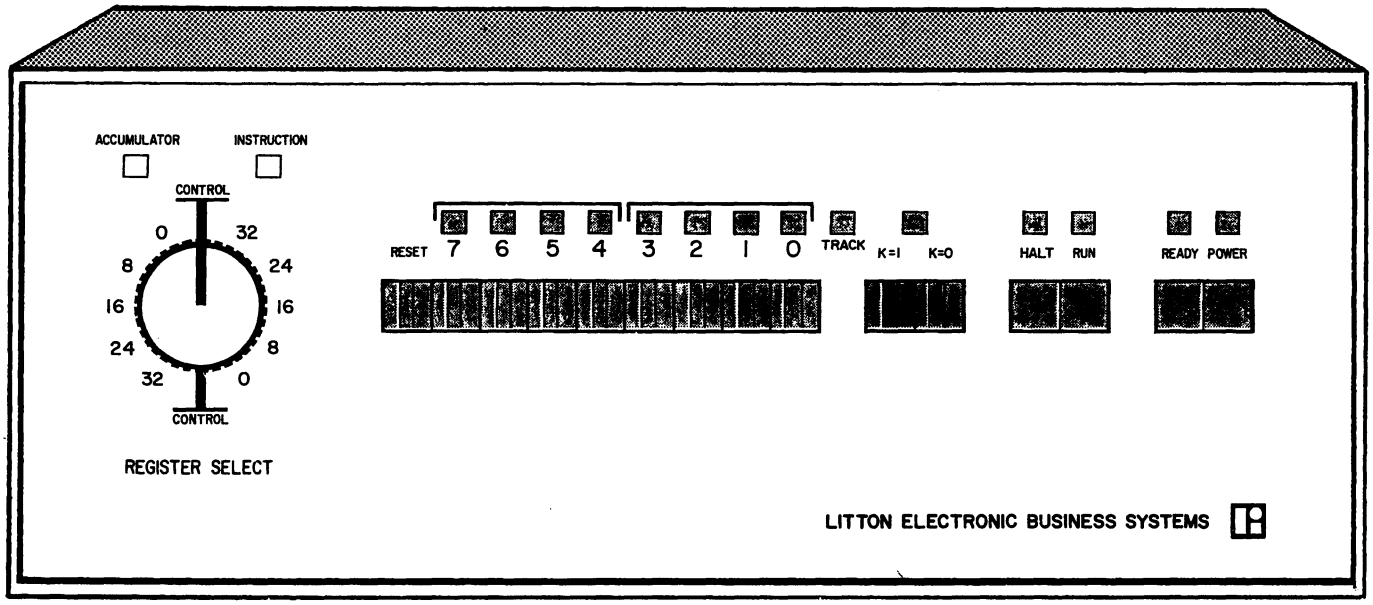
EBS 314
Revision No. 1

EBS/1231 System Programming Manual



AUTOMATED BUSINESS SYSTEMS

1231

PROGRAMMING MANUAL

EBS-314

July 1969

Automated Business Systems
a division of Litton Industries

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

SECTION I

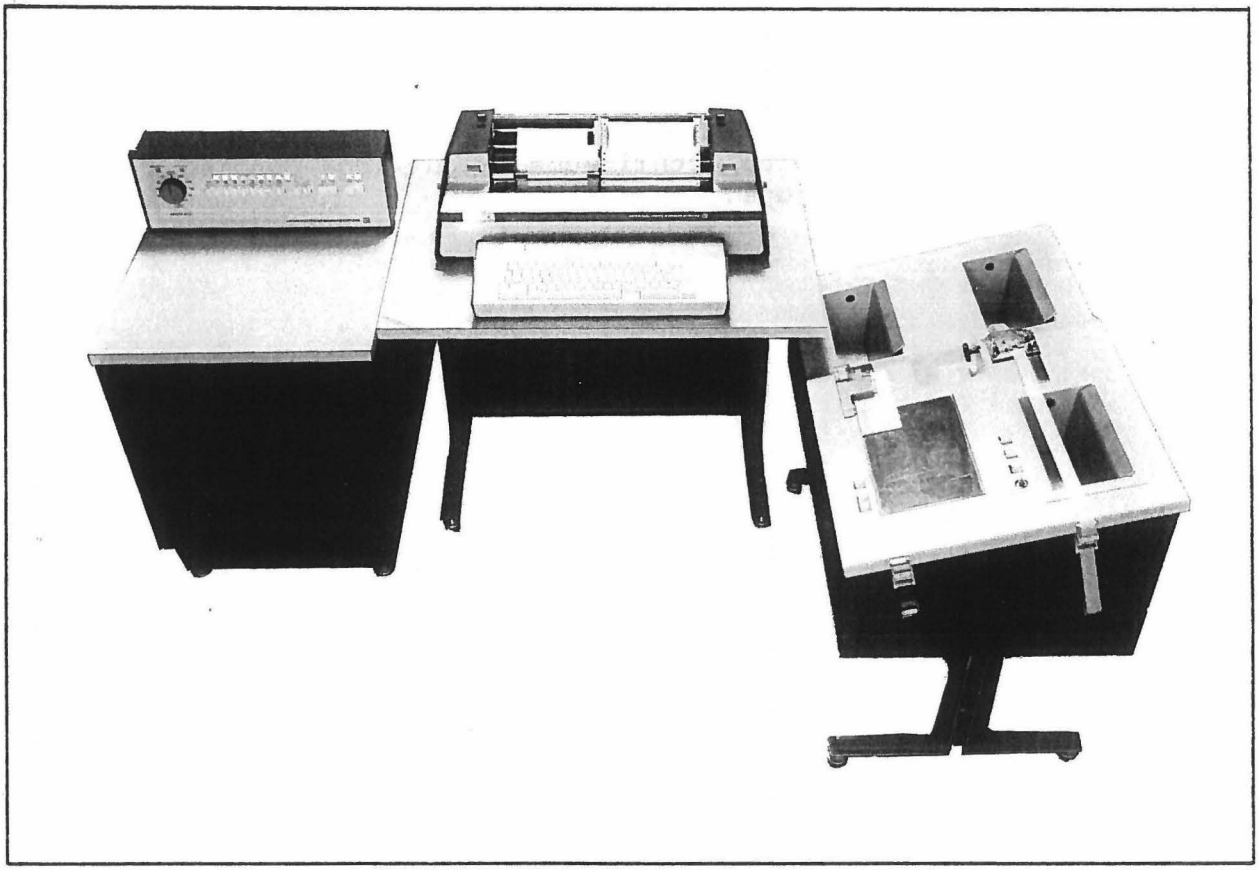
INTRODUCTION

I. INTRODUCTION

PURPOSE

This manual intends to supply a programmer with a description of the programming characteristics of the Litton EBS/1231 System and those of the EP31 Operating Program. The manual deals with basic programming as it pertains to the EBS/1231 System only.

Before the EBS/1231 can be programmed, its physical characteristics must be fully understood. The EBS/1231 Operator Manual, EBS-315, describes these characteristics and must be used in conjunction with this manual for a thorough knowledge of the EBS/1231.



GENERAL

The 1231-EP31 System is comprised of the Litton EBS/1231 System and the EP31 Operating Utility System (OPUS).

OPUS provides a series of functions which can be programmed to suit a particular application. OPUS also provides the essential service routines to assist the programmer in the creation and testing of application programs. OPUS is permanently stored in the memory unit of the EBS/1231 Processor making these functions and service routines available whenever required.

The Litton EBS/1231 System is comprised of the following components: a 1602 Processor, a Model 11 Keyboard, a Model 30 Printer, and a Model 60/70 Reader-Punch.

This System has extremely flexible forms and media capabilities.

The printer can handle: pressure fed roll, continuous or cut journals; tractor fed continuous forms, front fed ledger or cut forms.

The reader/punch can handle: continuous or cut edge-punched cards, as well as paper and Mylar tape.

SECTION II

PROGRAM AND DATA STORAGE

II. PROGRAM AND DATA STORAGE

The EBS/1231 Operating System Programs and Data are internally stored on a magnetic drum. The drum contains 694 registers for this purpose. A program register holds four instructions; a data register holds ten digits and a plus or minus sign.

The 694 registers have the following names and functions:

WORKING REGISTERS

A Register — The accumulator. All numeric data enter the computing unit through the A register. Up to ten digits can be entered with a minus sign to make the field negative. All numeric output data leave the computing unit from the A register. All arithmetic operations are performed in A. All numeric data are stored from A or retrieved from storage into A.

B Register — The B register holds the second factor in certain arithmetic and transfer operations.

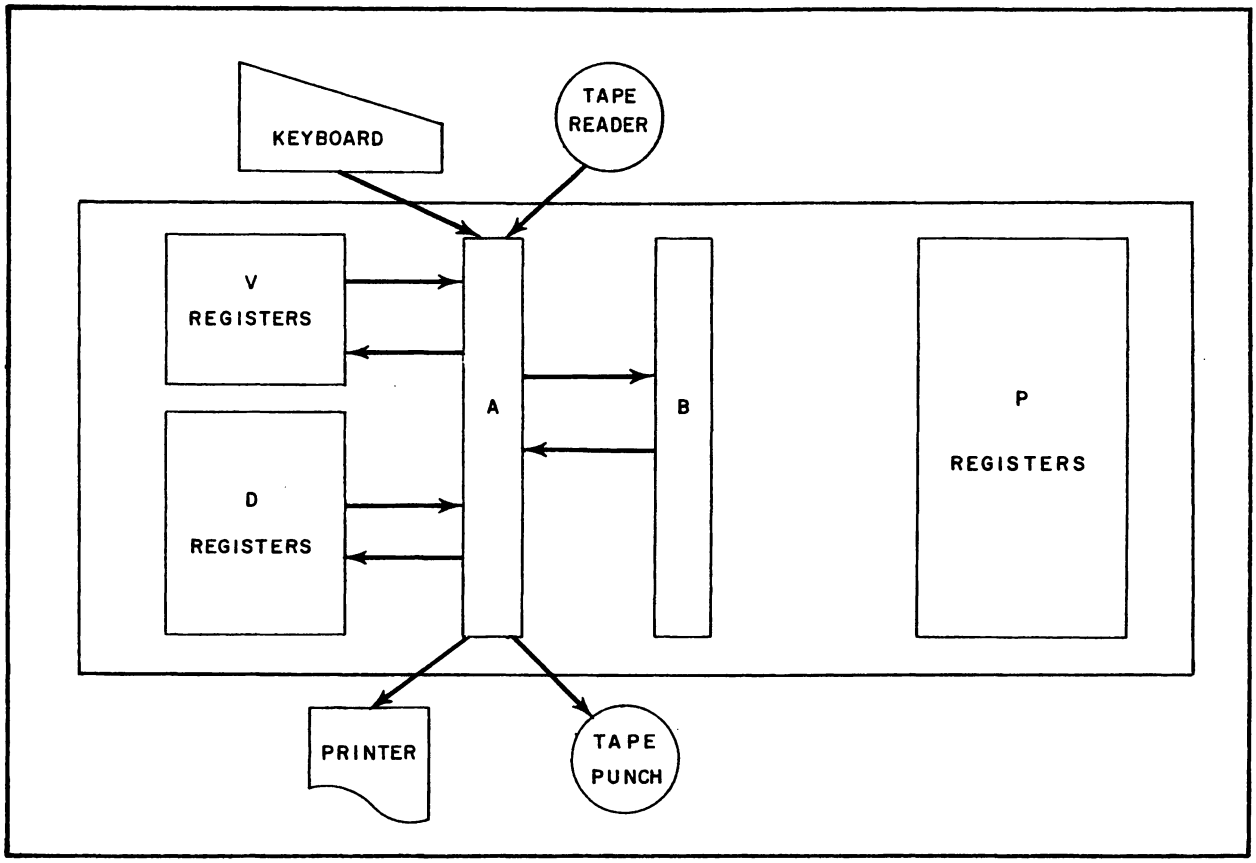
STORAGE REGISTERS

P Register — There are 128 program or "P" registers which hold the stored program. Each register holds four instructions, for a total of 512 program instructions in a single program.

V Register — There are 64 variable or "V" registers which are used for the storage of constants, accumulations, or as working areas. The contents of these registers can be brought into register A, or changed by storing from register A.

D Register — There are 500 distribution or "D" registers which are used to store accumulated totals or constants. The contents of these registers can be brought into register A, or changed by storing from register A only when specified distribution commands are utilized.

The diagram below shows the paths along which data can flow in the EBS/1231 operating system.



SECTION III

PROGRAMMING PROCEDURE

III. PROGRAMMING PROCEDURE

DESCRIPTION

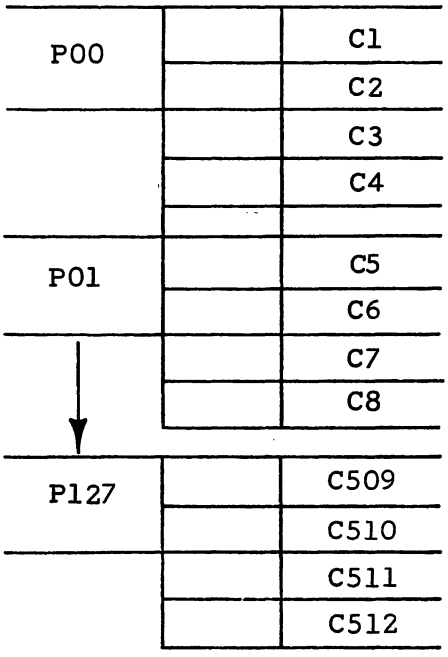
A program is the complete sequence of machine instructions necessary to solve a problem. An instruction directs the machine to perform a specific operation, such as add, multiply, print, etc.

The standard procedure for the EBS/1231 programming is first outlined below and then discussed in detail.

1. Define the program's objective.
2. Code the program in symbolic language.
3. Key-in and printout the program.
4. Punch the numeric instructions into a program tape.
5. Test the program.

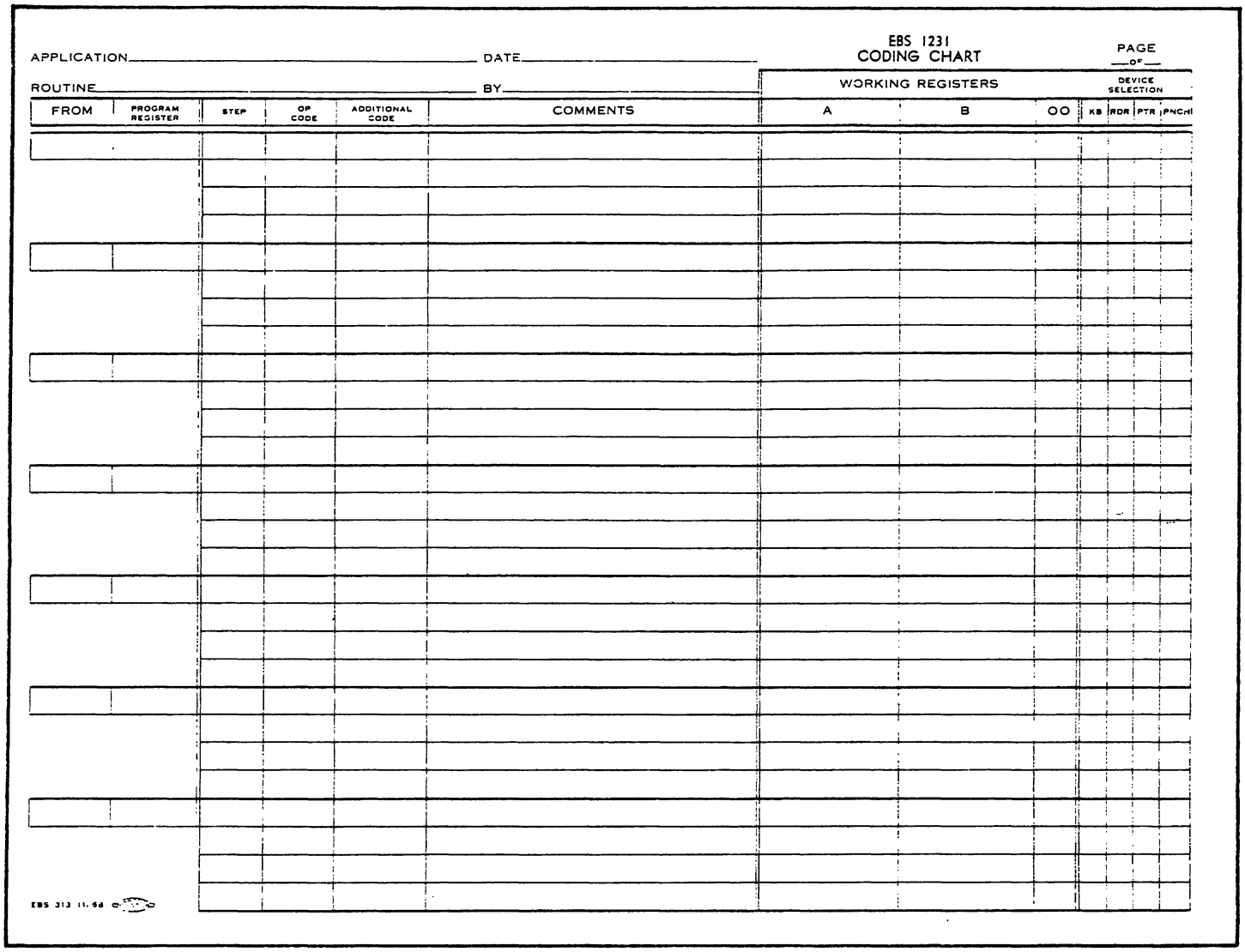
1. A program is designed to fit within the capabilities of the EBS/1231. The program objectives are defined within the limits of the devices available.

2. Coding is the writing of individual instructions in the sequence required by the application. For ease in coding, symbolic instructions are used to represent machine language instructions. Thus, BV05 is the symbolic instruction which commands the processor to Bring register V05 to the accumulator (A Register).



In coding, the instructions are assigned to the program registers which will ultimately hold them. Using C1, C2, C3, etc., to represent general illustrative instructions, the symbolic coding is done as follows:

Each box represents one register. Each register must contain four instructions. The number beside each box is the P-Register number which is, by definition, the register address. Programs always start with the first instruction in register P00. The processor executes each instruction in the register in order, and automatically sequences, after executing the fourth instruction, to the next register. The program continues, in a straight numeric sequence, from P00 through P127. This sequence can be altered by means of programmed jump instructions. A jump instruction must be programmed in P127.



APPLICATION _____ DATE _____

ROUTINE _____ BY _____

EBS 1231 CODING CHART

PAGE ___ OF ___

FROM	PROGRAM REGISTER	STEP	OP CODE	ADDITIONAL CODE	COMMENTS	WORKING REGISTERS A	B	OO	DEVICE SELECTION KB	RDR	PTR	PNCH

EBS 313 11-58

A printed form EBS-313 titled EBS/1231 Coding Chart should be used to construct the program register contents, instruction by instruction. As the program is coded, each V register used is listed so that no register will accidentally be misassigned. EBS-311 should be used for V register assignments. For all numeric output formats used, the edit words are constructed and written into the appropriate V registers on the "V" register sheet.

3. The symbolic instructions and "V" register contents are keyed into the EBS/1231 using the appropriate service routines in OPUS. The program is then printed out and checked for correct entry.

4. The program tape is then punched and verified using OPUS service routines. This program tape contains all of the program instructions and constants in proper format for re-reading at a later time. Characters in a program tape are ODD parity.

5. The program is ready to be tested to see whether it will operate as designed.

SUMMARY OF SYMBOLIC INSTRUCTIONS FOR THE EBS/1231 OPERATING SYSTEM

TRANSFER COMMAND:		OPERATING CODE	ADDITIONAL CODE
Clear	(A=0; B unchanged)	CLR	
Exchange AB	(A→B; B→A)	XCB	
Exchange V00	(A→V00; V00→A)	XCV	
Bring	(A→B; V→A; V unchanged)	BV	00- 63
Store	(A→V; A&B unchanged)	SV	00- 63

ARITHMETIC COMMAND:			
Add	(A+B→A; B unchanged)	ADD	
Negate A	(The sign in A is reversed)	NGA	
Negate B	(The sign in B is reversed)	NGB	
Update	(A+V→V; A&B unchanged)	UV	00- 63
Accumulate	(A+V→A; B&V unchanged)	ACC	00- 63
Multiply/Divide	(AxB÷V→A; B&V unchanged)	MDV	00- 31

JUMP COMMANDS:

JUMP AUTOMATIC- AJ
Go to first instruction of next program register.

JUMP UNCONDITIONAL- JUP 00-127
Go to first instruction of specified program register.

JUMP ZERO- JZP 00-127
Go to first instruction of specified program register if A is zero. Subtract 1 from A Register and go to next instruction if A is not zero.

JUMP POSITIVE- JPS
Go to second next instruction if A is zero or positive. Go to next instruction if A is negative.

JUMP MARK- JMK 00-127
Mark this place and go to first instruction of specified program register.

JUMP RETURN- JR
Go to instruction immediately following the last Jump Mark executed.

INPUT-OUTPUT COMMANDS:		OPERATING CODE	ADDITIONAL CODE
Channel Selection	(No registers affected)	SEL	00- 77
Input	(A→B; Input→A)	IN	01- 10
Skip		SKIP	
Output	(A→Output;) (A&B unchanged)	OUT	00- 31
Duplicate	(No register affected)	DUP	01-190
Character Output Refer to table of codes.	(No registers affected)	CO	00- 77
Single Character Input	(Input→A; B unchanged)	SCI	
Single Character Output	(A→output;) (B unchanged)	SCO	
Tab	(No registers affected)	TAB	01-190
Alpha Input	(5 alpha char. A)	ALFI	
Alpha Output	(A→5 alpha char. output)	ALFO	
Input ASCII	(Numeric ASCII→A; A→B)	INA	

DISTRIBUTION COMMANDS:	OPERATING CODE
Clear Distribution Registers (D registers set to zero;) (A and B are destroyed)	DCLR
Bring a Distribution Register (A→B; D→A; D unchanged)	DGET
Store to a Distribution Register (A→D; A&B unchanged)	DPUT
Distribute a Tape as specified, to the 500 Distribution Registers.	DIST
Search the Distribution Registers for a Non-Zero Value (Value→A; D Register Address→V007)	SCAN
Bring a Split Distribution Register	SGET
Store a Split Distribution Register	SPUT

SPECIAL COMMANDS:		
Program Interrupt		OPUS
Calculate		CALC
Check-Digit Verification		CDV
Convert and Duplicate	(Even Parity Input)	DUPE
	(Odd Parity Input	DUPO
	(Special Code Output)	SPEC

SECTION IV

INSTRUCTIONS

IV. INSTRUCTIONS

DESCRIPTION

The instructions used in programming the EBS/1231 are divided into six groups:

1- TRANSFER - Those instructions concerned with the movement of data into and out of specific registers.

2- ARITHMETIC - Those instructions which do the actual computations.

3- JUMP - Those instructions which make it possible to branch out of the normal sequence of program steps.

4- INPUT/OUTPUT - Those instructions which govern the flow of data from keyboard and reader into the processor and from the processor to the printer and punch.

5- DISTRIBUTION - Those instructions which control the use of the 500 distribution registers.

6- SPECIAL - Those instructions whose functions do not fall in any of the above groups.

In describing the operation of each EBS/1231 instruction, the following format is used:

Instruction The name of the instruction.

Symbolic The symbol used in writing a program. When the symbolic instruction is given in the form BV00-BV63, it means that this instruction has sixty-four forms, BV00, BV01, ...BV63.

Description An explanation of the operation of the instruction.

Example An example is given to show how the values change in registers affected by an instruction or to show how an instruction might be programmed. In the examples, C1, C2, etc., are representative non-jump instructions.

TRANSFER INSTRUCTIONS

The instructions detailed in this section are those which handle the transfer of data between the A register, the B register and the 64 V registers.

Instruction Clear

Symbolic CLR

Description Register A is cleared to zero.

Example CLR

REGISTER	A	B
Before	+0000985683	Not affected
After	+0000000000	

Instruction Exchange AB

Symbolic XCB

Description The contents of register A are transferred to register B. The previous contents of register B are transferred to register A.

Example XCB

REGISTER	A	B
Before	+0000000097	-0000000063
After	-0000000063	+0000000097

Instruction Exchange V00

Symbolic XCV

Description The contents of register A are transferred to register V00. The previous contents of register V00 are transferred to register A.

Example XCV

REGISTER	A	B	V00
Before	+0000001234	Not Affected	-0000000011
After	-0000000011		+0000001234

Instruction Bring

Symbolic BV00 - BV63

Description The contents of register A are transferred to register B. The contents of the V register specified in the BRING instruction are transferred to register A. The contents of the V register are preserved.

Example BV09

REGISTER	A	B	V09
Before	+0000000890	-0000098765	+0000000006
After	+0000000006	+0000000890	+0000000006

Instruction Store

Symbolic SV00 - SV63

Description The contents of register A are stored in the V Register specified by the STORE instruction. The previous contents of that register are lost. The contents of A are preserved.

Example SV12

REGISTER	A	B	V12
Before	-0000000606	Not affected	+0000008888
After	-0000000606		-0000000606

ARITHMETIC INSTRUCTIONS

The instructions detailed in this section provide the four arithmetic processes of addition, subtraction, multiplication and division. The absolute value of the maximum number that can be held internally is 34 359 738 367, that is, $2^{35} - 1$. However, the largest number at the output can only be 9 999 999 999, that is, a maximum of ten significant digits.

Instruction Add

Symbolic ADD

Description The number in register B is added to the number in register A. The sum is stored in register A. The contents of B are preserved.

Example ADD

REGISTER	A	B
Before	+0005634721	+0007469887
After	+0013104608	+0007469887
Before	+0007552845	-0000856338
After	+0006696507	-0000856338

Instruction Negate A

Symbolic NGA

Description The sign of the number in register A is reversed. If the number was positive, it becomes negative. If the number was negative, it becomes positive. However, if the number is zero, it is always treated as a positive number.

Example NGA

REGISTER	A	B
Before	+0000009876	Not affected
After	-0000009876	
Before	-0000000017	
After	+0000000017	

Instruction Negate B

Symbolic NGB

Description The sign of the number in register B is reversed. If the number is zero, it is treated as a positive number.

Example NGB

REGISTER	A	B
Before	Not affected	+00000012345
After		-00000012345

Instruction Update

Symbolic UV00 - UV63

Description The contents of register A are added to the contents of the V register specified in the UPDATE instruction. The sum is stored in the V register. The contents of A are preserved.

Example UV13

REGISTER	A	B	V13
Before	+0000000042	Not affected	+0000000336
After	+0000000042		+0000000378
Before	-0000000005		+0000000450
After	-0000000005		+0000000445

Instruction Accumulate

Symbolic ACC00 - ACC63

Description The contents of the V register specified in the instruction are added to the contents of the A register. The sum is stored in the A register. The contents of the V register are preserved.

Example ACC14

REGISTER	A	B	V14
Before	+0000000324	Not affected	+0000045244
After	+0000045568		+0000045244
Before	-0000000050		+0000000026
After	-0000000024		+0000000026

Instruction Multiply - Divide

Symbolic MDV00 - MDV31

Description The multiply-divide instruction has the form $\frac{AxB}{V} = Q$, that is, the contents of register A are multiplied by the contents of register B and the product is divided by the contents of the V register specified in the instruction. The result of the operation is automatically rounded off and stored in register A. The contents of the B and V registers are preserved.

All factors (A, B, or V) can be up to 10 decimal digits, either positive or negative, and correct results, including sign, will be obtained. If the A, B, or V register is zero, computation does not take place and a zero is recorded in A.

If the result (Q), as computed, is larger than the maximum number that can be held internally, a zero is recorded in A.

Example MDV15

REGISTER	A	B	V15
Before	+0000000246	+0000000000	+0000000003
After	+0000000000	+0000000000	+0000000003
Before	-0000000688	+0000000001	+0000000007
After	-0000000098	+0000000001	+0000000007
Before	-0000000689	-0000000001	+0000000007
After	+0000000098	-0000000001	+0000000007
Before	-0000000530	-0000000004	-0000000001
After	-0000002120	-0000000004	-0000000001
Before	-0000000690	+0000000001	-0000000007
After	+0000000099	+0000000001	-0000000007
Before	+0000065000	+0000000003	-0000000100
After	-0000001950	+0000000003	-0000000100

JUMP INSTRUCTIONS

The normal sequence of instructions is to process the first instruction in a program register, followed by the second, third, and fourth instructions. After the fourth instruction, the EBS/1231 then executes an automatic jump to the first instruction of the next program register. However, there are times when it is desirable to break this sequence and transfer control to another section of the program.

A jump instruction is used to change the sequence of instructions. Any one of the four instructions in a program register can jump to any one of the 128 P registers. Register P127 must always contain a programmed jump as it is the last register in the sequence.

Instruction Automatic Jump

Symbolic AJ

Description If program sequencing requires less than four instructions in one register, the balance of the register must be filled in with automatic jump instructions. As soon as an automatic jump instruction is interpreted, the program sequences to the first instruction of the next P Register (except in register P127 which must contain a programmed jump).

Instruction Jump Unconditional

Symbolic JUP00 - JUP127

Description The program jumps to the first instruction of the P register specified by the jump instruction. No register contents are changed.

Example

P05	C1	
	C2	
	C3	
	JUP	24

P24	C4	
	C5	
	C6	
	C7	

Instruction Jump Zero

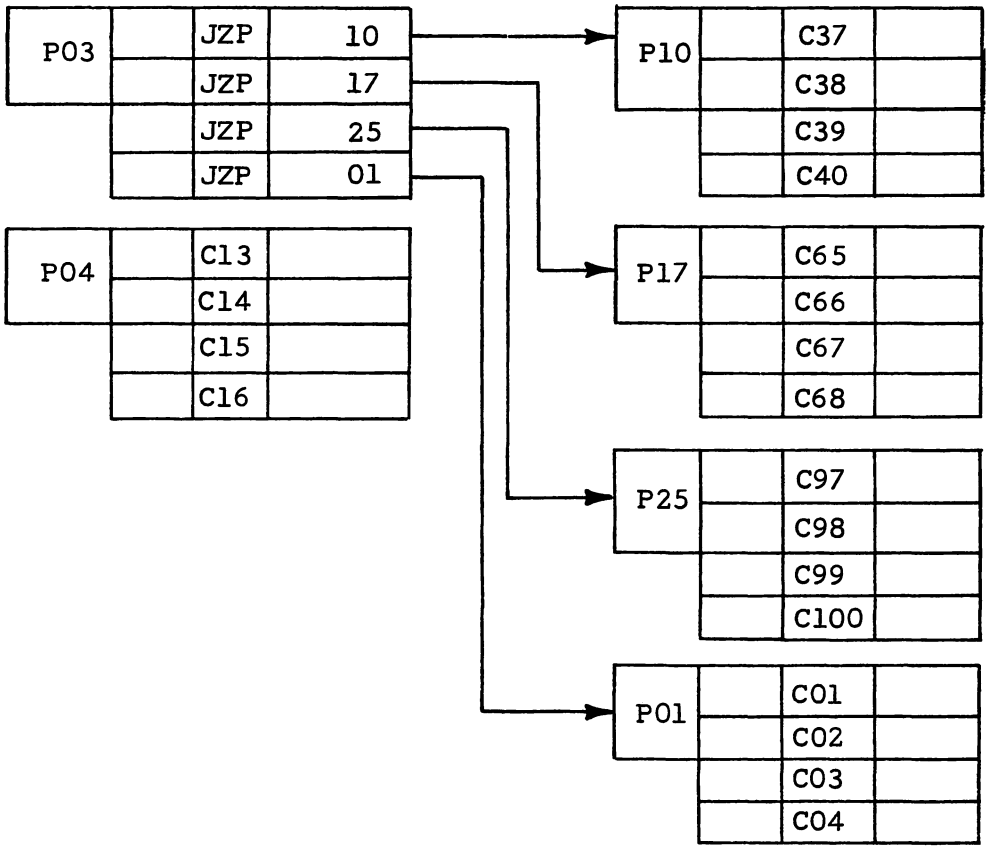
Symbolic JZP00 - JZP127

Description If register A = 0, the program jumps to the first instruction of the specified register. If A is not equal to zero, -1 is added to A and the program sequences to the next instruction.

Example

Instruction Being Executed	A Before	A After	Next Instruction Executed
JZP10	2	1	JZP17
JZP17	1	0	JZP25
JZP25	0	0	C97

(affects only the A register)



Instruction Jump Positive

Symbolic JPS

Description If the value in register A is positive or zero, the program sequences to the second next instruction. If the value in register A is negative, the program sequences to the next instruction. No register contents are changed.

Example

P01		C1	
		JPS	
		C3	
		C4	

(1) If A is positive or zero, C4 is executed.

(2) If A is negative, C3 is executed.

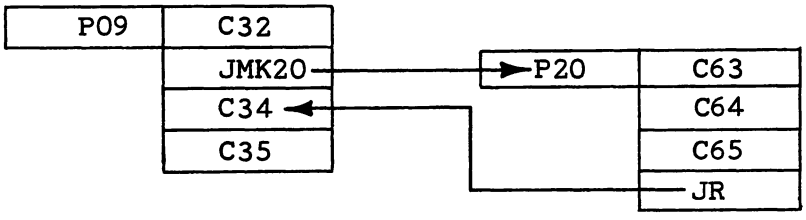
NOTE: JPS is ineffective as the fourth instruction of a register since the next instruction, the automatic jump, will lead directly to the second next instruction. That is, regardless of whether register A is zero, positive or negative, the program will sequence to the first instruction of the next register.

Instruction Jump Mark

Symbolic JMK00-JMK127

This instruction marks the spot where the jump mark command is used. The sequence of the instructions then jumps to the first instruction of the specified register. When the program finds a jump return instruction, it then jumps back to the instruction immediately following the last jump mark executed. The jump return instruction will only recognize the last jump mark executed. Thus, a jump return should be executed before another jump mark is processed.

Example JMK20

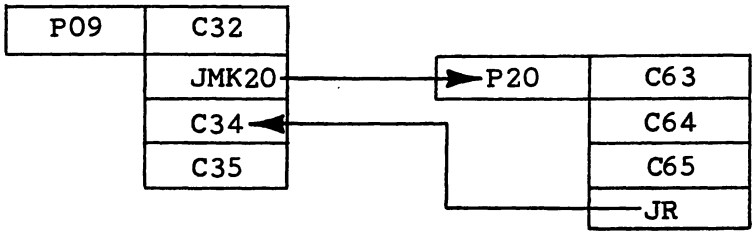


Instruction Jump Return

Symbolic JR

Description The program jumps to the instruction immediately following the last jump mark executed.

Example JR



INPUT AND OUTPUT

Data entered into the processor from either the keyboard or the tape reader is defined as Input.

Similarly, data transferred from the processor to the printer or tape punch is defined as Output.

The input and output devices are hooked into the 1231 system at specific points, referred to as "channels".

In the 1231 System, the keyboard is connected to channel 1 input; the printer is connected to channel 1 output; the reader is connected to channel 2 input; and the punch is connected to channel 2 output. Channel 3 input and output are available for an additional input device and/or output device.

Instruction Select Channels

Symbolic SEL 00 - SEL 77

Description The selection of channels determines from which channel(s) data can be input and through which channel(s) data can be output. The select channels instruction consists of two parts. The first digit selects (turns on or off) the input channel(s). The second digit controls the output channel(s). When one set of input or output channels is turned on, the previous set is automatically turned off, unless they have been re-selected. This instruction can select up to three input channels and three output channels, individually or simultaneously.

Example

	INPUT		OUTPUT
0	All input channels OFF	0	All output channels OFF
1	Channel 1 ON (Keyboard)	1	Channel 1 ON (Printer)
2	Channel 2 ON (Reader)	2	Channel 2 ON (Punch)
3	Channels 1 and 2 ON (Keyboard and Reader)	3	Channels 1 and 2 ON (Printer and Punch)
4	Channel 3 ON	4	Channel 3 ON
5	Channels 1 and 3 ON	5	Channels 1 and 3 ON
6	Channels 2 and 3 ON	6	Channels 2 and 3 ON
7	Channels 1, 2 and 3 ON	7	Channels 1, 2 and 3 ON

NOTE: When an output is attempted to a channel that is not selected, the program will hang-up. The corrective procedure is to depress HALT, READY, and RUN buttons on the console, and to correct the error in the program.

Instruction Input

Symbolic IN1 - IN10

Description The contents of register A are transferred to register B. Register A is cleared. The following characters are accepted as described:

0-9: These digits are entered into register A, up to a maximum field size as specified by the input instructions. These are the only characters which effect the field size count (leading spaces are ignored).

MINUS: (◊ or -) Makes the number in register A negative. The diamond (◊) or hyphen key (-) can be depressed anytime before the termination key is depressed.

CLEAR: Clears register A and allows re-entry of the field, if depressed before the termination key.

All other characters are ignored by this instruction.

The following characters cause the input instruction to terminate. These characters are the control and program selection keys on the keyboard. All these characters affect V00 except control key I.

I	No effect on V00
II	V00 set to 1
III	V00 set to 2
IIII	V00 set to 3
P0	V00 set to 4
P1	V00 set to 5
P2	V00 set to 6
P3	V00 set to 7
P4	V00 set to 8

Instruction INPUT (P-03 Compatibility)

Symbolic IN 01 - IN 10

Description The input command will also read a tape produced from the P-03 ADD PUNCH. The P-03 ADD PUNCH punches fields on tape containing specific start-of-field codes and end codes depending on the key (motor bar) that is depressed.

Each field punched to tape, contains digits (0-9) and space codes that are processed by the Input command in the same manner as described on page 4-14.

On the following page is a chart of start codes and end codes produced by each motor bar on the P-03. The chart shows the EBS 1231 code equivalent of each P-03 code, how the Input command processes each field, and what effect the end code of each type of field has on register V00.

NOTES:

1. The Input command now recognizes the CP1 code and the Line Feed Left code as end codes to a numeric field. Therefore, these two codes should not be output to tape as a by-product of processing any program. Their use should be limited to the printer only, or punched to tape only when it is certain they will ultimately be read under the DUP command.

2. When duplicating a P-03 tape (DUP command), the printer should not be selected.

P-03		EBS 1231		
MOTOR BAR	DESCRIPTION	CODE EQUIVALENT	EFFECT ON INPUT	EFFECT ON VOO
INT CYCLE NA	Start Code End Code	Asterisk (*) Control II	Ignored Terminates Input, with data field in Reg. A.	Set to 1
—	Start Code End Code	Minus (-) CP1	Sets input routine for a negative entry. Terminates Input, with negated field in Reg. A.	No effect
DES CYCLE NA	Start Code End Code	Clear Line Feed Left	Ignored. Terminates input, with data field in Reg. A.	Set to 2
AMT +	Start Code (None) End Code	CP1	Terminates input, with data field in Reg. A.	No effect
SUB TOTAL	Start Code (None) End Code	Line Feed Right	Ignored	No effect
TOTAL	Start Code (None) End Code	Percent (%)	Ignored	No effect
	Tape Leader	Carriage (Open/Close)	Ignored	

Instruction Skip Field from Tape

Symbolic SKIP

Description The SKIP command is used to read and ignore a field (or fields) from tape. It replaces the INU (Input unlimited) command. The INU command is removed from the EP31 command list and should not be used.

The SKIP command will read and ignore characters from tape until a control I key is recognized. No output will occur even though output channels may be selected. Registers A and B are unaffected.

Instruction Output

Symbolic OUT00 - OUT31

Description The contents of register A are output according to the format contained in the V register specified by the instruction. The contents of the A, B and V registers remain unchanged. The format specifies the field size, the use of the symbols and whether the sign and field end characters are to be output. Leading zeroes are output as spaces, zeroes, or asterisks. A minus is output as a hyphen (-). A plus is output as a space.

The control key I tape character may be output at the end of every field. This gives the numeric output field the correct format for subsequent use as a numeric input field. The field size used must be large enough to accommodate the largest number expected in that field. The maximum field size for numeric output is $10^{10}-1$ (ten digits). If the number in register A is greater than this, the output is incorrect.

Example OUT02 V02 = 455 556 466 360

FORMAT = X,XXX.XX (with sign)

CONTENTS OF A	CHARACTERS OUTPUT
+0000012345	123.45
-0000987654	9,876.54-
+0000000007	.07

CONSTRUCTION OF OUTPUT FORMAT CONSTANTS (EDIT WORDS)

A numeric field cannot be output without specifying the output format. The output instruction refers to a specified V register to find the format. The constant in that V register is created as follows:

To determine the edit word constant for a specific format, write down the format desired for the field to be output, showing all characters to be printed, including symbols (/,.);

X,XXX.XX

Add an (S) to the right if a sign is to be output.

X,XXX.XX(S)

Fill in to the left with S's so that the S's and X's total 10. Positions identified with S will be suppressed, that is, not printed at all. The S's must be used because all 10 possible positions must be coded. (This does not include the (S) for sign.)

SSSSX,XXX.XX(S)

Add an asterisk (*) to the left if leading spaces are to print as asterisks.

*SSSSX,XXX.XX(S)

Add a "Z" to the left if leading spaces are to print as zeroes.

ZSSSSX,XXX,XX(S)

Add a Roman numeral I to the left if an end code (Control I Key) is to be output following the field.

I*SSSSX,XXX.XX(S)

This format is used to construct the edit word constant for the V register, using the following tables. An edit word of 12 characters must be constructed. The first character (on the left) controls the field end code and the use of asterisks, zeroes, or spaces to be output for non-significance.

0 = Leading spaces; no end code

1 = Leading zeroes; no end code

2 = Leading asterisks (*); no end code

4 = Leading spaces with end code

5 = Leading zeroes with end code

6 = Leading asterisks (*) with end code

The next 10 characters control the 10 possible positions to be output:

5 = S (suppress). It is written for all positions to be suppressed.

2 = (/) output a slant followed by the next digit. All digits following the slant are considered to be significant.

4 = (,) output a comma followed by the next digit. If no significant digit has been output, a space, an asterisk, or a zero is output followed by the next digit.

3 = (.) output a decimal point followed by the next digit. All digits following the decimal point are considered to be significant.

6 = output any digit. A space, an asterisk, or a zero is output for leading zeroes.

7 = Used only with programs converted command-for-command from the EBS/1230. For these converted programs, this code represents the first digit.

The last character (on the right) controls the sign and end of output.

0 = Output the sign of the number; space for positive, hyphen for negative, and terminate the instruction.

1 = No sign is output. Terminate the instruction.

The edit word constant for the example format is:

Format	I*SSSSX,XXX.XX(S)
Edit Word	655 556 466 360

A table of commonly used formats and edit words is contained in the appendix.

Instruction Duplicate

Symbolic DUP1 - DUP190

Description Descriptive data are entered from the keyboard or tape reader, one character at a time, and output immediately to the channel(s) selected. This instruction is terminated with the control key I character only. The control I character is not automatically output and must be generated via the character output instruction. During this instruction, if the code for the RETURN key is recognized, the print wheel is automatically moved to the position specified in this instruction; a line feed is not automatically output. Every third printer position may be addressed, from 1 through 190 (i.e., 1, 4, 7, 10....190). All valid characters are handled by this instruction, either alpha-numeric or printer position (refer to appendix for complete list).

Example DUP16

1. All characters are duplicated until a control key I code is recognized which terminates the instruction.
2. If the RETURN key code is recognized during this instruction, the print wheel moves to position 16.

Instruction Character Output

Symbolic C0 00 - C0 77

Description This instruction is used to directly output any one of 64 alpha-numeric characters. See the appendix for a complete list of characters. No registers are affected by this instruction.

Instruction Single Character Input

Symbolic SCI

Description Any single character is accepted from any selected input channel and stored in A. B is not affected. Only one character is read by this instruction. No end code is required. Reference the table in the appendix for the internal values of characters read by this instruction.

Instruction Single Character Output

Symbolic SCO

Description The program will output one character from the A register. B is not affected.

Instruction Tab

Symbolic TAB 01 - TAB 190

Description The program will tab to the specified position. Every third position may be addressed, from 1 to 190 (i.e., 1, 4, 7, 10, ...190).

Instruction Alpha-Numeric Input

Symbolic ALFI

Description This command accepts the input of five characters only. After the fifth character is entered, control moves to the next instruction in the program. The five alpha-numeric characters remain in Register A after leaving this command. The previous contents of Register A are destroyed. The contents of Register B are unchanged. Any character is accepted by this command, including the Control keys, the "P" keys and tab codes (which use the "shift" key in conjunction with another key on the keyboard).

Example ALFI

Entry of the word STOP.

REGISTER	A	B
BEFORE	+0000012479	-0000000124
AFTER	STOP (P4)	-0000000124

NOTE: The P4 key is recommended for use as a filler code for fields less than five characters long. The P4 code is a non-printing code.

Instruction Alpha-Numeric Output

Symbolic ALFO

Description This command outputs the contents of Register A as five alpha-numeric characters. Register A and B are unchanged. The command is exited after the 5th character is output.

Example ALFO

REGISTER	A	B
BEFORE	STOP (P4)	-0001020411
AFTER	STOP (P4)	-0001020411

Instruction Input of ASCII-Coded Tape

Symbolic INA

Description The contents of Register A are transferred to Register B. This command will read and accept digits (0-9), and the minus code. When the ESC code is recognized, the command is exited with the numeric data in Register A.

The minus code will reverse the sign of the value in Register A. The rub-out code will clear Register A when recognized. All other ASCII codes (including the RS code) are ignored.

DISTRIBUTION INSTRUCTIONS

The instructions outlined in this section are those that control the input and processing of data in the 500 distribution registers.

The 500 distribution registers are considered to have addresses in the range 000 to 499.

Instruction Clear Distribution Registers

Symbolic DCLR

Description The 500 distribution registers are set to zero. After clearing is completed, the program returns to the instruction following the DCLR command. A and B are destroyed.

Instruction Bring a Distribution Register

Symbolic DGET

Description The contents of register A are placed in register B. The contents of the distribution register whose address is stored in V07 are placed in register A. The "D" register is unchanged. The program returns to the instruction following the DGET command. The address of the specified "D" register must be placed in V07 prior to execution of the DGET command.

Example DGET

REGISTER	A	B	D05	V07
Before	+0000000123	+0000123456	+0000011111	+0000000005
After	+0000011111	+0000000123	+0000011111	+0000000005

Instruction Store to a Distribution Register

Symbolic DPUT

Description The contents of register A are stored in the "D" register specified by the address in V07. Registers A and B are unchanged.

Example DPUT

REGISTER	A	D05	V07
Before	+0000145111	-0000000112	+0000000005
After	+0000145111	+0000145111	+0000000005

Instruction Search for a Non-zero Value

Symbolic SCAN

Description The program sequentially searches the 500 distribution registers for a register containing a non-zero value. If a non-zero value is found, the value is placed in register A; the address of the "D" register containing the value is placed in V07; and control is returned to the instruction following the SCAN command. The programmer may then process the entry, place a new address in V07, and return to the SCAN instruction. When V07 becomes larger than 499, control returns to the second instruction following the SCAN command. Prior to the use of this command, V07 must contain the address of the first "D" register to be scanned. This command is ineffective when used as the fourth command in a program register.

Instruction Distribute

Symbolic DIST

Description This command will read a formatted data tape, select the proper address and amount fields, distribute the data to assigned distribution registers, and accumulate a sum of all distributed amounts in register V63. (V63 should be cleared prior to entry into this routine.) This command selects device 2 input only. It makes decisions on the selection of address and amount fields based on five values stored in advance by the programmer.

The locations and functions of these five values are as follows:

V01 Contains the start character of the address field to be selected on this pass. This start character is the identification code used by the program to accept or reject the field as a proper address field. Tab positions should be used as start characters (i.e., 1, 4, 7, 10....190).

V02 Contains the start character of the amount field to be selected on this pass. This start character is the identification code used by the program to accept or reject the field as a proper amount field. Tab positions should be used as start characters (i.e., 1, 4, 7, 10,....190).

V04 Contains the 6-digit base number for this pass. The base number represents the lowest address that will be accepted during the pass. For example, if the range of address values on the tape was 000 to 1999, register V04 would be set to zero for the first pass, and addresses 000 through 499 would be distributed. The value of V04 would then be changed to 500 for the second pass and addresses 500 through 999 would be distributed.

To illustrate this example:

	V04	Range of Distribution
First Pass	000	000 - 499
Second Pass	500	500 - 999
Third Pass	1000	1000 -1499
Fourth Pass	1500	1500 -1999

Other examples:

V04	Range of Distribution
600	600 - 1099
250	250 - 749
10	10 - 509
123400	123400 -123899

V05 Contains the mode switch. The setting of this mode switch permits the routine to distribute under two different tape formats.

Mode 1 - Controlled by setting the value in V05 to 1.

This mode causes the routine to scan the tape until an acceptable address field is found. An acceptable amount field is then sought. After the amount field is accepted and distributed, the routine will re-cycle to search for the next address field. Thus, Mode 1 searches the tape, alternately locating an address field and then an amount field.

Mode 2 - Controlled by setting the value in V05 to 2.

This mode looks for an acceptable address field, then an acceptable amount field. However, rather than re-cycling back to look for the next acceptable address field, the routine will look for either an address field or another amount field. As a result, multiple amount fields could be distributed before another acceptable address field is located.

TAPE FORMATS

Mode 1 - Address Field; Amount Field; Address Field; Amount Field; etc...

Mode 2 - Address Field; Amount Field; Amount Field; Amount Field; Address Field; Amount Field; Address Field; Amount Field; etc...

V06 Contains the edit word that defines the positioning of the digits of the address field. It is possible to have 10-digit address fields entered. However, six digits are the maximum number of digits that this routine uses to base its decision on acceptance of the address field. Therefore, the routine uses this edit word to determine which six digits in the field to use in judging acceptance.

The edit word is made up of octal edit codes with the following functions:

Octal Code	Function
0	Ignore the digit.
1	This digit is the one hundred thousands digit position of the field.
2	This digit is the ten thousands digit position of the field.
3	This digit is the thousands digit position of the field.
4	This digit is the hundreds digit position of the field.
5	This digit is the tens digit position of the field.
6	This digit is the units digit position of the field.
7	This code identifies the end code of the address field.

TO CONSTRUCT A DISTRIBUTION EDIT WORD

1. Determine the exact number of digits that make up the address field. (For example: assume an eight digit address field. ABCDEFGH.)

2. Write the address field down. A B C D E F G H (H is the units digit of the address field and A is the high-order digit).

3. Select the proper edit code for each digit position. Write these edit codes below the corresponding positions in the field. (Assume the six low-order digits are selected to be the significant distribution digits in the field)

```
A B C D E F G H
      1 2 3 4 5 6
```

4. If the address field contains digit-positions that will not affect distribution, assign ignore-codes (code zero) to these digit-positions. Place code 7 to the right of the field and fill in enough zeroes to the right of code 7 to develop a total of thirteen octal codes for the edit word. The edit word must contain 13 octal codes.

```
A B C D E F G H
0 0 1 2 3 4 5 6 7 0 0 0 0
```

Thus, the edit word 0012345670000 would be stored in V06.

NOTE: Address fields may contain symbols among the digits in the field (such as decimal point, comma, or slash). The field may also contain a sign at the end of the field. These symbol and sign positions must be considered as digit-positions, with an ignore-code assigned to them.

In the example above, if the eight-digit address field actually was:

```
ABCDEF.GH(S)          (S) = Sign
```

Then the edit word would be:

```
A B C D E F . G H (S)
0 0 1 2 3 4 0 5 6  0 7 0 0
```

If symbol and sign positions are not represented by ignore-codes in the edit word, they will be interpreted by the DIST command as illegal and will cause the program to HALT.

A COMPREHENSIVE USE OF THE DIST COMMAND

BACKGROUND: Consider a distribution tape that was punched while processing a billing application. The specifications call for the following analysis:

Sales (Dollars) by each of 400 products
Sales (Dollars) by each of 30 salesmen
Sales Tax (Dollars) by each of 25 states

DESIGNING THE SYSTEM: In designing the system, the following decisions were agreed upon:

1. Range of Distribution Fields:

   Products numbered 1150 through 1499 falling in the four high-order digits of a nine-digit field. (XXXX00000)

   Salesmen numbered 1 through 30 in a two-digit field. (XX)

   States numbered 1 through 25 falling in two low-order digits of a three-digit field. (0XX)

2. Start-of-Field Characters:

   Products (Tab 4)
   Salesmen (Tab 61)
   States (Tab 52)
   Sales (Dollars) (Tab 100)
   Sales Tax Dollars (Tab 103)

3. The Distribution tape contains the following sequence of fields for each invoice processed:

   Salesman Number (XX)
   Product Number (XXXX00000) }
   Sales (Dollars) (XXXXX.XX±) }
   Product Number (XXXX00000) } VARIABLE
   Sales (Dollars) (XXXXX.XX±) }
   Product Number (XXXX00000) }
   Sales (Dollars) (XXXXX.XX±) }
   State Code (0XX)
   State Sales Tax (XXX.XX±)

4. Sequence of Distribution analysis:

   Pass 1: Sales by Product
   Pass 2: Sales by Salesman
   Pass 3: Sales Tax by State

PROCESSING THE ANALYSIS:

Pass 1: Sales by Product

V01	Address Field Start Character	(4)
V02	Amount Field Start Character	(100)
V04	Base Number	(1150)
V05	Mode Switch	(1)
V06	Edit Word (Octal)	(3456000007000)

COMMENT: The DIST command will search the tape for a code corresponding to the tab position 4 code in V01. It will ignore all other codes. When the address field start-character is located, it will read in the nine-digit field immediately following the code and will pull out the four digits and arrange them in the order dictated by the edit word in V06.

The base number in V04 will be compared with this four-digit number. This number must equal the base number or fall between the base number and the base number plus 500. If it does, the base number in V04 is subtracted from the four-digit field. The difference represents the address of the register where the sales amount will be distributed.

The DIST command then searches the tape for the code corresponding to the tab position 100 code in V02 (Amount field start character). It will ignore all other codes. When the amount field start-character is located, the field is read and the amount is distributed.

Since the mode switch in V05 is set to Mode 1, the DIST command then searches for the next tab position 4 code (address field start character).

This sequence of operations is maintained until all sales dollars have been distributed. The DIST command affects input only. The printout of sales by product can then be custom-written.

NOTE: If the range of products had been 1150 through 1999, the DIST command would have only accepted address fields whose edited four digits would fall between 1150 and 1649. Any products numbered higher than 1649 would be distributed in another pass where the values in V01, V02, V05 and V06 would be the same, but the base number in V04 would be 1650.

Pass 2: Sales by Salesman

V01 Address Field Start Chqracter (61)
V02 Amount Field Start Character (100)
V04 Base Number (1)
V05 Mode Switch (2)
V06 Edit Word (Octal) (5670000000000)

COMMENT: The sequence of operations is the same as in pass 1. However, because V05 contained a mode 2, for each salesman address-field located, the DIST command would search for multiple amount fields (Sales Dollars) to distribute to that salesman's total.

Pass 3: Sales Tax by State

V01 Address Field Start Character (52)
V02 Amount Field Start Character (103)
V04 Base Number (1)
V05 Mode Switch (1)
V06 Edit Word (0567000000000)

COMMENT: The sequence of operations is the same as pass 1.

CONCLUSIONS: This illustration was used to demonstrate the flexibility of the edit word and the mode switch. It was not intended to be an example of a good system. With better specifications, the analysis in this example could be accomplished with one pass of the tape. The point is, the flexibility of the DIST command is available, but should be used wisely.

Instruction Load A Split Distribution Register

Symbolic SGET

Description The contents of the distribution register whose address is stored in V07 are placed in registers A and B in the following manner:

A Register - Contains the low order half of the distribution register with the correct sign.

B Register - Contains the high order half of the distribution register with the correct sign.

The highest decimal value permitted in a half of a register is 500,000±.

The previous contents of registers A and B are destroyed.

Example SGET

REGISTER	V07	D422		A	B
Before	+0000000422	+500000	-001234	+0000001111	+0000124760
After	+0000000422	+500000	-001234	-0000001234	+0000500000

Instruction Store a Split Distribution Register

Symbolic SPUT

Description The SPUT instruction combines the contents of registers A and B. The combined value is then stored in a distribution register specified by the address in V07.

The A register must contain the value and sign to be stored in the low-order half of the distribution register. The B register must contain the value and sign to be stored in the high-order half of the distribution register.

The values in the A and B registers must not exceed 500000±.

After processing the SPUT instruction, register A contains the packed value that was stored in the distribution register. The value in register B is unchanged.

Example SPUT

REGISTER	A	B	V07	D261
Before	+ +0000001236	-0000014210	+0000000261	-005332 ¦ +000975
After	-014210 ¦ +001236	-0000014210	+0000000261	-014210 ¦ +001236

SPECIAL INSTRUCTIONS

Instruction Program Interrupt

Symbolic OPUS

Description This command, when temporarily inserted into the coding of an application program, causes a jump out of the application program and into the Operating Utility System (OPUS). This command is not used in the normal processing of an application program, but is used primarily to aid in program debugging.

Instruction Calculate

Symbolic CALC

Description This command is made up of ten routines, each with the ability to be micro-programmed into different calculating sequences. It can contain Federal, State, and City tax calculations; Social Security and SUI calculations.

A more in-depth description of this command can be found in the literature on the Payroll Application.

Instruction Check-Digit Verification

Symbolic CDV

Description Many companies assign numbers to their customers which contain a check-digit in the low order digit position.

Credit card numbers generally contain a check-digit. One method of determining the check-digit assigned to a number is the "Modulo 10" method (also known as the LUHN algorithm). This is the method used by the CDV command.

This command verifies the value in register A for a valid check-digit. If the A register contained a valid number, upon exiting this command, the A register will be set to zero, and register B will contain decimal ten. If the A register contained an invalid number, upon exiting this command, the A register will contain a negative number, and the B register will contain a positive number that would make the original value in A a valid number.

Assume the value 610157802 is a valid number. The low order digit (2) is the check-digit.

REGISTER	A	B
Before	610157802	UNIMPORTANT
After	ZERO	10

Assume the value 610157803 is an invalid number.

REGISTER	A	B
Before	610157803	UNIMPORTANT
After	-1	9

The CDV command can also be used to generate a check-digit for a new number. This number must not exceed nine digits. For example, if it is desired to generate a check-digit for the value 830754926, add a zero to the low-order end (8307549260) and place in register A. The CDV command will place the valid check-digit in register B upon exiting the command.

REGISTER	A	B
Before	8307549260	UNIMPORTANT
After	-6	4

Thus, if the value in register A was saved, it can now be added to the check-digit in register B, producing the valid number 8307549264.

If, however, the valid check-digit is zero, the CDV command will clear register A, and place a decimal 10 in register B.

CONVERSION AND DUPLICATING INSTRUCTIONS

Three commands are provided by OPUS to facilitate the processing of data in codes other than the basic 1231 code:

DUPE - To duplicate data entered with even parity.

DUPO - To duplicate data entered with odd parity.

SPEC - To output any code without parity control.

To use the DUPE and DUPO commands, a conversion table must be constructed. During processing, this conversion table is stored in the first 128 D-registers. This table contains the code system for the input data and the converted codes to be output.

NOTE: To use the DUPE or DUPO commands in an application program, the mode key on the punch console must be set to conform to the parity of the desired output codes, that is -

Even Parity Output - Position #1

Odd Parity Output - Position #3

An input code can be handled in one of three ways:

1. It can be converted

   The code is entered and located in the conversion table. Its corresponding converted output code is then output to the devices selected by the application program. The next code is then entered.

2. It can be ignored

   The code is entered, ignored, and the next code is then entered.

3. It can be designated as an "exit" character

   An "exit" character when recognized on input will terminate the command and return control to the application program. Register A will contain a value that identifies the particular "exit" character that caused the command to terminate. This identifying character is determined at the time the table is being constructed. Thus, more than one code can be classified as an "exit" character,

with the application program determining which "exit" character was entered by the value placed in register A.

NOTE: Parity errors detected during the entry of data will be treated in the same manner as described under parity errors during DUP with the reader selected.

The DUPE and DUPO commands can do two basic operations; duplicate codes or skip fields.

1. Duplicate Codes:

   If the A-register contains a (-1) when DUPE or DUPO are executed, these commands will convert and duplicate or ignore codes until an "exit" character is recognized. Control then returns to the application program.

2. Skip Fields:

   If the A-register contains zero (0) when DUPE or DUPO are executed, the commands will act as a SKIP command and read and ignore input data until an "exit" character is recognized. Control then returns to the application program.

HOW TO CONSTRUCT A CONVERSION TABLE

A. EXPLANATION:

The construction of a conversion table has been simplified by the use of a program entitled "Conversion Table Service Routine". This program enables the programmer to:

1. Select one of four possible input/output parity conditions.

   odd in - even out
   odd in - odd out
   even in - odd out
   even in - even out

2. Select which characters are to be ignored.

3. Select which characters are to be converted.

4. Select which characters are to be "exit" characters.

NOTE: The DUPE command is used in an application program where even-parity input data is converted. The output can be even or odd parity. The DUPO command is used when odd-parity input data is converted. The output can be even or odd parity.

B. THE CODE CONVERSION CHART

Before constructing a conversion table, it is necessary to prepare a Code Conversion Chart, listing each input code that is not to be ignored and its corresponding output code. These codes must reflect the exact tape representation of each character in the input and output set, using ones to represent the holes and zeroes the absence of holes. The eighth channel of the tape is shown on the left. After the ones and zeroes are laid out for a code, these eight binary numbers are sectioned into three octal codes: (00 000 000). For example, the ASCII code for B is 01 000 010 or octal 102. The ASCII code for C is 11 000 011 or octal 303. In the 1231 code system, B is 01 100 010 or 142 while C is 01 110 011 or 163.

NOTE: It is mandatory that the parity channel of the tape be shown in its correct position.

The Code Conversion Chart must also contain the input codes designated as "exit" characters. The Output column of the chart for "exit" characters should reflect an "E" followed by a decimal number (0-999) reflecting the value of that exit character which will be placed in the A register on exiting the command when that character is recognized.

Once the Code Conversion Chart is completed, the Conversion Table can then be constructed.

C. TO CONSTRUCT A CONVERSION TABLE

The "Conversion Table Service Routine" is read into memory by depressing the P1 key. Also read in at this time is the Master Conversion Table. The Master Conversion Table uses 200 D-registers starting at D 200.

This table is used by the Service Routine to translate the octal representation of the input code into the correct address for the Conversion Table being constructed. The master table also translates the octal representation of the corresponding output code into the correct form for the 1231 processor.

1. TO IGNORE

Since the Conversion Table contains 128 possible codes to be converted, most code systems that will be converted allow for the ignoring of more than half of the system codes. This Service Routine, therefore, eliminates the necessity of entering each code to be ignored by enabling the programmer to begin constructing the table by ignoring all 128 possible codes. Thus, only the codes to be converted or designated as "exit" characters need be entered.

To "Ignore" the table, after Control IIII has been depressed:

Depress the Control I key. "IGNORE" will print.

Hold down the SHIFT key and depress the P4 key. The table will be set to the ignore condition.

2. TO SELECT THE PROPER ENTRY SET-UP

ENTER	THEN DEPRESS	SET-UP DESCRIPTION
1	CONTROL I	EVEN IN - ODD OUT
2	CONTROL I	ODD IN - EVEN OUT
3	CONTROL I	EVEN IN - EVEN OUT
4	CONTROL I	ODD IN - OUT OUT

3. Enter the three-digit octal representation of the input character. Terminate the entry as follows:

CONTROL I - To be converted and output.
"P" will print.
Enter the three-digit octal representation of the output code. Depress Control I.

CONTROL II - To be an "exit" character.
"E" will print.
Enter the three-digit decimal number that identifies this exit character. Depress Control I.

CONTROL III - To be ignored.
"I" will print.
No further entry is required.

4. Repeat Step 3 above for each character.

NOTE: 1. If an octal representation is entered higher than 377, "ILLEGAL" will print. Return to Step 3 above.

2. If the parity of the octal representation of either the input or output field is not correct according to the select code entered at Step 2 above, "PARITY" will print. Return to Step 3 above.

D. TO TEST THE TABLE:

1. Restart the program (HALT, READY, RUN CONTROL IIII).

2. Place a tape to be tested on the reader.

V. THE OPERATING UTILITY SYSTEM (OPUS)

INTRODUCTION

The previous section of this manual describes the 1231 command list. This section of the manual describes the various service routines available to the programmer and the operator.

When an instruction is used in an application program, a function-routine is activated to supply the internal coding necessary to make that instruction operate correctly. That function-routine, along with all the other function-routines in the EP/31 command list, is found in the section of memory known as the Operating Utility System (OPUS).

When a service routine is selected to provide assistance to the programmer in the creation and testing of an application program, that service routine is found in the Operating Utility System (OPUS).

When an operator is ready to process an application program, a service routine is available to process at the beginning of the application program. This service routine is found in the Operating Utility System (OPUS).

OPUS provides the following features of importance to the programmer and the operator:

1. OPUS is permanently stored in memory, making the service routines and function-routines available whenever required.

2. The programmer, when creating an application program, need only enter the symbolic instructions. Conversion to machine-language codes is handled internally by OPUS.

3. When OPUS senses an error during the processing of an application program, it will display a pattern of lights on the console to indicate the type of error. Thus, corrective action can be quickly taken, and processing can resume in a minimal amount of time.

OPUS provides service routines for manually entering, through the keyboard, program instructions and data into the 1231's memory; printing program instructions and data from the 1231's memory; punching program instructions and data from the 1231's memory onto paper tape; examining individual program instructions and data for possible change; reading of program instructions and data from punched paper tape into memory; and the execution of a stored application program.

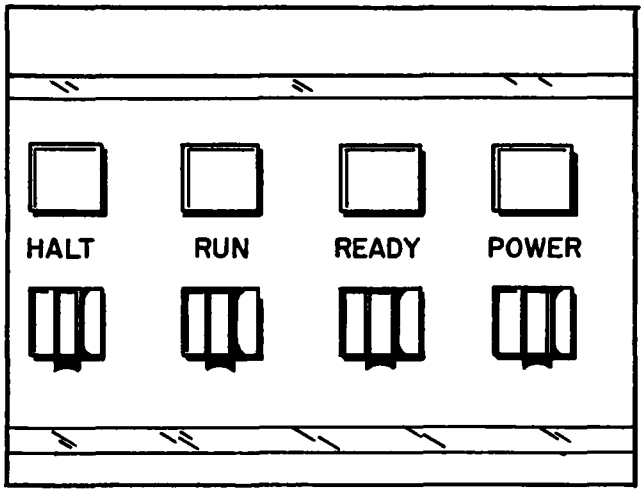
TO USE OPUS

Entry into OPUS is automatic once the EBS/1231 START-UP procedure or RESTART procedure is accomplished.

ART-UP

Depress the POWER button. (The light above the POWER and HALT buttons will light. When the memory drum reaches the proper speed for processing, the READY light will be lit.)

Depress (in this order) the HALT, READY, and RUN buttons. (The light above each button will light as it is depressed. The light above the HALT button will go out when the RUN button is depressed.)



EBS/1231 Console

The keyboard SELECTED light will begin flashing. Whenever the keyboard SELECTED light flashes, it is an indication that OPUS has been entered and is waiting for the depression of a selection key to indicate which OPUS service routine is desired.

RESTART

The RESTART procedure is used when power is already on and a return to OPUS is desired. The RESTART procedure follows the same procedures as steps 2 and 3 of the START-UP procedure.

RIPTION OF OPUS SERVICE ROUTINES

This section will list the name of the service routine; the selection key to depress to enter the service routine; and a description of how the routine operates.

These service routines can be grouped into three types of routines:

1. Program Creation Routines
2. Program Testing Routines
3. Program Operation Routines

1. Program Creation Routines

KEYBOARD KEYS	NAME AND DESCRIPTION
(R)	Reset Memory to the Origin-Pattern

This routine stores a unique pattern of bits in each program (P) and storage (V) register. These bit-patterns, or origin-patterns, do not resemble any program instructions or legal data. The origin-pattern is different in each register.

Since this routine destroys the previous contents of every register quickly, the depression of selection key R will not, by itself, activate this routine. When R is depressed, the word RESET will print. If it is truly desired to reset memory, the SHIFT key is held down with the left hand and the P4 key is depressed. The origin-pattern will then be established.

If selection key R was accidentally depressed, and it is not desired to reset memory, depress any key. ERR will print. Control will return to OPUS.

This routine <u>should</u> be selected prior to manually entering the instructions of a new application program. As instructions or data are entered, the origin-pattern in each used register is destroyed. This will indicate to OPUS which registers should be punched into the program tape, without the need for entering the beginning and ending register numbers.

(P) OR (V) Register Mode Control.

Prior to the selection of most service routines, the programmer must indicate which type of register (program or storage) the service routine is to affect. Once this indication is made, OPUS will maintain this mode until a different type of register is to be acted on. That is, a return to OPUS from a service routine will cause the same type of registers to be automatically selected for subsequent utility routines.

KEYBOARD KEYS	NAME AND DESCRIPTION
(O) OR (N)	Octal (O) or Decimal (N) Format.

If the register mode control (described previously) was set to storage (V) registers, the programmer has the option of entering or printing data in either octal (base 8) or decimal (base 10) format. Selection key O or N is depressed to indicate the format desired. Once this indication is made, OPUS will maintain the selected format until the other format is desired.

NOTE: OPUS monitors the use of all its routines. If it senses a violation to any rule governing its use, ERR will print and control will return to OPUS. The register mode control is automatically set to handle "P" registers and the format is set to octal (O).

(S) Store Instructions or Data.

This routine allows for the keyboard entry of symbolic program instructions into "P" registers, and data into "V" registers.

R = RESET
SHIFT + P4 (P REG.)

P-REGISTER ENTRY

5000 # SEC SUB 33 I
DUP SUB 001 I
CO SUB 34 I
VUP SUB 00 ←

1. Enter the numeric address (000-127) of the register whose instructions are to be keyed-in. The digits will print as they are entered.
2. Depress "#" key. This key indicates the end of an address entry. The # symbol prints, followed by a tab to position 19.
3. Enter the operation (symbolic) code of the instruction. The letters print as they are entered.
4. Depress the SPACE bar. This key indicates the end of the operation code.
5. Enter the additional code (if applicable) for the instruction. The additional code prints as it is entered.

6. Depress one of the following keys:

[I]

If another instruction is to be keyed into this same register. (The printer will line feed and tab to position 19 awaiting the entry of the next instruction.) Return to Step 3.

If this is the 4th line in the register, the printer will line feed, print an S, print out the address of the next register, print #, and tab to position 19 awaiting the entry of the first instruction. Return to Step 3.

[II]

This key stores the instructions into the specified register, line feeds, prints an S, and prints out the address of the next register. It then tabs to position 19 awaiting entry of the first instruction. Return to Step 3.

If this key is used and this was not the 4th instruction in the register, the unused instruction areas will be filled with automatic jumps before being stored into memory.

[|←]

If no more instructions are to be keyed-in and a return to OPUS is desired. Any unused instruction areas in this register will be filled with automatic jumps prior to storing the register. Control will return to OPUS.

NOTE: If "ERR" prints, the entire register's contents must be re-entered. Probably an operation code or additional code was not within the required limits monitored by OPUS. Control returns to OPUS.

NAME AND DESCRIPTION

V-REGISTER ENTRY

1. Be certain the correct format (octal or decimal) is entered. The only legal digits for entering data in the octal format are digits 0 through 7. This format is primarily used for edit words. It may be used for positive constants but it must not be used for negative constants.

   In the decimal format, only digits 0 through 9 are legal. This format should be used for both positive and negative constants.

2. Enter the numeric address (00-63) of the register whose data is to be keyed-in. The digits will print as they are entered.

3. Depress the # key. The # symbol will print followed by a tab to position 25.

4. Enter the data. Each digit prints as it is entered. When entering constant data (either positive or negative) only the significant digits need be entered. OPUS will fill the positions to the left of the digits with zeroes. If a storage register is to contain all zeroes, one zero must be entered. When a negative value is entered, the minus sign may be entered any time prior to the entry of the end code. If a mistake is made while entering data in either the octal or decimal format, depress the HALT, READY, and RUN buttons. Control returns to OPUS. The routine can then be re-selected.

5. Depress one of the following end keys:

   [I]

   The contents of this register are stored in memory, the printer will line feed, S will print, the address of the next storage register will print followed by a tab to position 25 awaiting the entry of data to the next register. Return to step 4, above.

The contents of this register are stored in memory, and control returns to OPUS.

NOTE: If "ERR" prints, an illegal digit or key was depressed. Control returns to OPUS. The data must be re-entered.

KEYBOARD KEY | NAME AND DESCRIPTION

Print Out Program or Storage Registers.

This routine prints out the contents of the specified registers. The printout of program registers is in the same symbolic form that it was entered. The printout of storage registers can be in the octal or decimal format selected.

TO PRINT OUT REGISTERS

1. Be certain the correct register mode is selected (selection key P or V). Then select the W key (print out routine).

2. Enter the numeric address of the first register to be printed. (Program registers 000-127; Storage registers 00-63.) The digits print as they are entered.

3. Depress the # key. (# prints.)

4. Enter the numeric address of the last register to be printed. The digits print as they are entered.

5. Depress the # key (# prints, followed by a line feed, a tab to position 1, and one space. The address of the first register prints, # prints, and a tab to position 25). The contents of the register prints out. This is repeated for each register in the range specified. When the contents of the last register have been printed, control returns to OPUS.

The format of an octal printout is:
X XXX XXX XXX XXX
The format of a decimal printout is:
XXX.XXX.XXX.XXX

NOTE: During the printout of P-registers, when OPUS recognizes the origin-pattern as the contents of the program register, the register address is printed, followed by the printing of the word "empty". The printout continues to the next register address. If the origin-pattern is recognized while printing storage registers, the pattern is printed octally or decimally depending on the mode selected.

KEYBOARD KEYS	NAME AND DESCRIPTION
(L)	Punch Tape Leader. This routine punches about four inches of tape leader automatically.
(T)	Punch Program Into Tape. This routine will punch, in sequence, the contents of all program and storage registers which vary from their origin-pattern. Only those registers which contain program instructions and data will be punched into tape.
(P2)	Verification of Program Tape. With the newly created program tape on the reader (device 2 input), this routine will compare the contents of the registers punched in tape with the corresponding contents in memory. If a discrepancy is found, "COMP ERR" will print and control will return to OPUS. The tape should be thrown away and another one punched.

2. Program Testing Routines

KEYBOARD KEY	NAME AND DESCRIPTION
@	Change the Contents of a Register.

This routine allows the printing of the contents of a specific program or storage register. The option is then available to change or accept the printed contents. The contents of program registers will be in symbolic format. The contents of storage registers will be either octal or decimal depending on the format selected.

TO PRINT A PROGRAM REGISTER

1. Enter the numeric address (000-127) of the register to be printed. The digits print as they are entered.

2. Depress the # key. (# prints, followed by a tab to position 19.)

3. The first instruction in the register is printed, followed by a tab to position 37.

4. If the instruction is acceptable, depress one of the following:

   I

   The next instruction will print.

   II

   The contents of this register will be stored in memory. The first instruction of the next register will print.

   ←

   The contents of this register will be stored in memory and control will return to OPUS.

NAME AND DESCRIPTION

5. If the instruction is to be changed, enter the operation code; enter a space; enter the additional code (if applicable) enter either the control I, control II or return key as described in step 4, above.

TO PRINT A STORAGE REGISTER

1. Enter the numeric address (00-63) of the register to be printed. The digits print as they are entered.

2. Depress the # key. (# prints, followed by a tab to position 25.)

3. The data print, followed by a tab to position 37.

4. If the data are acceptable, depress one of the following keys:

The contents of this register will be stored in memory and the data in the next register will print.

The contents of this register will be stored in memory and control will return to OPUS.

5. If the data are to be changed, enter the new data. The digits print as they are entered.

6. Depress the control I or the return key as described in step 4, above.

NOTE: If a register will no longer be used by the application program and it is desirable to delete it from the next program tape to be punched, refer to the origin pattern list in the appendix to find out what origin pattern to enter. Setting the register back to its own origin-pattern indicates to OPUS not to punch it in tape. Setting a storage register to zero will not keep it from being punched to tape.

KEYBOARD KEYS	NAME AND DESCRIPTION
(X)	Print Out Register A. This routine will print out the contents of register A with a decimal format.
(Y)	Print Out Register B. This routine will print out the contents of register B with a decimal format.

NOTE: When testing a program, it may be desirable to print out the contents of register A (X); register B (Y); or a storage register (Q) immediately after an arithmetic function has been performed. The special instruction "OPUS" may be substituted for a program instruction which immediately follows the arithmetic function. When the program reaches the instruction "OPUS", control will return to OPUS. The appropriate test can then be selected.

3. Program Operation Routines

KEYBOARD KEYS	NAME AND DESCRIPTION
(PI)	Read Program Tape Into Memory. This routine will read a program tape and store the instructions and data in the P and V registers.
IIII	To Process an Application Program. This routine will transfer control out of OPUS and to the first instruction in P00. The 1231 will then begin processing the application program.

NOTE: During the testing of an application program, if the program executes a jump to a program register which contains the origin-pattern, "empty ERR" prints, and control returns to OPUS. The error can then be corrected.

SUMMARY

Sequence of operations to create, test and operate an application program:

1. Enter OPUS via start-up restart procedure.
2. Reset memory - key R, then shift key and P4 key.
3. Select register mode if not already selected. Key P, for program registers; key V, for storage registers.
4. Select format (octal or decimal) if key V was entered and the desired format was not previously selected. Key O for octal format; key N for decimal format.
5. Select the STORE routine. Key S.
6. Enter the numeric address and the # key of the desired register.
7. Enter the symbolic program instructions or data as outlined in the description of the STORE routine.
8. Continue to store instructions and/or data until the RETURN key is depressed and control is returned to OPUS.
9. Printout P and V registers for verification of correct entry. Key W.
10. Punch Tape Leader - key L.
11. Punch out program tape - key T.
12. Punch Tape Leader at the end of the tape - key L.
13. Verify the punched tape - key P2.
14. Test the program using key Q for register changes; key W for V-register printout; key X and Y for registers A and B printout.
15. Operate the program -- Using key P1 to read the program into memory; and control key IIII to pass control to the first instruction in P00 to begin processing the application program.

SECTION VI

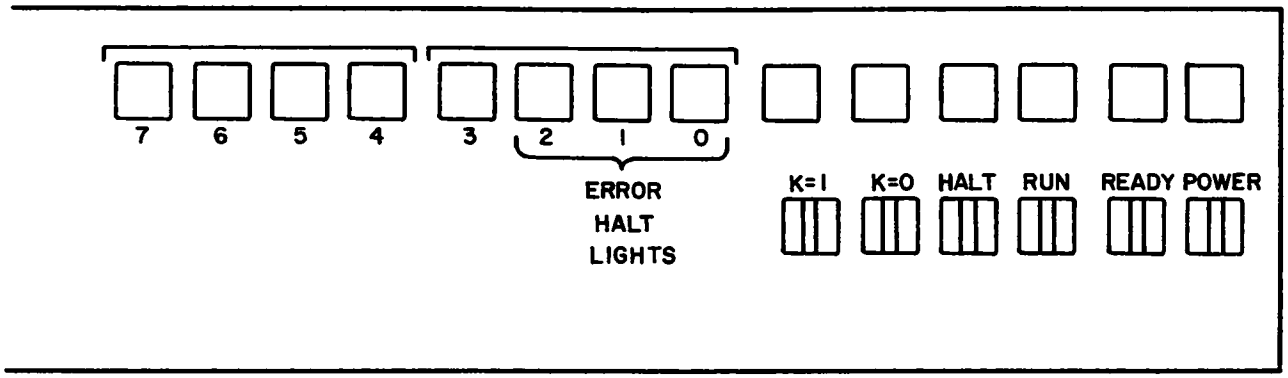
ERROR HALTS

VI. ERROR HALTS

PARITY AND OTHER ERRORS

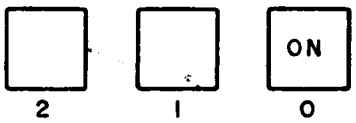
The various EP/31 instructions contain controls that check for odd parity on input and the entry of an excessive number of digits on input. There are also input controls on reading and distributing a data tape.

The EBS/1231 will automatically halt the program when one of these controls is violated. A pattern of lights will be displayed on the 1231 console indicating the reason for the halt. The following is the 1231 console panel of lights and switches.



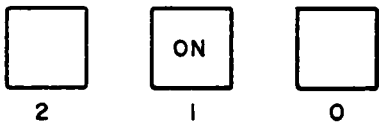
When the program halts, the RUN light will go out. Lights numbered 3 through 7 will go out; and a variable pattern of lights (numbered 0 through 2) will appear lit on the console.

The lights numbered 0 through 2 will be lit in the following pattern:

LIGHT PATTERN	REASON FOR HALT
	Parity error or an excessive number of digits entered in the input command. (Keyboard selected.)

CORRECTION PROCEDURE

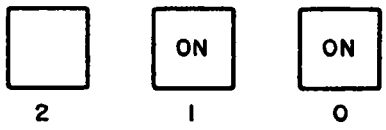
1. Depress keyboard RELEASE button.
2. Depress the CLEAR key.
3. Depress the RUN button on the console.
4. Re-enter the entire field.



Parity error on the SCI command. (Key board selected.)

CORRECTION PROCEDURE

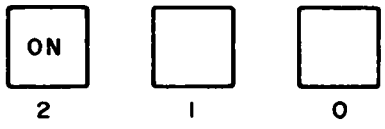
1. Depress keyboard RELEASE button.
2. Depress the RUN button on the console.
3. Re-enter the character.



Parity error detected during keyboard entry of alpha-numeric data. (DUP instruction.)

CORRECTION PROCEDURE

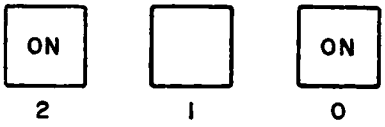
1. Depress keyboard RELEASE button.
2. Depress the RUN button on the console.
3. Re-enter the character.



A parity error was detected while reading a numeric field from tape.

CORRECTION PROCEDURE

1. Depress the REVERSE FEED button on the reader once.
2. Depress the RUN button on the console.
3. If the error halt is repeated, a parity error (even parity) is actually present in the tape. Refer to the error correction procedures of the application program involved.



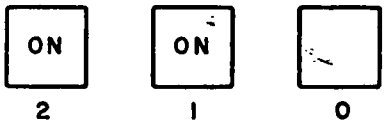
An excessive number of digits were read from tape during the execution of the input command.

CORRECTION PROCEDURE

If both keyboard and reader are selected, make the correct entry through the keyboard as follows:

1. Open the reader tight-tension switch.
2. Move the tape in the reader so that the sensing pins are under the first code of the next field in the tape.
3. Depress the CLEAR key on the keyboard.
4. Depress the RUN button on the console.
5. Manually re-enter the field.
6. Close the tight-tension switch on the reader. Processing will continue.

If the keyboard is not selected, refer to the error correction procedures of the application program involved.

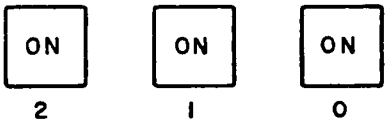


A parity error was detected while reading a SCI character or while reading an alpha-numeric field from tape (DUP instruction).

CORRECTION PROCEDURE

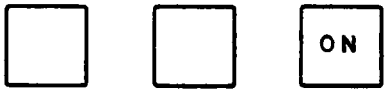
Re-read the character as follows:

1. Depress the REVERSE FEED button on the reader once.
2. Depress the RUN button on the console.
3. If the error halt is repeated, a parity error is present in the tape. Refer to the error correction procedures of the application program involved.



This error halt designates a control was violated while executing a distribution instruction. To determine the specific error involved, depress the HALT button on the console once. A new display of lights will appear. Refer to the section on Distribution Error Halts for the specific error and correction procedure.

DISTRIBUTION ERRORS



The DIST instruction has detected one of two errors:

1. A non-numeric character was detected in an address field that did not have an ignore code in the corresponding digit-position of the distribution edit word.

2. An end code in the distribution edit word was processed, and there was no end character in the corresponding position in tape.

CORRECTION PROCEDURE

1. Depress the STEP REVERSE switch on the reader once.

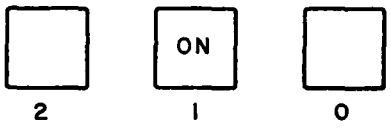
2. Depress the RUN button on the console. The character on tape will be re-read. If the character was accepted this time, processing will continue.

   If the character was in error again, processing will halt.

   A) Mark the tape so that it can be checked after the run.

   B) Depress the K=1 switch on the console.

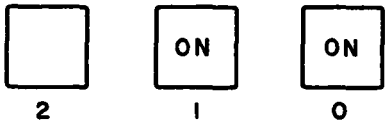
   C) Depress the RUN button on the console. The program will search the tape for the start code of the next address field.



The DIST instruction has detected a parity error (even parity in the tape).

CORRECTION PROCEDURE

Same as the correction procedure for the Error Halt on the preceding page.



The DGET or DPUT instruction found a D-register address outside the range of 000 to 499 in register V07.

CORRECTION PROCEDURE

Depress the RUN button on the console. Control returns to OPUS.

OUTPUT PARITY ERROR

When the EBS/1231 recognizes the output of EVEN parity (parity error), the following sequence occurs:

1. The PARITY LIGHT on the Model 70 Punch comes ON.
2. Processing halts.

CORRECTION PROCEDURE

1. Depress the RAPID ADVANCE SWITCH. Produce enough tape leader so that a proper identification of the error can be written on the tape.
2. Label the tape with a reference to the record being processed.

   EXAMPLE: Employee number, Invoice number, Product number, etc.

3. Open the TAPE-TENSION SWITCH on the punch.
4. Turn the MODE KEY on the punch to the middle position. The PARITY LIGHT goes OFF.
5. Turn the MODE KEY back to the right position.
6. Close the TAPE-TENSION SWITCH on the punch. Processing continues.

SECTION VII

APPENDICES

APPENDIX I

EDIT WORD FORMATS

FORMAT	EDIT WORD CONSTANT
X	055-555-555-561
XX	055-555-555-661
XXX	055-555-556-661
XXXX	055-555-566-661
XXXXX	055-555-666-661
XXXXXX	055-556-666-661
XXXXXXX	055-566-666-661
XXXXXXXX	055-666-666-661
XXXXXXXXX	056-666-666-661
XXXXXXXXXX	066-666-666-661
X.XX	055-555-556-361
XX.XX	055-555-566-361
XXX.XX	055-555-666-361
X,XXX.XX	055-556-466-361
XX,XXX.XX	055-566-466-361
XXX,XXX.XX	055-666-466-361
X,XXX,XXX.XX	056-466-466-361
XX,XXX,XXX.XX	066-466-466-361
XX/XX/XX	055-556-626-261

All edit words shown contain leading spaces, no sign, and no field code.

APPENDIX II

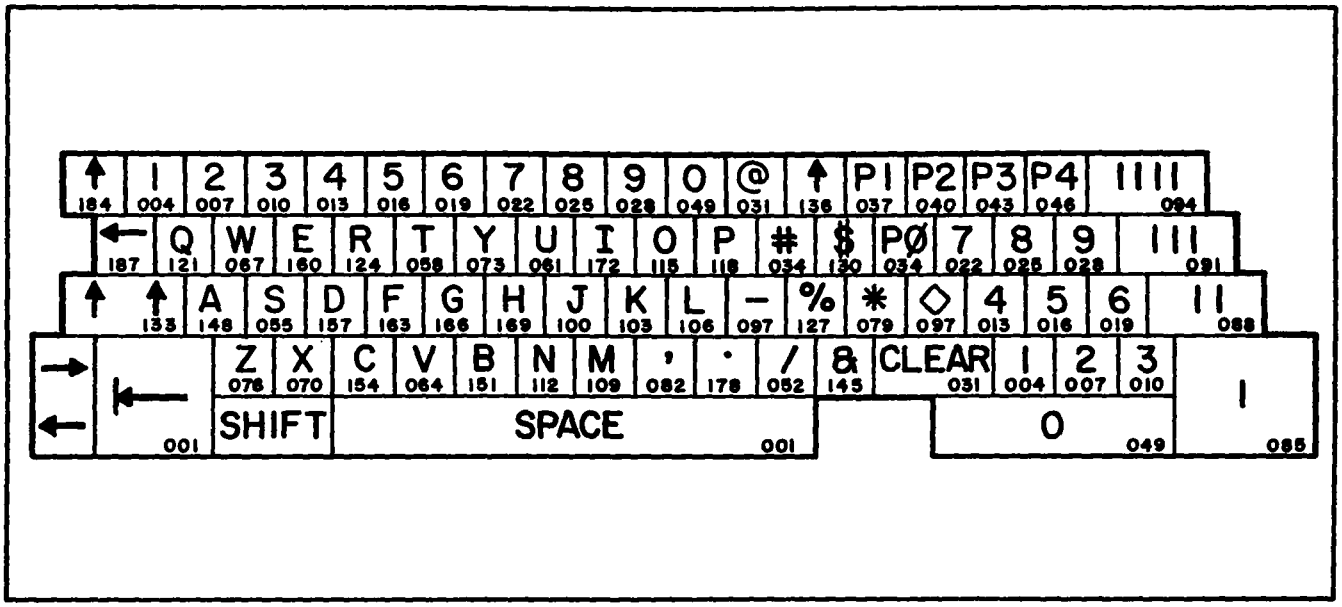
TABLE OF CHARACTER OUTPUT CODES

CHARACTER AND CODE		KEY AND CODE		SPECIAL CHAR. AND CODES	
A	061	P0	013	space	000
B	062	P1	014	*	032
C	063	P2	015	,	033
D	064	P3	016	/	021
E	065	P4	017	@	012
F	066			#	013
G	067			-	040
H	070	I	034	%	052
I	071	II	035	$	053
J	041	III	036	&	060
K	042	IIII	037	.	073
L	043				072
M	044			CLR	012
N	045				
O	046				
P	047				
Q	050				
R	051				
S	022				
T	023				
U	024				
V	025				
W	026				
X	027				
Y	030				
Z	031				
0	020				
1	001				
2	002				
3	003				
4	004				
5	005				
6	006				
7	007				
8	010				
9	011				

FUNCTIONS	CODES
Line Feed Left	075
Line Feed Right	055
Line Feed Both	054
Form-Up	057
Black Ribbon	056
Red Ribbon	074
Backspace	076
Carriage (Open,Close)	077

APPENDIX III

MODEL 11 KEYBOARD LAYOUT



NOTE: Subscripted numbers indicate the print head position obtained when "SHIFT" and the selected key are depressed. The following print head positions cannot be obtained from the keyboard:
139, 142, 175, 181, and 190.

APPENDIX IV

ORIGIN-PATTERNS FOR P AND V REGISTERS

There will be times when it is desired to reset selected program or storage registers to their origin-pattern, without resetting the entire memory. This is desirable, because setting unused registers back to their origin-patterns reduces the length of a punched program tape.

The following are the steps for re-setting origin-patterns:

1. Depress the HALT, READY and RUN buttons on the console. Control is transferred to OPUS.

2. Depress the "H" key on the keyboard.

3. Depress the "S" key on the keyboard.

4. Refer to the P or V origin-pattern lists.

   a. Find the line containing the register address to be reset.

   b. Enter the three-character internal address through the keyboard. (Do not enter the register address.)

   c. Depress the # key on the keyboard.

   d. Enter the eight character origin-pattern.

   e. Enter the end code as follows:

I	The origin pattern is stored in the specified register. The internal address of the next register prints. Return to step 4d above.
⇤	The origin-pattern is stored in the specified register. Control returns to OPUS. If no other registers are to be reset, depress the HALT, READY and RUN buttons on the console.

APPENDIX IV (CONTINUED)

P - REGISTER ORIGIN-PATTERNS

P REG. ADR.	INTERNAL ADR.	ORIGIN PATTERN	P REG. ADR.	INTERNAL ADR.	ORIGIN PATTERN	P REG. ADR.	INTERNAL ADR.	ORIGIN PATTERN
000	300	07 07 E300	043	32B	07 07 E32B	086	356	07 07 E356
001	301	07 07 E301	044	32C	07 07 E32C	087	357	07 07 E357
002	302	07 07 E302	045	32D	07 07 E32D	088	358	07 07 E358
003	303	07 07 E303	046	32E	07 07 E32E	089	359	07 07 E359
004	304	07 07 E304	047	32F	07 07 E32F	090	35A	07 07 E35A
005	305	07 07 E305	048	330	07 07 E330	091	35B	07 07 E35B
006	306	07 07 E306	049	331	07 07 E331	092	35C	07 07 E35C
007	307	07 07 E307	050	332	07 07 E332	093	35D	07 07 E35D
008	308	07 07 E308	051	333	07 07 E333	094	35E	07 07 E35E
009	309	07 07 E309	052	334	07 07 E334	095	35F	07 07 E35F
010	30A	07 07 E30A	053	335	07 07 E335	096	360	07 07 E360
011	30B	07 07 E30B	054	336	07 07 E336	097	361	07 07 E361
012	30C	07 07 E30C	055	337	07 07 E337	098	362	07 07 E362
013	30D	07 07 E30D	056	338	07 07 E338	099	363	07 07 E363
014	30E	07 07 E30E	057	339	07 07 E339	100	364	07 07 E364
015	30F	07 07 E30F	058	33A	07 07 E33A	101	365	07 07 E365
016	310	07 07 E310	059	33B	07 07 E33B	102	366	07 07 E366
017	311	07 07 E311	060	33C	07 07 E33C	103	367	07 07 E367
018	312	07 07 E312	061	33D	07 07 E33D	104	368	07 07 E368
019	313	07 07 E313	062	33E	07 07 E33E	105	369	07 07 E369
020	314	07 07 E314	063	33F	07 07 E33F	106	36A	07 07 E36A
021	315	07 07 E315	064	340	07 07 E340	107	36B	07 07 E36B
022	316	07 07 E316	065	341	07 07 E341	108	36C	07 07 E36C
023	317	07 07 E317	066	342	07 07 E342	109	36D	07 07 E36D
024	318	07 07 E318	067	343	07 07 E343	110	36E	07 07 E36E
025	319	07 07 E319	068	344	07 07 E344	111	36F	07 07 E36F
026	31A	07 07 E31A	069	345	07 07 E345	112	370	07 07 E370
027	31B	07 07 E31B	070	346	07 07 E346	113	371	07 07 E371
028	31C	07 07 E31C	071	347	07 07 E347	114	372	07 07 E372
029	31D	07 07 E31D	072	348	07 07 E348	115	373	07 07 E373
030	31E	07 07 E31E	073	349	07 07 E349	116	374	07 07 E374
031	31F	07 07 E31F	074	34A	07 07 E34A	117	375	07 07 E375
032	320	07 07 E320	075	34B	07 07 E34B	118	376	07 07 E376
033	321	07 07 E321	076	34C	07 07 E34C	119	377	07 07 E377
034	322	07 07 E322	077	34D	07 07 E34D	120	378	07 07 E378
035	323	07 07 E323	078	34E	07 07 E34E	121	379	07 07 E379
036	324	07 07 E324	079	34F	07 07 E34F	122	37A	07 07 E37A
037	325	07 07 E325	080	350	07 07 E350	123	37B	07 07 E37B
038	326	07 07 E326	081	351	07 07 E351	124	37C	07 07 E37C
039	327	07 07 E327	082	352	07 07 E352	125	37D	07 07 E37D
040	328	07 07 E328	083	353	07 07 E353	126	37E	07 07 E37E
041	329	07 07 E329	084	354	07 07 E354	127	37F	07 07 E37F
042	32A	07 07 E32A	085	355	07 07 E355			

APPENDIX IV (CONTINUED)

V- REGISTER ORIGIN-PATTERNS

V REG. ADR.	INTERNAL ADR.	ORIGIN PATTERN	V REG. ADR.	INTERNAL ADR.	ORIGIN PATTERN
000	380	07 07 E380	032	3A0	07 07 E3A0
001	381	07 07 E381	033	3A1	07 07 E3A1
002	382	07 07 E382	034	3A2	07 07 E3A2
003	383	07 07 E383	035	3A3	07 07 E3A3
004	384	07 07 E384	036	3A4	07 07 E3A4
005	385	07 07 E385	037	3A5	07 07 E3A5
006	386	07 07 E386	038	3A6	07 07 E3A6
007	387	07 07 E387	039	3A7	07 07 E3A7
008	388	07 07 E388	040	3A8	07 07 E3A8
009	389	07 07 E389	041	3A9	07 07 E3A9
010	38A	07 07 E38A	042	3AA	07 07 E3AA
011	38B	07 07 E38B	043	3AB	07 07 E3AB
012	38C	07 07 E38C	044	3AC	07 07 E3AC
013	38D	07 07 E38D	045	3AD	07 07 E3AD
014	38E	07 07 E38E	046	3AE	07 07 E3AE
015	38F	07 07 E38F	047	3AF	07 07 E3AF
016	390	07 07 E390	048	3B0	07 07 E3B0
017	391	07 07 E391	049	3B1	07 07 E3B1
018	392	07 07 E392	050	3B2	07 07 E3B2
019	393	07 07 E393	051	3B3	07 07 E3B3
020	394	07 07 E394	052	3B4	07 07 E3B4
021	395	07 07 E395	053	3B5	07 07 E3B5
022	396	07 07 E396	054	3B6	07 07 E3B6
023	397	07 07 E397	055	3B7	07 07 E3B7
024	398	07 07 E398	056	3B8	07 07 E3B8
025	399	07 07 E399	057	3B9	07 07 E3B9
026	39A	07 07 E39A	058	3BA	07 07 E3BA
027	39B	07 07 E39B	059	3BB	07 07 E3BB
028	39C	07 07 E39C	060	3BC	07 07 E3BC
029	39D	07 07 E39D	061	3BD	07 07 E3BD
030	39E	07 07 E39E	062	3BE	07 07 E3BE
031	39F	07 07 E39F	063	3BF	07 07 E3BF

APPENDIX V

EBS/1231 SYSTEM CODE CHART

TAPE CHARACTER								INTERNAL CODE	KEYBOARD CHARACTER	PRINTER CHARACTER	PRINT WHEEL POSITION	FUNCTION
1	2	3	4	5	6	7	8					
				x				000	Space	Space		
x								001	1	1		
	x							002	2	2		
x	x			x				003	3	3		
		x						004	4	4		
x		x		x				005	5	5		
	x	x		x				006	6	6		
x	x	x						007	7	7		
			x					010	8	8		
x			x	x				011	9	9		
	x		x	x				012	@Clear	@		
x	x		x					013	#P0	#		
		x	x	x				014	P1			
x		x	x					015	P2			
	x	x	x					016	P3			
x	x	x	x	x				017	P4			
					x			020	0	0		Numeric Zero
x				x	x			021	/	/		
	x			x	x			022	S	S		
x	x				x			023	T	T		
		x		x	x			024	U	U		
x		x			x			025	V	V		
	x	x			x			026	W	W		
x	x	x		x	x			027	X	X		
			x	x	x			030	Y	Y		
x			x		x			031	Z	Z		
	x		x		x			032	*	*		
x	x		x	x	x			033	,	,		
		x	x		x			034	I			
x		x	x	x	x			035	II			
	x	x	x	x	x			036	III			
x	x	x	x		x			037	IIII			
						x		040	◇	-		
x				x		x		041	J	J		
	x			x		x		042	K	K		
x	x					x		043	L	L		
		x		x		x		044	M	M		
x		x				x		045	N	N		
	x	x				x		046	O	O		
x	x	x		x		x		047	P	P		
			x	x		x		050	Q	Q		
x			x			x		051	R	R		
	x		x			x		052	%	%		

APPENDIX V (CONTINUED)

EBS/1231 SYSTEM CODE CHART (CONTINUED)

TAPE CHARACTER 1	2	3	4	5	6	7	8	INTERNAL CODE	KEYBOARD CHARACTER	PRINTER CHARACTER	PRINT WHEEL POSITION	FUNCTION
x	x		x	x		x		053	$	$		
		x	x			x		054	↑↑			Index Left + Right
x		x	x	x		x		055	↑(R)			Index Right
	x	x	x	x		x		056				Black Ribbon Print
x	x	x	x			x		057				Formup
				x	x	x		060	&	&		
x					x	x		061	A	A		
	x				x	x		062	B	B		
x	x			x	x	x		063	C	C		
		x			x	x		064	D	D		
x		x		x	x	x		065	E	E		
	x	x		x	x	x		066	F	F		
x	x	x			x	x		067	G	G		
			x		x	x		070	H	H		
x			x	x	x	x		071	I	I		
	x		x	x	x	x		072				
x	x		x		x	x		073	.	.		
		x	x	x	x	x		074				Red Ribbon Print
x		x	x		x	x		075	↑(L)			Index Left
	x	x	x		x	x		076	←			Backspace
x	x	x	x	x	x	x		077				Carriage Open or Close
							x	100	⇤		1	Return
x				x			x	101	SHIFT 1		4	
	x			x			x	102	SHIFT 2		7	
x	x						x	103	SHIFT 3		10	
		x		x			x	104	SHIFT 4		13	
x		x					x	105	SHIFT 5		16	
	x	x					x	106	SHIFT 6		19	
x	x	x		x			x	107	SHIFT 7		22	
			x	x			x	110	SHIFT 8		25	
x			x				x	111	SHIFT 9		28	
	x		x				x	112	SHIFT @ or Clear		31	
x	x		x	x			x	113	SHIFT # or P0		34	
		x	x				x	114	SHIFT P1		37	
x		x	x	x			x	115	SHIFT P2		40	
	x	x	x	x			x	116	SHIFT P3		43	
x	x	x	x				x	117	SHIFT P4		46	
				x	x		x	120	SHIFT 0		49	Numeric Zero
x					x		x	121	SHIFT /		52	
	x				x		x	122	SHIFT S		55	
x	x			x	x		x	123	SHIFT T		58	
		x			x		x	124	SHIFT U		61	
x		x		x	x		x	125	SHIFT V		64	

APPENDIX V (CONTINUED)

EBS/1231 SYSTEM CODE CHART (CONTINUED)

TAPE CHARACTER 1	2	3	4	5	6	7	8	INTERNAL CODE	KEYBOARD CHARACTER	PRINTER CHARACTER	PRINT WHEEL POSITION	FUNCTION
	x	x		x	x		x	126	SHIFT W		67	
x	x	x			x		x	127	SHIFT X		70	
			x		x		x	130	SHIFT Y		73	
x			x	x	x		x	131	SHIFT Z		76	
	x		x	x	x		x	132	SHIFT *		79	
x	x		x		x		x	133	SHIFT ,		82	
		x	x	x	x		x	134	SHIFT I		85	
x		x	x		x		x	135	SHIFT II		88	
	x	x	x		x		x	136	SHIFT III		91	
x	x	x	x	x	x		x	137	SHIFT IIII		94	
				x		x	x	140	SHIFT-or ◇		97	
x						x	x	141	SHIFT J		100	
	x					x	x	142	SHIFT K		103	
x	x			x		x	x	143	SHIFT L		106	
		x				x	x	144	SHIFT M		109	
x		x		x		x	x	145	SHIFT N		112	
	x	x		x		x	x	146	SHIFT O		115	
x	x	x				x	x	147	SHIFT P		118	
			x			x	x	150	SHIFT Q		121	
x			x	x		x	x	151	SHIFT R		124	
	x		x	x		x	x	152	SHIFT %		127	
x	x		x			x	x	153	SHIFT $		130	
		x	x	x		x	x	154	SHIFT ▲▲		133	Index Left & Right
x		x	x			x	x	155	SHIFT ▲ (R)		136	Index Right
	x	x	x			x	x	156			139	
x	x	x	x	x		x	x	157			142	
					x	x	x	160	SHIFT &		145	
x				x	x	x	x	161	SHIFT A		148	
	x			x	x	x	x	162	SHIFT B		151	
x	x				x	x	x	163	SHIFT C		154	
		x		x	x	x	x	164	SHIFT D		157	
x		x			x	x	x	165	SHIFT E		160	
	x	x			x	x	x	166	SHIFT F		163	
x	x	x		x	x	x	x	167	SHIFT G		166	
			x	x	x	x	x	170	SHIFT H		169	
x			x		x	x	x	171	SHIFT I		172	
	x		x		x	x	x	172			175	
x	x		x	x	x	x	x	173	SHIFT .		178	
		x	x		x	x	x	174			181	
x		x	x	x	x	x	x	175	SHIFT ▲ (L)		184	Index Left
	x	x	x	x	x	x	x	176	SHIFT ←		187	Backspace
x	x	x	x		x	x	x	177			190	

OPUS EP/31 SERVICE ROUTINES

Key

R "RESET" sh+P4

?/V Selects STORAGE REG.

/▲ OCTAL / DECIMAL (STORAGE REGISTER DATA)

S STORE: AAA # OP▲AD { I SEQUENTIAL INPUT; II NEXT REGISTER (P-REGS ON.; ← OPUS
AA # -DDDDDDDD-

W WRITE: FIRST LAST
AAA#AAA#
AA #AA#

L LEADER, PAPER TAPE.

T TAPE PROGRAM-from STORAGE

P2 VERIFY ABOVE ON (ch.2) READER "COMP ERR", if defective.

Q ALTER: AAA# "OP▲AD" {OP▲AD} { I; II; ← }
AA# "DATA" {DATA}

X PRINTS A_{10}

Y PRINTS B_{10}

P1 READ PROG TAPE INTO MEMORY

~~JMP~~ JUMP TO P000, + EXECUTE.

TO INITIALIZE TO OPUS:

POWER; ("READY"), HALT, READY, RUN. (KEYBOARD SEL flashing

STERS:

WORKING: A (ACC) ALL I/O, THRU 'A' REG.
A ⇄ 'A'

STORAGE: P (PROG.) 128 REG, (4) = 512 INSTR., 00 → 127,
A127 MUST HAVE A JUMP.
V (VARI.) 64, , ⇄ 'A' REG. 00 → 63

D (DISTR.) 500, , ⇄ 'A' REG. 000 → 499

(1602 PROCESSOR)

LITTON

EBS/1231 INSTRUCTION SET

AND TYPE	OP CODE		
BUTION:	DCLR	ALL = Ø, A=B=?	CLEAR D-REGS
	DGET	B=A, A=D_{V07}	BRING A D-REG
	DPUT	D_{V07}=A	STORE = =
	*DIST		DISTRIBUTE TAPE TO D
	*SCAN	A = VALUE, V_{007} = FIRST D-REG. ADDR.	SEARCH FOR ≠0 VALUE
	SGET		BRING A SPLIT D-RE
	SPUT		STORE = = =
IAL	OPUS		PROG. INTER.
	CALC		CALCULATE
	CDV		CHECK DIGIT VER.
	DUPE	INPUT	CONV. + DUP. (EVEN ADR
	DUPO	INPUT	= = = (ODD =
	SPEC	OUTPUT	= = = (SPECIAL

ES:

* SCAN ① N-OP'ED when 4th Comm. IN Prog. Reg.
② if V_{07} > 499 → SKIPS NEXT (SINGLE) INSTR.

* DIST ① V_{01} = START CHAR of ADDR. field (TAB *#'s.
V_{02} = = = = AMOUNT " "
V_{04} = BASE NO. (for lowest ADDR.) (000000-999
V_{05} = {1 → (MODE 1) → MUST HAVE FIRST OR ADDR, THEN OR AMNT.
{2 → (MODE 2) →
V_{06} = ADDR field EDIT WORD.

* TAB #S = ~~D~~ (TAB#) − 1 = EVENLY DIVISIBLE by 3. {1, 4, 7, ... n 190

1231 INSTRUCTION SET

MAND PE	OP CODE	OPTIONAL ADDR.	SYMBOLIC EXPLANATION	NOTES
VSFER:	CLA	–	A = Ø	
	XCB	–	A ⇄ B	EXCHANGE
	XCV	–	A ⇆ V00	
	BV	00-63	B = A, A = V	BRING
	SV	00-63	V = A	STORE
th:	ADD		A = A + B	
	NGA		A = ⁻A	NEGATE
	NGB		B = ⁻B	
	UV	00-63	V = V + A	UPDATE
	ACC	00-63	A = A + V	ACCUMULATE. A=0 if {ANY R. RESULT
	MDV	00-31	A = A · B ÷ V	MULTIPLY / DIVIDE
Jump:	AJ	–		AUTOMATIC JUMP TO PROG
	JUP	00-127		JUMP UNCOND, TO PROG
	JZP	00-127	if A {=0 → JUMP. ≠0 → A = A-1.	JUMP ZERO TO PROG
-OP if 4,8,12...	JPS	–	if A ≥ Ø → SKIP NEXT INSTR.	JUMP POS SKIP ONE,
	JMK	00-127		JUMP MARK TO PROG
	JR	–		JUMP RETURN TO LAST M'K'D INSTR
I/O:	SEL	00-77		CHANNEL SELECTION
	IN	01-10	B = A, A = INPUT.	[redacted]
	SKIP	01-10		SKIPS TAPE INPUT UNTIL 'CNTL I'
	OUT	00-31 1,4,7,10,13...	A → OUTPUT. FORMAT=U 00-31 TERM BY 'CNTL I' ONLY.	
	DUP	01-190	→ RETN → TAB 01-190, NO LF	DUPLICATE (WON'T AUTO. OUTPUT 'C
	CO	00-77		CHAR. OUTPUT
	SCI	–	A = INPUT.	SINGLE CHAR. INPUT
	SCO	–	A → OUTPUT	" " OUTPUT
	TAB	1,4,7,10,13...m 01-190		TABULATE
	ALFI	–	XXXXX → A	ALFA INPUT
	ALFO	–	A → XXXXX	ALFA OUTPUT
v)	INA	–	A→B Num. ASCII → A, [redacted]	{INPUT ASCII ESC → EXIT Rub-out → CLEAR

I/O

INPUT DEVICE	CHANNEL #	OUTPUT DEVICE
KEYBOARD	1	PRINTER
READER	2	PUNCH
RES'D.	3	RES'D.

MAND

SEL 00-77 : SEL XX

INPUT ch. CODE — OUTPUT ch CODE

'0' → off
'1' → on

CHANNEL CODE	ch. 3	ch. 2	ch. 1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

IN 01-10 INPUTS 01-10 SIG. DIGITS

'0' or '-' KEYS (ANYWHERE) INDICATE NEG. NO.

CLEAR KEY → TO RE-ENTER

I ($V_{00} = V_{00}$) UNCHANGED

II $V_{00} = 1$

III $V_{00} = 2$

IIII " = 3

A0 " = 4

A1 " = 5

A2 " = 6

A3 " = 7

nmAND

LT 00-31 A → OUTPUT, FORMAT = V 00-31.

EDIT FORMAT: (CHECK ARGTECT.)

{ { * or Z } S S S X / X , X X X . X X }

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

SUPPRESSED DIGITS

SPECIAL CHARS.

↑ MOST ZERO SUPPR

EDIT WORD:

{ (S) }

OUTPUTS SIGN.

NOTES:

① # of S's & X's MUST = 10.

② { ~ } INDICATES OPTION.

EDIT WORD CONSTANT

V XX = [D DDDDDDDDDD, D]

0 → if + → ' ' .

1 → if − → '−' .

	END SUPR	LEADING *'S	LEADING ●'S
0 =	0	0	0
1 =	0	0	1
2 =	0	1	0
3 =	−	−	−
4 =	●	0	0
5 =	●	0	1
6 =	●	1	0

00 = LEADING SPACES

0 =	−	−	−	
1 =	−	−	−	
2 =	0	1	0	→ / D (SIG
3 =	0	1	1	→ . D (SIG.
4 =	1	0	0	→ , D
5 =	1	0	1	→ SUPPRESSED
6 =	1	1	0	→ D
7 =	1	1	1	→ CONV. PROG. 1ST DIGIT

MMAND

N01-10

DATA from		first char / subs. char (code equivalent)	effect on V_{00} =	negates/terminates? first char.	negates/terminates? subsequent char
Keyboard	Tape				
	INT. CYCLE NA	*/II	V_{00} = 1	IGNORED	TERMINATES
	—	-/CP1	—	NEG.	NEG. + TERM.
	DES CYCLE NA	CLR/LF left	V_{00} = 2	IG.	TERM.
	AMT +	Φ/CP1	—		TERM.
	SUB TOTAL	Φ/LF Right	—		IG.
	TOTAL	/%	—		IG
	TAPE LEADER	CARR. OPEN+CLOSE	—		IG

APPLICATION ________________ DATE ________________

ROUTINE ________________ BY ________________

EBS 1231
CODING CHART

PAGE ___ OF ___

FROM	PROGRAM REGISTER	STEP	OP CODE	ADDITIONAL CODE	COMMENTS	WORKING REGISTERS A	B	OO	DEVICE SELECTION KB	RDR	PTR	PNCH

EBS 313 11.6d