SNAP I ASSEMBLER

PBC 1018

pb Packard Bell Computer
A DIVISION OF PACKARD BELL ELECTRONICS
1905 ARMACOST AVENUE • LOS ANGELES 25, CALIFORNIA • GRANITE 8-4247

June 1, 1962

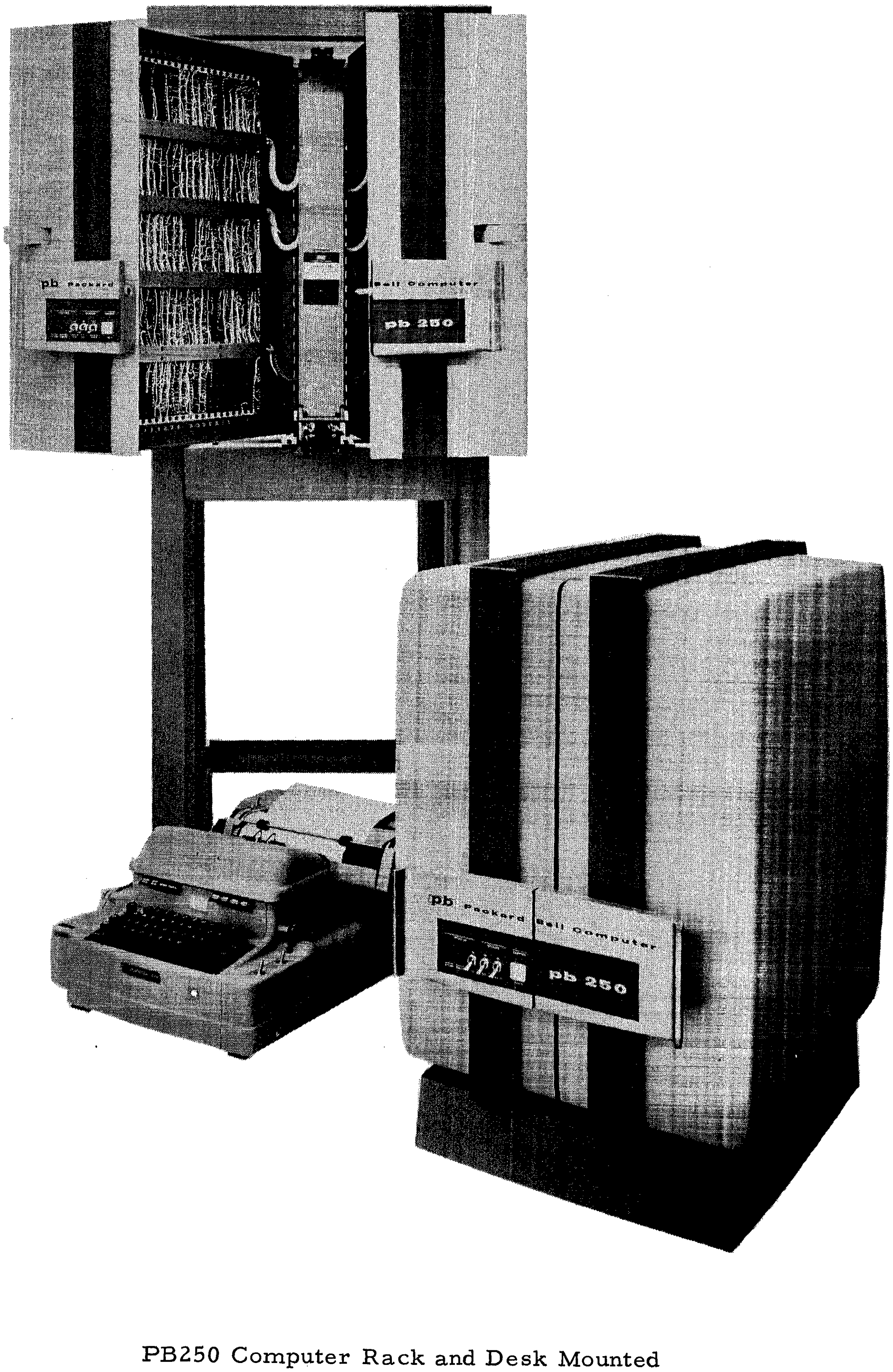
CONTENTS

CONTENTS (Continued)

INTRODUCTION

The SNAP I, Symbolic Non-optimizing Assembly Program, enables the programmer to code all instruction operation codes and addresses in symbolic language. This symbolic language program is translated by the assembler into Octal Utility Package listable format and punched on paper tape. The symbolic tape may be read by either the Flexowriter or by the modified HSR-1 High-Speed Reader, and the output tape may be punched by either the Flexowriter or by the HSP-1 High-Speed Punch.

This manual contains a description of the PB250 Computer as a decimal device and provides instructions for programming using symbolic coding. A list of symbolic codes to be used by the programmer is located in Section III. A sample program using these codes is located in Appendix A.



PB250 Computer Rack and Desk Mounted

I. DESCRIPTION OF EQUIPMENT

A. GENERAL

This chapter contains a description of the PB250 Computer and the Flexowriter. A description of their controls and indicators is also provided.

B. DESCRIPTION OF PB250 COMPUTER

The Packard Bell PB250 is a high-speed, completely solid-state general purpose digital computer in which both the data and the commands required for computation are stored in a homogeneous memory. The storage medium is a group of nickel steel magnetostrictive lines along which acoustical pulses are propagated. At one end of each of these lines is a writing device for translating electrical energy into acoustical energy. At the other end of each line is a reading device for translating acoustical energy back into electrical signals. By rewriting the stored information as it is read, information continuously circulates without alteration except for alterations which result from the execution of the computer program.

The PB250 provides a repertoire of more than 50 commands flexible enough to permit coding of a very broad range of scientific and engineering problems. Double precision commands are provided for operating upon large numbers. Commands to normalize and scale numbers facilitate floating point operation. Square root, and variable length multiplication and division operations are available in the command list. Other features include input-output buffering, and a large number of optional peripheral units such as punched card equipment, tape handlers, shaft encoders, photo readers, and analog-to digital and digital-to analog converters.

B-1. MEMORY ORGANIZATION

The memory of the basic PB250 contains 10 lines, numbered decimally from 00 through 09, which hold both data and instructions. Each long line, 01 through 09 contains 256 locations, also called sectors, that are numbered 000 through 255. Since the information in any location can be either data or a command, the generic term "word" is used to cover both. The location of any word is specified by a line and sector number, and these together are called an address. Line 00 is a 16-word fast access line. Since line 00 is 1/16 the length of a long word line, any unit of information contained in it is available 16 times during each complete circulation of the 256-word lines. Thus, any word in the fast access line can be identified by one of 16 sector addresses. For example, sector 000 of line 00 can be identified by the following addresses: 00000, 01600, 03200, 04800,..... 24000.

Fifty-three additional lines, each of which may have from 1 to 256 words, can be added. These lines are numbered 10 through 30 and 32 through 63. Line number 31 is used as an Index register. If all of the additional lines are used, and if all hold 256 words, the memory capacity of the PB250 is extended to 15,888 words.

Commands can be executed only from lines 00 through 15; these lines are therefore designated "Command Lines".

B-2. Data Word Configuration

Every number stored in the PB250 is represented by a series of pulses which correspond to a series of zeroes and ones that are the digits of the binary number system. The term "binary digit" is usually contracted to the word "bit".

A number stored in a location in the PB250 consists of 21 bits (Figure 1-1) that represent magnitude, and a 22nd bit to indicate sign. A negative number has a one in position zero, whereas a positive number has a zero in position zero. Negative numbers are expressed in their 2's complement form.

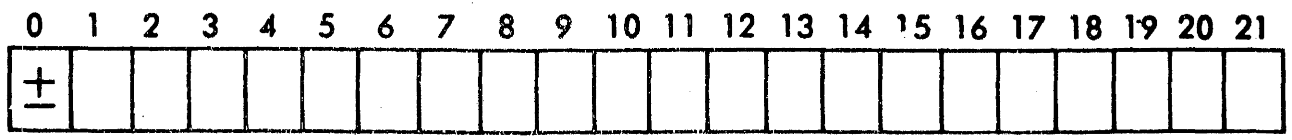


Figure 1-1. Data Word Configuration

These 22 bits can represent any decimal number less than 2,097,152. Larger numbers may easily be represented by using the double precision features of the computer.

B-3. Arithmetic Registers

Three arithmetic registers, A, B, and C, are provided for arithmetic operations and information manipulation. Each register has exactly the same format as a memory location, including the sign, and all are available to the programmer. Double precision commands treat A and B as a double-length register; information may be interchanged between A, B, and C. The contents of a register may be tested for non-positive values or compared against the contents of any memory location. A record may be kept in one register of operations performed on the others.

B-4. COMMAND WORD CONFIGURATION

Information in any memory location may be either data or a command. When the information is a command, it has a definite configuration, or format, as illustrated in Figure 1-2.

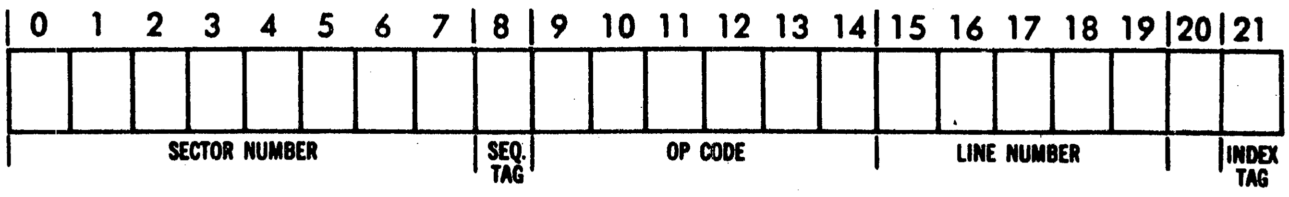


Figure 1-2. Command Word Configuration

Each subdivision, or field, of the command word is uniquely identified. The subdivisions are the sector number, sequence tag, op code, line number, and index tag fields. There will be frequent references in subsequent discussions to the address field of a command. Although the address is made up of a sector and a line number, these numbers are not contiguous in the command format. The address field, however, is considered as a single entity. The address 03204 refers to sector 032, line 04. The contents of the address field in a command do not always designate a memory location. For example, the shifting commands use the address field to indicate the number of places to shift.

The sequence tag field may contain either a one or a zero.

The op code field contains a numeric code which specifies one of the PB250 commands.

The index tag field may contain either a one or a zero. When a one is placed in this field, the contents of the Index register are used; a zero in the field indicates no use of that register.

Bit position 20 contains a one only when referring to a line number of 32 or greater.

B-5. INDEX REGISTER

The Index register contains a line number for use with commands which are index-tagged. When used, the contents of the Index register replace the line number of the address in the command. This replacement is made during the reading of the command, but does not change the command as it stands in memory.

Line number 31 is reserved to designate the Index register. Addresses 00031 through 25531 all apply to this register, and bit positions 16 through 20 are the useful positions for the line number.

B-6. COMMAND TIMING

The PB250 reads and executes commands from the circulating command lines. The words of the long lines are read serially in sector number sequence (000, 001, 002 --- 254, 255, 000, 001 ---). The time for each word to pass through a reading device is 12 microseconds; therefore, the time for all 256 words of a long line is 3072 microseconds. The commands are read and executed in numerical order from a given line and a given sector: 000 of line 01 (00001, 00101, 00201, ---) or 011 of line 05(01105, 01205, 01305, ---). The performance of each command involves four phases:

Phase I	Wait to read next command.
Phase II	Read next command.
Phase III	Wait to execute command.
Phase IV	Execute command.

For example, a command in 00001 to store A in 03004 will be read (Phase II) in sector 000, held for execution (Phase III) in sectors 001 through

029, executed (Phase IV) in sector 030, and held while waiting to read the next command (Phase I) in sectors 031 through 000. Phase II will follow in sector 001 to read the next command in 00101.

There are four classes of commands in which the nature of Phase IV differs.

CLASS 1.

In this class of commands, execution always follows the reading of the command by skipping Phase III. This class of commands consists of all those which require an extended interval of execution such as block transfer, shifting, and multiplication. The execution time for this class of command varies with the required duration. For example, block transfers require 12 microseconds per word, shifting requires 12 microseconds per bit, and multiplication requires 12 microseconds per multiplier bit.

CLASS 2.

In this class of commands, execution is always completed in the sector specified by the sector number of the command. This class consists of all one-sector operations such as load, store, add, and clear. All commands of this class require 12 microseconds to execute.

CLASS 3.

Class 3 is an extension of class 2 to handle double precision operations. As in class 2, execution always starts in the sector specified by the sector number of the command but the execution phase is always extended into the following sector. All commands of this class require 24 microseconds to execute.

CLASS 4.

Class 4 consists of commands for conditional and unconditional transfer of control. The condition for a conditional transfer is tested in Phase II and, if the condition is met, the next command is read from the address specified by the command. If the condition is not met, the command directly following transfer of control command is read. A conditional transfer where the condition is not met, thus requires no execution time. The unconditional transfer selects the next command with no restrictions. The execution time when control is transferred is 12 microseconds per sector for the interval between the transfer of control command and the next command.

B-7. THE COMPUTER CONSOLE

The console of the PB250, shown in Figure 1-3, is composed of lights and switches arranged in two rows. The top row has three sets of lights: six for OPERATION, five for OPERAND, and three for COMMAND. OPERATION specifies which OP code is being executed, i. e. , ADD, LOAD A, etc. Using 1 to indicate light on, and 0 for a light off, the pattern 001100 represents the command ADD (Command 12 in decimal). OPERAND specifies the line number portion of the address, and COMMAND indicates from which command line the command is being executed.

On the second row are the O'FLOW light, PARITY light, FILL switch, TEST switches, and POWER button. The O'FLOW light is on if an overflow has occurred. The PARITY light indicates a parity check error. Computation may be started by depressing the ENABLE and BREAKPOINT SWITCHES on the Flexowriter to clear the parity flip-flop. The FILL switch is used for loading a "bootstrap loading program". The TEST switches are used for maintenance of the system. The POWER button is an alternating type, turning the power to the computer on or off.

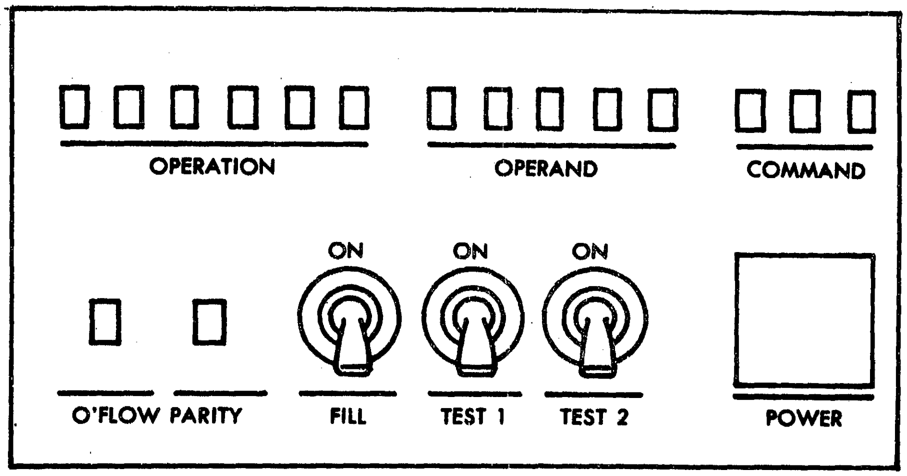


Figure 1-3. PB250 Computer Console

C. DESCRIPTION OF FLEXOWRITER

A Model FL Flexowriter is used as the control unit for the PB250. This machine is also used to prepare, duplicate, and read tapes. The Flexowriter can be used on-line (Flexowriter under control of computer), or off-line (Flexowriter under control of operator). The general appearance and operation of the Flexowriter are similar to a standard electric typewriter. (See diagram of Flexowriter keyboard in Figure 1-4.) Such features as space lever, paper release lever, platen knobs, margin release lever, ribbon position lever, margin and tab stops, and type guide, are used in exactly the same manner as for a standard typewriter.

The Flexowriter prepares tape by punching coded holes across the width of the tape. The punched tape is prepared manually from the keyboard (off-line) or by output commands from the computer (on-line), and is described as having channels and characters. The channels run lengthwise along the tape while characters are across its width. The PB250 uses six channels of an eight-channel tape; the code used in this tape is pictured in Figure 1-5. The tape reader can read and type out the information contained on a coded tape.

When the Flexowriter is under the control of the computer, its paper-tape reader, typewriter keyboard, and paper-tape punch are used as input-output devices by the computer. Each typewriter character has a specific code, which is sent to the computer as a pattern of six bits.

The command READ PAPER TAPE will cause the tape reader to read a single character and load it into the input buffer of the computer. The command READ TYPEWRITER KEYBOARD will turn on the INDICATING LIGHT of the Flexowriter, after which a typewriter key must be depressed to load the input buffer. The INDICATING LIGHT is turned off by the loading operation.

The command WRITE OUTPUT CHARACTER provides information to either the typewriter or the punch. It is possible to prepare a tape with as many as eight channels by this command and read such a tape back into the computer with the READ PAPER TAPE command.

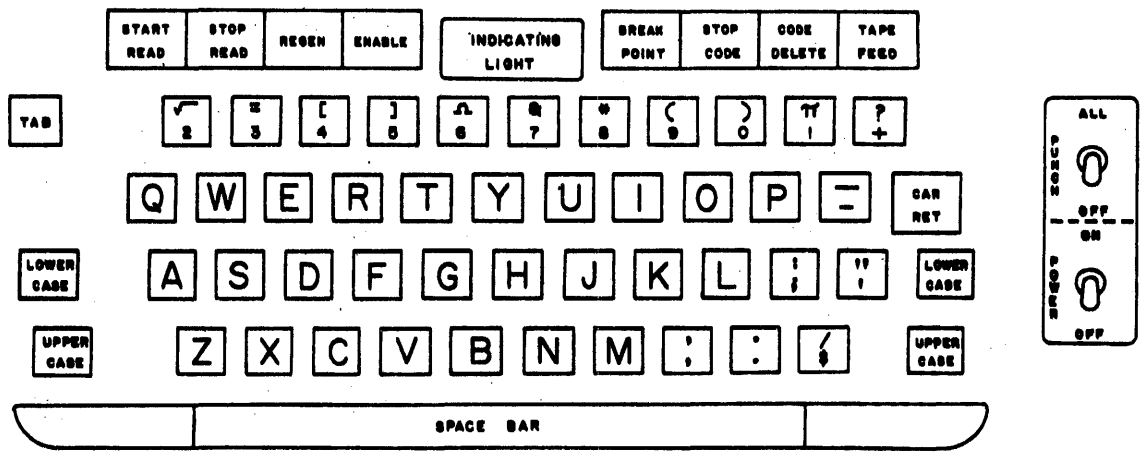


Figure 1-4. Flexowriter Keyboard

	CHANNEL					
	A	B	8	4	2	1
A	●					●
B	●				●	
C	●	●			●	●
D	●			●		
E	●	●		●		●
F	●	●		●	●	
G	●			●	●	●
H	●		●			
I	●	●	●			●
J		●				●
K		●			●	
L					●	●
M		●		●		
N				●		●
O				●	●	
P		●		●	●	●
Q		●	●			
R			●			●
S	●	●			●	
T	●				●	●
U	●	●		●		
V	●			●		●
W	●			●	●	
X	●	●		●	●	●
Y	●	●	●			
Z	●		●			●

ALPHABETICAL CHARACTERS AVAILABLE IN BOTH UPPER & LOWER CASE

		CHANNEL					
Upper Case	Lower Case	A	B	8	4	2	1
)	0	●					
π	1						●
$\sqrt{}$	2					●	
=	3		●			●	●
[	4				●		
]	5		●		●		●
Ω	6		●		●	●	
&	7				●	●	●
*	8			●			
(	9		●	●			●
?	+		●	●	●	●	
—	−		●	●	●	●	●
:	;	●	●				
”	’		●	●		●	●
,	,	●	●	●		●	●
.	.	●		●		●	●
/	$	●	●				●

NUMERICAL & SPECIAL CHARACTERS

	A	B	8	4	2	1
Upper Case	●	●	●		●	
Lower Case	●	●	●	●		
Tab	●	●	●	●	●	
Carriage Return	●		●	●	●	
Stop Code			●		●	●
Delete	●	●	●	●	●	●
Space		●				

CONTROL CHARACTERS

Figure 1-5. Flexowriter Code

The keyboard of the Flexowriter is similar to a standard typewriter keyboard and may serve many of the same purposes. The numeric, alphabetic, and symbolic keys require no explanation other than the alphanumeric code for the computer as given in Figure 1-3. The TAB KEY, CAR RET (Carriage return), LOWER CASE, UPPER CASE, and SPACE BAR are self-explanatory and are analogous to controls on a standard typewriter. The

REGEN SWITCH, when depressed, permits exact duplication of the tape which is in the Flexowriter tape reader.

Certain switches and keys on the Flexowriter are used to control the computer.

ENABLE SWITCH:

1. Interrupts computation.
2. Conditions the use of other switches and keys of the Flexowriter.

BREAKPOINT SWITCH:

1. Sends signal to computer which may be tested by the command TRANSFER ON EXTERNAL SIGNAL.
2. With ENABLE SWITCH clears parity flip-flop (indicated by PARITY light on).

I Key: With ENABLE SWITCH causes the computer to execute the command in memory location 00001.

C Key: With ENABLE SWITCH causes the computer to cycle by one command.

II. THE ASSEMBLY LANGUAGE

A. GENERAL

This chapter describes the instruction format to be used for coding programs symbolically and describes the principles of programming using SNAP. All numbers used in this section are radix 10.

SNAP instructions consist of four parts, or fields, separated by tabs, with the entire instruction terminated with a carriage return. The fields are designated as location, operation code (op code), address, and tag and are described in paragraphs B through E below. The only field required for each instruction is the op code, however, many op codes require an address (see paragraphs C and D below). A typical instruction is shown below.

Location		Op Code		Address		Tag	
START	Tab	ADD	Tab	$ + 5	Tab	S	Carriage Return

B. LOCATION FIELD

Programs assembled using SNAP are free of many of the clerical chores required for octal coding. This is accomplished by the SNAP function, which assigns sequential location to instructions once an absolute location is specified. If an initial absolute location is not specified, instructions are assigned sequential locations, beginning with sector 000 of line 02. Instructions will be assigned sequentially within the same command line, with sector 000 following sector 255, until a new absolute location is specified by the user. When a location other than 00002 is desired, the absolute location is inserted in the location field.

A symbolic location field may contain from one to five alphanumeric characters, at least one of which must be alphabetic. Alphanumeric characters include A through Z and the numbers 0 through 9. Special characters such as $, +, -, etc., should not be used in the location field.

It is also permissible to specify an absolute decimal location of the form SSSLL, where SSS is a three-digit decimal sector number from 000-255, and LL is a two-digit decimal number from 01-21. When an absolute location field is encountered, the location counter in the assembler is set to the new value, and instructions are assigned sequential locations from there. If an absolute location is specified, to which an instruction has been assigned previously by the assembler, the location will be flagged as a memory overlap error, and an error printout will occur. The location counter will be set to the new location, however, and the assembler will assign instructions sequentially from there.

The location field must be terminated by a tab. If a blank location field is desired, a tab alone should be typed, followed by the operation code

C. OP CODE FIELD

The operation code field (op code) contains the operation to be performed, such as "add" or "subtract". A list of the operations to be performed is given in Table 3-1. Each op code must consist of one of the symbolic codes shown in the table.

Operations are divided into two categories:

1) Standard Operations, which are those operations which can be performed by the PB250 Computer and which are

described in PBC1004, PB250 Programming Manual. Use of one of these op codes results in a command format word.

2) Pseudo Operations, which are those additional operations which can be performed using SNAP and which are described in Section III of this manual. Some pseudo operations result in the storage of a data format word, while others provide control functions.

Most operations require an address field. The Reference column of Table 3-1 refers to the paragraph which defines the address requirements for each op code. These requirements are described in Section III of this manual.

When an op code requires an address, the op code field is terminated with a tab and the address is inserted as described in paragraph D below. When the op code does not require an address, the op code field is terminated with a carriage return, indicating end of instruction, and the program continues.

D. ADDRESS FIELD

It is not necessary to use the address field for each instruction, this is determined by the op code (see Section III of this manual). When the address field is required, use one of the following forms:

1) An absolute decimal consisting of five decimal digits of the form SSSLL where SSS is a sector number from 000 to 255, and LL a line number from 00 through 63.

2) A symbolic address consisting of from one to five alphanumeric characters, at least one of which must be alphabetic.

Alphanumeric characters include the letters A through Z, and the numbers 0 through 9.

3) A relative address, which refers to an address relative to an established symbolic location, or an address relative to the location of the instruction being written. Assuming that a symbolic address of NOT12 has been established previously, and that it is desired to refer to the fourth sector location following that assigned to NOT12, it is permissible to write in the address field NOT12+4. Any decimal integer up to 225 may be used. To refer to sectors preceding NOT12, a minus sign should be used instead of the plus.

If the desired address is in the next sector location, a $+1 is required: where $ means, "this location." Thus, $ - 1 means this location plus one location, or the next location. Any + or - integer may be used depending on the location desired.

An address may be relative to a symbolic address or to the location of the instruction. An address relative to an absolute address should not be used. The + or - will be ignored by SNAP, and SNAP will function on the absolute address listed, resulting in error or false computation. The relative address will, in all cases, have the same line number as the base address.

4) A decimal integer less than 256. The number used represents the duration of certain operations such as shifts. Assuming that the number 6 is used in the address field of one of the shift op codes, the AB registers will shift once for each unit of the number.

5) An address field used as defined in Section III for certain pseudo operations.

6) An address field left blank for op codes not requiring an address. Where an op code, such as CLA or IAC, does not normally require an address, but an address is specified, only the sector portion is used, and the line portion is set to 00.

E. TAG FIELD

The tag field contains an S if an instruction is to be sequence-tagged, an I if index-tagged, or SI if both sequence and index tagging are to be used. No instruction in SNAP is required to have either a sequence tag or an index tag. Certain pseudo ops use the tag field to specify control data. This usage is described in Section III.

III. SNAP COMMANDS

A. GENERAL

This section describes the addressing requirements of the PB250 Computer and the SNAP pseudo operations. Table 3-1 lists the symbolic coding data alphabetically, gives it designation and octal values, and describes their functions. Reference is provided to paragraphs which contain specific data for each code.

B. ADDRESSING REQUIREMENTS

The addressing requirements of each of the commands are described in the following paragraphs.

B-1. The commands listed below require an address field which is either symbolic, absolute decimal in the form SSSLL, or which uses the notation $. A relative address (Section II, paragraph D) may also be used.

ADD	EBP	LDP	TAN
AMC	EXF	STA	TBN
AOC	LAI	STB	TCN
CAM	LDA	STC	TOF
DPA	LDB	STD	TRU
DPS	LDC	SUB	

B-2. The commands listed below do not require an address. If an address is specified, only the sector portion is used with the line address being set at 00, except in the case of HLT, in which the line address is also inserted into the command.

If no address is specified, the assembler inserts sector and line addresses of 000 and 00, respectively, except in the case of MAC, in which the correct sector address will be formed.

Execution occurs during the sector address specified, except in the cases of NOP and HLT in which execution occurs up to, but not including, the specified sector address.

CIB	DIU	IBC	ROT
CLA	GTB	MAC	RPT
CLB	HLT	NOP	RTK
CLC	IAC	RFU	

Example:

Symbolic	Machine Word
IBC	000 0200;

B-3. The commands listed below require an address consisting of a decimal integer less than 256, which indicates the number of bit positions the AB registers are to be shifted.

Execution occurs during the next N sectors, where N is the number specified in the address field.

LRS	NOR	SBR
LSD	RSI	SLT
NAD	SAI	SRT

Example:

Symbolic	Machine Word
00003 SLT 6	007 2110;

B-4. Certain commands do not require an address. When an address is specified, it must be a decimal number indicating the number of sector times the command is to operate.

When no address is specified, a full-length command will be formed. A full-length DIV, DVR, or MUP command requires 22 sector times for execution, while a full-length SQR requires 21 sector times for execution.

Example:

Symbolic	Machine Word
00002 MUP	027 3200;

B-5. The commands listed below require an address of the form LL, M, where LL is a 2-digit decimal line number and M is a memory location of the form described under paragraph B-1 above. In forming the command, only the sector address of M is used, with the line address coming from LL. Separate LL from M by a comma.

Execution starts in the sector following the command and continues up to, and including, the sector address of M.

BSI	IAM	MLX
BSO	MCL	PTU

Example:

Symbolic	Machine Word
00005 MCL 00, $+5S	006S7105;

B-6. The TES command has an address of the form LL, M as described in paragraph B-5 above. In the case of TES, however, when the specified signal is present, transfer is made to M, which must be located in the same memory line as the TES command itself.

Use signal numbers 21 through 31 to specify the following:

Signal Number	Function
21-24	Arbitrary input signals
25	High-speed punch sync. signal
26	Magnetic tape gap signal
27	Magnetic tape reader clock input signal
28	Photo tape reader sprocket input signal
29	BREAKPOINT switch input signal
30	Typewriter or paper tape reader "character input complete" signal
31	"Typewriter not ready for output character" signal

Example:

Symbolic	Machine Word
TES 29, ALPHA	025 7735;

B-7. The WOC command requires an address of the form X, M (where X is the particular output character and M is the location in which the command is to be executed). Only the sector address of M is used. X must be separated from M by a comma.

It is not necessary to specify an address of execution. When one is not designated, a sector address of 000 will be inserted.

The character, X, may be any alphanumeric or special character, including space. Control characters other than space should not be used directly. The special codes listed below should be used.

To form a dummy WOC command, terminate the WOC op code with a carriage return.

Example: To output the letter A, the command would be as follows:

Symbolic	Machine Word
WOC A	000 6101;

Control Character	Special Code
Carriage Return	CR
Tab	TB
Upper Case	UC
Lower Case	LC
Stop Code	ST
Code Delete	DE
Blank (Tape Feed)	BL

C. PSEUDO OPERATIONS

Each of the pseudo operations used with SNAP are described below.

C-1. BINARY-CODED DECIMAL (BCD)

Information in the address field is stored as six-bit bcd data, preceded by a leading octal zero and a positive sign, as shown below. The address field can contain any combination of up to three alphanumeric characters, including space. Control characters other than space, must be represented by two-letter codes (carriage return = CR, tab = TB, upper case = UC, lower case = LC, stop code = ST, delete = DE). When a special two-letter code is used, it must be the only information in the address field of the particular bcd instruction and the tag field of the instruction must contain an X.

Example: Message: END OF JOB

Op	Addr	Tag	Machine Word
BCD	END		+0650544
BCD	OF		+0200666
BCD	JO		+0202106
BCD	B		+0420000
BCD	CR	X	+0560000

C-2. BLOCK STARTED BY SYMBOL (BSS)

This pseudo op code reserves consecutively the number of words specified in the address field, up to 256, beginning with the location specified by the BSS instruction. The reserve block must lie entirely within one memory line.

Loc	Op	Addr	Machine Word
TABLE	BSS	100	Locations TABLE through TABLE + 99, inclusive, are reserved. Location TABLE will be defined as the location of the BSS instruction.
20003	BSS	10	Sectors 200 - 209 of line 03 are reserved.

C-3. DECIMAL (DEC)

This pseudo op code is used to enter decimal data. The decimal number can contain a sign, decimal point, and no more than seven decimal digits.

The magnitude of the decimal number, ignoring decimal point, must be less than 2,097,152. If the decimal number is entirely fractional, the maximum number of digits possible is six. The decimal number will be stored as a binary machine word at specified binary scaling, Q. Q is expressed decimally and must lie within the range $0 \leq Q \leq 21$. The Q value must be placed in the tag field. If no Q is specified, the number will be assumed an integer and stored at 21. For example, to convert the number 100.25 to binary and store it at a Q of 7:

Op	Addr	Tag
DEC	100.25	7

C-4. END

This pseudo op code terminates the program. The address field is ignored and the op code is terminated by a carriage return. A "W" will be punched at the end of the listable assembled tape as an Octal Utility Package control code. This pseudo op code is nongenerative.

C-5. EQUALS (EQU)

This pseudo op code is used to define symbols. The location field must be symbolic and contain a symbol which does not appear in the location field of any other instruction. This pseudo op code is nongenerative.

Examples:

Loc	Op	Addr	Machine Word
ALPHA	EQU		The symbol ALPHA is entered into the symbol table as equal to sector 001 of line 07.
SS2	EQU	TAB2	SS2 is now defined to have the same location as TAB2. TAB2 must have been previously defined.
FIN	EQU	DATA+20	FIN is assigned to the 20th sector following that previously assigned to the symbol DATA.

C-6. OCTAL (OCT)

This pseudo op code is used to enter octal data. The address field may contain a sign and from one to seven octal digits. If no sign is present, plus is assumed. If less than seven digits are present, the machine word will be right-justified with leading octal zeros and the appropriate sign. A minus sign will not generate a complemented number. Example:

Op	Addr	Machine Word
OCT	+4217077	+4217077
OCT	-20	-0000020
OCT	1	+0000001

C-7. PROGRAM ORIGIN (ORG)

The location counter is set to the absolute address appearing in the location field. Instructions are assigned sequentially from this point. This pseudo op code is nongenerative.

C-8. SKIP (SKIP Y)

The location counter is advanced Y + 1 sectors within the same command line. This pseudo op code does not reserve any memory sectors, nor does it generate any instructions.

D. SYMBOLIC PROGRAMMING CODE

Table 3-1 lists all of the operations used with SNAP, briefly describes their functions, and gives their class. The Reference column lists the paragraph which outlines the addressing requirements of the op code. Symbols used in the table are described as follows:

Symbol	Meaning
( )	contents of
$\longrightarrow$	replaces
NC	next command
-( )	complemented contents of
$\overline{}$	logical NOT
+	sum or logical OR
Mm	Memory sector locations M and M + 1
PSEU	pseudo operation

Table 3-1. (Sheet 1 of 3)

SYMBOLIC PROGRAMMING CODE

Operation	Code	Octal Value	Function	Class	Reference
Add	ADD	14	(M) + (A) →A	2	B-1
AND M and C	AMC	42	(C) ∧ (M) →B	2	B-1
AND OR Combined	AOC	46	MC + $\overline{M}$B →B	2	B-1
Binary-Coded Decimal	BCD	-	Alphanumeric Address	PSEU	C-1
Block Serial Input	BSI	73	Transfers (external register) to M	1	B-5
Block Serial Output	BSO	72	Transfers (M) to External Register	1	B-5
Block Started by Symbol	BSS	-	Reserve data block	PSEU	C-2
Compare A and M	CAM	56	Set Overflow if (A) = (M)	2	B-1
Clear Input Buffer	CIB	57	Clear Input Buffer	2	B-2
Clear A	CLA	45	Clear (A) to Zero	2	B-2
Clear B	CLB	43	Clear (B) to Zero	2	B-2
Clear C	CLB	44	Clear (C) to Zero	2	B-2
Decimal	DEC	-	Decimal Number	PSEU	C-3
Disconnect Input Unit	DIU	50	Disconnects input buffer	2	B-2
Divide	DIV	31	(AB) ÷ (C) →B, R in A	1	B-4
Double Precision Add	DPA	16	(mM) + (AB) →AB	3	B-1
Double Precision Subtract	DPS	17	-(mM) + (AB) →AB	3	B-1
Divide Remainder	DVR	31	(AB) ÷ (C) →(B)	1	B-4
Extend Bit Pattern	EBP	40	(M) lock (A) →A	2	B-1
End	END	-	Terminates Program	PSEU	C-4
Equals	EQU	-	Defines Symbols	PSEU	C-5
Extract Field	EXF	47	($\overline{M}$) ∧ (B) →(B)	2	B-1
Gray to Binary	GTB	41	(A) from gray to binary	2	B-2
Halt	HLT	00	Stops Computation	1	B-2
Interchange A and C	IAC	01	(A) ⟷(C)	2	B-2
Interchange A and M	IAM	25	(A) ⟷(M)	1	B-5
Interchange B and C	IBC	02	(B) ⟷(C)	2	B-2

Table 3-1. (Sheet 2 of 3)

SYMBOLIC PROGRAMMING CODE

Operation	Code	Octal Value	Function	Class	Reference
Load A from Input Buffer	LAI	55	(Input Buffer) $\wedge$ (M) + (A) $\wedge$ ($\overline{\text{M}}$) $\rightarrow$A	2	B-1
Load A	LDA	05	(M) $\rightarrow$(A)	2	B-1
Load B	LDB	06	(M) $\rightarrow$(B)	2	B-1
Load C	LDC	05	(M) $\rightarrow$(C)	2	B-1
Load Double Precision	LDP	07	(mM) $\rightarrow$(AB)	3	B-1
Logical Right Shift	LRS	33	(AB) Shifted S places	1	B-3
Left Shift and Decrement	LSD	21	Shift (AB) left Decrement (C)	1	B-3
Merge A and C	MAC	00	(A) to (C), (C) = 1 when (A) or (C) = 1	2	B-2
Move Command Line Block	MCL	71	Command line to Line M_L	1	B-5
Move Line X to Line 7	MLX	26	(M_L) $\rightarrow$Line 07	1	B-5
Multiply	MUP	32	(B) X (C) $\rightarrow$(AB)	1	B-4
Normalize and Decrement	NAD	20	Normalize (AB) decrement (C)	1	B-3
No Operation	NOP	24	Continue in Command Sequence	1	B-2
Normalize	NOR	20	Normalize (AB) Decrement (C)	1	B-3
Octal	OCT	-	Enter Octal data	PSEU	C-6
Program Origin	ORG	-	Set location counter	PSEU	C-7
Pulse to Specified Unit	PTU	70	Starts - Stops External Equipment	1	B-5
Read Fast Unit	RFU	53	Enable fast character input	2	B-2
Rotate A, B, C	ROT	03	$\begin{tikzcd} (A) \arrow[dr, bend left] & \\ (B) \arrow[u, bend left] & (C) \arrow[l] \end{tikzcd}$	3	B-2
Read Paper Tape	RPT	52	(Paper tape) to input buffer	2	B-2
Right Shift and Increment	RSI	22	Shift (AB) right, Increment C	1	B-3
Read Typewriter Keyboard	RTK	51	(Keyboard) to input buffer	2	B-2
Scale Right and Increment	SAI	23	Scale (AB) right, Increment C	1	B-3
Shift B Right	SBR	33	Shift (AB) right S positions	1	B-3
Skip	SKP	-	Advance location counter	PSEU	C-8
Shift Left	SLT	21	Shift (AB) left S positions	1	B-3
Square Root	SQR	30	$\sqrt{(AB)}$ $\rightarrow$C	1	B-4
Shift Right	SRT	22	Shift (AB) right S positions	1	B-3
Store A	STA	11	(A) $\rightarrow$(M)	2	B-1

Table 3-1. (Sheet 3 of 3)

SYMBOLIC PROGRAMMING CODE

Operation	Code	Octal Value	Function	Class	Reference
Store B	STB	12	(B) $\rightarrow$(M)	2	B-1
Store C	STC	10	(C) $\rightarrow$(M)	2	B-1
Store Double Precision	STD	13	(AB) $\rightarrow$(mM)	3	B-1
Subtract	SUB	15	-(M) + (A) $\rightarrow$(A)	2	B-1
Transfer if A Negative	TAN	35	NC from (M) if (A) negative	4	B-1
Transfer if B Negative	TBN	36	NC from (M) if (B) negative	4	B-1
Transfer if C Negative	TCN	34	NC from (M) if (C) negative	4	B-1
Transfer on External Signal	TES	77	When external signal is present, transfer to specified address. When external signal is not present, take next instruction.	4	B-6
Transfer on Overflow	TOF	75	NC from (M) if Overflow	4	B-1
Transfer Unconditionally	TRU	37	NC from (M)	4	B-1
Write Output Character	WOC	6X	Character Output	1	B-6

IV. USE OF ASSEMBLER

A. GENERAL

Since SNAP is a two-pass assembler, the symbolic tape must be read in twice for a complete assembly. The first pass generates a symbol table and checks for memory overlaps, errors in the location field, and certain operation code errors. The second pass checks for illegal operation codes and certain address field errors. Any absolute addresses used in the symbolic program must be in decimal.

B. LOADING THE ASSEMBLER

SNAP requires 13 long memory lines in the PB250 Computer. The assembly program has its own bootstrap and is self-loading.

To load the program, insert the tape in the Flexowriter tape reader and raise the FILL switch. If the PARITY light is on, depress the ENABLE and BREAKPOINT switches. After the tape starts to move, raise the ENABLE and the BREAKPOINT switches. When the bootstrap section has been loaded, the computer will halt with the parity light on . To read the tape, depress the ENABLE and BREAKPOINT switches, strike the I key, and raise the switches.

The SNAP tape is prepared with a 6- to 8-inch space between sections. As each section of the assembler is loaded, a check sum is formed and compared against the one punched on the SNAP tape. If the two sums agree, loading continues; if not, the computer halts with an octal line number of 37 displayed on the OPERAND lights of the computer. Should this happen, back up the SNAP tape to the beginning of the particular section involved, depress the ENABLE and BREAKPOINT switches, and raise the switches to attempt to read the section again. If a parity error occurs on the first binary section of the tape, it will be necessary to restart loading from the bootstrap section of the tape.

C. INITIALIZING THE ASSEMBLER FOR PHASE 1

When the assembler has been properly loaded, the following heading will be printed out:

```
PB250 ASSEMBLY
IDENTIFICATION:
```

At this time, information may be typed to identify the particular symbolic program being assembled. When a carriage return is typed, the assembly will continue. When no identification is desired, strike the carriage return to continue.

The assembler will not print:

```
PHASE 1
INPUT:
```

Load the symbolic tape to be assembled into either the Flwxowriter or modified HSR-1 and type the desired input mode as follows:

For Flexowriter input, type FLEX

For photo reader input, type HSR

When the proper input mode has been specified, type a carriage return. The assembly program proceeds automatically through phase 1.

D. INITIALIZING THE ASSEMBLER FOR PHASE 2

Upon completion of phase 1, the assembler will print:

PHASE 2

OUTPUT:

Reload the symbolic tape into the same input unit used for phase 1 and specify whether the output tape is to be punched on the Flexowriter or High Speed Punch (HSP-1). The output mode is specified as follows:

For Flexowriter output, type FLEX

For High-Speed Punch output, type HSP

When the proper output mode has been specified, type a carriage return. The assembly program proceeds automatically through phase 2 and produces an output tape. Upon completion of phase 2, the assembler prints END OF JOB and halts with an octal line number of 37 displayed on the OPERAND lights of the console.

E. RESTARTING THE ASSEMBLER

After a normal end-of-job halt, SNAP may be restarted for a new assembly by depressing, then raising, the ENABLE and BREAKPOINT switches.

At any time during phase 1 or phase 2, except during input, the assembly program may be restarted at phase 1 by depressing the ENABLE and BREAKPOINT switches, striking the I key, then raising the switches.

During phase 2 the assembly may be restarted in the phase 2 mode by depressing the BREAKPOINT switch. Phase 2 will be restarted after the assembler completes the current instruction being processed, including any input-output in progress. Raise the BREAKPOINT switch after phase 2 restarts.

V. ERROR DETECTION

A. GENERAL

SNAP contains an error diagnostic which has been designed as an integral part of the assembler itself. The symbolic program is checked for a large number of possible errors and these errors are printed on-line as soon as they are detected.

When an error is detected and printed out, the assembler will continue to process the symbolic program, if possible. All errors detected during phase 1 should be corrected before going on to phase 2.

An error in the operation or address field will cause a word of zeroes, preceded by a space, to be punched on the output tape if phase 2 is performed. This format causes faulty instructions to be readily discernible.

The error printout will be of the form

SSSLL$XXSp

where SSSLL is the current setting of the location counter and XX is a special two-letter code, followed by a space, indicating the type of error detected. Following this, normally, will be the field which was in error.

B. ERROR CODES

B-1. ADDRESS FIELD ERROR (AD)

The printed address field contains an error. The possible errors are (1) an absolute sector number greater than 255, (2) an absolute line number greater than 63, (3) illegal special code for control character, (4) no comma in instruction as required, and (5) a BCD pseudo op code containing more than three characters in address field.

B-2. FIELD LENGTH (FL)

Indicates that an excessively long field has been found in an instruction. The remainder of the instruction will be ignored.

B-3. LOCATION FIELD ENDS IN C/R (LC)

Indicates that a location field has been terminated in a carriage return instead of the required tab.

B-4. LOCATION FIELD ERROR (LO)

An error is present in the location field printed out. The location field will be ignored. The possible errors are: $, field contains more than 6 characters, field contains + or -, absolute sector greater than 255, absolute line equal 00 or greater than 21, and numeric or blank location field used with an EQU instruction.

B-5. MULTIPLE DEFINITION OF SYMBOL (MD)

The symbol printed out has been used in the location field of more than one instruction. The symbol will be reassigned to the current setting of the location counter.

B-6. MEMORY OVERLAP (MO)

The program has assigned an instruction or reserved a data block in a sector to which an instruction has been previously assigned or which has previously been reserved as part of a data block. Each overlapping sector will be printed out; however, the assembler will assign the sectors as indicated in the symbolic program.

B-7. NO ADDRESS (NA)

An address has been omitted in an instruction which requires an address. If possible, the op code will be printed out. The entire instruction will be set to +0000000.

B-8. NO OP CODE (NO)

The instruction contains no operation code.

B-9. OP CODE ERROR (OP)

The operation code printed out is in error. The possible errors are: op code contains more than three characters, or an illegal mnemonic.

B-10. SCALING ERROR (SC)

In DEC pseudo op, designated number can not be held at specified scaling.

B-11. SYMBOL TABLE FULL (SF)

The capacity of the symbol table (256 entries) has been exceeded. No further symbols will be assigned to the table. The symbol printed out and all symbols following will therefore be undefined.

B-12. TAG FIELD ERROR (TG)

Illegal character in tag field.

B-13. TAPE FORMAT (TF)

Indicates that a carriage return has been omitted from a symbolic instruction in the general area printed out. The assembly will halt with an octal line number of 11 displayed on the console. It will be necessary to correct the error and restart the entire assembly.

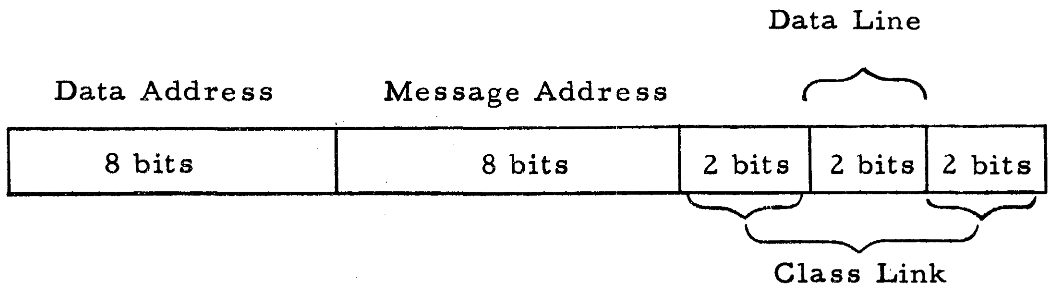
B-14. UNDEFINED SYMBOL (UD)

The symbol used in the address field has not been defined by being in the location field of an instruction.

APPENDIX A

SAMPLE SYMBOLIC PROGRAM

The following sample program is used to assemble interpretive control words from data read on paper tape. The data represents several variables required to compute a special function, where the variables are constantly changing. The function of the program is to accept octal inputs, and when a carriage return is sensed, to establish the fields according to the predetermined format shown below.



Immediately following the sample program Table A-1 shows the symbolic coding prepared by the programmer as compared to the machine coding prepared by the computer.

pb SNAP SYMBOLIC CODING SHEET

PROBLEM: D. P. C-W Assembly DATE:

PROGRAMMER: PAGE 1 of 3

LOCATION	OP	ADDRESS	TAG	REMARKS
START	LDP	$+1	S	
	TOF	A1		
	IBC			
	STB	SPACE		Set space link-1
	STA	PHASE		Set phase link-1
	CLA	READ	S	
	LAI	MASK	S	Character → A
READ	RPT	$-2	S	
	RPT	$+1		Reject last character
	TES	30, $-1		
	TES	30, READ		Wait for next character
	CIB	$-2	S	
MASK	OCT	77		
	CAM	$+1	S	
	OCT	56		
	TOF	CR		Exit if CR
	CAM	$+1	S	
	OCT	20		
SPACE	TOF	A1		Exit to space link
PHASE	IBC			Exit to phase link
	SRT	3		Phase link-1
	IAC			Assemble by 3
	SRT	19		
	CLA	READ	S	Return to read
A1	SLT	14		Space link-1
	STB	M1		Save last 8 bits
	LDA	$+1	S	Set space link-2
	TOF	A2		

pb SNAP SYMBOLIC CODING SHEET

PROBLEM: D. P. C-W Assembly DATE:

PROGRAMMER: PAGE 2 of 3

LOCATION	OP	ADDRESS	TAG		REMARKS
	STA	SPACE			
	CLA	READ	S		Return to read
A2	EXF	$+1	S		Space link-2
	OCT	-7777400			
	SLT	6			Save last 8 bits
	STB	M1+1			
	LDP	$+1	S		Set space link-3
	TOF	A3			and phase link-2
	IBC	CW-1	S		
	STD	SPACE			
	CLA	READ	S		Return to read
CW	SRT	2			Phase link-2
	IAC				Assemble by 2
	SRT	20			
	CLA	READ	S		Return to read
A3	EXF	$+1	S		Space link-3
	OCT	-7777774			
	SLT	2			Save last 2 bits
	STB	M1+2			
	CLA	READ	S		Return to read
CR	EXF	$+1	S		Save last 4 bits
	OCT	-7777760			
	SLT	20			High order 2→A
	CLC				Save in C
	IAC				
	SRT	2			Split the 2
	IAC				Merge
	SLT	4			Position in A

pb SNAP SYMBOLIC CODING SHEET

PROBLEM: D. P. C-W Assembly DATE:

PROGRAMMER: PAGE 3 of 3

LOCATION	OP	ADDRESS	TAG	REMARKS
	ADD	M1		Add remaining
	ADD	M1+1		Fields
	ADD	M1+2		
STORE	STA	00014		Store in output
	LDA	STORE		Advance Store
	ADD	$+1	S	
	OCT	+0040000		
	STA	STORE		
	TRU	START	3	Return for next word
M1	BSS	3		
	END			

Table A-1. (Sheet 1 of 2)

MACHINE VS SYMBOLIC CODING

Machine Coding	Symbolic Coding			
PB250 ASSEMBLY				
IDENTIFICATION: CONTROL LIST ASSEMBLY				
PHASE 1				
INPUT:FLEX				
PHASE 2				
OUTPUT:FLEX				
END OF JOB				
00002$001S0702;	START	LDP	$+1	S
00102$030 7502;		TOF	A1	
00202$000 0200;		IBC		
00302$022 1202;		STB	SPACE	
00402$023 1102;		STA	PHASE	
00502$007S4500;		CLA	READ	S
00602$014S5502;		LAI	MASK	S
00702$005S5200;	READ	RPT	$-2	S
01002$011 5200;		RPT	$+1	
01102$010 7736;		TES	30, $-1	
01202$007 7736;		TES	30, READ	
01302$011S5700;		CIB	$-2	S
01402$+0000077;	MASK	OCT	77	
01502$016S5602;		CAM	$+1	S
01602$+0000056		OCT	56	
01702$060 7502;		TOF	CR	
02002$021S5602;		CAM	$+1	S
02102$+0000020		OCT	20	
02202$030 7502;	SPACE	TOF	A1	
02302$000 0200;	PHASE	IBC		
02402$030 2210;		SRT	3	
02502$000 0100;		IAC		
02602$052 2210;		SRT	19	
02702$007S4500;		CLA	READ	S
03002$047 2110;	A1	SLT	14	
03102$101 1202;		STB	M1	
03202$033S0502;		LDA	$+1	S
03302$036 7502;		TOF	A2	
03402$022 1102;		STA	SPACE	
03502$007S4500;		CLA	READ	S

Table A-1. (Sheet 2 of 2)

MACHINE VS SYMBOLIC CODING

Machine Coding	Symbolic Coding			
03602$037S4702;	A2	EXF	$+1	S
03702$-7777400		OCT	-7777400	
04002$047 2110;		SLT	6	
04102$102 1202;		STB	M1+1	
04202$043S0702;		LDP	$+1	S
04302$053 7502;		TOF	A3	
04402$046S0200;		IBC	CW-1	S
04502$022 1302;		STD	SPACE	
04602$007S4500;		CLA	READ	S
04702$052 2210;	CW	SRT	2	
05002$000 0100;		IAC		
05102$076 2210;		SRT	20	
05202$007S4500;		CLA	READ	S
05302$054S4702;	A3	EXF	$+1	S
05402$-7777774		OCT	-7777774	
05502$060 2110;		SLT	2	
05602$103 1202;		STB	M1+2	
05702$007S4500;		CLA	READ	S
06002$061S4702;	CR	EXF	$+1	S
06102$-7777760		OCT	-7777760	
06202$107 2110;		SLT	20	
06302$000 4400;		CLC		
06402$000 0100;		IAC		
06502$070 2210;		SRT	2	
06602$000 0100;		IAC		
06702$074 2110;		SLT	4	
07002$101 1402;		ADD	M1	
07102$102 1402;		ADD	M1+1	
07202$103 1402;		ADD	M1+2	
07302$000 1116;	STORE	STA	00014	
07402$073 0502;		LDA	STORE	
07502$076S1402;		ADD	$+1	S
07602$+0040000		OCT	+0040000	
07702$073 1102;		STA	STORE	
10002$000S3702;		TRU	START	S
W	M1	BSS	3	
		END		

APPENDIX B

FAST LINE

RECIRCULATION CHART

Table B-1.

FAST LINE RECIRCULATION CHART

F00	F01	F02	F03	F04	F05	F06	F07	F08	F09	F10	F11	F12	F13	F14	F15
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255