entry for P I C  
.00 .02.24.4.25.02

.30 .88.96.6.02.25 Entry for T A B  
.96 .w0.33.0.29.28

.74	.75.77.0.02.24	10's telescoped
.77	.06.84.0.24.31	
.84	.87.87.3.23.31	all but D <sub>1</sub>
.87	.06.94.0.24.31	
.94	.97.97.3.23.31	all but D <sub>1</sub> D <sub>2</sub>
.97	.06.u4.0.24.31	
.u4	.u7.u7.3.23.31	all but D <sub>1</sub> D <sub>2</sub> D <sub>3</sub>
.u7	.06.26.0.24.31	go to .06
.26	.29.29.3.23.31	all but D <sub>1</sub> → D <sub>6</sub>
.29	.06.36.0.24.31	
.36	.37.38.0.26.28	
.38	.40.67.0.02.27	

.67 u.44.44.0.00.00

.68	.68.68.0.28.31	Entry for PHE
.69	u.70.86.0.29.19	
.86	.u7.25.0.28.19	
.25	.27.44.0.09.31	

.37	.40.54.0.29.02	Entry for BDI
.54	.57.57.0.23.31	
.57	.59.98.0.02.25	10 <sup>7</sup> .2-28
.98	.99.u1.0.28.26	
.u1	.57.55.1.25.31	
.55	.64.65.0.24.28	

.31	.31.31.0.28.31	Entry for WRV
.32	u.44.44.0.00.00	

**Bendix Computer Division**

LOS ANGELES 45, CALIFORNIA

