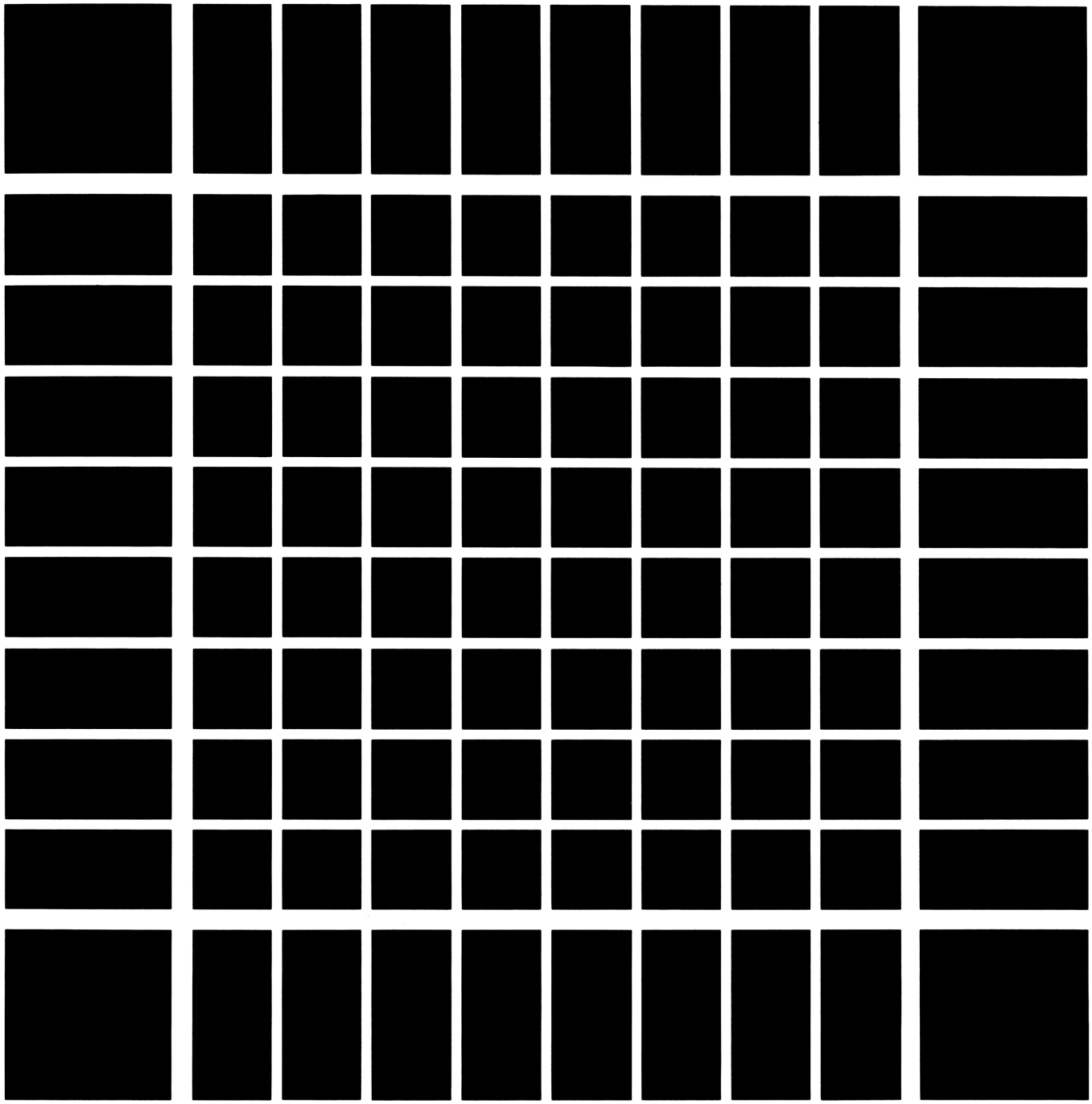
HEWLETT-PACKARD

Creating Your Own HP-41 Bar Code MANUAL



Creating Your Own HP-41 Bar Code Manual

March 1981

82153-90019

Printed in U.S.A.

Contents

Introduction 5

Section 1: Description of Bar Code Types 7
- Program Bar Code 7
- Direct Execution (Complete Function) Bar Code 8
- Data Bar Code 9
  - Standard Data 9
  - Sequenced Data 10
- Paper Keyboard Bar Code 11

Section 2: Physical Specifications 13
- Size and Shape 13
- Paper and Ink Considerations 15

Section 3: Sample Software 17
- Generating Program Type Bar Code 17
  - PRGMBR Syntax 17
  - PRGMBR Program Listing 19
- Generating Other Bar Code Types 32
  - FULFCN Syntax 32
  - FULFCN Program Listing 33

Section 4: Algorithms 43
- Checksum Algorithms 43
  - Eight-bit (One byte) Checksum 43
  - Four-bit Checksum 44
- Program Bar Code Algorithms 44
  - Labels 45
  - Execute 46
  - Go To 46
  - Store and Recall 47
  - Alpha Text Strings 48
  - End 48
  - Header Algorithms 48
- Other Bar Code Algorithms 50
  - Direct Execution (Complete Function Code) Algorithms 50
  - Numeric Data Functions 52
  - Alpha Data Functions 54

Appendix A: Flowcharts 55
- PRGMBR 56
- FULFCN 63

Appendix B: HP-85A Adaptations of PRGMBR and FULFCN 67
- PRGMBR 68
- FULFCN 75

Figures

1 Program Type Bar Code 8
2 Direct Execution Type Bar Code 8
3 Numeric Data Type Bar Code 9
4 Alpha Data Type Bar Code 9
5 Sequenced Data Bar Code (Rev. F and Following) 10
6 Configuration of HP-41 Bar Code 13
7 20-Mil Bar Code Having Excessive Stroke-width Error 14
8 Preferred Wand Orientation 15
9 Different Module Width Bar Codes 16

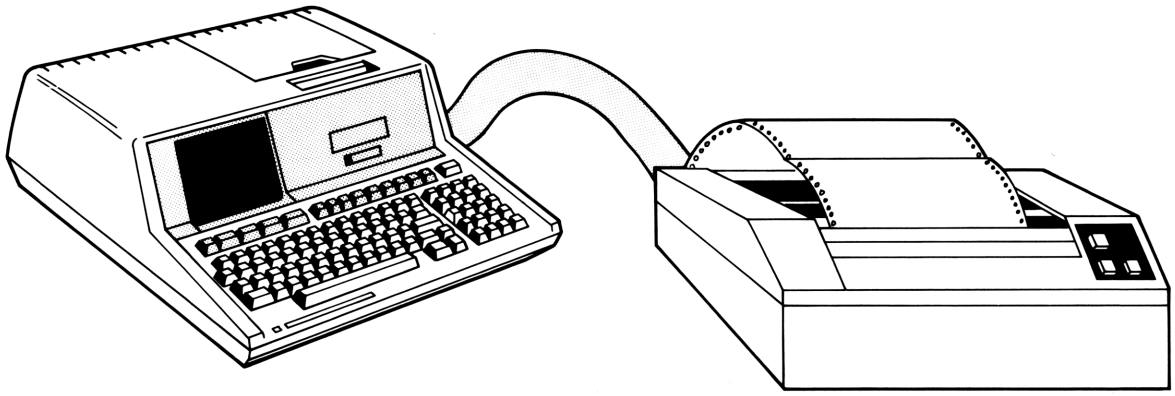
Tables

I Bar Size Specifications 14
II Commands for Program "PRGMBR" 18
III HP-41 Function Table 45
IV Numeric Values for A-J, a-e, and The Stack 50
V HP-82143A Printer Character Set 54

Introduction

The objective of this manual is to provide a technical base for generating HP-41 bar code, that is, the information necessary for you to develop your own bar code printing capability tailored to your specific computer-printer/plotter system.

The minimum system needed is as simple as a minicomputer with a plotter or a printer. An example system would be an HP 85A Personal Computer using an HP 7225A Plotter or an HP 2631G dot matrix printer.



Any mini (or larger) computer with a BASIC compiler that has 16K bytes of user memory will be able to compile and run, with modifications for specific BASIC implementations, the sample software listed in this manual. The input needed to generate the desired bar code for an HP-41 program listing and/or HP-41 functions can be entered through a terminal. As an alternative input method, the generation program may be altered to accept punched cards or paper tape.

If you use a plotter it must be able to create solid, dense lines at least .015″ in width for a narrow bar and at least .030″ in width for a wide bar. Alphanumeric capability is also desirable but not necessary.

The software provided in the main body of this manual was written in HP-9845A BASIC, and was used to print bar code on a Diablo 1650 Daisy Wheel Printer with a "Titan 10" 96-character wheel. Appendix A contains flow charts for these programs, and Appendix B contains the listings of the same programs adapted into HP-85A BASIC. A comparison of the differences between the HP-9845A and HP-85A BASICs may help you with your custom adaptations.

You probably will have a different hardware configuration than was used to produce the two programs in this manual. In that case you will not only have to adapt the language used but also it will be necessary to rewrite the sections that actually print the bar code. For PRGMBR the sections that deal with the actual printing begin on line 1650 and line 4460; for FULFCN that section begins on line 5760.

You may find that at some time you will need a higher quality bar code than your system can produce. If this situation arises it is recommended that you contact:

George Lithograph
620 Second Street
San Francisco, CA 94107
(415) 397-2400

Section 1
Description of Bar Code Types

There are four categories of bar code used by the HP-41 and HP-82153A Wand:

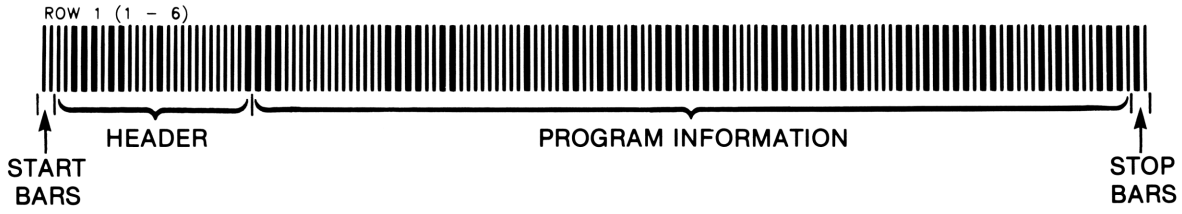
- Program Bar Code
- Direct Execution Bar Code (Complete Key Phrases)
- Data Bar Code
- Paper Keyboard Bar Code (Individual Keystrokes)

Every bar code row has overhead (header) information included in the left-most positions. This header provides the means for error checking and signals the HP-41 as to which type of code the Wand is reading. A header can be as short as 4 bits (single byte Paper Keyboard), or as long as 24 bits (program and sequential data). In this section the physical makeup of each type is outlined.

Every row also has two start bars at the beginning of the bar code row, and two stop bars at the end of the row. These are used by the circuitry within the wand module to determine scan direction. These start and stop bars are not included in the figures for this section, but you must be sure to include them in your finished bar code row.

Program Bar Code

Program type bar code contains the necessary information to load program steps into the HP-41's memory. Each row contains the actual program steps, preceded by a three-byte header.



You have the option to make your program bar code private if you wish. The HP-41 and the wand can work together to provide a security system that will prevent accidental alteration or reproduction of important programs. If a program is made "private" in bar code, when it is read into the HP-41, it cannot be viewed, altered, or reproduced through normal operations.

In every Program bar code row, the first field in the header contains the checksum. This is a running eight-bit sum with end-around carry of the current row and all preceding rows. The second field, containing the next four bits in the header, indicates the type of program; non-private or private. If the field is set to "1," the program is non-private; if the field is set to "2" the program is private. The third field which specifies the row sequence number (modulo 16), is four-bits wide. This field triggers an error message in the HP-41 if a row of Program bar code is scanned out of sequence. The fourth field, which contains four bits, specifies the number of bytes at the beginning of the current row that are part of a multiple byte function carried over from the previous row. The fifth field concludes the header. This field specifies the number of bytes at the end of the current row that are part of a multiple byte calculator function that begins, but is not completed, in that row. The remainder of the bars in the row contain the actual HP-41 program information.

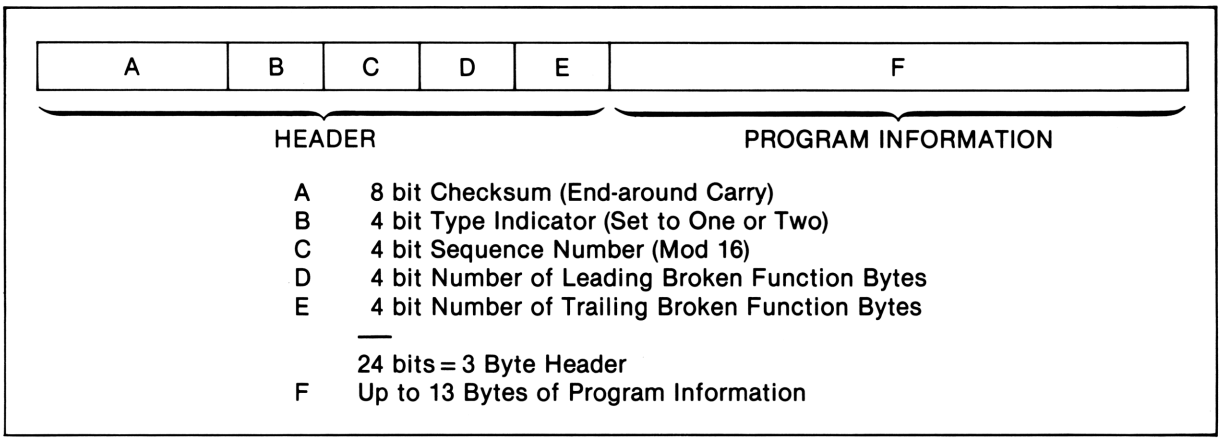


Figure 1. Program Type Bar Code

Direct Execution (Complete Function) Bar Code

Many HP-41 functions require multiple keystrokes. With Direct Execution type bar code, those keystrokes can be replaced by one row of bar code. A row of Direct Execution bar code has two bytes of header information plus up to twelve bytes of the function code.

As with other types of bar code, the first eight bits of the header contain the checksum. This is a local, end-around carry checksum. The next four bits in the header are set to the value "4" to indicate the Direct Execution type. The last four bits of the header field are not used, and should be set to zero.

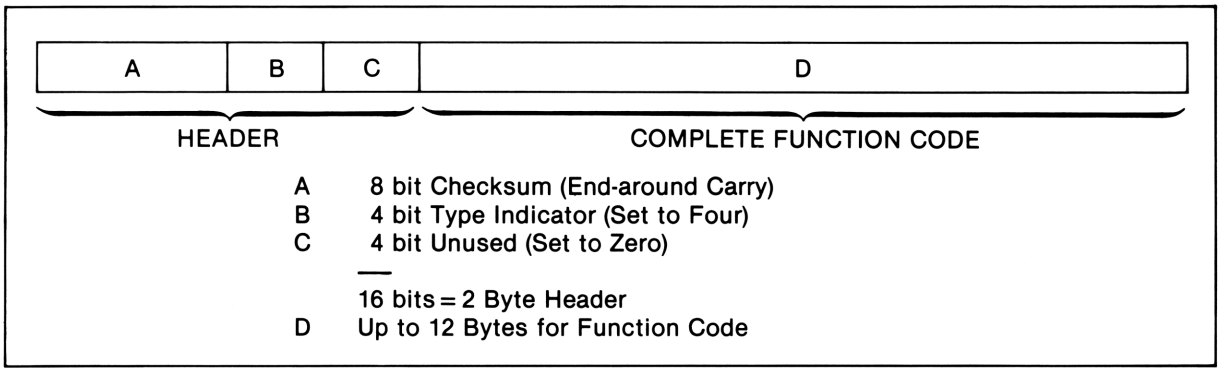


Figure 2. Direct Execution Type Bar Code

Data Bar Code

Data type bar code can contain either a number or an alphanumeric string. A number may have up to 29 digits, however internally the HP-41 has only ten digits in the mantissa and two digits in the exponent. An alphanumeric string may be as long as fourteen characters (ASCII bytes).

Standard Data

When numeric data is read it is loaded into the X-register. Alphanumeric data is either appended to the current contents of the Alpha register, or the Alpha register is cleared before the string is loaded. The wand function [WNDDTX] overrides the normal destination of the data by storing what is scanned into the HP-41 data storage registers indicated by the contents of the X-register. (Refer to page 11, [WNDDTX] , in the HP 82153A Wand Owners Manual.)

The Data bar code overhead information is contained in the leftmost twelve bits for the numeric data type and the leftmost sixteen bits for the Alpha data types. For all data types, the first byte is an end-around carry checksum. The type field (next four bits) is set to "6" for numeric, "7" for alpha-replace, and "8" for alpha-append. Figure 3 illustrates the numeric data bar code. The Alpha types (7 and 8) have an additional four bits of overhead which contain the number of Alpha characters in the bar code string. (Refer to Figure 4.)

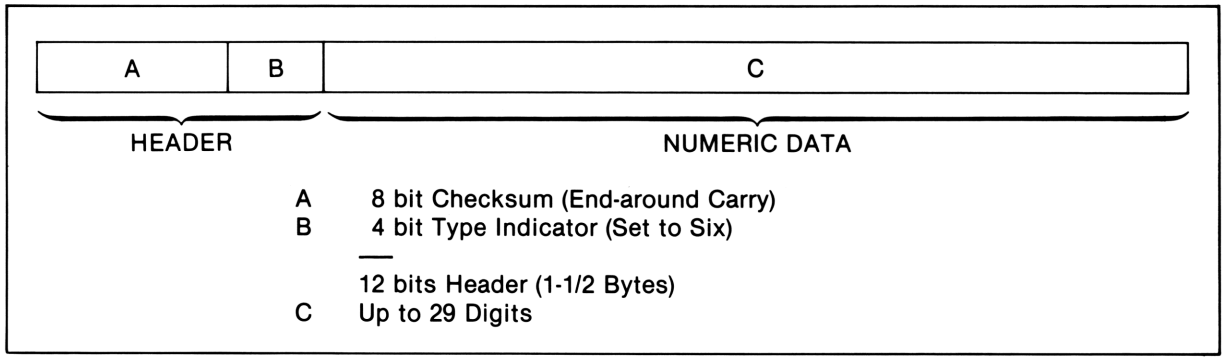


Figure 3. Numeric Data Type Bar Code

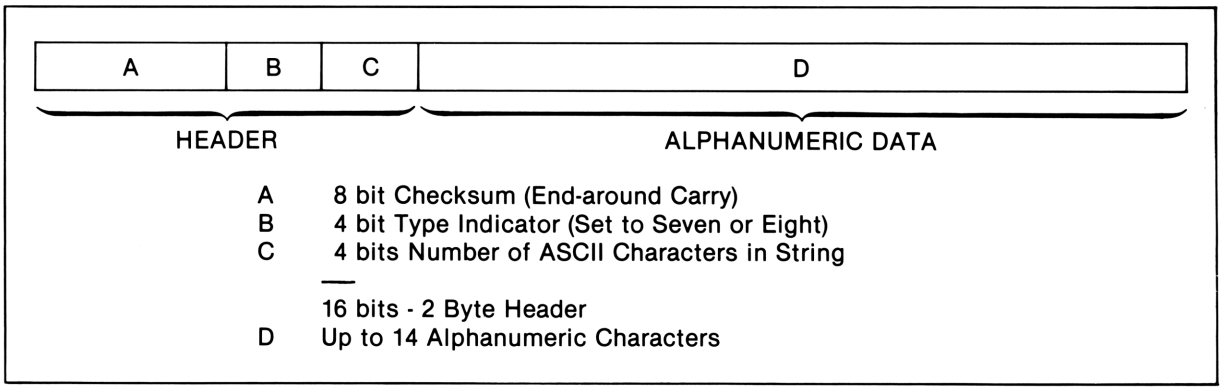


Figure 4. Alpha Data Type Bar Code

Sequenced Data

Sequenced data is an additional format for all data bar code types. It has in the header of each row a twelve-bit sequence number that is used by [WNDDTX] to prevent loading data out of order. Sequenced data bar code assures that the data is always loaded in the intended order. (Using the wand function [WNDDTX], this format is readable by revision "F" and later wands.*)

The first byte of sequenced data bar code is the checksum. The next four-bits contain the type indicator, which is set to "9" for numeric, "10" for alpha-replace, or "11" for alpha-append. Next is the twelve-bit sequence field. This field is ignored by the Wand when the bar code is scanned normally (not under [WNDDTX] control). Please note that the alpha types, 10 and 11, do not have a character count within the header.

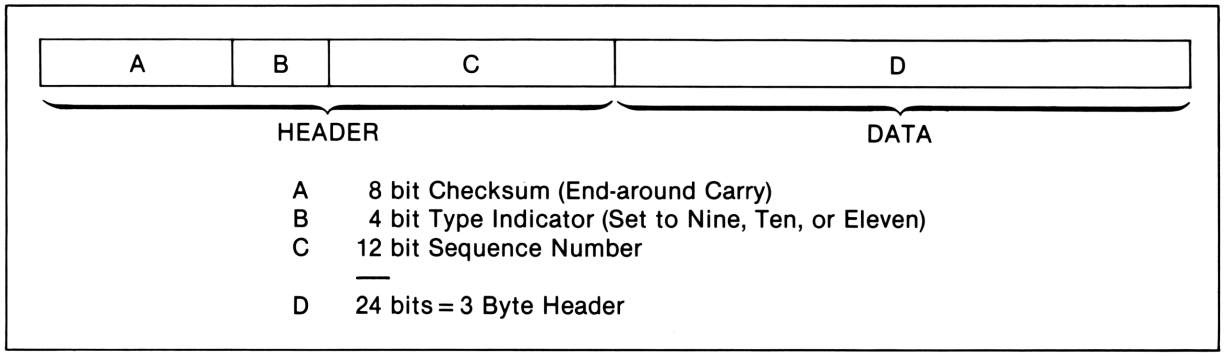


Figure 5. Sequenced Data Bar Code (Rev. F and Following)

*The first marketed version of the HP 82153A Wand was revision *E*. There is no functional difference between revisions *E* and *F* except for the use of the additional sequenced data bar code types. To determine the revision letter of your wand remove all modules from your HP-41, then plug in the wand and press [Shift], [Catalog] 2 [R/S]. The revision letter will appear in the display.

Paper Keyboard Bar Code

This type of bar code represents one keystroke. The number pad keys are represented by a one-byte (8 bits) row. All other keys and one byte functions are represented by a two-byte row.

The error checking is accomplished for these shorter bar code rows by using a slightly different configuration than the other bar code types. For the one-byte code, four bits represent the keystroke and the other four bits are mirror symmetric to the keystroke representation pattern. The two byte rows have twelve bits of data and four bits of checksum, computed as a sum of the data in four bit nybbles with end-around carry.

A sample program is not included to generate Paper Keyboard bar code. Since these are short rows, with values that do not change, a simple program using a table lookup is recommended. The physical description for all the keys follows.

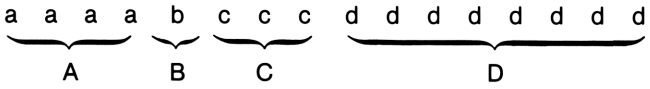
ONE BYTE PAPER KEYBOARD

a a a a b b b b

A B

FUNCTION	VALUE OF A	VALUE OF B
0	0000	0000
1	1000	0001
• • •	• • • •	• • • •
• • •	• • • •	• • • •
9	1001	1001
.	0101	1010
EEX	1101	1011
CHS	0011	1100
←	1011	1101

TWO BYTE PAPER KEYBOARD



1. Programmable Functions: Found in the following locations in the HP-41 Function Table (Table III, page 45).
   Row 4 column 0 through Row 9 column 15
   Row 10 columns 8 through 14
   Row 12 columns 0, 14, and 15
   Row 13 column 0
   Row 14 column 0
   A = Four bit Checksum (End-around Carry)
   B = 0
   C = 000
   D = Eight bit Function Code
   (such as ABS, Row 4 Column 2 = 01000010)

2. Alpha Characters (Keys)
   A = Four bit Checksum (End-around Carry)
   B = 0
   C = 001
   D = Eight bit ASCII with Most Significant Four Bits Doubled. (For example the ASCII for "A" is 01000001. With most significant nybble doubled the bar code value for the character A becomes 10000001.)

3. Indirect
   A = 10 (1010)
   B = 0
   C = 2 (010)
   D = 128 (10000000)

4. Non-Programmables
   A = Four bit Checksum (End-around Carry)
   B = 0
   C = 4 (100)

	FUNCTION	VALUE	FUNCTION	VALUE
D =	CAT	0	SST	8
	GTOL	1	STAYON	9
	DEL	2	PACK	10
	COPY	3	DELETE	11
	CLP	4	ALPHA	12
	R/S	5	PRGM	13
	SIZE	6	USER	14
	BST	7	ASN	15

5. XROM
   A = Four bit Checksum (End-around Carry)
   B = 1
   C = f f f
   D = g g h h h h h h
   f f f g g = ROM I.D. Number
   h h h h h h = ROM Function Number

   For Example: [WNDSCN] has ROM I.D. = 27 and ROM Function = 5. Therefore fffgg = 11011, and hhhhhh = 000101.

Section 2
Physical Specifications

There is a wide variation in the appearance of readable bar code. However, some properties are shared, and cannot vary beyond certain limits. This section contains a description of the characteristics of readable bar code so that you will be able to judge the acceptability of HP-41 bar code.

The performance of bar code depends heavily on print quality. The printed characters must exhibit high contrast and good definition. Stroke width and location should be maintained as closely as possible to their nominal values. Hewlett-Packard Corvallis Division concurs in general with the recommendations of ANSI X3.17-1977 "Characters Set and Print Quality for Optical Character Recognition" (OCR-A)*. Many of the recommendations in that document are "based largely upon expert opinion and judgement and are not fully substantiated by extensive test data and measurement". Also contained in this section is a discussion of the special characteristics of HP-41 bar code which differ from the recommendations of OCR-A.

Size and Shape

All four categories of bar code that are acceptable to the HP 82153A Wand look essentially like that shown in Figure 6. The information contained in the bars is encoded using narrow bars for "zeroes" and bars twice as wide for "ones". These bars are separated by spaces which have the same width as the narrow bars. The bars containing the information are always preceded by two "0" bars and followed by a "1-0" combination. These "start" and "stop" sequences are used by the decoding electronics to determine scan direction. This allows HP-41 bar code to be scanned either forwards or backwards.

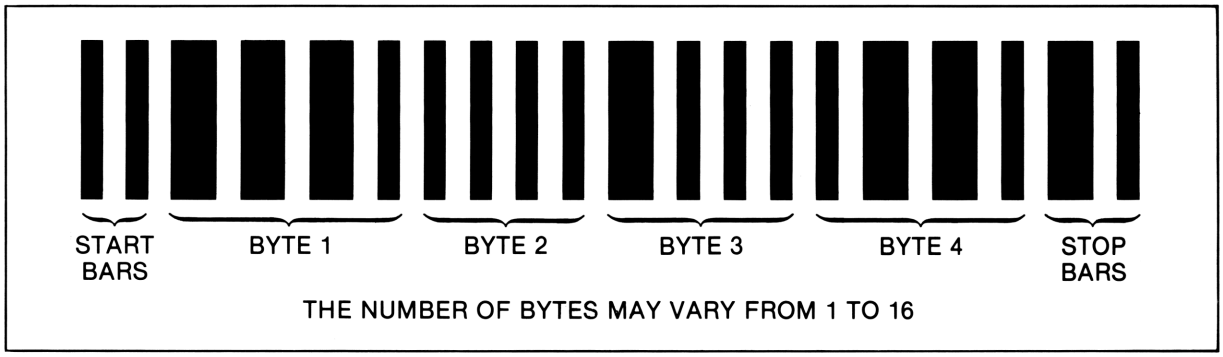
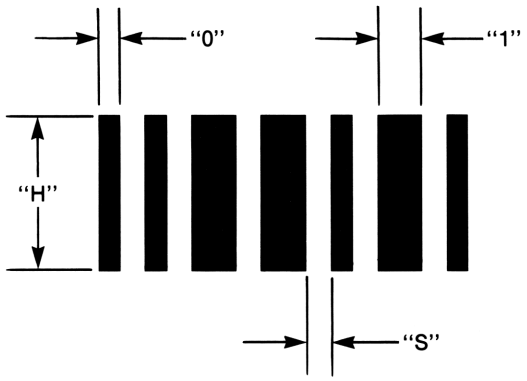


Figure 6. Configuration of HP-41 Bar Code

There will always be an integral multiple of eight-bit bytes between the start and stop bars, but there cannot be more than sixteen bytes in any given row. If there are too many or too few bars in a row of bar code, the decoder in the wand module discards what it has read.

*Available from American National Standards Institute, Inc., 1430 Broadway, New York, NY 10018.



	WIDTH*	TOLERANCE	SUGGESTED VALUES
"S"	W + .002″	W – 0.005″ / 5	0.022″ ± 0.003″
"0"	W – .002″	W – 0.005″ / 5	0.018″ ± 0.003″
"1"	2W – .002″	W – 0.005″ / 5	0.038″ ± 0.003″
"H"			H > 0.35″

*Bar widths are arbitrarily reduced by .002″ to allow for the differences between bar width perception of the wand and the human eye.

Table I. Bar Size Specifications

The dimensions shown in Table I may be varied widely, but we have found them to be a very good compromise between compact code (which occupies little area but is difficult to read) and extensive codes (which is easy to read but requires a lot of space). Printers have found that it is easier to maintain location than stroke width. If there were no error in placing the bar, and no voids in the bars or spots in the spaces, then the bar width could vary as much as 10% of the width of a zero bar without severe loss of readibility, and as much as 20% if degraded readability were acceptable. Figure 7 shows examples of bar code having excessive stroke-width error.

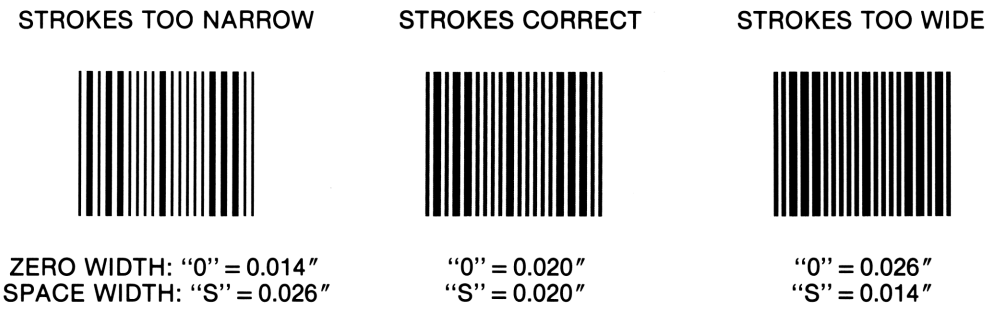
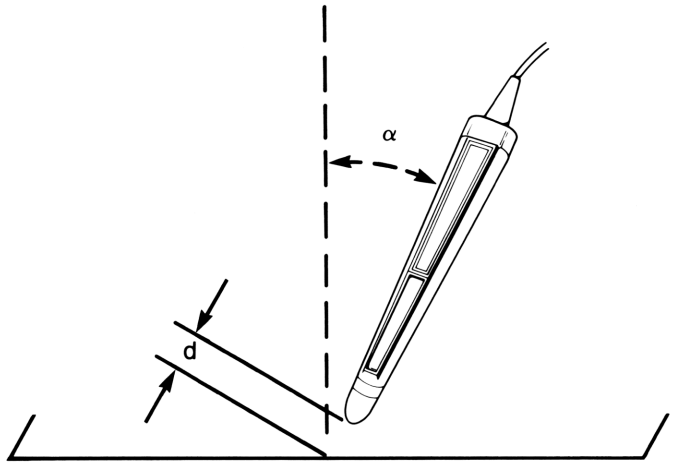


Figure 7. 20-Mil Bar Code Having Excessive Stroke-Width Error

It might seem reasonable to make the bars and spaces narrower and narrower, but the optics in the photodetector is such that the wand "sees" a spot between 0.011" to 0.016" in diameter. Bars smaller than this will often read, because a bar only 0.004" to 0.009" wide will sometimes be detected. However, such bars usually read only when the wand is held in a particular orientation. To read very narrow bars, the wand must be tilted as shown in Figure 8 until the spot of light it produces is in the best possible focus. All bar code is more easily read when the wand is held in the "preferred orientation", but marginally-narrow bar code cannot be read in any other way.



$d \leq 0.5$ mm
$0 \leq \alpha \leq 30°$ MAX
PREFERRED ORIENTATION IS:
$10° \leq \alpha \leq 20°$
AND TIP SHOULD TOUCH THE PAPER DURING READING

Figure 8. Preferred Wand Orientation

Paper and Ink Considerations

There is yet another reason to hold the wand in the preferred orientation; it has been found that the surface of some paper is specular or mirror-like. Bar code printing on such paper is likely not to read if the wand is held perpendicular to the paper. The smooth surface reflects so much light that even black bars appear to be white. Accordingly, it is wise to choose a paper that does not have a glossy surface. Unfortunately, it is also necessary to avoid rough-surfaced papers, because excessive variation in paper reflectance can be troublesome as well.

The wand electronics determines the difference between bars and spaces by measuring their relative contrast. Print Contrast Signal or PCS is defined to be the difference between white and black reflectance divided by white reflectance and expressed as a percentage where Rw = White Reflectance and Rb = Black Reflectance:

$$PCS = \frac{Rw - Rb}{Rw} \times 100\%$$

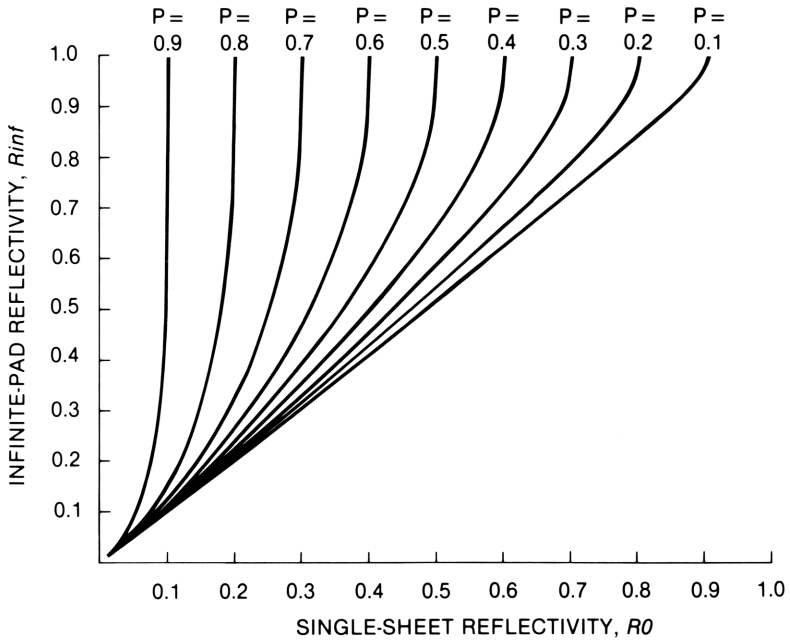
It is desirable to maintain PCS as high as possible. The wand's response to bar code is also dependent on battery voltage. A battery level near the low-level-detect threshold might prevent reading bar code having a PCS below 60%, while fresh batteries might allow 30%-PCS bar code to be read. Bar code having a PCS as low as 20% can sometimes be read with fresh batteries by paying particular attention to wand orientation.

Since people see a broad range of colors, it is not usually possible to judge PCS by eye. The light-emitting diode in the wand produces light having a wavelength of 700 nanometers, close to standard wavelength B680 mentioned in OCR-A.

Variation of reflectivity within bars and spaces can cause misreads. The amount of permissible reflectivity variation is between 10 and 20 percent. Reflectivity variation, an inherent property of paper and ink, also results from the presence of voids and spots. Voids in inked areas and spots in un-inked areas should be minimized, but we have found that voids of not more than 0.005″ spaced no closer than 0.009″ usually do not cause a misread. Bar code containing voids and spots will not be readable with as much freedom of orientation and speed as will clean bar code. Because the wand's response to low-level signals is a function of battery voltage, noisy bar code is usually easier to read with low voltage batteries.

The opacity of paper determines whether or not bar code can be adversely affected by printing on the reverse side of the paper. Opacity of paper suitable for optical character recognition exceeds 60%, but Hewlett-Packard Corvallis Division recommends the use of paper exceeding 85% opacity. Paper porosity is probably important for reverse show-through also, because the effective opacity will be reduced if ink can move into the paper.

The following plot shows the PCS that perfect bar code would appear to have when viewed through paper having particular values of *R0* and *Rinf*. Since wands can often read bar code with a PCS as low as 30%, it is probably wise to avoid papers with reflectivity less than 70%. Papers of lower reflectivity might have excessive show-through.



All of these characteristics interact in ways that are very difficult to quantify. The best way to test bar code readability is to try to read bar code samples with the wand held in various orientations and with battery voltages ranging from 4.1 to 6 volts.

Figure 9. Different Module Width Bar Codes

Section 3
Sample Software

Generating Program Type Bar Code (PRGMBR)

PRGMBR allows you to type in numbered HP-41 instructions and will insert the instructions into a text string for later use. The line number of the HP-41 instruction may be from 1 to 2233. This number determines the order of execution of the instructions. The number 2233 is maximum because it is the largest number of bytes available in program memory. If an HP-41 program is very long, a renumbering command can be used to create gaps in the numbering scheme to allow for later insertion of HP-41 instructions. Instructions can also be deleted or replaced. In addition, you can save an entire HP-41 program in a file for later use. Other features include the single word commands to compile and generate bar code for an HP-41 program, save and retrieve the compiled HP-41 machine language, and list or delete the entire program. The syntax and action of each command is given in Table II, page 18, and will be printed out by the program if a "??" is typed in response to the prompt symbol.

PRGMBR is structured with a main routine that generates the prompts and decodes the input. A jump table in the main routine is used to call a series of other routines which perform the command functions. The limited decoding done by the main routine is performed only to determine if a command or an HP-41 instruction has been given. The subroutines will then be able to retrieve the HP-41 program by a linear inspection of the pointer array. Replacement, deletion, and renumbering of instructions, therefore, only involve manipulation of the pointer array. (Insertion of an instruction requires that the instruction number—an integer—must fall between two existing instruction numbers.)

PRGMBR Syntax

In almost all cases, the syntax of the HP-41 instructions (recognized by "PRGMBR") follows that of the HP 82143A thermal printer and of the HP-41 program listings distributed by the HP User's Library. The few exeptions, dictated by the difference between ASCII and the HP-41 character set are shown below:

HP-41C Character	Substitute ASCII Character
Σ	"&"
⊿	"@"
≠	"#"

Single quotes instead of double quotes are used for inputting HP-41 Alpha text and Alpha labels. The alpha append instruction is indicated by an "A" preceding the single quotes and character string. See Table V, page 54, for the character set and their decimal values.

Commands for Program "PRGMBR"

COMPILE	— Compiles the current HP-41 program and loads the compiled code into the array *M*.
DELETE *n*	— Deletes the instruction given by *n* from the current program.
EXIT	— Halts execution of the bar code generator or of the line number generator.
GETPROG	— Retrieves compiled code from a file on cassette tape (routine prompts for a file name).
GETTEXT	— Retrieves program instructions from a file on cassette tape (routine prompts for a file name).
LIST	— Lists the entire current HP-41 program.
NUMBER	— Automatically generates instruction numbers for HP-41 program entry. The starting number and size of the increment are requested by the routine. This routine is halted by typing "EXIT".
RENUMBER	— Renumbers current HP-41 program instructions (routine prompts for the old starting number, new starting number and size of the increment).
RUN	— Generates the bar code from the compiled code (will not run unless compiled code has been generated).
RUNPRIVATE	— Generates bar code for a private program.
SAVEPROG	— Stores compiled code for the current program on cassette tape (routine prompts for a file name).
SAVETEXT	— Stores instructions of the current program on cassette tape (routine prompts for a file name).
SCRATCH	— Erases the current program.
??	— Displays a list of available commands and syntax rules.

Table II.

PRGMBR* Program Listing

```
10   !
15   ! HP-41 USER LANGUAGE COMPILER AND BAR CODE GENERATION PROGRAM
20   !     THIS PROGRAM PROMPTS FOR NUMBERED HP41 INSTRUCTIONS AND STORES THEM
25   !     FOR LATER COMPILATION (INITIATED ON COMMAND). THE INSTRUCTION
30   !     NUMBERS MAY BE FROM 1 TO 2240 (THE TOTAL NUMBER OF BYTES AVAILABLE
35   !     FOR USE IN A PROGRAM) AND MUST BE INTEGERS. THE COMPILED CODE WILL
40   !     BE USED TO DRIVE A BAR CODE GENERATION ROUTINE WHICH WILL CALCULATE
45   !     THE BIT PATTERN FOR A ROW OF BAR CODE.  THIS BIT PATTERN WILL APPEAR
50   !     WITHIN A LOOP, AT WHICH POINT THE USER WILL BE ABLE TO CALL HIS OWN
55   !     PLOTTING ROUTINES TO GENERATE HP-41 BAR CODE.  THE COMMANDS
60   !     AVAILABLE IN THIS PROGRAM ARE:
70   !     1)"NUMBER":  GENERATES LINE NUMBERS FOR HP-41C INSTRUCTIONS
75   !     2)"LIST":  LISTS THE INSTRUCTIONS CURRENTLY ENTERED
80   !     3)"RUN":  CHECKS FOR PRESENCE OF COMPILED CODE AND PRODUCES
85   !       THE BAR CODE BIT PATTERN.
90   !     4)"COMPILE":  COMPILES THE CURRENT TEXT INTO MACHINE CODE
95   !     5)"RENUMBER": ALTERS THE 41 INSTRUCTION NUMBER SEQUENCE
100  !     6)"SAVETEXT": SAVES THE CURRENT TEXT ON CASSETTE TAPE
105  !     7)"GETTEXT": RETRIEVES THE TEXT FROM CASSETTE TAPE
110  !     8)"SAVEPROG": SAVES THE COMPILED MACHINE CODE ON TAPE
115  !     9)"GETPROG": RETRIEVES THE COMPILED CODE FROM TAPE
120  !     10)"EXIT": HALTS THE USER LANGUAGE COMPILER PROGRAM
125  !     11)"DELETE": DELETES THE NAMED INSTRUCTION NUMBER FROM THE
130  !        CURRENT TEXT.
135  !     12)"SCRATCH": ERASES THE CURRENT INSTRUCTIONS ENTERED
140  !     13)"RUNPRIVATE": GENERATES BAR CODE FOR A PRIVATE PROGRAM
150  ! ***************************************************************
160  INTEGER I,J,K,L,V,R,R1,C,C1,C2(60),T,P(2240),M(2240),M1,P1,P2,K1(2240)
165  INTEGER F1,F2,F3,F4,F5,F6,F7,F8,F9,E,E1,E2,E3(15),E4,I1(103),D,X,Y,Z
170  INTEGER I5,B1,B2,B3,T5,H1,H2,P5,L5,C5,S3,S1(16),B(132),V1,V2,V3
175  DIM T$[60],T1$[30],T2$[30],S$[50],A$[1500],B$[915]
180  DIM S1$(27)[1],I$(104)[6],H1$[3],H2$[3],H4$[9],C$(60)[1]
190  ! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
200  !  MAIN PROGRAM:  WRITES PROMPT FOR TEXT OR COMMAND ENTRY AND EITHER
205  !    DECODES THE INSTRUCTION NUMBER AND ENTERS THE TEXT INTO THE TEXT
210  !    ARRAY, OR USES THE COMMAND JUMP TABLE TO JUMP TO THE CORRECT COMMAND
220  ON ERROR GOTO 4250
225  H1$=CHR$(27)&CHR$(31)&CHR$(2)      !SET UP DIABLO CONTROL CODES
230  H2$=CHR$(27)&CHR$(83)
235  H4$="          "
237  T$=T1$=T2$=""
240  !
245  FOR I=1 TO 26          !READ LOCAL LABELS & STACK REGISTER MNEMONICS INTO S1$
250   READ S1$(I)
255  NEXT I
260  FOR I=1 TO 103         !READ SORTED INSTRUCTION MNEMONICS INTO I$, INSTRUC.
265   READ I$(I),I1(I)      !VALUES INTO I1 FOR TABLE DRIVER
270  NEXT I
275  FOR I=1 TO 60          !READ IN VALID CHARACTER TABLES FOR CHARACTER CHECK
280   READ C$(I),C2(I)      !CHARACTERS IN C$; CHARACTER CODE IN C2
285  NEXT I
290  !
295  FOR I=1 TO 2240
300   P(I)=-1               !INITIALIZE ARRAY OF POINTERS INTO TEXT STRING
305   M(I)=-1               !INITIALIZE COMPILED PROGRAM ARRAY
310  NEXT I
315  A$=""
317  T=F6=F8=0             !INITIALIZE TEXT STRING POINTER AND FLAGS
320  !
322  INPUT "DO YOU WANT A LIST OF THE AVAILABLE COMMANDS?",T1$
323  IF (T1$="N") OR (T1$="NO") THEN 350
324  IF T$="SCRATCH" THEN 350
325  GOTO 3500              !PRINT OUT REFERENCE TABLE
* Courtesy BYTE Publications, Inc. Copyright 1980.
```

```
340  !          BEGIN PROMPTER SECTION:  ASK FOR COMMAND OR INSTRUCTION
350  INPUT ">",T$
355  I=POS(T$," ")          !SET I TO FIRST WORD OF INPUT
360  V=K=D=0
365  P5=1                   !SET PRIVACY FLAG TO INDICATE A NON-PRIVATE PROGRAM
370  IF I=0 THEN 530        !ONE WORD COMMAND
380  T1$=T$[1,I-1]          !EXTRACT FIRST WORD OF INPUT
385  IF T1$<>"DELETE" THEN 405   !CHECK FOR A DELETE COMMAND
390  T1$=TRIM$(T$[I+1])
395  I=LEN(T1$)+1
400  D=1
405  IF I-1>4 THEN 510        !CALCULATE INSTRUCTION NUMBER VALUE
410  FOR J=I-1 TO 1 STEP -1
415    IF (T1$[J,J]<"0") OR (T1$[J,J]>"9") THEN 500
420    V=V+(NUM(T1$[J,J])-48)*10^K
425    K=K+1
430  NEXT J
435  IF V>2240 THEN 510
450  IF D<>1 THEN 465      !DELETE INSTRUCITON IF FLAG IS SET
455  P(V)=-1
460  GOTO 350
465  T$=TRIM$(T$[I+1])     !ENTER TEXT AND DELIMITER INTO TEXT ARRAY AND STORE
470  A$=A$&T$&"!"          !POINTER AT THE INDEX GIVEN BY INSTRUCTION NUMBER
475  P(V)=T
480  T=T+LEN(T$)+1
485  GOTO 350
495       ! ERROR MESSAGES
500  PRINT "?? - GIVE NUMBERED STATEMENT OR A COMMAND"
505  GOTO 350
510  PRINT "STATEMENT NUMBER VALUE TOO LARGE"
515  GOTO 350
525  !     ***  COMMAND JUMP TABLE  ***
530  IF T$="NUMBER" THEN 670
535  IF T$="LIST" THEN 785
540  IF T$="RUN" THEN 925
545  IF T$="COMPILE" THEN 1855
550  IF T$="??" THEN 3500
555  IF T$="RENUMBER" THEN 3710
560  IF T$="SAVETEXT" THEN 3845
565  IF T$="GETTEXT" THEN 3935
570  IF T$="SAVEPROG" THEN 4055
575  IF T$="GETPROG" THEN 4140
580  IF T$="SCRATCH" THEN 295
585  IF T$="EXIT" THEN STOP
590  IF T$<>"RUNPRIVATE" THEN 605
595   P5=2
600   GOTO 925
605  PRINT "?? - UNRECOGNIZABLE COMMAND"
610  GOTO 350                      !GO BACK TO PROMPTER
620  ! * * * * * * * * *  END OF MAIN PROGRAM * * * * * * * * * * * * *
645  ! * * * * * * * * * 'AUTO' ROUTINE * * * * * * * * * * * * * * * * *
655  !  THIS ROUTINE AUTOMATICALLY NUMBERS THE 41C INSTRUCTIONS AND
660  !  THEM INTO THE TEXT ARRAY.  TO LEAVE THIS ROUTINE, TYPE 'EXIT'.
670  INPUT "GIVE THE STARTING VALUE AND SIZE OF THE INCREMENT",V,X1
675  IF V>2240 THEN 730
680  PRINT ">";V                    !PRINT THE PROMPT AND THE LINE NUMBER
685  INPUT T$
690  IF T$="EXIT" THEN 340          !LEAVE ROUTINE AND GO TO NORMAL PROMPT
700  A$=A$&T$&"!"                   !ENTER INSTRUCTION INTO TEXT ARRAY
705  P(V)=T
710  T=T+LEN(T$)+1
715  V=V+X1
720  GOTO 675
730  PRINTER IS 16                  !ERROR MESSAGE FOR STATEMENT NUMBER
735  PRINT "STATEMENT NUMBER VALUE TOO LARGE"
```

```
740  GOTO 340                          !RETURN TO NORMAL PROMPT
750  ! * * * * * * * * * * END OF 'AUTO' ROUTINE * * * * * * * * * * * * *
775  ! * * * * * * * * * *  "LIST' ROUTINE * * * * * * * * * * * * * * * *
781  ! THIS ROUTINE LISTS THE CURRENT PROGRAM HELD IN THE TEXT STRING
785  FOR I=1 TO 2240              !STEP THROUGH POINTER TABLE AND PRINT
790   IF P(I)<0 THEN 810          !OUT TEXT IF A VALID POINTER IS SEEN
800   T1$=FNI$(A$,P(I))
805   PRINT I;T1$
810  NEXT I
815  GOTO 350
825  ! * * * * * * * * * * END OF 'LIST' ROUTINE * * * * * * * * * * * * *
845  ! * * * * * * * * * * * * 'RUN' ROUTINE * * * * * * * * * * * * * * * *
855  ! BAR CODE DATA GENERATION ROUTINE: THIS ROUTINE TAKES  COMPILED PROGRAM
860  ! HELD IN THE 'M' ARRAY AND CONVERTS IT INTO THE BIT PATTERN REPRESENTING
865  ! HP-41 BAR CODE.  THE BIT PATTERN APPEARS WITHIN A LOOP IN 16 BYTE SEG-
870  ! MENTS, INCLUDING 3 BYTES OF HEADER DATA AND START AND STOP BITS. OTHER
875  ! INFORMATION SEEN AT THAT POINT WILL BE:
885  !      1)THE NUMBER OF BYTES IN THE CURRENT SEGMENT (HELD IN 'B2')
890  !      2)THE LINE NUMBER OF THE FIRST INSTRUCTION IN THE CURRENT SEGMENT
895  !        (HELD IN 'L5')
900  !      3)THE LINE NUMBER OF THE LAST INSTRUCTION SEEN IN THE CURRENT
905  !        SEGMENT (HELD IN 'I5')
910  !      4) THE BAR CODE ROW NUMBER ('HELD IN 'S3')
925  IF F8=1 THEN 940              !CHECK FOR PREVIOUS COMPILATION
930  PRINT "A PROGRAM MUST BE COMPILED FIRST!"
935  GOTO 350
940  INPUT "ENTER THE TITLE OF THE PROGRAM (60 CHARACTERS OR LESS)",T$
942  ! ******THIS SECTION WRITES OUT TO THE DIABLO 1650 **********
943  PRINTER IS 9                 !SET PRINTER TO DIABLO LU
945  PRINT CHR$(12)
950  PRINT USING "10X,50A";T$
955  PRINT " "
960  X=S2 DIV 7                   !S2 CONTAINS THE NUMBER OF BYTES IN THE PROGRAM
965  IF S2 MOD 7>0 THEN X=X+1
970  PRINT "          PROGRAM REGISTERS NEEDED: ";X
975  PRINT " "
977  PRINTER IS 16                !RESET PRINTER BACK TO CRT
985  I5=0                         !START HP-41 INSTRUCTION COUNTER AT 0
990  B1=0                         !START COMPILED DATA ARRAY 'M' POINTER AT 0
995  B2=3                         !START BAR CODE ROW BYTE POINTER AT 3
1000 B3=0                         !START INSTRUCTION LENGTH COUNTER AT 0
1005 T5=0                         !START # OF WORDS SINCE LAST INST. COUNTER AT 0
1010 S3=0                         !START BAR CODE ROW COUNTER AT 0
1015 H1=0                         !START LEADING PARTIAL FCN. BYTE COUNTER AT 0
1020 H2=0                         !START TRAILING PARTIAL FCN. BYTE COUNTER AT 0
1025 L5=1                         !START FIRST INSTRUCTION OF ROW COUNTER AT 0
1030 C5=0                         !START CHECKSUM COUNTER (SUM MOD 256) AT 0
1035 FOR I=1 TO 132               !ZERO OUT THE BIT PATTERN ARRAY
1040  B(I)=0
1045 NEXT I
1050 B$=""
1065 ! INSTRUCTION TRANSLATION SECTION:  LOAD INSTRUCTIONS INTO A BAR CODE
1070 !   ROW AND KEEP COUNTERS FOR THE HEADER DATA (NOTE THAT B3 IS SET TO THE
1075 !   NUMBER OF BYTES EXPECTED TO COMPLETE THE CURRENT INSTRUCTION, AND
1080 !   SERVES AS A FLAG FOR THE BEGINNING OF THE NEXT INSTRUCTION)
1095 S1(B2+1)=M(B1+1)             !TRANSFER WORD FROM 'M'INTO 16 BYTE BUFFER S1
1100 B1=B1+1                      !UPDATE VARIABLES
1105 B2=B2+1
1110 B3=B3-1
1120 IF B3<>0 THEN 1135           !IF B3=0 THEN INSTRUCTION ENDS; RESET COUNTER
1125 T5=0
1130 GOTO 1435
1135 IF B3<0 THEN 1155           !IF B3>0 THEN INSTRUCTION CONTINUES
1140 T5=T5+1
1145 GOTO 1435
1155 IF S1(B2)<>0 THEN I5=I5+1 !IF B3<0 THEN INSTRUCTION IS STARTING; INCREMENT
```

```
1160 IF M(B1)>143 THEN 1300    !COUNTER IF NON-NULL INST. AND CHECK FOR LENGTH
1170 !              **PROCESS ONE BYTE INSTRUCTIONS**
1175                           ! CHECK FOR AN ALPHA EXECUTE OR A GOTO ALPHA
1180 !                         !INSTRUCTION.  (GET SIZE FROM SECOND BYTE)
1185 IF (M(B1)<>29) AND (M(B1)<>30) THEN 1215
1190 B3=M(B1+1) MOD 16+1
1195 T5=T5+1
1200 GOTO 1435
1210                           ! CHECK FOR A DIGIT ENTRY INSTRUCTION
1215 IF (M(B1)<16) OR (M(B1)>28) THEN 1270
1220 I=B1+1
1225 IF (M(I)<16) OR (M(I)>28) THEN 1240
1230 I=I+1                     !STEP THROUGH INSTRUCTIONS UNTIL END OF
1235 GOTO 1225                 !DIGIT ENTRY HAS BEEN SEEN
1240 B3=I-B1-1
1245 T5=T5+1
1250 IF B3<>0 THEN 1435        !CHECK FOR SINGLE DIGIT INSTRUCTION
1255 T5=0
1260 GOTO 1435
1265 !
1270 B3=0                      !PROCESS ALL OTHER ONE BYTE INSTRUCTIONS
1275 T5=0
1280 GOTO 1435
1290 !              **PROCESS TWO BYTE INSTRUCTIONS**
1300 IF M(B1)>207 THEN 1370    !CHECK FOR ALPHA LABEL AND END INSTRUCTIONS
1305 IF (M(B1)<192) OR (M(B1)>205) THEN 1345
1310 T5=T5+1
1315 IF B1+2<S2 THEN 1330      !CHECK FOR THE END INSTRUCTION
1320 B3=2
1325 GOTO 1435
1330 B3=M(B1+2) MOD 16+2       !ALPHA LABEL:GET SIZE FROM THIRD BYTE
1335 GOTO 1435
1345 B3=1                      !PROCESS SHORT FORM GTO'S (TWO BYTES LONG)
1350 T5=T5+1
1355 GOTO 1435
1365 !              **PROCESS THREE BYTE INSTRUCTIONS**
1370 IF M(B1)>240 THEN 1395    !PROCESS LONG FORM GTO'S AND XEQ'S
1371 IF M(B1)=240 THEN 1270
1375 B3=2
1380 T5=T5+1
1385 GOTO 1435
1395 B3=M(B1) MOD 16           !PROCESS ALPHA DATA ENTRY INSTRUCTIONS
1400 T5=T5+1                   !(GET LENGTH FROM FIRST BYTE)
1415 !     BAR CODE ROW SETUP SECTION:  TAKE DATA FOR THIS ROW AND
1420 !           CALCULATE THE HEADER DATA AND OTHER VARIABLES
1430                           ! CHECK FOR A COMPLETED ROW (A FILLED BUFFER)
1435 IF (B2<16) AND (B1<S2) THEN 1095
1445 H1=H2                     !UPDATE PARTIAL FUNCTION COUNTERS
1450 H2=B3
1455 S1(3)=H1 MOD 16*16+T5 MOD 16
1456 IF H1<=B2-3 THEN 1470
1457 H1=B2-3
1458 S1(3)=H1 MOD 16*16+H1 MOD 16
1465                           ! ENTER PRIVACY VALUE AND SEQUENCE NUMBER
1470 S1(2)=P5*16+S3 MOD 16
1480 FOR I=2 TO B2             !CALCULATE CHECKSUM (A CUMULATIVE SUM MOD 256)
1485  C5=C5+S1(I) MOD 256
1490  IF C5>=256 THEN C5=C5 MOD 256+1
1495 NEXT I
1500 S1(1)=C5
1515 !     CONVERSION SECTION: CONVERT THE CURRENT ROW OR SEGMENT INTO A
1520 !          BIT PATTERN REPRESENTING THE BAR CODE.
1530 S3=S3+1
1535 FOR I=1 TO B2             !CONVERT DECIMAL DATA TO A BINARY PATTERN
1540  X=S1(I)
1545  FOR Y=2+I*8 TO 3+(I-1)*8 STEP -1
```

```
1550   B(Y)=X MOD 2
1555   X=X DIV 2
1560  NEXT Y
1565 NEXT I
1575 B(1)=0                     !ADD START AND STOP BITS
1580 B(2)=0
1585 B(B2*8+3)=1
1590 B(B2*8+4)=0
1605 !    AT THIS POINT, THE ARRAY 'B' HOLDS A SERIES OF 1'S AND 0'S
1610 !    REPRESENTING A SINGLE ROW OF 41C PROGRAM BAR CODE, INCLUDING
1615 !    THE START AND STOP BITS. OTHER DATA WILL BE FOUND IN THE
1620 !     VARIABLES B2,S3,L5 AND I5 AS EXPLAINED ABOVE.
1630 !    ***THIS IS THE BAR CODE GENERATION AND OUTPUT SECTION USED BY***
1635 !    ***HP FOR BAR CODE GENERATION ON A DIABLO 1650 DAISY WHEEL******
1640 !    ***PRINTER WITH A TITAN 10 96-CHARACTER METALLIC DAISY WHEEL****
1650 T1$=FNP$(S3+1-1,L5,I5)
1652 PRINTER IS 9               !SET PRINTER TO DIABLO LU
1655 PRINT USING "3X,20A";T1$
1660 L=B2*8+4
1665 GOSUB 4460                 !GENERATE BAR PATTERN FROM BIT PATTERN
1670 PRINT USING "3X,3A,10A,915A,2A";H1$,H4$,B$,H2$
1675 PRINT " "
1680 IF S3 MOD 18=0 THEN PRINT CHR$(12)
1682 PRINTER IS 16              !RESET PRINTER TO CRT
1683 !          *****    END OF DIABLO OUTPUT SECTION   *****
1690 !     CLEANUP SECTION:  THIS SECTION RESETS VARIABLES TO PREPARE FOR
1695 !          GENERATION OF THE NEXT ROW OF BAR CODE.
1705 FOR I=1 TO 16              !ZERO OUT THE 16 BYTE BAR CODE ROW BUFFER
1710  S1(I)=0
1715  B(I)=0
1720 NEXT I
1725 FOR I=17 TO 132            !ZERO OUT THE BIT PATTERN ARRAY
1730  B(I)=0
1735 NEXT I
1740 B2=3                       !RESET S1 BUFFER POINTER TO 3
1745 L5=I5                      !UPDATE FIRST INSTR. COUNTER TO CURRENT INSTR.
1750 IF B3=0 THEN L5=L5+1       !CHECK FOR THE START OF A NEW INSTRUCTION
1755 IF B1<S2 THEN 1095
1762 PRINT "BAR CODE GENERATION COMPLETED"
1765 GOTO 350                   !GO BACK TO PROMPT IF BAR CODE GENERATION HAS
1770                            ! BEEN COMPLETED
1780 ! * * * * * * * * * * * * END OF 'RUN' ROUTINE * * * * * * * * * * * * *
1805 ! * * * * * * * * * * * * 'COMPILE' ROUTINE * * * * * * * * * * * * * * *
1815 !  HP-41 INSTRUCTION INTERPRETATION ROUTINE:  THIS ROUTINE INTERPRETS
1820 !  THE INSTRUCTIONS ENTERED IN THE TEXT ARRAY AND LOADS THE MACHINE
1825 !  CODE INTO THE 'M' ARRAY.  THE INTERPRETER IS TABLE DRIVEN EXCEPT
1830 !  FOR THOSE INSTRUCTIONS WHOSE OPERANDS CHANGE THE LENGTH OF THE
1835 !  INSTRUCTION (GTO'S,LBL'S OR XEQ'S), DIGIT ENTRY INSTRUCTIONS,
1840 !  OR INSTRUCTIONS INVOLVING ALPHANUMERIC TEXT. THE PROCESS MAY BE
1842 !  ABORTED IF AN ERROR IS ENCOUNTERED BY TYPING 'ABORT' IN RESPONSE
1843 !  TO THE ERROR MESSAGE.
1855 M1=1                        !START MACHINE CODE ARRAY INDEX AT 1
1875 FOR J=1 TO 2240            !LOOP THROUGH THE INSTRUCTION POINTER ARRAY
1880  IF P(J)<0 THEN 3335       !AND LOOK FOR A VALID POINTER
1885  T$=FNI$(A$,P(J))          !THEN EXTRACT THE INSTRUCTION FROM THE TEXT ARRAY
1887  E4=M1                     !SAVE CURRENT MACHINE CODE ARRAY POINTER
1895 !     SCANNER SECTION: THIS SECTION SCANS THE INSTRUCTION AND SENDS
1900 !     THE DECODED TEXT TO THE INTERPRETING SECTION.  IT ALSO SETS
1905 !     SEVERAL FLAGS (F1-F6) FOR THE FOLLOWING CONDITIONS,RESPECTIVELY:
1910 !     ALPHA APPEND INSTRUCTION, ANY TEXT INSTRUCTION, ANY ONE WORD
1915 !     INSTRUCTION, ANY ALPHA OPERAND,ANY INDIRECT OPERAND, AND ANY
1920 !     DIGIT ENTRY INSTRUCTION.
1930 T1$=T2$=" "                !INITIALIZE FLAGS AND TEXT VARIABLES
1935 P1=P2=0
1937 V=-1
```

```
1940 IF F6<>1 THEN 1955          !ADD A NULL INSTRUCTION BETWEEN ADJACENT DIGIT
1945 M(M1)=0                     !ENTRY INSTRUCTIONS
1950 M1=M1+1
1955 F1=F2=F3=F4=F5=F6=0
1960 S$=T$
1965 IF (T$="END") OR (T$=".END.") THEN 3355
1970                             ! CHECK FOR A TEXT ENTRY INSTRUCTION
1975 IF (T$[1,1]<>"'") AND (T$[1,2]<>"A'") THEN 2055
1980 IF T$[1,2]<>"A'" THEN 2000
1985 T$=T$[2]                    !CHECK FOR ALPHA APPEND TEXT INSTRUCTION
1990 F1=1
2000 L=LEN(T$)                   !FIND END OF TEXT AND CHECK FOR ERRORS
2005 IF L<18 THEN 2020
2010  PRINT "ALPHA STRING TOO LONG IN LINE # ";J
2015  GOTO 2230
2020 IF T$[L,L]="'" THEN 2040
2025 PRINT "ERROR IN ALPHA REGISTER ENTRY INSTRUCTION AT LINE # ";J
2035 GOTO 2230
2040 F2=1                        !SET TEXT FLAG
2045 GOTO 2275
2055 FOR I=1 TO LEN(T$)          !CHECK FOR A DIGIT ENTRY INSTRUCTION
2060  T1$=T$[I,I]
2065  IF (T1$>="0") AND (T1$<="9") THEN 2085
2070  IF ((T1$="+") OR (T1$="-")) AND (LEN(T$)>1) THEN 2085
2075  IF (T1$=" ") OR (T1$="E") OR (T1$=".") THEN 2085
2080  GOTO 2105                  !NOT A DIGIT ENTRY INSTR.; CONTINUE SCAN
2085 NEXT I
2090 F6=1                        !SET DIGIT ENTRY FLAG
2095 GOTO 2275
2105 P1=POS(T$," ")              !LOOK FOR AN OPERAND OF THE INSTRUCTION
2110 IF P1<>0 THEN 2130          !SET FLAG AND RETURN IF ONLY ONE WORD LONG
2115 F3=1
2120 GOTO 2275
2130 T1$=TRIM$(T$[P1+1])         !GET FIRST OPERAND AND CHECK FOR AN ALPHA STRING
2135 T$=T$[1,P1-1]
2140 L=LEN(T1$)
2145 IF (T1$[1,1]<>"'") OR (T1$[L,L]<>"'") THEN 2175
2150 IF L>9 THEN 2010            !CHECK FOR LENGTH OF OPERAND
2155 T1$=T1$[2,L-1]
2160 F4=1
2165 GOTO 2275
2175 P2=POS(T1$," ")             !GET SECOND OPERAND AND CHECK FOR INDIRECTION
2180 IF P2=0 THEN 2220
2185 T2$=T1$[1,P2-1]
2190 IF T2$="IND" THEN 2210
2195 PRINT "OPERAND ERROR IN LINE # ";J
2205 GOTO 2230
2210 F5=1                        !SET INDIRECTION FLAG AND EXTRACT NUMERIC OPERAND
2215 T1$=TRIM$(T1$[P2+1])
2220 IF LEN(T1$)<=2 THEN 2275
2225 PRINT "ERROR IN NUMERIC OPERAND IN LINE # ";J
2230 GOSUB 3400
2235 GOTO 1930
2250 !     INTERPRETING SECTION:  THIS SECTION TAKES THE DECODED TEXT HELD IN
2255 !     T$,T1$ AND T2$, INTERPRETS THE INSTRUCTION AND ENTERS THE MACHINE
2260 !     CODE INTO THE ARRAY 'M' AT THE POSITION GIVEN BY THE POINTER 'M1'.
2265 !     AN ERROR CAUSES A MESSAGE TO BE PRINTED WHICH REQUESTS A CORRECTION.
2275 IF F2<>1 THEN 2395          !CHECK FOR A TEXT ENTRY INSTRUCTION
2285 M(M1)=240+L-2               !ENTER LENGTH OF TEXT
2300 X=1
2305 Y=59
2315 IF F1<>1 THEN 2333          !
2316 M(M1)=M(M1)+1               !ADD COUNTER FOR EXTRA BYTE IF APPEND INST.
2317 M1=M1+1
2320 M(M1)=127                   !PUT IN 2ND BYTE
```

```
2333 M1=M1+1
2335 FOR I=2 TO L-1              !CHECK FOR VALID CHARACTERS AND ADD TO INST.
2340  Z=FNS(X,Y,(T$[I,I]),C$(*))
2345  IF Z<>0 THEN 2370
2350   PRINT "INVALID CHARACTER IN LABEL OR TEXT"
2365   GOTO 2230                          !IF ERROR EXISTS
2370  M(M1)=C2(Z)                         !ENTER VALID CHARACTER
2372  M1=M1+1
2375 NEXT I
2380 GOTO 3335
2395 IF F6<>1 THEN 2660          !CHECK FOR DIGIT ENTRY INSTRUCTION
2400 F9=0
2410 IF (T$[1,1]<>"+") AND (T$[1,1]<>"-") THEN 2435
2415 IF T$[1,1]<>"-" THEN 2430 !CHECK FOR MINUS SIGN
2420  M(M1)=28
2425  M1=M1+1
2430 T$=T$[2]
2435 L1=POS(T$," ")              !LOOK FOR EXPONENT
2440 L2=POS(T$,"E")
2445 IF L1=0 THEN L1=LEN(T$)
2450 IF L2<>0 THEN L1=L2-1
2460 FOR I=1 TO L1               !ENTER MANTISSA INTO MACHINE CODE ARRAY
2465  IF T$[I,I]<>"." THEN 2495!CHECK FOR THE DECIMAL POINT
2470  IF F9=1 THEN 2530
2475  F9=1
2480  M(M1)=26
2485  M1=M1+1
2490  GOTO 2510
2495  IF (T$[I,I]<"0") OR (T$[I,I]>"9") THEN 2490
2500  M(M1)=NUM(T$[I,I])-32
2505  M1=M1+1
2510 NEXT I
2520 IF (L1=LEN(T$)) AND (L2=0) THEN 3335!CHECK FOR ERRORS IN MANTISSA
2525 IF (I=L1) OR (I=L2) THEN 2555
2530 PRINT "DIGIT ENTRY INSTRUCTION ERROR IN LINE # ";J
2535 GOSUB 3400
2545 GOTO 2400
2555 T$=TRIM$(T$[I])             !ENTER EXPONENT IF IT EXISTS
2560 IF T$[1,1]<>"E" THEN 2530
2565 M(M1)=27
2570 M1=M1+1
2575 T$=T$[2]
2580 IF T$[1,1]<>"-" THEN 2600 !CHECK FOR NEGATIVE EXPONENT
2585 M(M1)=28
2590 M1=M1+1
2600 IF (T$[1,1]="-") OR (T$[1,1]="+") THEN T$=T$[2]
2605 T$=TRIM$(T$)
2610 L=LEN(T$)                   !ADD EXPONENT TO MACHINE CODE ARRAY
2615 IF L>2 THEN 2530
2620 FOR I=1 TO L
2625  IF (T$[I,I]<"0") OR (T$[I,I]>"9") THEN 2530
2630  M(M1)=NUM(T$[I,I])-32
2635  M1=M1+1
2640 NEXT I
2645 GOTO 3335
2660 L=LEN(T1$)                   !OTHER INSTRUCTIONS: CALCULATE VALUE OF OPERANDS
2665 IF F3=1 THEN 3265           !ONE WORD FLAG
2670 IF F4=1 THEN 2805           !ALPHA OPERAND FLAG
2680 IF L<=2 THEN 2705           !NUMERIC OPERAND SEEN
2685 PRINT "NUMERIC OPERAND TOO LONG IN LINE # ";J
2695 GOTO 2230
2705 IF L<2 THEN 2740            !CHECK FOR DOUBLE DIGIT OPERAND
2710 V=(NUM(T1$[1,1])-48)*10+NUM(T1$[2,2])-48
2715 IF (V>=0) AND (V<=99) THEN 2805
2720 PRINT "INCORRECT NUMERIC OPERAND IN LINE # ";J
```

```
2730 GOTO 2230
2740 V=0
2745 FOR I=1 TO 26              !CALCULATE STACK REGISTER OR LOCAL LABEL VALUE
2750  IF T1$=S1$(I) THEN V=I+101
2755 NEXT I
2760 IF V<>0 THEN 2805
2765 V=NUM(T1$)-48              !CALCULATE SINGLE DIGIT OPERAND VALUE
2770 IF (V>=0) AND (V<=9) THEN 2805
2775 PRINT "INCORRECT STACK OR SINGLE DIGIT OPERAND IN LINE # ";J
2785 GOTO 2230
2800                            ! CHECK FOR SPECIAL COMMANDS
2805 IF F5=1 THEN V=V+128       !ADD INDIRECTION BIT (HIGH ORDER BIT) TO OPERAND
2815 IF (T$<>"GTO") AND (T$<>"XEQ") THEN 3020!CHECK FOR 'GTO' OR 'XEQ' COMMAND
2825 IF F5<>1 THEN 2860         !CHECK FOR 'GTO IND' OR 'XEQ IND'
2830 M(M1)=174
2835 IF T$="GTO" THEN V=V-128   !SET HIGH BIT FOR 'XEQ IND'
2840 M(M1+1)=V
2845 M1=M1+2
2850 GOTO 3335
2860 IF F4<>1 THEN 2955         !START WITH 'GTO/XEQ ALPHA LABEL' COMMAND
2870 X=1
2875 Y=59
2880 M(M1)=29
2885 IF T$="XEQ" THEN M(M1)=30
2890 L=LEN(T1$)
2895 M(M1+1)=240+L              !ADD LENGTH TO SECOND BYTE (HI ORDER BITS= F HEX)
2900 FOR I=1 TO L
2905  Z=FNS(X,Y,(T1$[I,I]),C$(*))
2910  IF Z<>0 THEN 2935         !CHECK FOR VALID CHARACTERS IN ALPHA STRING
2915   PRINT "INVALID CHARACTER IN ALPHA LABEL"
2930   GOTO 2230
2935  M(M1+I+1)=C2(Z)
2936 NEXT I
2940 M1=M1+L+2
2945 GOTO 3335
2955 IF (V>14) OR (T$="XEQ") THEN 2985   !SHORT FORM (2 BYTE) NUMERIC GTO:
2960 M(M1)=177+V                !SECOND BYTE CONTAINS UNCOMPILED POINTER
2965 M(M1+1)=0
2970 M1=M1+2
2975 GOTO 3335
2985 M(M1)=208                  !LONG FORM (3 BYTE) NUMERIC GTO OR NUMERIC XEQ:
2990 IF T$="XEQ" THEN M(M1)=224
2995 M(M1+1)=0                  !SECOND BYTE AGAIN CONTAINS POINTER
3000 M(M1+2)=V                  !THIRD BYTE CONTAINS REGISTER NUMBER
3005 M1=M1+3
3010 GOTO 3335
3020 IF T$<>"STO" THEN 3080     !CHECK FOR STORE INSTRUCTION
3030 IF V>15 THEN 3055          !SHORT FORM (ONE BYTE) STORE INSTRUCTION
3035 M(M1)=48+V
3040 M1=M1+1
3045 GOTO 3335
3055 M(M1)=145                  !LONG FORM (2 BYTE) STORE INSTRUCTION
3060 M(M1+1)=V
3065 M1=M1+2
3070 GOTO 3335
3080 IF T$<>"RCL" THEN 3140    !CHECK FOR RECALL INSTRUCTION
3090 IF V>15 THEN 3115         !SHORT FORM (1 BYTE) RECALL INSTRUCTION
3095 M(M1)=32+V
3100 M1=M1+1
3105 GOTO 3335
3115 M(M1)=144                 !LONG FORM (2 BYTE) RECALL INSTRUCTION
3120 M(M1+1)=V
3125 M1=M1+2
3130 GOTO 3335
3140 IF T$<>"LBL" THEN 3265
3150 IF F4<>1 THEN 3210        !ALPHA LABEL INSTRUCTION
```

```
3160 X=1
3165 Y=59
3170 M(M1)=192                  !UNCOMPILED LABEL CHAIN POINTER IN SECOND BYTE
3175 M(M1+1)=0
3180 L=LEN(T1$)
3185 M(M1+2)=241+L
3190 M(M1+3)=0                  !KEY ASSIGNMENT BYTE (SET TO ZERO FOR BAR CODE)
3195 M1=M1+2
3200 GOTO 2900                  !ADD CHARACTER DATA TO MACHINE CODE ARRAY
3210 IF V>14 THEN 3235          !SHORT FORM (ONE BYTE) NUMERIC LABEL
3215 M(M1)=1+V
3220 M1=M1+1
3225 GOTO 3335
3235 M(M1)=207                  !LONG FORM (TWO BYTE) NUMERIC LABEL
3240 M(M1+1)=V
3245 M1=M1+2
3250 GOTO 3335
3265 X=1                                !ALL OTHER COMMANDS ARE TABLE DRIVEN
3270 Y=103
3275 Z=FNS(X,Y,T$,I$(*))
3280 IF Z<>0 THEN 3310
3290 PRINT "UNRECOGNIZABLE INSTRUCTION GIVEN IN LINE # ";J
3300 GOTO 2230
3310 M(M1)=I1(Z)                !STORE INSTRUCTION TYPE IN MACHINE CODE ARRAY
3312 M1=M1+1
3313 !                          !CHECK FOR CORRECT ONE WORD INSTRUCTION
3314 IF (I1(Z)<64) OR (I1(Z)>143) OR F3 THEN 3320
3316 PRINT "ERROR: EXTRANEOUS OPERAND IN INSTRUCTION"
3318 GOTO 2230
3320 IF I1(Z)<144 THEN 3335     !CHECK FOR TWO BYTE INSTRUCTION
3322 IF (I1(Z)<144) OR (I1(Z)>191) OR (V>=0) THEN 3327
3323 PRINT "ERROR: MISSING OPERAND"
3325 GOTO 2230
3327 M(M1)=V
3330 M1=M1+1
3335 NEXT J                     !***END OF THE INSTRUCTION DECODE LOOP***
3345 ! *         *         *         *         *         *         *
3355 M(M1)=192                  !ADD FINAL END INSTRUCTION: UNCOMPILED POINTER
3360 M(M1+1)=0                  !IN SECOND BYTE
3365 M(M1+2)=47
3370 S2=M1+2
3375 PRINT "COMPILATION COMPLETED"
3377 F8=1                       !SET COMPILATION DONE FLAG
3380 GOTO 350
3395 !        ***ERROR CORRECTION SUBROUTINE***
3400 PRINTER IS 16
3402 M1=E4                      !RESET MACHINE CODE ARRAY TO OLD VALUE
3405 PRINT "THE INSTRUCTION GIVEN WAS: ";S$
3407 PRINT "GIVE THE CORRECTED INSTRUCTION (WITHOUT LINE NUMBER)  "
3410 INPUT " (TO ABORT THIS COMPILATION, TYPE 'ABORT'): ",T$
3415 IF T$="ABORT" THEN 350
3420 A$=A$&T$&"!"
3425 P(J)=T
3430 T=T+LEN(T$)+1
3435 RETURN
3450 ! * * * * * * * * * END OF 'COMPILE' ROUTINE * * * * * * * * * * * * * *
3475 ! * * * * * * * * * PROGRAM COMMANDS LIST ROUTINE * * * * * * * * * * *
3485 !  THIS ROUTINE LISTS OUT THE COMMANDS AVAILABLE IN THIS PROGRAM AND
3490 !  THE SYNTAX OF THE COMMANDS AND OF INSTRUCTION ENTRY
3500 PRINT " "
3505 PRINT "COMMANDS AVAILABLE IN THIS PROGRAM:"
3510 PRINT " "
3515 PRINT "  COMPILE       - COMPILES THE HP-41 PROGRAM CURRENTLY ENTERED"
3520 PRINT "  DELETE nn     - DELETES THE NUMBERED INSTRUCTION FROM THE PROGRAM"
3525 PRINT "  EXIT          - HALTS THIS PROGRAM OR STOPS NUMBER GENERATOR"
```

```
3530 PRINT "  GETPROG       - RETRIEVES THE COMPILED CODE FROM CASSETTE TAPE"
3535 PRINT "  GETTEXT       - RETRIEVES THE PROGRAM INSTRUCTIONS FROM TAPE"
3540 PRINT "  LIST          - LISTS THE ENTIRE HP-41 PROGRAM CURRENTLY IN MEMORY"
3542 PRINT "  NUMBER        - GENERATES HP-41 INSTRUCTION NUMBERS - STOPPED BY "
3543 PRINT "                  TYPING 'EXIT'"
3545 PRINT "  RENUMBER      - RENUMBERS THE HP-41 PROGRAM INSTRUCTION NUMBERS"
3550 PRINT "  RUN           - GENERATES THE BAR CODE FROM THE COMPILED CODE"
3555 PRINT "  RUNPRIVATE    - GENERATES THE BAR CODE FOR A PRIVATE PROGRAM"
3560 PRINT "  SAVEPROG      - STORES THE COMPILED CODE ON CASSETTE TAPE"
3565 PRINT "  SAVETEXT      - STORES THE PROGRAM LISTING ON CASSETTE TAPE"
3570 PRINT "  SCRATCH       - ERASES THE ENTIRE HP-41 PROGRAM"
3575 PRINT "  ??            - LISTS OUT THE COMMANDS AVAILABLE "
3580 PRINT " "
3585 PRINT "SYNTAX FOR INSTRUCTION ENTRY:"
3590 PRINT " A)INSTRUCTION FORMAT: "
3595 PRINT "      n <HP-41 INSTRUCTION>"
3600 PRINT "      (BLANKS ARE USED AS DELIMITERS)"
3605 PRINT " B)SPECIAL SYMBOLS:"
3610 PRINT "      1) USE '&' INSTEAD OF THE SIGMA SIGN"
3615 PRINT "      2) USE '@' INSTEAD OF THE ANGLE SIGN"
3620 PRINT "      3) USE '#' INSTEAD OF THE 'NOT EQUAL' SIGN"
3625 PRINT "      4) USE SINGLE QUOTES (') INSTEAD OF DOUBLE QUOTES"
3630 PRINT " C)TEXT FORMAT:       '<TEXT ENTRY>'     (NOTE SINGLE QUOTES)"
3635 PRINT "                OR    A'<TEXT ENTRY>'     (FOR APPENDING TEXT)"
3640 GOTO 350
3650 ! * * * * * * * * * * END OF COMMAND LIST ROUTINE * * * * * * * * * *
3675 ! * * * * * * * * * * 'RENUMBER' ROUTINE * * * * * * * * * * * * * * *
3685 !  THIS ROUTINE RENUMBERS THE HP-41 INSTRUCTIONS BY REARRANGING THE
3690 !  ARRAY OF POINTERS INTO THE TEXT STRING.  THE STARTING OLD VALUE, NEW
3695 !  STARTING VALUE AND INCREMENT ARE REQUESTED, AND AN NUMBER OVERFLOW
3700 !  (>2240) OR REWRITING OVER EXISTING INSTRUCTIONS WILL ABORT ROUTINE
3710   INPUT "ENTER THE OLD STARTING #,NEW STARTING #, AND INCREMENT: ",V1,V2,V3
3711   FOR I=V2+1 TO V1-1                !CHECK FOR OVERWRITTEN INSTRUCTIONS
3712    IF P(I)=-1 THEN 3715
3713    PRINT "ERROR - ATTEMPT MADE TO OVERWRITE EXISTING INSTRUCTIONS"
3714    GOTO 350
3715   NEXT I
3717   FOR I=1 TO 2240                 !CREATE TEMPORARY POINTER ARRAY
3720    K1(I)=P(I)
3725    IF I>=V1 THEN P(I)=-1
3730   NEXT I
3735   K=V1-1
3740   FOR I=V2 TO 2240 STEP V3 !TRANSFER VALID POINTERS BACK TO POINTER ARRAY
3745    K=K+1
3750    IF K>2240 THEN 350      !CHECK FOR END OF PROCESSING
3755    IF K1(K)<0 THEN 3745
3760    P(I)=K1(K)
3765   NEXT I
3770   PRINT "ERROR: INSTRUCTION NUMBER OUT OF BOUNDS"
3775   FOR I=1 TO 2240
3780    P(I)=K1(I)
3785   NEXT I
3790   GOTO 350
3795 ! * * * * * * * * * * END OF 'RENUMBER' ROUTINE * * * * * * * * * * *
3820 ! * * * * * * * * * * 'SAVETEXT' ROUTINE * * * * * * * * * * * * * * *
3830 !   THIS ROUTINE SAVES THE TEXT OF THE HP-41 INSTRUCTIONS (THE SOURCE
3835 !   FILE) ON CASSETTE TAPE.
3845 INPUT "GIVE THE NAME OF THE FILE TO BE SAVED: ",T1$
3850 CREATE T1$,T DIV 64+50
3855 ASSIGN #1 TO T1$
3860 PRINT #1;P(*)             !SAVE THE POINTER ARRAY AND THE TEXT STRING
3865 PRINT #1;A$,END
3870 ASSIGN * TO #1
3875 GOTO 350
3885 ! * * * * * * * * * * END OF 'SAVETEXT' ROUTINE * * * * * * * * * * *
```

```
3910 ! * * * * * * * * * * 'GETTEXT' ROUTINE * * * * * * * * * * * * * * * *
3920 !   THIS ROUTINE RETRIEVES THE TEXT INSTRUCTIONS (SOURCE FILE) FROM
3925 !   CASSETTE TAPE AND RESETS THE END OF TEXT POINTER.
3935 INPUT "GIVE THE NAME OF THE FILE TO BE READ",T1$
3940 ASSIGN T1$ TO #1,I
3945 IF I<>1 THEN 3960
3950 PRINT "FILE NAME NOT FOUND"
3955 GOTO 350
3960 IF I=0 THEN 3980
3965 PRINT "FILE IS PROTECTED OR OF WRONG TYPE"
3970 GOTO 350
3980 READ #1;P(*)              !RETRIEVE THE POINTER ARRAY AND TEXT STRING
3985 READ #1;A$
3990 T=LEN(A$)                 !RESTORE POINTER TO END OF TEXT
3995 GOTO 350
4005 ! * * * * * * * * * * * END OF 'GETTEXT' ROUTINE * * * * * * * * * * *
4030 ! * * * * * * * * * * 'SAVEPROG' ROUTINE * * * * * * * * * * * * * * *
4040 !   THIS ROUTINE SAVES THE COMPILED CODE (THE JOB FILE) IN THE 'M' ARRAY
4045 !   ON CASSETTE TAPE FOR LATER USE.
4055 INPUT " GIVE THE NAME OF THE FILE TO BE SAVED: ",T1$
4060 CREATE T1$,50
4065 ASSIGN T1$ TO #1
4070 PRINT #1;S2               !SAVE THE NUMBER OF BYTES IN THE PROGRAM
4075 PRINT #1;M(*)
4080 ASSIGN * TO #1
4085 GOTO 350
4095 ! * * * * * * * * * END OF 'SAVEPROG' ROUTINE * * * * * * * * * * *
4125 ! * * * * * * * * *'GETPROG' ROUTINE * * * * * * * * * * * * * * * * *
4130 !   THIS ROUTINE RETRIEVES THE COMPILED PROGRAM FROM CASSETTE TAPE
4140 INPUT "GIVE THE NAME OF THE FILE TO BE READ",T1$
4145 F8=1                      !SET THE COMPILED PROGRAM PRESENT FLAG
4150 ASSIGN T1$ TO #1,I
4155 IF I<>1 THEN 4170         !CHECK FOR FILE ERRORS
4160 PRINT "FILE NOT FOUND"
4165 GOTO 350
4170 IF I=0 THEN 4190
4175 PRINT "FILE IS PROTECTED OR OF WRONG TYPE"
4180 GOTO 350
4190 READ #1;S2                !GET THE NUMBER OF BYTES IN THE MACHINE CODE ARRAY
4195 READ #1;M(*)
4200 ASSIGN #1 TO *
4205 GOTO 350
4215 ! * * * * * * * * * * END OF 'SAVEPROG' ROUTINE * * * * * * * * * * *
4240 ! * * * * * * * * ERROR CONDITION HANDLING ROUTINE * * * * * * * * * *
4250 E1=ERRN                          !SAVE ERROR NUMBER
4255 E2=ERRL                          !SAVE LINE NUMBER WHERE ERROR OCCURED
4260 IF E1<>80 THEN 4275
4265 PRINT "NO TAPE IN TAPE DRIVE.  PLEASE INSERT TAPE"
4270 GOTO 350
4275 IF E1<>64 THEN 4290
4280 PRINT "NOT ENOUGH ROOM ON TAPE. PLEASE USE ANOTHER TAPE"
4285 GOTO 350
4290 IF E1<>83 THEN 4305
4295 PRINT "TAPE IS WRITE PROTECTED. PLEASE FIX"
4300 GOTO 350
4305 IF E1<>53 THEN 4320
4310 PRINT "IMPROPER FILE NAME (SHOULD BE SIX CHARACTERS OR LESS)"
4315 GOTO 350
4320 IF E1<>54 THEN 4360
4325 PRINT "DUPLICATE FILE NAME"
4330 IF (E2<>3850) AND (E2<>4060) THEN 4345
4335 INPUT "DO YOU WISH TO USE THE OLD FILE? ",T2$
4340 IF (T2$="Y") OR (T2$="YES") THEN 4350
4345 GOTO 350
4350 IF E2=3850 THEN 3855
4355 IF E2=4060 THEN 4065
```

```
4360 PRINT "ERROR # ";E1;"SEEN AT LINE # ";E2
4365 GOTO 350
4375 ! * * * * * * * * * * * END OF ERROR ROUTINE * * * * * * * * * * * * *
4400 ! * * * * * * * * * * BAR PATTERN STRING GENERATOR * * * * * * * * * * *
4410 !   THIS PROCEDURE GENERATES A STRING REPRESENTING THE BAR CODE PATTERN
4415 !   AS IT WILL BE WRITTEN ON THE DIABLO 1650 DAISY WHEEL PRINTER.  IT
4425 !   DECODES THE BIT PATTERN FROM THE ARRAY 'B' AND CONCATENATES THE
4430 !   CORRECT NUMBER OF VERTICAL BARS AND BLANKS ONTO THE STRING.  THE
4435 !   PATTERN USED IS:
4440 !     1)NARROW BAR:  2 VERTICAL BARS
4445 !     2)WIDE BAR:  4 VERTICAL BARS
4450 !     3)SPACE:  3 BLANKS
4460  B$=" "
4465 FOR I=1 TO L
4470 !  THE VERTICAL BAR ON THE TITAN 10 CHARACTER WHEEL IS AN '>' IN ASCII
4475  IF B(I)=0 THEN B$=B$&">>   "
4480  IF B(I)=1 THEN B$=B$&">>>>   "
4485 NEXT I
4490 RETURN
4500 ! * * * * * * * * * END OF BAR PATTERN GENERATOR * * * * * * * * * * * *
4520 ! * * * * * * * * * * * * * DATA * * * * * * * * * * * * * * * * * * * *
4530 !    ***LOCAL LABEL AND STACK REGISTER CHARACTERS***
4535 DATA A,B,C,D,E,F,G,H,I,J,T,Z,Y,X,L
4540 DATA " "," "," "," "," "," "," ",a,b,c,d,e
4550 !    ***INSTRUCTION MNEMONICS AND NUMERIC VALUES***
4555 DATA %,76,%CH,77,&+,71,&-,72,&REG,153
4557 DATA *,66,+,64,-,65,/,67,1/X,96,10^X,87,ABS,97
4560 DATA ACOS,93,ADV,143,AOFF,139,AON,140,ARCL,155,ASHF,136,ASIN,92
4565 DATA ASTO,154,ATAN,94,AVIEW,126,BEEP,134,CF,169,CHS,84,CL&,112
4570 DATA CLA,135,CLD,127,CLRG,138,CLST,115,CLX,119,COS,90
4575 DATA D-R,106,DEC,95,DEG,128,DSE,151,ENG,158,ENTER^,131,E^X,85
4580 DATA E^X-1,88,FACT,98,FC?,173,FC?C,171,FIX,156,FRC,105
4585 DATA FS?,172,FS?C,170,GRAD,130,HMS,108,HMS+,73,HMS-,74,HR,109
4590 DATA INT,104,ISG,150,LASTX,118,LN,80,LN1+X,101,LOG,86
4595 DATA MEAN,124,MOD,75,OCT,111,OFF,141,P-R,78,PI,114
4600 DATA PROMPT,142,PSE,137,R-D,107,R-P,79,RAD,129,RCL,144,RDN,117
4605 DATA RND,110,RTN,133,R^,116,SCI,157,SDEV,125,SF,168
4610 DATA SIGN,122,SIN,89,SQRT,82,ST*,148
4615 DATA ST+,146,ST-,147,ST/,149,STO,145,STOP,132,TAN,91,TONE,159
4620 DATA VIEW,152,X#0?,99,X#Y?,121,X<0?,102,X<=0?,123,X<=Y?,70
4625 DATA X<>,206,X<>Y,113,X<Y?,68,X=0?,103,X=Y?,120,X>0?,100
4630 DATA X>Y?,69,X^2,81,Y^X,83
4640 !    ***VALID 41C CHARACTERS AND CHARACTER CODE***
4645 DATA " ",32,#,29,$,36,%,37,&,126,*,42,+,43,",",44,-,45,.,46,/,47
4650 DATA "0",48,"1",49,"2",50,"3",51,"4",52,"5",53,"6",54,"7",55,"8",56
4655 DATA "9",57,:,58,";",59,<,60,=,61,>,62,?,63,@,13,A,65,B,66,C,67,D,68,E,69
4660 DATA F,70,G,71,H,72,I,73,J,74,K,75,L,76,M,77,N,78,O,79,P,80,Q,81,R,82,S,83
4665 DATA T,84,U,85,V,86,W,87,X,88,Y,89,Z,90,^,94,a,97,b,98,c,99,d,100,e,101
4670 END
4675 ! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
4685 !                ******  FUNCTION DEFINITIONS ******
4695 ! INSTRUCTION EXTRACTION FUNCTION:  THIS FUNCTION LOOKS AT THE TEXT
4700 ! ARRAY AND EXTRACTS THE INSTRUCTION POINTED TO BY THE POINTER AT THE
4705 ! INDEX PASSED TO THE FUNCTION.
4715 DEF FNI$(A$,INTEGER I)
4720   INTEGER J
4725   DIM S$[50]
4730   FOR J=1 TO 50
4735    IF A$[I+J;1]="!" THEN 4745
4740   NEXT J
4745   S$=A$[I+1,I+J-1]          !GET TEXT UP TO THE DELIMITER
4750   RETURN S$
4755 FNEND
4775 ! *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
4785 !   NUMBER FORMAT FUNCTION:  THIS FUNCTION CONVERTS A NUMBER INTO A
```

```
4790 !   CHARACTER STRING.  IT IS USED ONLY IN THE DIABLO PRINTOUT SECTION.
4792 !   PARAMETER NEEDED IS:
4795 !    1) AN INTEGER NUMBER TO BE CONVERTED INTO A CHARACTER STRING
4805 DEF FNF$(INTEGER N)
4810  INTEGER I,J,K
4815  DIM N$[5]
4820  IF N>=10 THEN 4840          !CONVERT ONE DIGIT NUMBERS
4825  N$=CHR$(N+48)
4830 RETURN N$
4840  IF N>=100 THEN 4860         !CONVERT TWO DIGIT NUMBERS
4845  N$=CHR$(N DIV 10+48)&CHR$(N MOD 10+48)
4850  RETURN N$
4860  IF N>1000 THEN 4890
4865  I=N DIV 100
4870  J=N MOD 100 DIV 10
4875  N$=CHR$(I+48)&CHR$(J+48)&CHR$(N MOD 10+48)
4880  RETURN N$
4890  I=N DIV 1000
4895  J=N MOD 1000 DIV 100
4900  K=N MOD 100 DIV 10
4905  N$=CHR$(I+48)&CHR$(J+48)&CHR$(K+48)&CHR$(N MOD 10+48)
4910  RETURN N$
4915 FNEND
4925 ! *   *    *    *   *    *   *   *   *   *   *   *   *   *
4935 !   ROW AND INSTRUCTION PRINTOUT FUNCTION:  THIS FUNCTION
4940 !   CREATES A STRING CONTAINING THE ROW NUMBER AND BEGINNING
4945 !   AND ENDING FUNCTION NUMBERS. IT IS USED ONLY IN THE DIABLO
4947 !   PRINTOUT SECTION.  PARAMETERS NEEDED ARE:
4950 !    1)ROW NUMBER
4955 !    2)FIRST INSTRUCTION NUMBER
4960 !    3)LAST INSTRUCTION NUMBER
4970 DEF FNP$(INTEGER R,INTEGER I,INTEGER F)
4975  DIM R$[30]
4980  R$="ROW "&FNF$(R)&" ("&FNF$(I)
4985  IF I=F THEN 4995
4990  R$=R$&" - "&FNF$(F)
4995  R$=R$&")"
5000  RETURN R$
5005 FNEND
5015 ! *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
5030 ! BINARY SEARCH FUNCTION:  FUNCTION SEARCHES PASSED CHARACTER ARRAY
5035 !   FOR PASSED KEY AND RETURNS INDEX OF KEY FOUND IN ARRAY OR 0 IF
5040 !   NO KEY WAS FOUND.  PARAMETERS REQUIRED ARE:
5045 !   1)INDEX OF FIRST POSITION (INTEGER)
5050 !   2)INDEX OF LAST POSITION (INTEGER)
5055 !   3)KEY TO BE FOUND (STRING)
5060 !   4)STRING ARRAY IN WHICH THE SEARCH IS MADE
5070 DEF FNS(INTEGER I,J,Q$,A$(*))
5075  INTEGER F,L,M
5080  F=I
5085  L=J
5090  M=(F+L) DIV 2               !FIND CENTER OF ARRAY
5095  IF Q$=A$(M) THEN RETURN M   !IF KEY HAS BEEN FOUND, RETURN INDEX
5100  IF Q$>A$(M) THEN F=M+1
5105  IF Q$<A$(M) THEN L=M-1
5110  IF F<=L THEN 5090           !CONTINUE SEARCH THROUGH APPROPRIATE HALF
5115  M=0                         !RETURN 0 IF SEARCH FAILS
5120  RETURN M
5125 FNEND
5140 ! ************ END OF BAR CODE GENERATION PROGRAM ******************
```

Generating Other Bar Code Types (FULFCN)

This program produces single rows of bar code containing complete key phrases (Direct Execution — type 4), numeric data (Types 6 and 9), and/or alpha data (Types 7, 8, 10, and 11). It prompts the user for input and responds to a "??" by displaying the instruction set on the CRT. FULFCN differs from PRGMBR in that a row of bar code is generated immediately after each completed line of input.

When inputting a complete HP-41 function the user types the function as it would appear in an HP-41 program line (without line number). If the user wishes to produce data bar code the input is typed as follows:

HP-41 Type Bar Code	Input Expected by FULFCN*
Numeric Data	DATA 123.45
Alpha-replace Data	DATA 'HELLO'
Alpha-append Data	DATA A'THERE'
Numeric Sequenced Data	DATASEQ 123.45 1
Alpha-replace Sequenced Data	DATASEQ 'HELLO' 2
Alpha-append Sequenced Data	DATASEQ A'THERE' 3

*The number 123.45 and the alpha letters are the "data;" the individual numbers 1, 2, and 3 in the sequenced data are the sequence numbers that will be placed in the bar code.

FULFCN is a table driven program with all the functions contained in a data table. (Each entry in the table includes the function plus four parameters.) When the data table is read in, the function name is placed in string "T$", the first parameter is placed in array "Datar", the second parameter into array "Datac", the third parameter into array "Arg", and the fourth parameter into array "Alpha". The first two parameters correlate the function to its position in the HP-41 Function Table (Table III, page 45). The third parameter indicates the number of arguments allowed for that function. The fourth parameter indicates whether or not an alpha argument is allowed.

When the input string is given, it is parsed into three strings using spaces as delimiters. At the end of this parse routine "F$" contains the function name, "A1$" contains the first argument, and "A2$" contains the second argument. If there were no arguments input, strings "A1$" and "A2$" are blank.

At this point "F$" is examined. If it contains a "??" then the instruction set is displayed, followed immediately by a prompt for input. If "F$" contains "DATA" or "DATASEQ" the program branches to the section that generates Data bar code. Otherwise the string "T$" (which contains the function names from the data table) is searched. If the function name in "F$" is not found in "T$", a message that the function was not found is displayed and the user is prompted for another instruction.

FULFCN Syntax

The syntax of the HP-41C instructions recognized by "FULFCN" is the same as that recognized by "PRGMBR". The same execeptions are found in this program: "Σ", "$\measuredangle$", and "$\neq$" are represented by the ASCII characters "&", "@", and "#" respectively. Also, single quotes instead of double quotes are used for texts and alpha labels. The Alpha-append instruction is indicated by an "A" preceding the single quotes and character string.

FULFCN Program Listing

```
20   !
30   ! THIS PROGRAM PROMPTS THE USER FOR HP-41 TYPE COMMANDS THAT WILL BE
40   ! TRANSLATED INTO A CHARACTER STRING REPRESENTING HP-41 BAR CODE.
70   ! ************************************************************************
100  ! VARIABLES:       I$ : INPUT STRING
110  !                  A$ : DUMMY STRING VARIABLE
120  !                  A1$ : FIRST ARGUMENT STRING
130  !                  A2$ : SECOND ARGMENT STRING
140  !                  F$ : FUNCTION NAME STRING
150  !                  B$ : OUTPUT STRING (1's AND 0's)
160  !                  Byte(N): DECIMAL VALUE OF BYTE(N) IN BAR CODE.
170  !                  Cksum: CHECK SUM
180  !                  N : NUMBER OF BYTES IN CODE
190  !                  L : LENGTH OF INPUT STRING
200  !                  Lf: LENGTH OF FUNCTION STRING
210  !                  La: LENGTH OF ARGUMENT STRING
220  !                  I : INDEX VARIABLE
230  !                  Type: BAR CODE TYPE NUMBER
240  !                  D(*): DIGITS FOR DATA TYPE
250  !                  J : INDEX VARIABLE
260  !                  K : INDEX VARIABLE
270  !                  X$ : DUMMY VARIABLE
280  !                  T$ : TABLE FOR FUNCTIONS NAMES
290  !                  Datar(*): DATA FOR FUNCTIONS, ROW NUMBER
300  !                  Datac(*): DATA FOR FUNCTIONS, COLUMN NUMBER
310  !                  Arg(*): NUMBER OF ARGUMENTS ALLOWED FOR FUNCTION
320  !                  Alpha(*): INDICATES IF ALPHA ARGUMENTS ARE ALLOWED
330  !                  Bar(*): BINARY BALUE OF OUTPUT STRING
340  !                  Dec(*): DECIMAL VALUE OF BYTE (*)
350  !                  B$: OUTPUT STRING
360  !                  Line: OUTPUT LINE NUMBER
380  ! ************************************************************************
400  !                          MAIN PROGRAM
420  ! ************************************************************************
450  OPTION BASE 1
460  PRINTER IS 9
470  DIM I$[90],A1$[50],A2$[50],F$[10],B$[915],X$[10],T$(256)[8],B$(915)[1]
480  DIM A$[50]
490  DIM Byte(16),D(30),Datar(256),Datac(256),Arg(256),Alpha(256),Dec(16)
500  DIM Bar(150)
510  ON ERROR GOTO 6130
520  Line=0
540            ! READ IN TABLE FOR HP-41 FUNCTIONS
560  FOR I=1 TO 256
570    READ T$(I),Datar(I),Datac(I),Arg(I),Alpha(I)
580    IF T$(I)="*END*" THEN 690
590  NEXT I  !IF THE LOOP EXITED NORMALLY THEN *END* NOT FOUND
630  DISP "ERROR IN TABLE DATA.  MISSING '*END*'"
640  STOP
670          ! COME HERE IF THE TABLE WAS READ CORRECTLY
690  Total=I-1                 ! TOTAL IS THE NUMBER OF ENTRIES IN TABLE
710          ! PROMPT USER FOR INSTRUCTION REQUEST
730  INPUT "WOULD YOU LIKE INSTRUCTIONS ?",X$
740  X$=X$[1,1]                ! JUST CHECK FIRST LETTER
750  IF X$="N" THEN 1000
770       ! IF INPUT IS ANYTHNG OTHER THAN 'NO' THEN DISPLAY INSTRUCTIONS
790  DISP "THIS PROGRAM PROMPTS THE USER FOR HP-41 BAR CODE TYPE"
800  DISP "COMMANDS.  JUST TYPE THE COMMAND AS YOU WOULD ON THE"
810  DISP "HP-41 WITH THE FOLLOWING EXCEPTIONS:"
820  DISP
830  DISP "      FOR NUMERIC DATA        : DATA **...** "
840  DISP "      FOR ALPHA REPLACE DATA  : DATA '**...**'"
850  DISP "      FOR ALPHA APPEND DATA   : DATA A'**...**'"
851  DISP "      FOR SEQUENCED DATA USE THE FOLLOWING FORMAT:"
```

```
852  DISP "                  DATASEQ DATA SEQ#"
853  DISP
854  DISP "     FOR INSTANCE, DATASEQ 1.234 45 OR DATASEQ 'TEST' 13"
860  DISP
870  DISP "SIMILARLY, ALL ALPHA TEXT STRING ARGUMENTS, LIKE XEQ 'SQR'"
880  DISP "ARE DELIMITED WITH 's."
890  DISP
900  DISP "FOR SIGMA,ANGLE, AND NOT EQUAL IN TEXT STRINGS USE &,@, AND"
910  DISP "# RESPECTIVELY."
911  DISP
912  DISP "FOR FUNCTIONS WITH THE GREEK LETTER SIGMA USE: CL&, &REG,"
913  DISP "&+, AND &-."
914  DISP "FOR THE NOT EQUAL SIGN USE #, AS IN X#0?, AND X#Y?"
920  DISP
930  DISP "ENTER ?? FOR INSTRUCTIONS AT ANY TIME."
940  DISP
941  DISP "ENTER *STOP TO TERMINATE THE PROGRAM"
942  DISP
950  DISP "TERMINATE ALL ENTRIES WITH THE CONTINUE BUTTON"
970            ! PROMPT USER FOR INSTRUCTION TO BE CODED
990  DISP
1000 INPUT "FUNCTION TO BE CODED ?",I$
1050 F$=""  ! INITIALIZE STRINGS IN CASE THEY ARE NOT USED
1060 A1$=""
1070 A2$=""
1080 A$=""
1100 ! ********** PARSE OUT THE FUNCTION AND ANY ARGUMENTS ******************
1120 I$=TRIM$(I$)              ! DELETE LEADING AND TRAILING BLANKS
1130 L=LEN(I$)
1131 IF I$="*STOP" THEN PRINTER IS 16
1132 IF I$="*STOP" THEN STOP
1140 IF L<=0 THEN 990
1150     FOR I=1 TO L          ! THIS LOOP PARSES OUT THE FUNCTION
1160        IF I$[I,I]=" " THEN 1220    ! SPACES ARE USED FOR DELIMITERS
1170        F$=F$&I$[I,I]
1180     NEXT I
1190 GOTO 1550                 ! IF HERE, THERE ARE NO ARGUMENTS ENTERED SO JUMP
1220 A$=TRIM$(I$[I,L])         ! DELETE ANY LEADING/TRAILING BLANKS FROM ARGUMENT
1230 L=LEN(A$)                 ! CHECK FOR TEXT STRING.  IF ONE IS FOUND THEN
1250                           ! JUMP AHEAD AND PROCESS IT.
1260 IF (A$[1,1]="'") OR (A$[1,1]="A") AND (A$[2,2]="'") THEN 1380
1270     FOR I=1 TO L          ! THIS LOOP PARSES OUT THE FIRST ARGUMENT
1280       IF A$[I,I]=" " THEN 1340
1290       A1$=A1$&A$[I,I]
1300     NEXT I
1310 GOTO 1550                 ! IF HERE THERE WAS ONLY ONE ARGUMENT ENTERED
1340 A2$=TRIM$(A$[I,L])        ! PARSE OUT THE SECOND ARGUMENT
1350 GOTO 1550                 ! ALL ARGUMENTS ARE NOW PARSED
1380 I=3                       ! THIS LOOP PARSES OUT TEXT STRINGS
1390 IF A$[1,1]="'" THEN I=2
1400    FOR I=I TO L-1         ! SEARCH TEST STRING FOR ENDING '
1410      IF A$[I,I]="'" THEN 1460
1420    NEXT I
1430 A1$=A$                    ! HERE IF THERE IS ONLY ONE ARGUMENT AND IT WAS
1450 GOTO 1550                 ! A TEXT STRING
1460 A1$=A$[1,I]               ! HERE IF THERE TWO ARGUMENTS AND THE FIRST WAS
1470 I=I+1                     ! A TEXT STRING.
1480 GOTO 1340
1500 ! ****************** END OF PARSE ROUTINE ******************************
1530 ! CHECK TO SEE IF THERE WAS INDEED AN INSTRUCTION
1550 IF LEN(F$)>0 THEN 1590
1560 DISP "ERROR IN ENTRY",I$
1570 GOTO 990
1590 Lf=LEN(F$)                ! LENGTH OF FUNCTION STRING
1600 La1=LEN(A1$)              ! LENGTH OF FIRST ARGUMENT
1610 La2=LEN(A2$)              ! LENGTH OF SECOND ARGUMENT
```

```
1700 IF F$="??" THEN 790       ! ?? MEANS TO REPEAT INSTRUCTIONS
1710 IF F$="DATA" THEN 3680    ! JUMP AHEAD IF IT IS A DATA STATEMENT
1711 IF F$="DATASEQ" THEN 3680
1730 !  ############ HERE FOR FUNCTION TYPE CODE ########################
1770 X=1  ! SEARCH FOR THE INSTRUCTION IN THE TABLE
1780 Y=Total
1790 Z=(X+Y) DIV 2             ! Z POINTS TO THE CORRECT TABLE POSITION
1800      IF F$=T$(Z) THEN 1920
1810      IF F$<T$(Z) THEN Y=Z-1
1820      IF F$>T$(Z) THEN X=Z+1
1830      IF X<=Y THEN 1790
1840                            ! IF THIS LOOP IS EXITED WITHOUT A MATCH THEN
1850                            ! FLAG THAT THE FUNCTION WAS NOT FOUND IN TABLE
1860 DISP "FUNCTION NOT FOUND IN TABLE",F$
1870 DISP
1880 GOTO 990                   ! PROMPT FOR ANOTHER INSTRUCTION
1900          ! HERE FOR A CORRECT FUNCTION
1920 IF Arg(Z)>0 THEN 2130        ! JUMP AHEAD FOR FUNCTIONS WITH ARGUMENTS
1940 ! 00000000000 HERE FOR FUNCTIONS WITH NO ARGUMENTS 0000000000000000000
1960 IF (La1=0) AND (La2=0) THEN 2010
1970  DISP "THERE SHOULD BE NO ARGUMENTS FOR THIS FUNCTION ",I$
1980  DISP "PLEASE ENTER THE FUNCTION AGAIN"
1990  GOTO 990
2010 Byte(2)=64                 ! Byte(2)=64 FOR FULL FCN CODE
2020 Byte(3)=Datar(Z)*16+Datac(Z)      ! Byte(3) IS THE FUNCTION CODE
2030 N=3                        ! N IS THE NUMBER OF BYTES IN THE CODE
2040 GOSUB 4580                 ! CALCULATE THE CHECKSUM
2050 GOSUB 5760                 ! CALCULATE OUTPUT STRING HERE
2060 GOTO 990                   ! RETURN TO PROMPT FOR ANOTHER INSTRUCTION
2090 ! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
2110           ! HERE FOR FUNCTIONS WITH ARGUMENTS
2130 GOSUB 4890                 ! CHECK THE TYPE OF ARGUMENTS
2140 IF (A1>4) OR (A2>4) THEN 990       ! CHECK FOR ERRORS IN ARGUMENTS
2150 IF Arg(Z)=2 THEN 3320      ! JUMP AHEAD FOR FUNCTIONS WITH 2 ARGUMENTS
2170 ! 111111111111 HERE FOR SINGLE ARGUMENT FUNCTIONS 1111111111111111111
2190 IF La1>0 THEN 2220       ! CHECK TO BE SURE AN ARGUMENT WAS ENTERED
2200 DISP "ARGUMENT REQUIRED,  PLEASE ENTER FUNCTION AGAIN";I$
2210 GOTO 990
2220   IF A1$="IND" THEN 3010 ! INDIRECT ARGUMENTS
2230   IF A1=0 THEN 2520       ! NUMERIC ARGUMENTS
2240   IF A1=3 THEN 2680       ! ALPHA NUMERIC ARGUMENTS (A-J)
2250   IF A1=3.5 THEN 2690     ! ALPHA NUMERIC ARGUMENTS (a-e)
2260   IF A1=4 THEN 2520       ! STACK ARGUMENTS
2270                           ! IF NONE OF THE ABOVE ARGUMENT TYPES WERE FOUND
2280                           ! THEN IT IS AN ALPHA ARGUMENT
2300 ! 1A1A1A1A1A1 HERE FOR SINGLE ARGUMENT, ALPHA TYPE A1A1A1A1A1A1A1A1A1A
2320 IF Alpha(Z)=1 THEN 2380  ! CHECK TO SEE IF ALPHA ARGUMENTS ARE O.K.
2330 DISP "ALPHA ARGUMENTS NOT ALLOWED FOR THIS FUNCTION"
2340 DISP "PLEASE ENTER FUNCTION AGAIN"
2350 GOTO 990
2380   Byte(2)=64
2390   Byte(3)=Datar(Z)*16+Datac(Z)
2400   Start=4                  ! Start IS THE Byte NUMBER FOR THE FIRST BYTE
2410                           ! OF THE ALPHA ARGUMENT TEXT STRING
2420   End=3+LEN(A1$)           ! End IS THE Byte NUMBER FOR THE LAST BYTE OF THE
2430                           ! ALPHA ARGUMENT TEXT STRING
2440   GOSUB 5610               ! CREATE ALPHA STRING
2450   N=3+LEN(A1$)             ! N IS THE NUMBER OF BYTES IN THE CODE
2480   GOTO 2040
2500 ! 1N1N1N1N1N HERE FOR SINGLE ARGUMENT, NUMERIC TYPE 1N1N1N1N1N1N1N1N1N1N
2520   IF A1<>4 THEN 2590       ! CHECK FOR A STACK OPERATION
2530    IF A1$="X" THEN X=115
2540    IF A1$="Y" THEN X=114
2550    IF A1$="Z" THEN X=113
2560    IF A1$="T" THEN X=112
2570    IF A1$="L" THEN X=116
```

```
2580    GOTO 2700
2590   IF F$="GTO." THEN 2870 ! CHECK FOR GTO.
2600   IF F$="SIZE" THEN 2870 ! CHECK FOR SIZE
2610   IF F$="DEL" THEN 2870  ! CHECK FOR DEL
2620   X=INT(VAL(A1$))
2630   IF (X>=0) AND (X<=99) THEN 2700  ! CHECK FOR RANGE OF ARGUMENT
2650   DISP " ARGUMENT OUT OF RANGE "
2660   GOTO 990
2680   IF A1=3 THEN X=NUM(A1$)+37      ! WEVE JUMPED HERE FOR ARG A-J
2690   IF A1=3.5 THEN X=NUM(A1$)+26    ! WEVE JUMPED HERE FOR ARG a-e
2700   Byte(2)=64
2710   Byte(3)=Datar(Z)*16+Datac(Z)    ! SET FUNCTION CODE
2720                           ! CHECK FOR PROPER FUNCTION CODES FOR FUNCTIONS
2730                           ! THAT HAVE DIFFERENT FCN CODES FOR DIFFERENT
2740                           ! ARGUMENT TYPES LIKE: XEQ, GTO, LBL
2750   IF F$="XEQ" THEN Byte(3)=224
2760   IF F$="GTO" THEN Byte(3)=208
2770   IF F$="LBL" THEN Byte(3)=207
2790   Byte(4)=X
2800   N=4                      ! N IS THE NUMBER OF BYTES IN CODE
2830   GOTO 2040
2850   ! %%%%%%%%%%%%%%% HERE FOR DEL,GTO. OR SIZE %%%%%%%%%%%%%%%%%%%%%%%%%%
2870   Byte(2)=64
2880   Byte(3)=16*Datar(Z)+Datac(Z)
2890   IF (F$="GTO.") AND (A1$=".") THEN Byte(4)=15 !
2900   IF (F$="GTO.") AND (A1$=".") THEN Byte(5)=255! SET CORRECT BITS FOR GTO..
2910   IF (F$="GTO.") AND (A1$=".") THEN 2940       !
2920   Byte(4)=VAL(A1$) DIV 256    !
2930   Byte(5)=VAL(A1$) MOD 256    ! SET CORRECT BITS FOR SIZE
2940   N=5                      ! 5 BYTES IN THE CODE
2970   GOTO 2040
2990   ! &&&&&&&&&&&&&& HERE FOR INDIRECT SINGLE ARGUMENT &&&&&&&&&&&&&&&&&&&&
3010   IF La2>0 THEN 3060      ! CHECK FOR ARGUMENT FOR IND
3030   DISP "MISSING ARGUMENT FOR 'IND' ",I$
3040   GOTO 990
3060   IF A2<>4 THEN 3130      ! CHECK FOR STACK ARGUMENT
3070   IF A2$="X" THEN X=115
3080   IF A2$="Y" THEN X=114
3090   IF A2$="Z" THEN X=113
3100   IF A2$="T" THEN X=112
3110   IF A2$="L" THEN X=116
3120   GOTO 3220
3130   IF A2=0 THEN 3200       ! CHECK FOR NUMERIC ARGUMENT.  ONLY STACK
3140                           ! AND NUMERIC ARGUMENTS ARE ALLOWED SO IF
3150                           ! ANYTHING ELSE IS ENCOUNTERED, FLAG THE ERROR
3170   DISP "IMPROPER ARGUMENT FOR 'IND' ",I$
3180   GOTO 990
3200   X=INT(VAL(A2$))
3210   IF (X<0) OR (X>99) THEN 3170    ! CHECK FOR RANGE FOR ARGUMENT
3220   Byte(2)=64
3230   Byte(3)=Datar(Z)*16+Datac(Z)
3231   IF (F$="XEQ") OR (F$="GTO") THEN Byte(3)=174
3240   Byte(4)=128+X
3241   IF F$="GTO" THEN Byte(4)=X
3250   N=4                      ! 4 BYTES IN CODE
3280   GOTO 2040
3300 ! 2222222222 HERE FOR TWO ARGUMENT FUNCTIONS 22222222222222222222
3320 IF (La1>0) AND (La2>0) THEN 3370  ! JUMP AHEAD IF THERE ARE 2 ARGUMENTS
3340 DISP "TWO ARGUMENTS ARE REQUIRED.  PLEASE ENTER FUNCTION AGAIN"
3350 GOTO 990
3370 IF A1=1 THEN 3420          ! FIRST ARGUMENT MUST BE ALPHA TEXT STRING
3380 IF (A1<>0) OR (A2<>0) THEN 3390
3381 IF F$<>"XROM" THEN 3390
3382 Byte(2)=64
3383 Byte(3)=Datar(Z)*16+Datac(Z)+VAL(A1$) DIV 4
```

```
3384 Byte(4)=VAL(A1$) MOD 4*64+VAL(A2$) MOD 64
3385 GOTO 3250
3390 DISP "IMPROPER FIRST ARGUMENT.  PLEASE ENTER FUNCTION AGAIN"
3400 GOTO 990
3420 IF A2=0 THEN 3470         ! SECOND ARGUMENT MUST BE NUMERIC
3440 DISP "IMPROPER SECOND ARGUMENT.  PLEASE ENTER FUNCTION AGAIN"
3450 GOTO 990
3470 Byte(2)=64
3480 Byte(3)=Datar(Z)*16+Datac(Z)
3490 Asn=VAL(A2$)
3491 IF ABS(Asn)<85 THEN 3500
3493 DISP "KEYCODE IS OUT OF RANGE ";A2$
3494 GOTO 990
3500 Up=ABS(Asn) DIV 10
3510 Dn=ABS(Asn) MOD 10
3520 IF Asn<0 THEN Up=(ABS(Asn) DIV 10+8) MOD 16
3530 Byte(4)=Up*16+Dn
3540 Start=5                   ! Start IS THE Byte NUMBER OF THE FIRST BYTE IN
3550                           ! THE TEXT STRING
3560 End=LEN(A1$)+4            ! End IS THE Byte NUMBER OF THE LAST BYTE IN THE
3570                           ! TEXT STRING
3580 GOSUB 5610                ! CREATE ALPHA STRING
3590 N=4+LEN(A1$)              ! N IS THE NUMBER OF BYTES IN THE CODE
3620 GOTO 2040
3640 ! ************************************************************************
3660                ! HERE FOR DATA TYPE BAR CODE
3680 GOSUB 4890                ! CHECK FOR THE ARGUMENT TYPE
3690 IF LEN(A1$)=0 THEN 4100
3700 IF (A1>4) OR (A2>4) THEN 990    ! CHECK FOR ERRORS IN ARGUMENTS
3710 IF A1=0 THEN 3800         ! NUMERIC TYPE DATA
3720 IF A1=1 THEN 4341         ! ALPHA REPLACE TYPE DATA
3730 IF A1=2 THEN 4171         ! ALPHA APPEND TYPE DATA
3750 ! N N N N N N N N N N NUMERIC DATA N N N N N N N N N N N N N N N N N N N N
3770                ! HERE FOR NUMERIC TYPE DATA
3790 !   SET THE DIGITS
3800 IF LEN(A1$)<>1 THEN 3805
3801 A1$="0"&A1$
3802 La1=2
3805 FOR I=1 TO La1
3810     X1=NUM(A1$[I,I])
3820     IF (X1>=48) AND (X1<=57) THEN D(I)=VAL(A1$[I,I])
3830     IF A1$[I,I]="." THEN D(I)=11
3840     IF A1$[I,I]="+" THEN D(I)=12
3850     IF A1$[I,I]="-" THEN D(I)=13
3860     IF A1$[I,I]="E" THEN D(I)=14
3870     IF (D(I)<0) OR (D(I)>14) THEN 4100   !EXIT LOOP ON IMPROPER DIGIT
3880 NEXT I
3881 IF F$="DATASEQ" THEN 4062
3882 IF LEN(A1$)>1 THEN 3890
3883 Byte(2)=6*16+10
3884 Byte(3)=10*16+D(1)
3885 N=3
3886 GOTO 2040
3890     Byte(2)=6*16+D(1)     ! SET TYPE CODE. SINCE TWO DIGITS CAN BE
3900                           ! PACKED INTO A SINGLE BYTE, A NULL HAS TO BE
3910                           ! APPENDED IF THERE ARE AN EVEN NUMBER OF DIGITS
3930     IF LEN(A1$) MOD 2=0 THEN Byte(2)=6*16+10  ! THIS ADDS A NULL
3940     J=2                                 ! J TELLS WHERE TO START IN D(*)
3950     IF LEN(A1$) MOD 2=0 THEN J=1  ! J TELLS WHERE TO START IN D(*)
3960     K=3                   ! K IS THE Byte POINTER
3980     FOR I=J TO LEN(A1$) STEP 2 ! THIS LOOP SETS THE DIGITS INTO Byte
3990         Byte(K)=16*D(I)+D(I+1)
4000         K=K+1
4010     NEXT I
4030     N=2+LEN(A1$) DIV 2   ! N IS THE NUMBER OF BYTES IN THE CODE
```

```
4060     GOTO 2040
4062 Seq=VAL(A2$)              ! HERE FOR DATASEQ NUMBERS
4063 Byte(2)=9*16+(Seq-1) DIV 256
4064 Byte(3)=(Seq-1) MOD 256  ! Byte(2) AND (3) ARE THE TYPE AND SEQ #'S
4065 K=4                      ! K IS THE Byte NUMBER
4066 J=1                      ! J IS THE POINTER INTO D(*)
4067 IF LEN(A1$) MOD 2=0 THEN 4071  !JUMP AHEAD FOR AN EVEN NUMBER OF DIGITS
4068 K=5                      ! HERE FOR AN ODD NUMBER OF DIGITS
4069 J=2
4070 Byte(4)=16*10+D(1)
4071 FOR I=J TO LEN(A1$) STEP 2   ! THIS LOOP SETS THE DIGITS INTO Byte(*)
4072     Byte(K)=16*D(I)+D(I+1)
4073     K=K+1
4074 NEXT I
4075 N=3+LEN(A1$) MOD 2+LEN(A1$) DIV 2  ! N IS THE LENGTH OF THE BAR CODE
4076 GOTO 2040                 ! JUMP BACK TO DO CHECKSUM AND OUTPUT
4080             ! ERROR IN NUMERIC DATA ARGUMENT
4100 DISP "ERROR IN NUMERIC DATA",I$
4110 GOTO 990
4130 ! A A A A A A A A A A A ALPHA APPEND A A A A A A A A A A A A A A A A
4150                 ! HERE FOR ALPHA APPEND DATA
4171 IF F$="DATASEQ" THEN 4311
4180 Byte(2)=128+LEN(A1$)     ! Byte(2) IS THE NUMBER OF CHARACTERS IN STRING
4200 ! SET THE REST OF THE BYTES WITH THE ASCII EQUIVALENT OF THE STRING
4220     Start=3              ! Start IS THE Byte NUMBER OF THE FIRST BYTE
4230                          ! IN THE TEXT STRING
4240     End=LEN(A1$)+2       ! End IS THE byte NUMBER OF THE LAST BYTE IN THE
4250                          ! TEXT STRING
4260     GOSUB 5610           ! SET THE Bytes WITH THE ASCII CHARACTERS
4270     N=LEN(A1$)+2         ! N IS THE LENGTH OF THE CODE IN BYTES
4300     GOTO 2040
4311 Seq=VAL(A2$)             ! HERE FOR DATASEQ ALPHA'S
4312 Byte(2)=11*16+(Seq-1) DIV 256
4313 Byte(3)=(Seq-1) MOD 256
4314 Start=4
4315 End=LEN(A1$)+3
4316 GOSUB 5610
4317 N=LEN(A1$)+3
4318 GOTO 2040
4320 ! A A A A A A A A A A A ALPHA REPLACE A A A A A A A A A A A A A A A A
4341 IF F$="DATASEQ" THEN 4441
4350     Byte(2)=112+LEN(A1$) ! Byte(2) IS THE NUMBER OF CHARACTERS
4360     Start=3              ! NUMBER OF FIRST Byte IN THE TEXT STRING
4370     End=LEN(A1$)+2       ! NUMBER OF THE LAST Byte IN THE TEXT STRING
4380     GOSUB 5610           ! ASSIGN ASCII CHARACTERS TO Bytes
4390     N=LEN(A1$)+2         ! N IS LENGTH OF CODE IN BYTES
4420     GOTO 2040
4441  Seq=VAL(A2$)
4442  Byte(2)=10*16+(Seq-1) DIV 256
4443  GOTO 4313
4460  ! ******************** END OF MAIN ROUTINE **************************
4500  !  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
4520  ! CHECKSUM CALCULATION ROUTINE
4540  ! THIS ROUTINE CALCULATES AN EIGHT BIT CHECKSUM FOR AN N BYTE
4550  ! BAR CODE.  N MUST BE INITIALIZED TO THE NUMBER OF BYTES IN THE CODE
4560  ! BEFORE THIS ROUTINE IS CALLED.
4580   Cksm=0
4590   FOR I=2 TO N           ! N IS THE NUMBER OF BYTES IN CODE
4600     Cksm=Cksm+Byte(I)
4610     DISP "BYTE ",I,Byte(I)
4620   NEXT I
4630   Cksm=Cksm DIV 256+Cksm MOD 256
4640   Cksm=Cksm DIV 256+Cksm MOD 256
4650   Byte(1)=Cksm
4660   DISP "CHECK SUM ",Byte(1)
4670   RETURN
```

```
4700 ! ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
4710 ! ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
4730 !                      ARGUMENT TYPE ROUTINE
4750 ! THIS ROUTINE DETERMINES THE TYPE OF THE TWO POSSIBLE ARGUMENTS AND
4760 ! RETURNS THIS INFORMATION TO THE MAIN PROGRAM IN A1 FOR THE FIRST
4770 ! ARGUMENT AND A2 FOR THE SECOND.  THE ARGUMENT TYPES ARE DEFINED AS
4780 ! FOLLOWS:
4800 !     TYPE               MEANING
4810 !      0                   NUMERIC DATA
4820 !      1                   ALPHA REPLACE DATA
4830 !      2                   ALPHA APPEND DATA
4840 !      3                   A THROUGH J
4850 !      3.5                 a THROUGH e
4860 !      4                   STACK ARGUMENTS
4870 !      5                   ERRORS IN ARGUMENTS
4890 A1=0
4900 A2=0
4910 IF La1=0 THEN RETURN
4911 IF F$<>"XROM" THEN 4920
4912 IF La1>2 THEN A1=5
4913 IF La2>2 THEN A2=5
4914 IF (A1<>5) AND (A2<>5) THEN 4920
4915 DISP "ARGUMENT/S TOO LARGE."
4916 RETURN
4920 IF A1$="IND" THEN 5240   ! FOR IND CHECK THE SECOND ARGUMENT
4930 IF (La1=1) AND (A1$="E") THEN 5000
4940 FOR I=1 TO La1
4950     X1=NUM(A1$[I,I])
4960     IF ((X1<43) OR (X1>57)) AND (X1<>69) THEN 4990
4970 NEXT I
4980 GOTO 5240            ! CHECK FOR SECOND ARGUMENT
4990     IF La1>1 THEN 5080
5000     IF (NUM(A1$)>64) AND (NUM(A1$)<75) THEN A1=3
5010     IF (NUM(A1$)>96) AND (NUM(A1$)<102) THEN A1=3.5
5020     IF A1$="X" THEN A1=4
5030     IF A1$="Y" THEN A1=4
5040     IF A1$="Z" THEN A1=4
5050     IF A1$="T" THEN A1=4
5060     IF A1$="L" THEN A1=4
5070     GOTO 5200
5080     IF (A1$[1,1]="A") AND (A1$[2,2]="/") AND (A1$[La1,La1]="/") THEN A1=2
5090     IF (A1$[1,1]="/") AND (A1$[La1,La1]="/") THEN A1=1
5100     IF A1>0 THEN 5160
5120        DISP "ERROR IN DATA.  CHECK FOR MISSING / OR IMPROPER NUMBER"
5130        A1=5
5150     RETURN
5160     IF A1=1 THEN A1$=A1$[2,La1-1]
5170     IF A1=2 THEN A1$=A1$[3,La1-1]
5180     IF A1=3 THEN A1$=A1$
5190     IF A1=3.5 THEN A1$=A1$
5200     IF ((F$="STO") OR (F$="RCL")) AND ((A1=3) OR (A1=3.5)) THEN 5120
5210     La1=LEN(A1$)
5220                    ! HERE FOR SECOND ARGUMENT
5240  IF La2=0 THEN RETURN
5250  FOR I=1 TO La2
5260     X1=NUM(A2$[I,I])
5270     IF ((X1<43) OR (X1>57)) AND (X1<>69) THEN 5300
5280  NEXT I
5290  GOTO 5470
5300     IF La2>1 THEN 5390
5310     IF A2$="X" THEN A2=4
5320     IF A2$="Y" THEN A2=4
5330     IF A2$="Z" THEN A2=4
5340     IF A2$="T" THEN A2=4
5350     IF A2$="L" THEN A2=4
5360     IF (NUM(A2$)>64) AND (NUM(A2$)<75) THEN A2=3
```

```
5370      IF (NUM(A2$)>96) AND (NUM(A2$)<102) THEN A2=3.5
5380      GOTO 5470
5390      IF (A2$[1,1]="A") AND (A2$[2,2]="'") AND (A2$[La2,La2]="'") THEN A2=2
5400      IF (A2$[1,1]="'") AND (A2$[La2,La2]="'") THEN A2=1
5410      IF A2>0 THEN 5470
5430         DISP "ERROR IN DATA.  CHECK FOR MISSING ' OR IMPROPER NUMBER"
5440         A2=5
5450      RETURN
5470      IF A2=1 THEN A2$=A2$[2,La2-1]
5480      IF A2=2 THEN A2$=A2$[3,La2-1]
5490      IF A2=3 THEN A2$=A2$
5500      IF A2=3.5 THEN A2$=A2$
5510      IF A2=4 THEN A2$=A2$
5520      La2=LEN(A2$)
5530      RETURN
5560   ! ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
5590   !                   ALPHA STRING PLACEMENT ROUTINE
5610     K=1
5620     FOR J=Start TO End
5630         Byte(J)=NUM(A1$[K,K])
5640         IF A1$[K,K]="@" THEN Byte(J)=13
5650         IF A1$[K,K]="#" THEN Byte(J)=29
5660         IF A1$[K,K]="&" THEN Byte(J)=126
5670         K=K+1
5680     NEXT J
5690     RETURN
5720   ! ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
5740   !                     OUTPUT STRING GENERATION
5760   FOR J=1 TO N
5770   Dec(J)=Byte(J)
5780      FOR I=J*8 TO (J-1)*8+1 STEP -1
5790          Bar(I)=Dec(J) MOD 2
5800          Dec(J)=Dec(J) DIV 2
5810      NEXT I
5820   NEXT J
5830   B$=""
5840   FOR I=1 TO 8*N
5850      IF Bar(I)=0 THEN B$=B$&">>   "
5860      IF Bar(I)=1 THEN B$=B$&">>>>   "
5870   NEXT I
5880   B$=">>   >>   "&B$&">>>>   >> "
5900                         ! PRINT TITLE
5920   PRINT I$,CHR$(27)&CHR$(31)&CHR$(2) ! ALSO SET HMI
5940                         ! PRINT BUFFER
5960   IF (F$="DATA") OR (F$="DATASEQ") THEN PRINT USING 6020;B$
5970   IF (F$="DATA") OR (F$="DATASEQ") THEN 6030
5980   IF Arg(Z)=0 THEN PRINT USING 6010;B$
5990   IF Arg(Z)=1 THEN PRINT USING 6020;B$
6000   IF Arg(Z)=2 THEN PRINT USING 6020;B$
6010   IMAGE 200A
6020   IMAGE 850A
6030   PRINT CHR$(27)&CHR$(83)
6050                       ! CHECK FOR A FILLED PAGE
6070   Line=Line+1
6080   IF Line<15 THEN RETURN  ! IF THERE ARE LESS THAN 15 LINES, RETURN
6090   PRINT CHR$(12)          ! SEND OUT A FORM FEED
6100   Line=0
6110   RETURN
6130   ! ********************************************************************
6150   !                            ERRORS
6170   IF ERRN<>18 THEN 6250
6180   DISP "INPUT STRING WAS TOO LONG, PLEASE CHECK IT."
6190   DISP "CHECK LINE NUMBER ",ERRL
6200   DISP "I= ",I
6210   DISP "LENGTH OF A$= ",LEN(A$)
6220   DISP "LENGTH OF A1$= ",LEN(A1$)
```

```
6230  DISP "LENGTH A$ SHOULD BE IS ",L
6240  GOTO 990
6250  IF ERRN<>32 THEN 6310
6260  IF ERRL=3170 THEN 3170
6270  IF ERRL=2590 THEN 2650
6280  DISP "POSSIBLE PROBLEM IN DATA TABLE."
6290  DISP "CHECK LINE NUMBER ",ERRL
6300  STOP
6310  DISP "ERROR NUMBER ",ERRN
6320  DISP "LINE NUMBER ",ERRL
6330  STOP
6350                          ! TABLE DATA
6361  DATA %,4,12,0,0
6362  DATA %CH,4,13,0,0
6363  DATA &+,4,7,0,0
6364  DATA &-,4,8,0,0
6365  DATA &REG,9,9,1,0
6370  DATA ABS,6,1,0,0
6380  DATA ACOS,5,13,0,0
6390  DATA ADV,8,15,0,0
6400  DATA AOFF,8,11,0,0
6410  DATA AON,8,12,0,0
6420  DATA ARCL,9,11,1,0
6430  DATA ASHF,8,8,0,0
6440  DATA ASIN,5,12,0,0
6450  DATA ASN,0,15,2,1
6460  DATA ASTO,9,10,1,0
6470  DATA ATAN,5,14,0,0
6480  DATA AVIEW,7,14,0,0
6490  DATA BEEP,8,6,0,0
6500  DATA BST,0,7,0,0
6510  DATA CAT,0,0,1,0
6520  DATA CF,10,9,1,0
6530  DATA CHS,5,4,0,0
6535  DATA CL&,7,0,0,0
6540  DATA CLA,8,7,0,0
6550  DATA CLD,7,15,0,0
6560  DATA CLP,0,4,1,1
6570  DATA CLRG,8,10,0,0
6590  DATA CLST,7,3,0,0
6600  DATA CLX,7,7,0,0
6610  DATA COPY,0,3,1,1
6620  DATA COS,5,10,0,0
6630  DATA D-R,6,10,0,0
6640  DATA DEC,5,15,0,0
6650  DATA DEG,8,0,0,0
6660  DATA DEL,0,2,1,0
6670  DATA DELETE,0,11,0,0
6680  DATA DSE,9,7,1,0
6690  DATA END,12,0,0,0
6700  DATA ENG,9,14,1,0
6710  DATA ENTER^,8,3,0,0
6720  DATA E^X,5,5,0,0
6730  DATA E^X-1,5,8,0,0
6740  DATA FACT,6,2,0,0
6750  DATA FC?,10,13,1,0
6760  DATA FC?C,10,11,1,0
6770  DATA FIX,9,12,1,0
6780  DATA FRC,6,9,0,0
6790  DATA FS?,10,12,1,0
6800  DATA FS?C,10,10,1,0
6810  DATA GRAD,8,2,0,0
6820  DATA GTO,1,13,1,1
6830  DATA GTO.,0,1,1,0
6840  DATA HMS,6,12,0,0
6850  DATA HMS+,4,9,0,0
6860  DATA HMS-,4,10,0,0
6870  DATA HR,6,13,0,0
6880  DATA INT,6,8,0,0
6890  DATA ISG,9,6,1,0
6900  DATA LASTX,7,6,0,0
6910  DATA LBL,12,13,1,1
6920  DATA LN,5,0,0,0
6930  DATA LN1+X,6,5,0,0
6940  DATA LOG,5,6,0,0
6950  DATA MEAN,7,12,0,0
6960  DATA MOD,4,11,0,0
6970  DATA OCT,6,15,0,0
6980  DATA OFF,8,13,0,0
6990  DATA ON,0,9,0,0
7000  DATA P-R,4,14,0,0
7010  DATA PACK,0,10,0,0
7020  DATA PI,7,2,0,0
7030  DATA PROMPT,8,14,0,0
7040  DATA PSE,8,9,0,0
7050  DATA R-D,6,11,0,0
7060  DATA R-P,4,15,0,0
7070  DATA R/S,0,5,0,0
7080  DATA RAD,8,1,0,0
7090  DATA RCL,9,0,1,0
7100  DATA RDN,7,5,0,0
7110  DATA RND,6,14,0,0
7120  DATA RTN,8,5,0,0
7130  DATA RUP,7,4,0,0
7135  DATA SCI,9,13,1,0
7140  DATA SDEV,7,13,0,0
7160  DATA SF,10,8,1,0
7190  DATA SIGN,7,10,0,0
7210  DATA SIN,5,9,0,0
7220  DATA SIZE,0,6,1,0
7230  DATA SQRT,5,2,0,0
7240  DATA SST,0,8,0,0
7250  DATA ST*,9,4,1,0
7260  DATA ST+,9,2,1,0
7270  DATA ST-,9,3,1,0
7280  DATA ST/,9,5,1,0
7290  DATA STO,9,1,1,0
7300  DATA STOP,8,4,0,0
7310  DATA TAN,5,11,0,0
7320  DATA TONE,9,15,1,0
7330  DATA VIEW,9,8,0,0
7340  DATA X#0?,6,3,0,0
7350  DATA X#Y?,7,9,0,0
7360  DATA X<0?,6,6,0,0
7370  DATA X<=0?,7,11,0,0
7380  DATA X<=Y?,4,6,0,0
7390  DATA X<>,12,14,1,0
7400  DATA X<>Y,7,1,0,0
7410  DATA X<Y?,4,4,0,0
7420  DATA X=0?,6,7,0,0
7430  DATA X=Y?,7,8,0,0
7440  DATA X>0?,6,4,0,0
7450  DATA X>Y?,4,5,0,0
7460  DATA XEQ,1,14,1,1
7465  DATA XROM,10,0,2,0
7470  DATA X^2,5,1,0,0
7480  DATA Y^X,5,3,0,0
7490  DATA *END*,0,0,0,0
```

Section 4
Algorithms

Checksum Algorithms

All rows of HP-41 bar code have a checksum in the leftmost positions of the row. All but the Paper Keyboard have an 8-bit end-around carry checksum. The two byte Paper Keyboard bar code uses only a 4-bit end-around carry checksum. Following are the algorithms for checksums stated in a block-structured language.

Eight-Bit (One Byte) Checksum

The checksums for all HP-41 bar code, except Program bar code, are local checksums and are calculated separately for each row. For Program bar code (Type 1 and 2), a running checksum is used. That is, the checksum for each row is the sum of that row plus the sum of all previous rows.

```
Let X be a running checksum. (This would be cleared at the beginning of each row for all but
      Types 1 and 2.)
Let N be the number of bytes in this row of bar code.
Let Y be an array holding bytes two through N.
Let Z be a temporary variable.
Let I be a pointer into the byte array Y
   BEGIN
      If TYPE < > 1 and TYPE < >2 Then X = 0
      For I = 2 to N Do
      BEGIN
         Z = X mod 256 + Y(I) mod 256
         X = Z mod 256
         If Z  > 255 then X = X + 1
   END
      Y(1) = X mod 256
   END
```

Four-Bit Checksums

Four-bit checksums are found only in Two Byte Paper Keyboard bar code. They use end-around carry but are summed in 4-bit nybbles.

```
Let SUM be the calculated checksum.
Let Y   be the array that holds the two bytes.
Let I   be the pointer to the Y array.
Let K   be the loop counter.
  BEGIN
    SUM = 0
    I = 1
    BEGIN
      If K > 1 Then I = 2
      If K = 2 Then J = Y(I) DIV 16
               Else J = Y(I) MOD 16
      SUM = SUM MOD 16 + J
      If SUM > 15 Then SUM = (SUM MOD 16) + 1
    END
    Y(1) = SUM*16 + Y(1) MOD 16
  END
```

Program Bar Code Algorithms

In order to write your own software to generate program bar code you will need a good understanding of the HP-41 and the internal structures of program memory. The HP-41 Owner's Manual and Programming Guide is a good resource, particularly appendix D, Program Memory Storage Requirements and Last X Operations. Other important references for this section are Table III, The HP-41 Function Code Table (page 45), and Table IV, The Numeric Values for A-J, a-e, and the Stack (page 50). It will also be helpful to refer back to program "PRGMBR" to see how these algorithms are implemented in a working program.

In a row of Program bar code, the code for a simple one byte function appears in program memory as a combination of the row and column location of that function in Table III. For most two-byte functions (for example, a function with a numeric argument) the first byte contains the function's location in the table; the second byte contains the value of the argument.

There are also several functions that are not as straightforward and need further explanations. These are enumerated following Table III.

HIGH ORDER BITS	LOW ORDER 4 BITS 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	NULL	LBL 00	LBL 01	LBL 02	LBL 03	LBL 04	LBL 05	LBL 06	LBL 07	LBL 08	LBL 09	LBL 10	LBL 11	LBL 12	LBL 13	LBL 14	
1	digit 0	1	2	3	4	5	6	7	8	9	.	EEX	(digit entry) CHS	GTO α	XEQ α		
2	RCL 00	RCL 01	RCL 02	RCL 03	RCL 04	RCL 05	RCL 06	RCL 07	RCL 08	RCL 09	RCL 10	RCL 11	RCL 12	RCL 13	RCL 14	RCL 15	
3	STO 00	STO 01	STO 02	STO 03	STO 04	STO 05	STO 06	STO 07	STO 08	STO 09	STO 10	STO 11	STO 12	STO 13	STO 14	STO 15	
4	+	−	*	/	X < Y?	X > Y?	X < = Y?	Σ+	Σ−	HMS+	HMS−	MOD	%	%CH	P−R	R−P	ONE BYTE
5	LN	X^2	SQRT	Y^X	CHS	e^X	LOG	10^X	e^X-1	SIN	COS	TAN	ASIN	ACOS	ATAN	DEC	
6	1/X	ABS	FACT	X ≠ 0?	X > 0?	LN(1+X)	X < 0?	X=0?	INT	FRAC	D−R	R−D	HMS	HR	RND	OCT	
7	CL	X < > Y	PI	CLST	R↓	RDN	LASTX	CLX	X=Y?	X ≠ Y?	SIGN	X < = 0?	MEAN	SDEV	AVIEW	CLD	
8	DEG	RAD	GRAD	ENTER↑	STOP	RTN	BEEP	CLA	ASHF	PSE	CLRG	AOFF	AON	OFF	PROMPT	ADV	
9	RCL nn	STO nn	ST+ nn	ST− nn	ST* nn	ST/ nn	ISG nn	DSE nn	VIEW nn	ΣREG nn	ASTO nn	ARCL nn	FIX n	SCI n	ENG n	TONE n	TWO BYTE
10	XROM	XROM	XROM	XROM	XROM	XROM	XROM	XROM	SF nn	CF nn	F?C nn	FC?C nn	FS? nn	FC? nn	GTO/XEQ IND		
11		GTO 00	GTO 01	GTO 02	GTO 03	GTO 04	GTO 05	GTO 06	GTO 07	GTO 08	GTO 09	GTO 10	GTO 11	GTO 12	GTO 13	GTO 14	
12	← ALPHA LABEL AND END INSTRUCTIONS														X < > nn	LBL nn	
13	← GTO nn															→	THREE BYTE
14	← XEQ nn															→	
15		TEXT 1	TEXT 2	TEXT 3	TEXT 4	TEXT 5	TEXT 6	TEXT 7	TEXT 8	TEXT 9	TEXT 10	TEXT 11	TEXT 12	TEXT 13	TEXT 14	TEXT 15	UP TO 16 BYTE

Table III.

Lables [LBL]

The [LBL] function is found in three different locations in Table III.

Short Form Numeric Labels. The first place it appears is at Row 0, columns 1 through 14. This is a short form numeric label ([LBL] 00 through [LBL] 14), which requires only one byte in program memory.

Long Form Numeric Labels. Row 12, Column 15 represents numeric labels with argument values greater than fourteen, (this includes the local Alpha Labels A-J, and a-e). The first byte in this two byte label contains the function location 12,15 (11001111) and the second byte contains the argument value. For the local Alpha Labels the values of A-J and a-e are found in Table IV, page 50.

Global Alpha Labels. These labels are found at Row 12, Columns 0 through 13. Their structure in program memory is as follows:

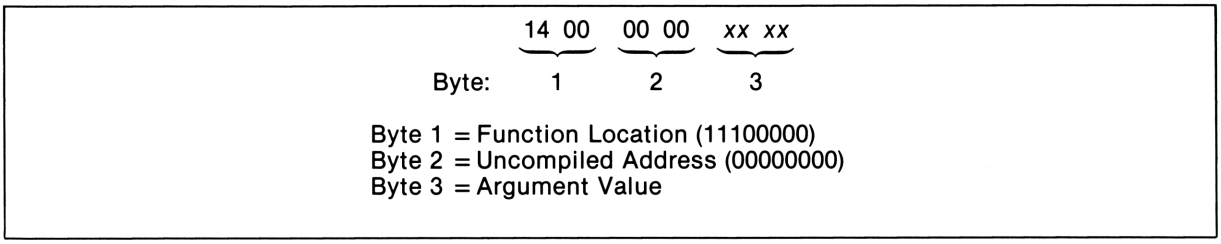
	12 00	00 00	15 xx	00 00	CCCC	CCCC	• • • •	• • • •	CCCC
Byte:	1	2	3	4	5	6			xx+3

Byte 1 = Function Location (11000000)
Byte 2 = Uncompiled Label Chain (00000000)
Byte 3 = String code + (No. of ASCII Chars. + 1)(1111xxxx)
Byte 4 = Reserved for Key Assignment (00000000)
Byte 5 = 1st Character (ASCII Value)
• • • •
• • • •
Byte xx + 3 = Last Character

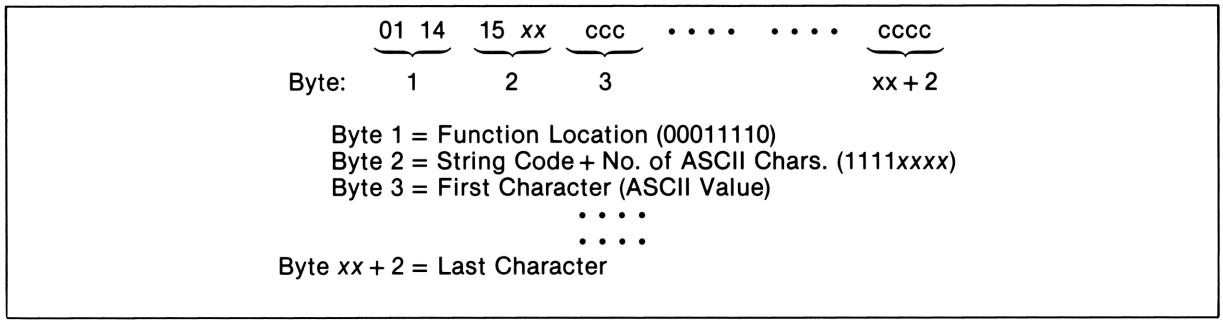
Execute [XEQ]

The [XEQ] function is also found in three different locations in the HP-41 Function Table (Table III).

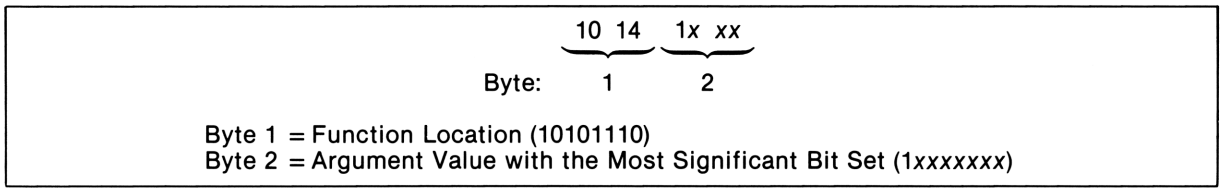
Numeric and Local Alpha Execute. This function ([XEQ] *nn*) is found at Row 14, Columns 0 through 15.



Global Alpha Execute. This function ([XEQ] α) is found at Row 1, Column 14.



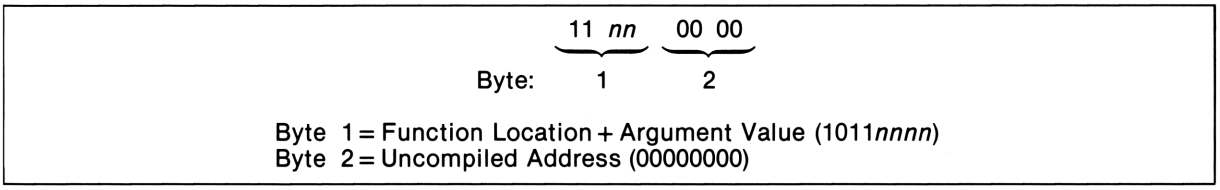
Indirect Execution. This function ([XEQ] [IND] *nn*) is found at Row 10, Column 14.



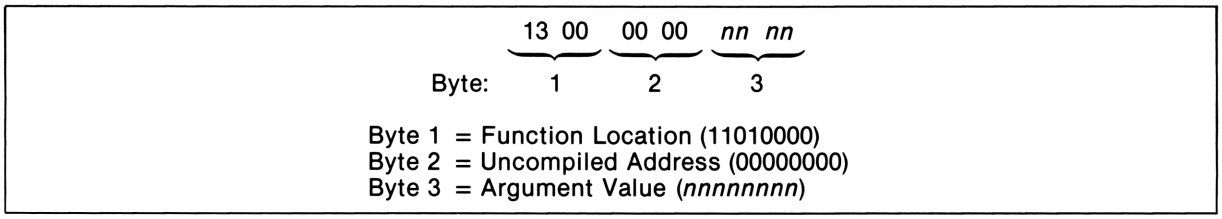
Go To [GTO]

The [GTO] function has four locations in the Function Table. It has both short and long form numeric, as well as [GTO] [IND] and [GTO] α.

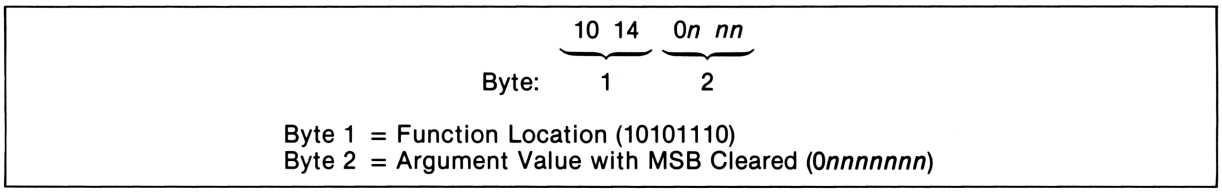
Short Form Numeric Go To. ([GTO] *nn* for *nn* <15). This is found at Row 11, Columns 1 through 15.



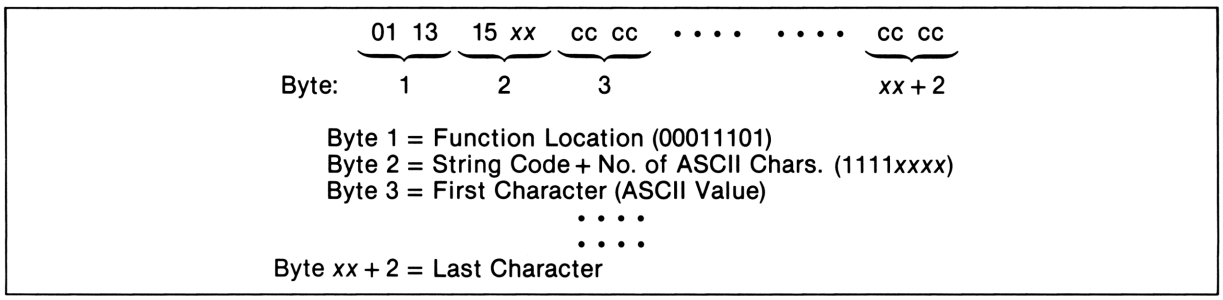
Long Form Numeric and Local Alpha Go To. ([GTO] *nn* for *nn* > 14). This is found at Row 13, Columns 0 through 15.



Indirect Go To. ([GTO] [IND] *nn*). This is found at Row 10, Column 14.



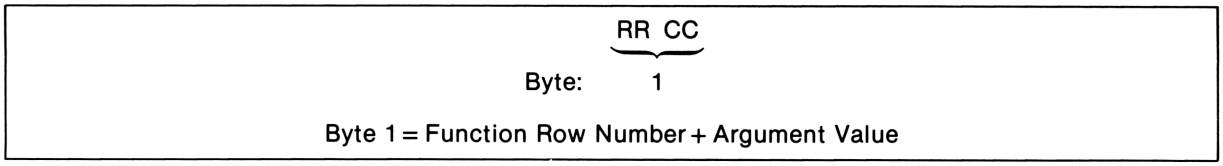
Global Alpha Go To. ([GTO] α). This is found at Row 1, Column 13.



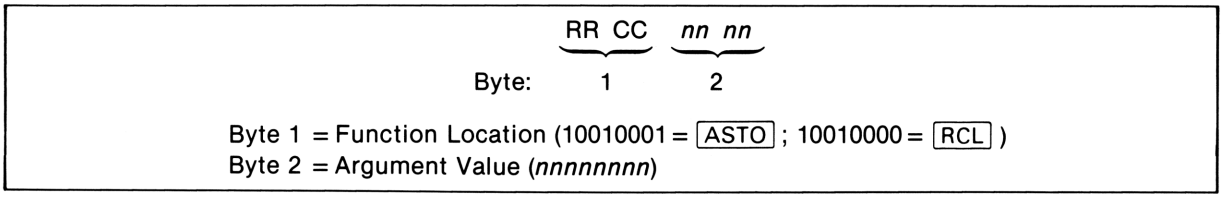
Store and Recall [STO], [RCL]

Both of these functions have two numeric locations in the table as well as one alpha location. In both cases to make the [STO] or [RCL] [IND] the most significant bit of the second byte (the argument value) is set to one.

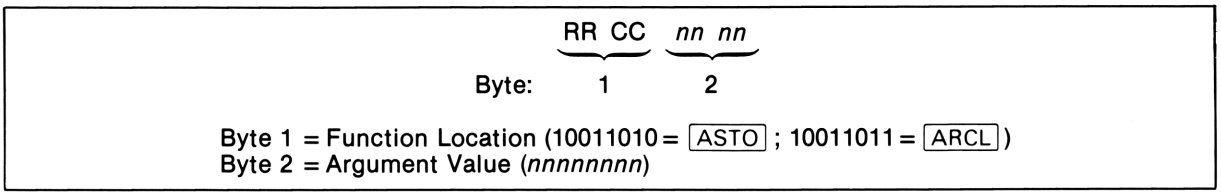
Short Form Numeric ([STO] *nn*, [RCL] *nn*; *nn* <= 15). These are found at Rows 3 and 2, Columns 0 through 15.



Long Form Numeric ([STO] *nn*, [RCL] *nn*; *nn* > 15). These are found at Row 9, Columns 1 and 0.

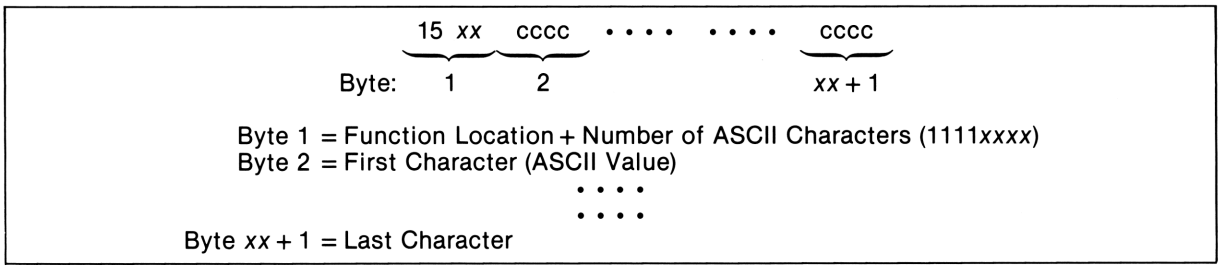


Alpha ([ASTO] *nn*, [ARCL] *nn*). These are found at Row 9, Columns 10 and 11.



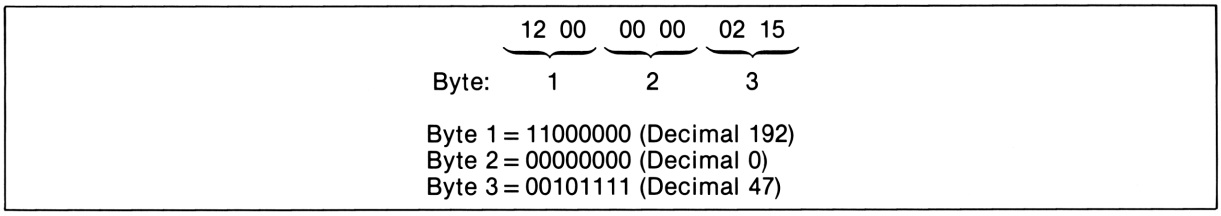
Alpha Text Strings

This function is found at Row 15, Columns 1 through 15.



End [END]

Each bar coded program must have an [END] function as the last three bytes of the program. It must be set to the following values:



Header Algorithms

The preceding text covers the HP-41 program memory configurations of the various calculator functions. At this point it is important to understand the specific algorithms needed to make the header of each program bar code row.

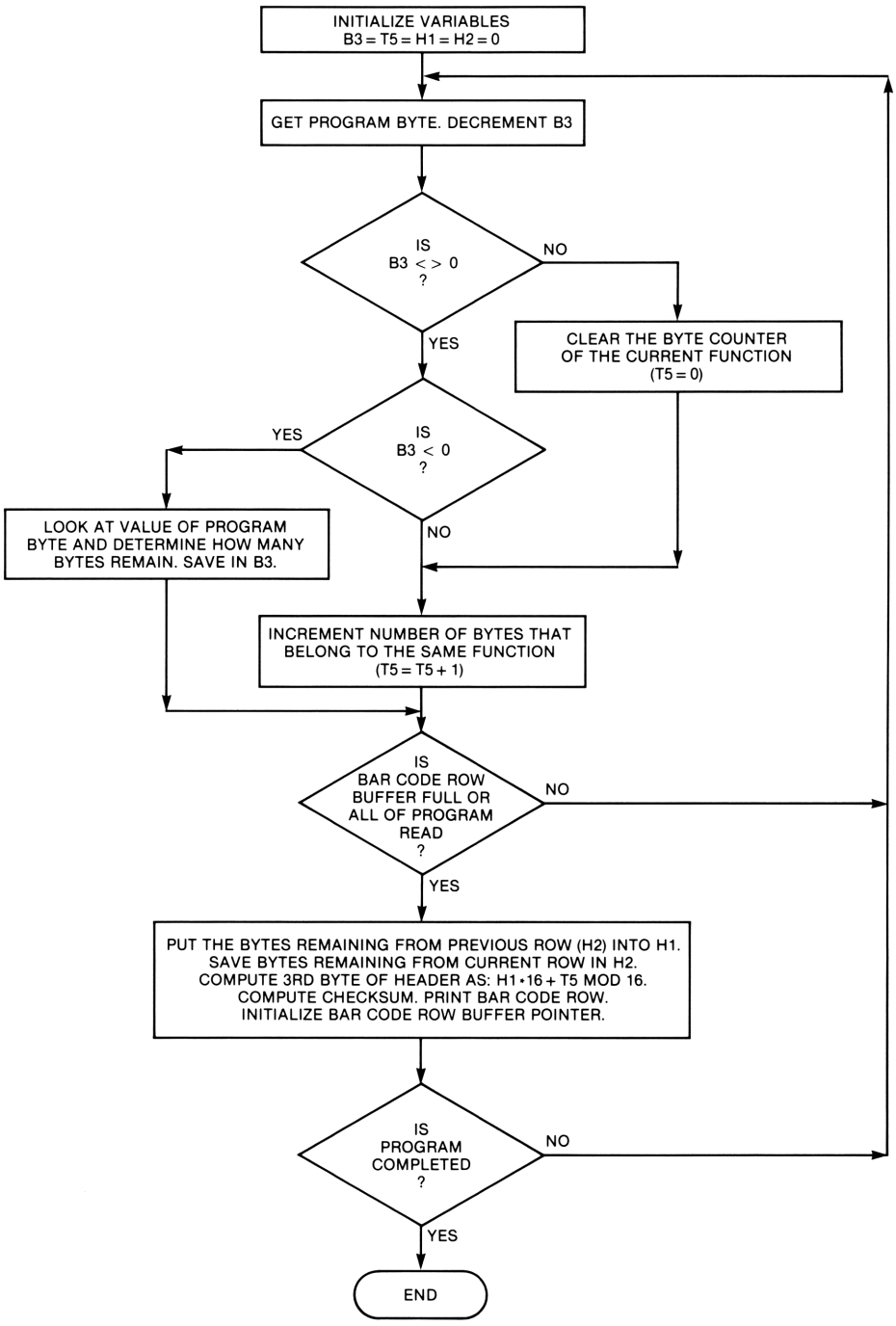
The first byte of the header is the checksum which was covered beginning on page 43. The second byte has the type field and the row sequence number in it (refer to Figure 1, page 8). The type field is set to "1" for non-private and to "2" for private bar code. In the sample program "PRGMBR" a counter is kept which labels each bar code row with a consecutive row number. Using this row number the internal sequence number is generated as follows:

Sequence Number = Row Number (Minus 1) Modulo 16.

In any Program type bar code row, the third and last byte of the header specifies the leading broken function code bytes and the trailing broken function code bytes of the row (page 7, Program Bar Code). In order to calculate these values, the following is needed:

1. One incrementing counter (T5);
2. One decrementing counter (B3);
3. One variable (H1) to hold the number of bytes left over from the previous row; and
4. One variable (H2) to hold the number of bytes left over from the current row.

These counters and variables are all initially set to zero. Here is a flow chart of this procedure.



Other Bar Code Algorithms

Direct Execution (Complete Function Code) Algorithms

The value of the bar code is derived for the most part directly from the position of the function in the HP-41 Function Table (Table III, page 45). All functions in rows 0 through 8 are one-byte functions and, with the exception of [GTO] α and [XEQ] α, require bar code rows of only three bytes in length (two bytes header and one byte function code).

Rows 0, 2 and 3 as shown in Table III are not used. These are short forms of the functions and they have analogous long forms that are to be used instead (Row 12, column 15; Row 9, column 0; and Row 9, column 1; respectively).

The digit entry functions contained in row 1, columns 0 through 13, are handled by type 6 or 9, Numeric Data bar code. Data bar code types 7, 8, 10, and 11 handle the alpha entry functions contained in Row 15.

For functions that are able to have an indirect argument, the indirection is shown by setting the most significant bit of the argument byte to one.

The numeric values for the stack arguments (X, Y, Z, T, and L) as well as *A* through *J* and *a* through *e* are found in Table IV, following:

Numeric Values for A – J, a – e, and the Stack

A	B	C	D	E	F	G	H	I	J
102	103	104	105	106	107	108	109	110	111
a	b	c	d	e	X	Y	Z	T	L
123	124	125	126	127	115	114	113	112	116

TABLE IV.

The first byte in the bar code row is always the checksum; the second byte is always the decimal value 64. Following is a breakdown of how the values for the remainder of the bytes are formed from the HP-41 Function Table (Table III) for the different functions.

The programmable function values are derived as follows:

Position in HP-41 Function Table			Contents of:			
Row	**Column**	**Function**	**Byte 3**	**Byte 4**	**• • • •**	**Byte N + 4**
1	13, 14	GTO, XEQ (Alpha)	(Row No.*16) + (Column Number)	1st ASCII Character		Nth (last) ASCII Character
4-8	0-15	Refer to Table III	(Row No.*16) + (Column Number)	N/A		N/A
9	0-15	Refer to Table III	(Row No.*16) + (Column Number)	Numeric Value of Argument		N/A
10	0-7	XROM	(Row No.*16) + [(Rom I.D. Number/4)mod 8]	(Rom I.D. mod 4) *64 + (Rom Func. No. mod 64)		N/A
10	8-13	Refer to Table III	(Row No.*16) + (Column Number)	Numeric Value of Argument		N/A
10	14	GTO/XEQ IND	(Row No.*16) + (Column Number)	Numeric Value of Argument (If GTO msb = 0, if XEO msb = 1)		N/A
12	0-13	LBL, END	(Row No.*16) + 13(LBL) or 0(END)	1st ASCII Character		Nth (last) ASCII Character
12	14, 15	Refer to Table III	(Row No.*16) + (Column Number)	Numeric Value of Argument		N/A
13,14	0-15	Refer to Table III	(Row No.*16) + 0	Numeric Value of Argument		N/A

The non-programmable function values are derived as follows:

Non-Programmable Function	Contents of: (N = The Number of ASCII Characters)					
	Byte 3	Byte Four	Byte Five	• • • •	Byte N + 3	Byte N + 4
CAT	0	Catalog Number	N/A		N/A	N/A
GTO.	1	Argument (right justified)			N/A	N/A
GTO..	1	15	255		N/A	N/A
DEL	2	Argument (right justified)			N/A	N/A
COPY	3	1st ASCII Character	2nd ASCII Character		Nth ASCII Character	N/A
CLP*	4	1st ASCII Character	2nd ASCII Character		Nth ASCII Character	N/A
R/S	5	N/A	N/A		N/A	N/A
SIZE	6	Argument (right justified)			N/A	N/A
BST	7	N/A	N/A		N/A	N/A
SST	8	N/A	N/A		N/A	N/A
ON	9	N/A	N/A		N/A	N/A
PACK	10	N/A	N/A		N/A	N/A
DELETE	11	N/A	N/A		N/A	N/A
ASN	15	A × 16 + B†	1st ASCII Character		N − 1 ASCII Character	Nth ASCII Character (N < 7)

*This three byte bar code by itself will clear the program where the PC (program counter) is located. If there is an argument it is an Alpha string.
†A = ABS(Keycode) DIV 10, B = ABS(Keycode) MOD 10. If Keycode < 0 then A = [ABS(Keycode) Div 1048] MOD 16

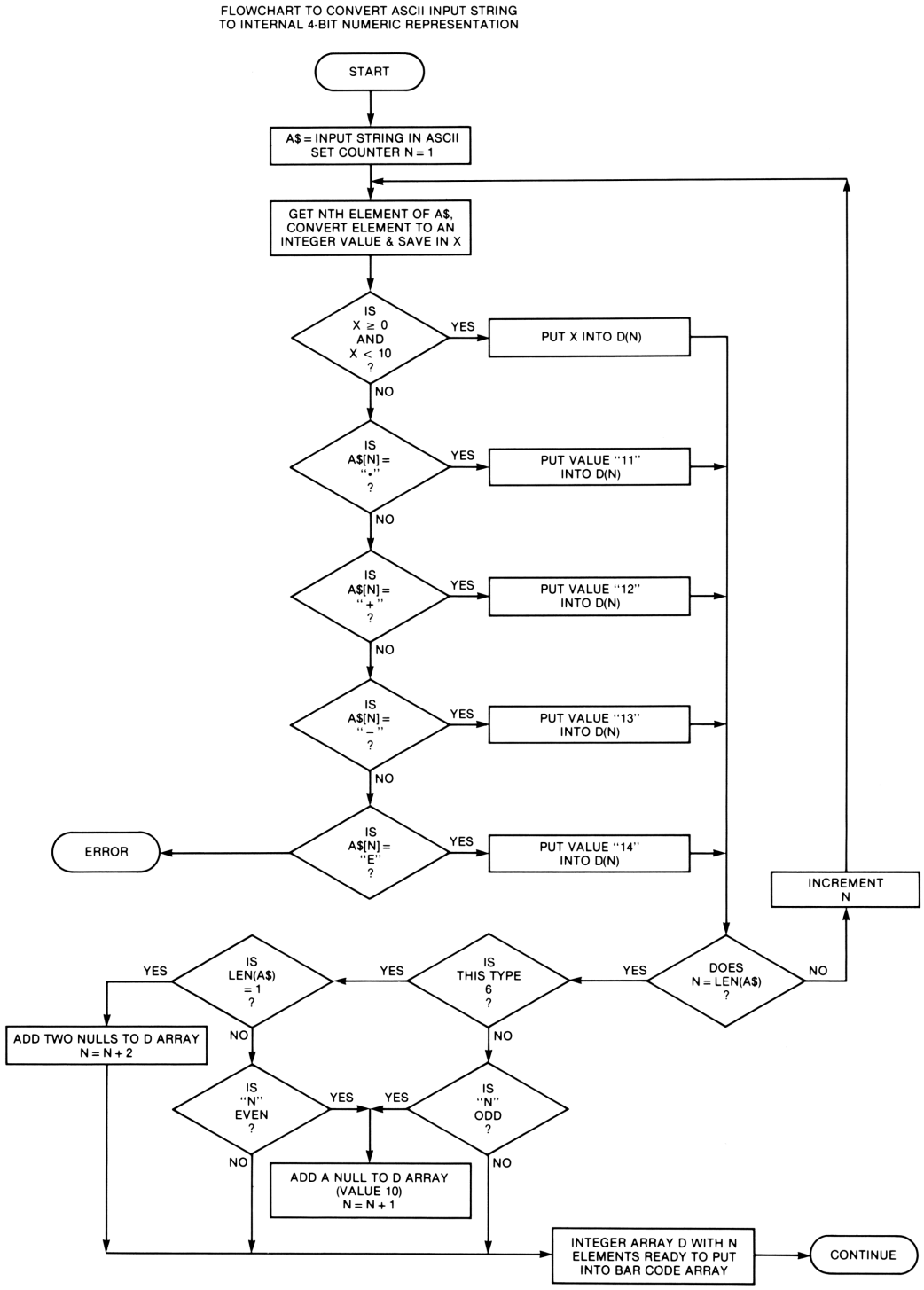
Numeric Data Functions

In order to pack as many digits in as short a bar code row as possible, the encoding scheme chosen for the digits uses only four bits per digit, as follows:

DIGIT	CODE
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
NULL	1010
"."	1011
"+"	1100
"−"	1101
"E"	1110
illegal	1111

Since we are using a four bit BCD code for these digits, it may be necessary to add a digit to make the bar code conform to the "n*8 bits" requirement (refer to Section 2). A special case is for standard numeric data (Type 6). When only one digit is input it is necessary to add two nulls so that the minimum bar code row length is three bytes.

In all other cases it is necessary to count the number of digits in the input string. If the data is standard numeric and the number of digits is even then add a null. If the data is sequenced numeric and the number of digits is odd then add a null.



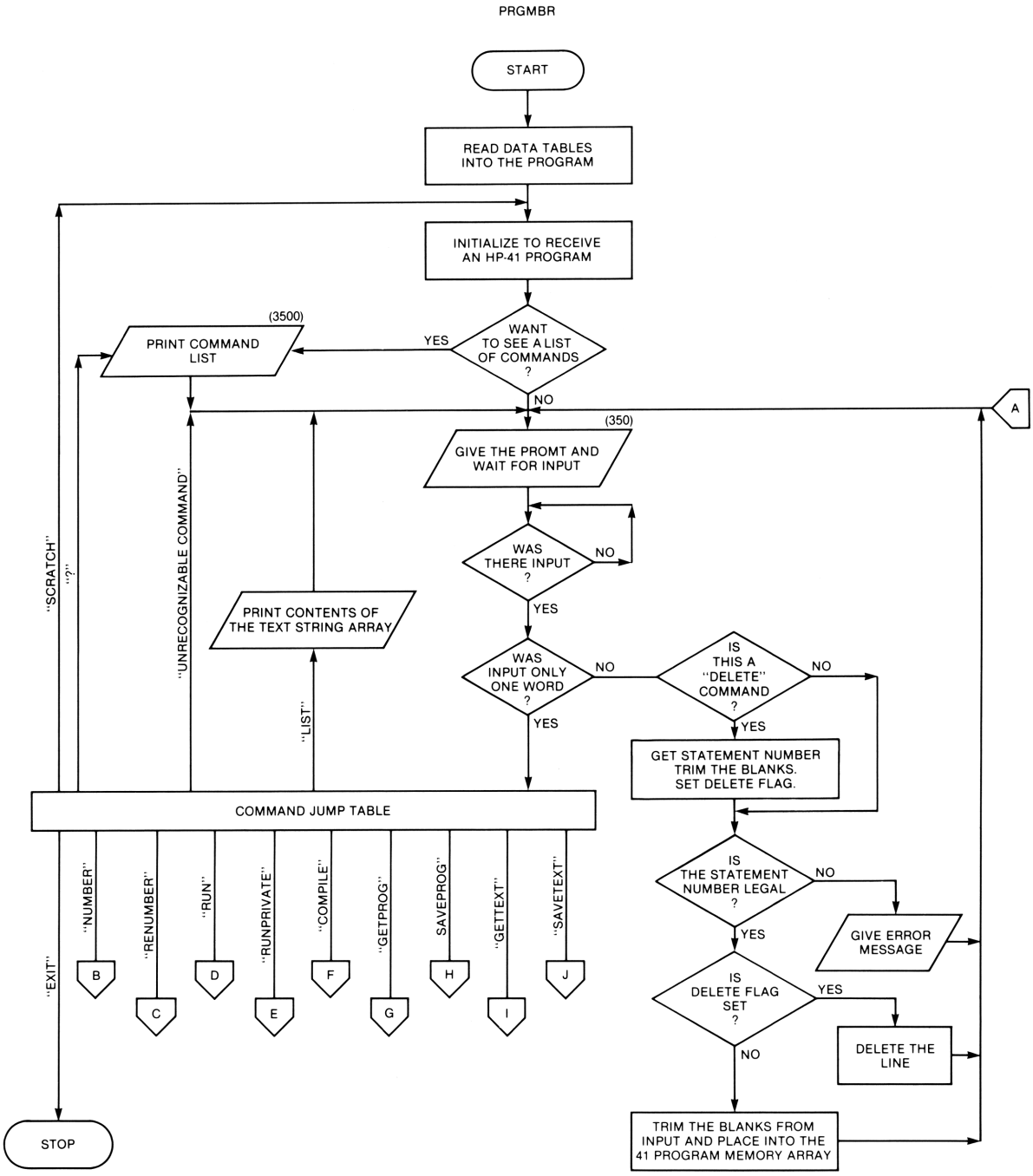
Alpha Data Encoding

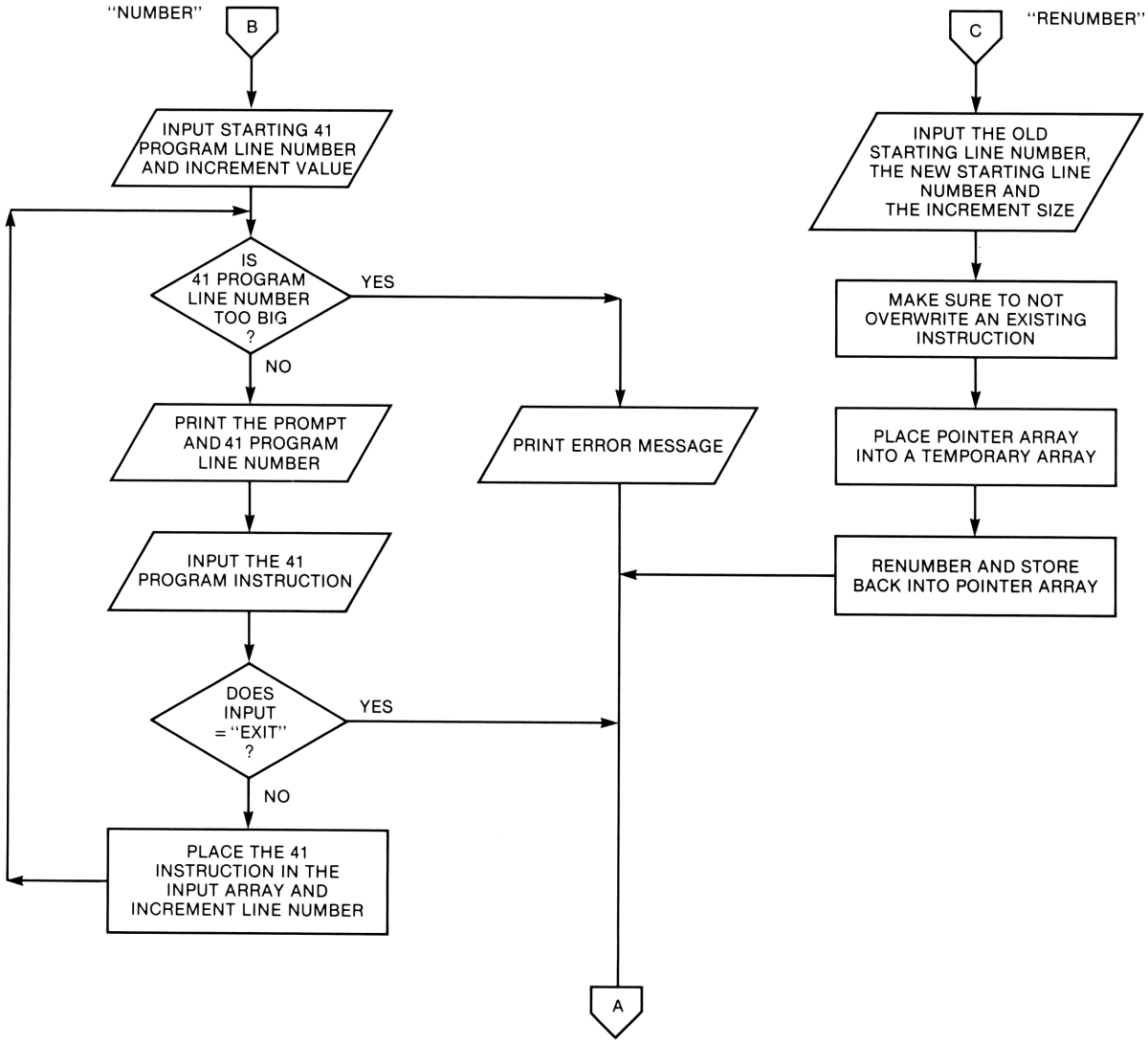
The encoding for the alpha characters is ASCII (with the few exceptions necessary to conform to the HP 82143A Printer character set). The values are listed in decimal form in Table V following:

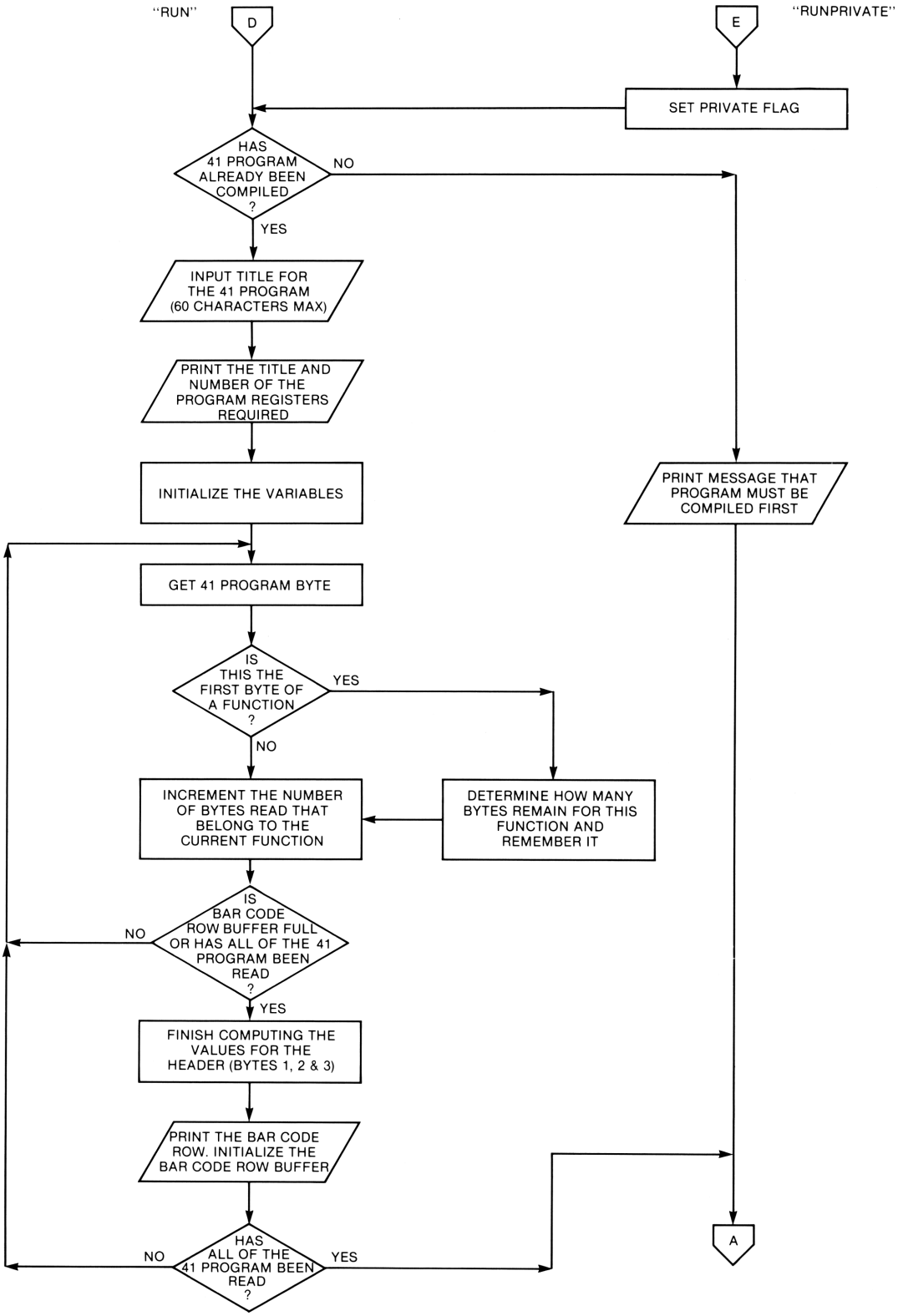
1. ×	44. ,	86. V
2. x̄	45. −	87. W
3. ←	46. .	88. X
4. α	47. /	89. Y
5. β	48. 0	90. Z
6. Γ	49. 1	91. [
7. ↓	50. 2	92. \
8. Δ	51. 3	93.]
9. σ	52. 4	94. ↑
10. ◆	53. 5	95. _
11. λ	54. 6	96. ‾
12. μ	55. 7	97. a
13. ∡	56. 8	98. b
14. τ	57. 9	99. c
15. ∉	58. :	100. d
16. θ	59. ;	101. e
17. Ω	60. <	102. f
18. δ	61. =	103. g
19. Ä	62. >	104. h
20. ä	63. ?	105. i
21. Ä	64. @	106. j
22. ä	65. A	107. k
23. Ö	66. B	108. l
24. ö	67. C	109. m
25. Ü	68. D	110. n
26. ü	69. E	111. o
27. ß	70. F	112. p
28. æ	71. G	113. q
29. ≠	72. H	114. r
30. £	73. I	115. s
31. ▒	74. J	116. t
32.	75. K	117. u
33. !	76. L	118. v
34. "	77. M	119. w
35. #	78. N	120. x
36. $	79. O	121. y
37. %	80. P	122. z
38. &	81. Q	123. π
39. '	82. R	124. |
40. (	83. S	125. →
41.)	84. T	126. Σ
42. *	85. U	127. ⊢
43. +		

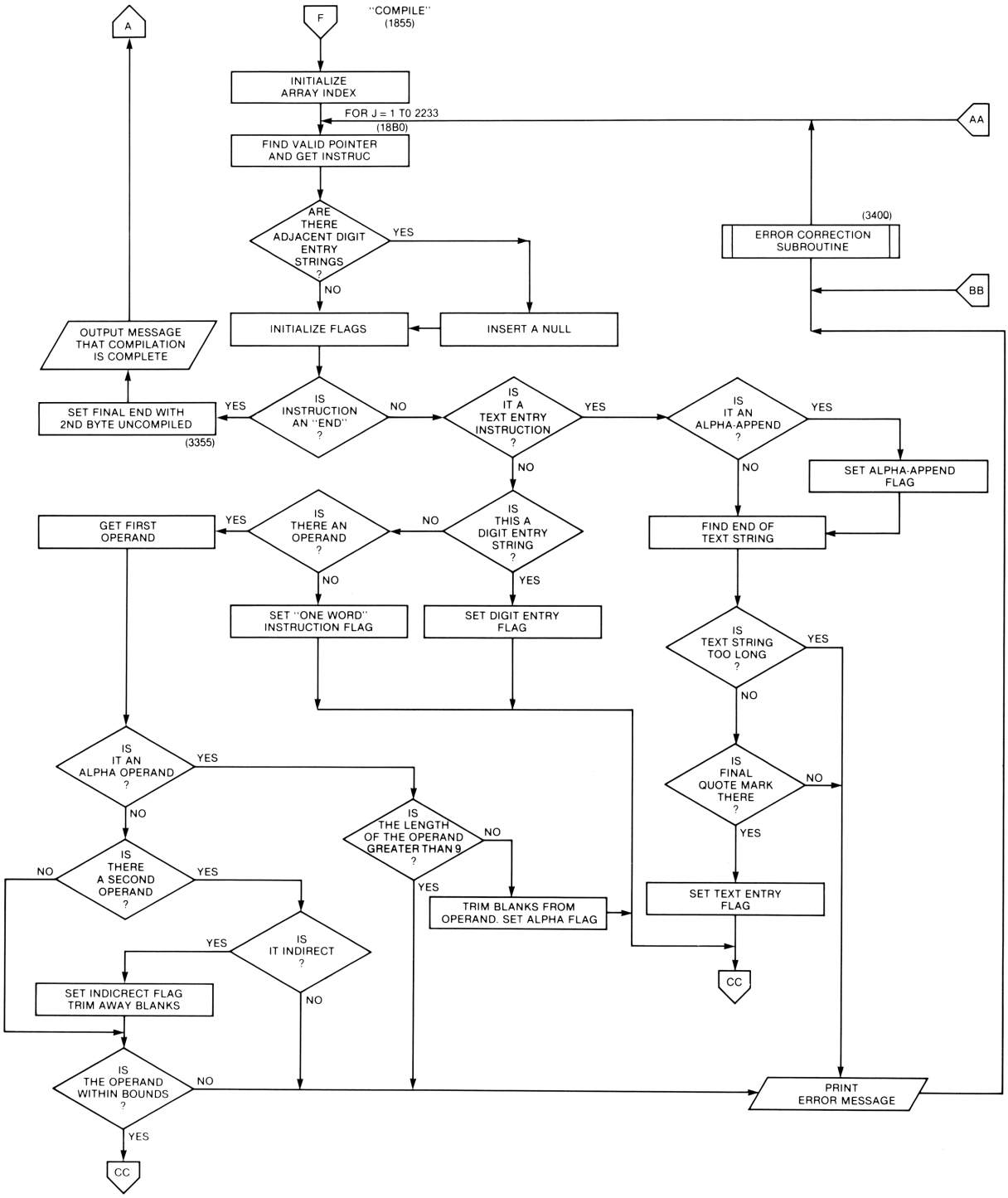
Table V. HP 82143A Printer Character Set

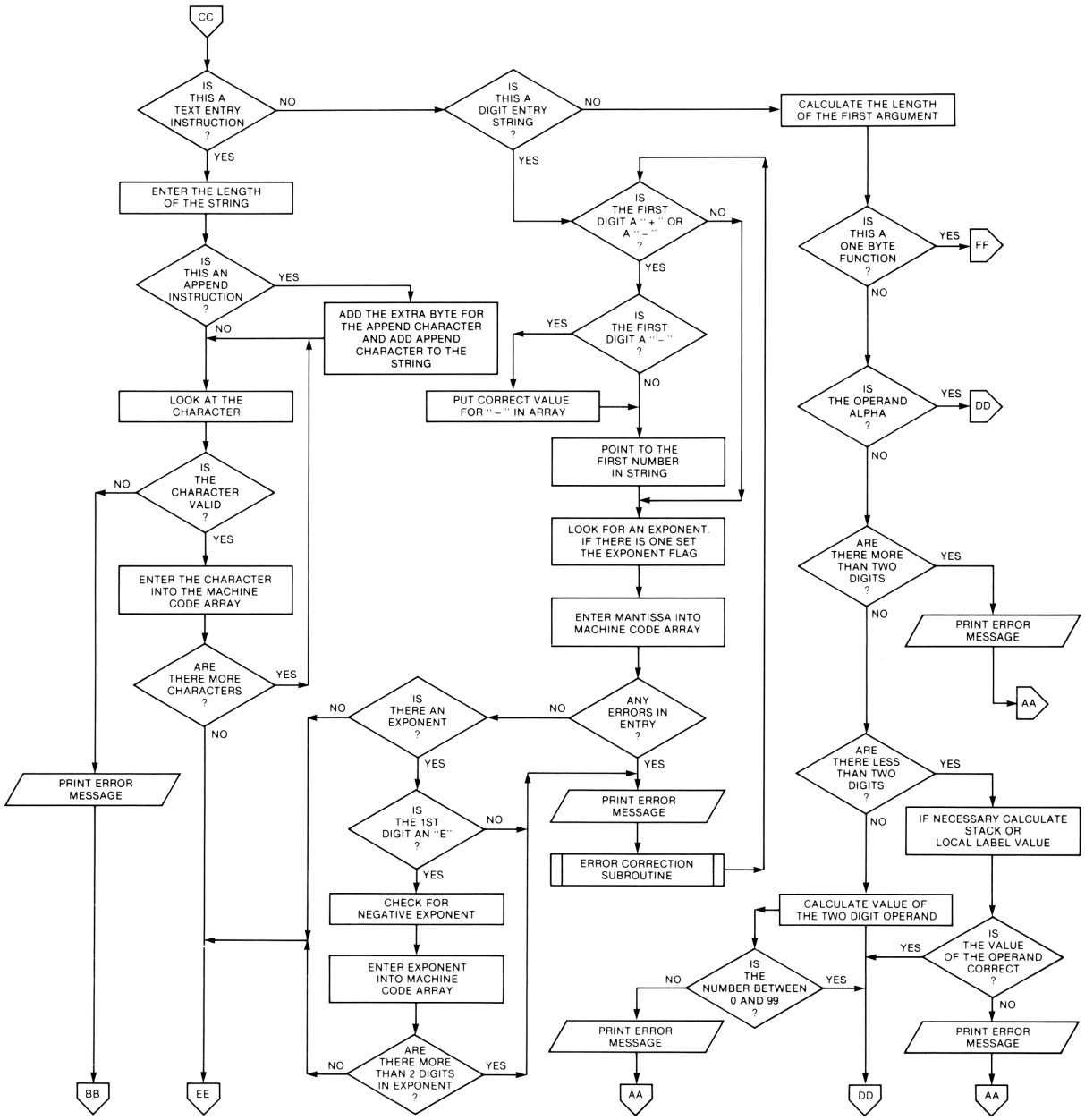
Appendix A
Flowcharts

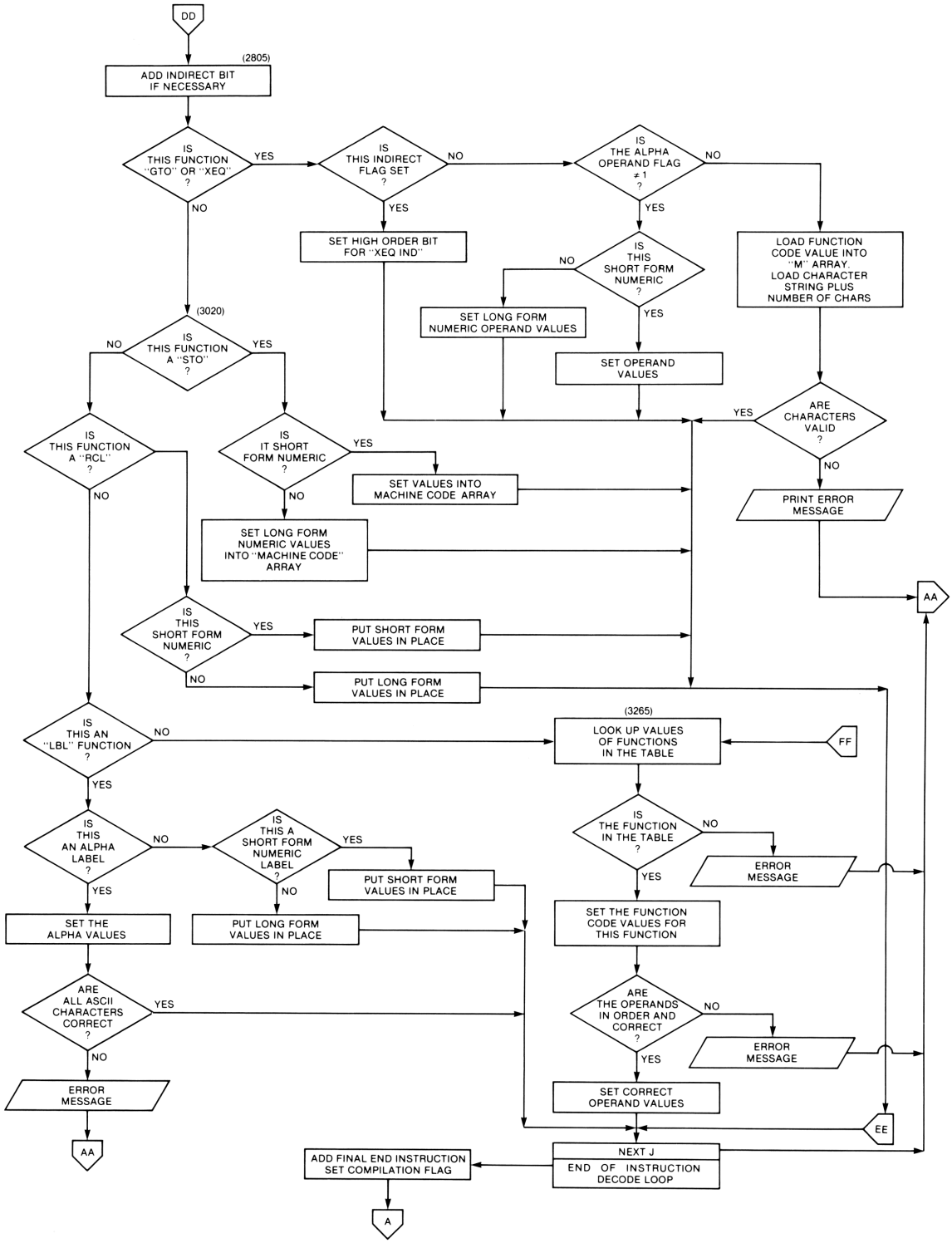

PRGMBR
START
READ DATA TABLES INTO THE PROGRAM
INITIALIZE TO RECEIVE AN HP-41 PROGRAM
(3500)
PRINT COMMAND LIST
YES
WANT TO SEE A LIST OF COMMANDS ?
NO
A
(350)
GIVE THE PROMT AND WAIT FOR INPUT
WAS THERE INPUT ?
NO
YES
"SCRATCH"
"?"
"UNRECOGNIZABLE COMMAND"
PRINT CONTENTS OF THE TEXT STRING ARRAY
WAS INPUT ONLY ONE WORD ?
NO
YES
"LIST"
IS THIS A "DELETE" COMMAND ?
NO
YES
GET STATEMENT NUMBER TRIM THE BLANKS. SET DELETE FLAG.
COMMAND JUMP TABLE
"EXIT"
"NUMBER"
"RENUMBER"
"RUN"
"RUNPRIVATE"
"COMPILE"
"GETPROG"
SAVEPROG"
"GETTEXT"
"SAVETEXT"
B
C
D
E
F
G
H
I
J
IS THE STATEMENT NUMBER LEGAL ?
NO
YES
GIVE ERROR MESSAGE
IS DELETE FLAG SET ?
YES
NO
DELETE THE LINE
TRIM THE BLANKS FROM INPUT AND PLACE INTO THE 41 PROGRAM MEMORY ARRAY
STOP

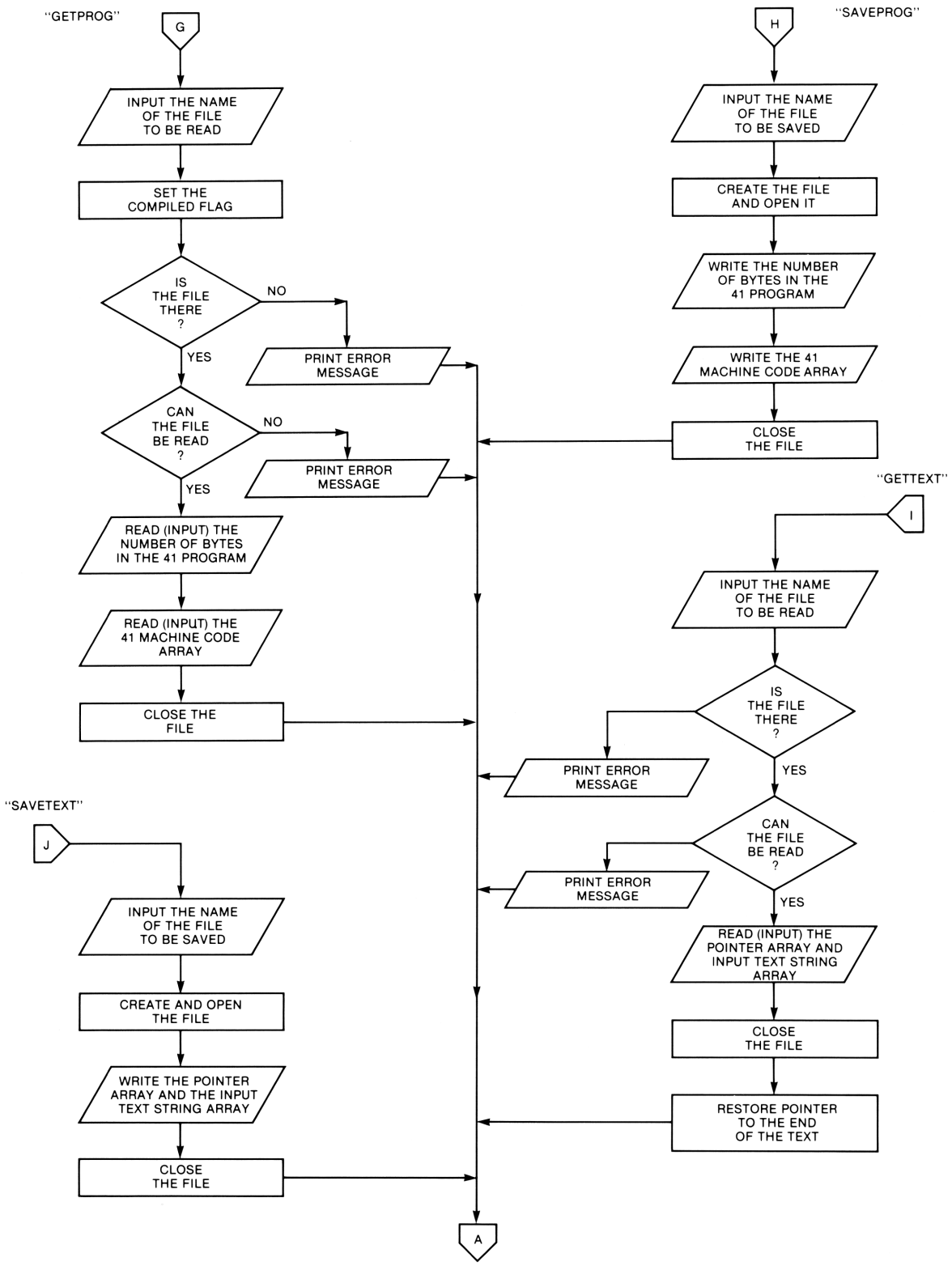

"NUMBER"
B
INPUT STARTING 41 PROGRAM LINE NUMBER AND INCREMENT VALUE
IS 41 PROGRAM LINE NUMBER TOO BIG ?
YES
NO
PRINT THE PROMPT AND 41 PROGRAM LINE NUMBER
INPUT THE 41 PROGRAM INSTRUCTION
DOES INPUT = "EXIT" ?
YES
NO
PLACE THE 41 INSTRUCTION IN THE INPUT ARRAY AND INCREMENT LINE NUMBER
PRINT ERROR MESSAGE
A
C
"RENUMBER"
INPUT THE OLD STARTING LINE NUMBER, THE NEW STARTING LINE NUMBER AND THE INCREMENT SIZE
MAKE SURE TO NOT OVERWRITE AN EXISTING INSTRUCTION
PLACE POINTER ARRAY INTO A TEMPORARY ARRAY
RENUMBER AND STORE BACK INTO POINTER ARRAY

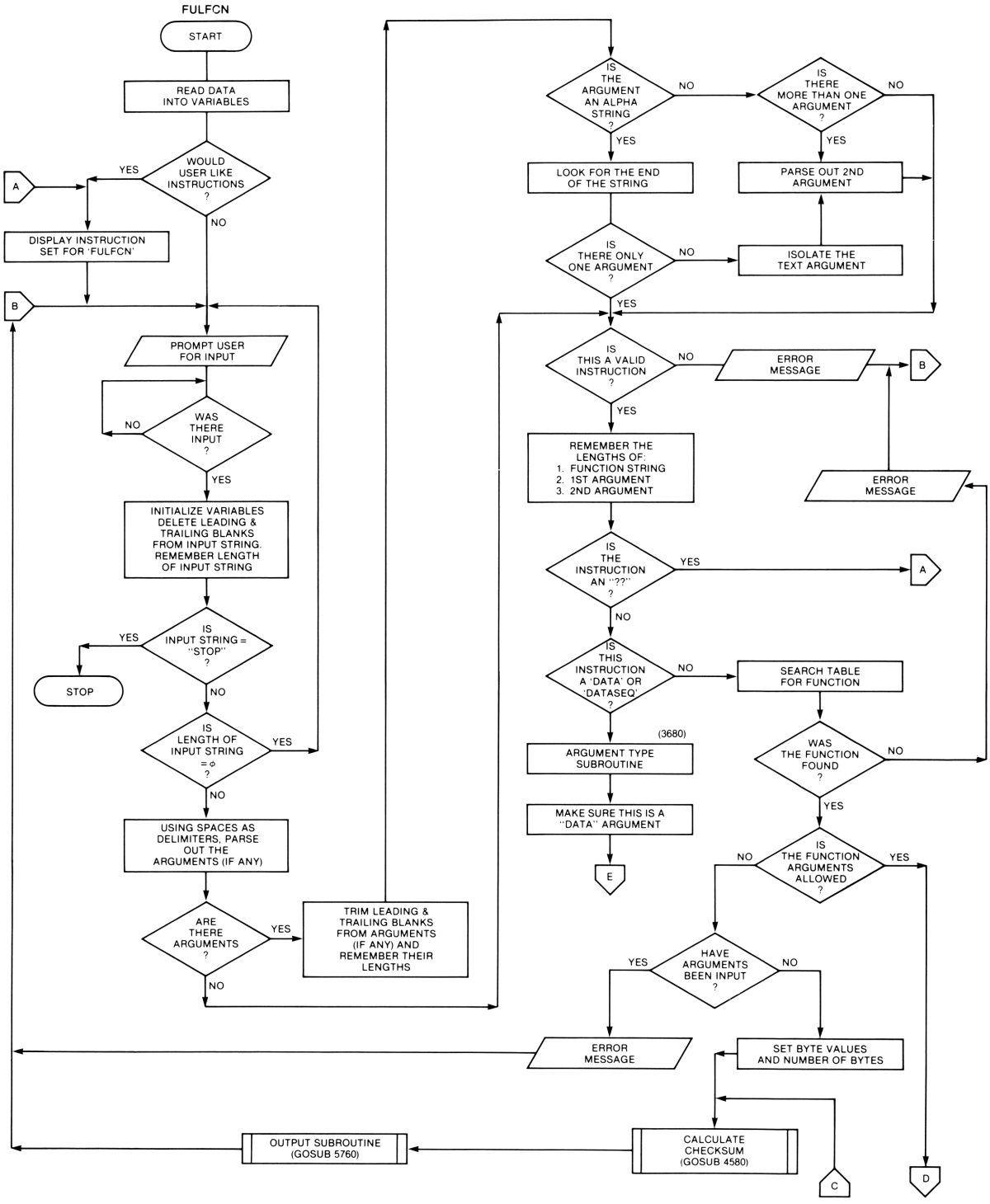

"RUN"
D
E
"RUNPRIVATE"
SET PRIVATE FLAG
HAS 41 PROGRAM ALREADY BEEN COMPILED ?
NO
YES
INPUT TITLE FOR THE 41 PROGRAM (60 CHARACTERS MAX)
PRINT THE TITLE AND NUMBER OF THE PROGRAM REGISTERS REQUIRED
INITIALIZE THE VARIABLES
PRINT MESSAGE THAT PROGRAM MUST BE COMPILED FIRST
GET 41 PROGRAM BYTE
IS THIS THE FIRST BYTE OF A FUNCTION ?
YES
NO
INCREMENT THE NUMBER OF BYTES READ THAT BELONG TO THE CURRENT FUNCTION
DETERMINE HOW MANY BYTES REMAIN FOR THIS FUNCTION AND REMEMBER IT
IS BAR CODE ROW BUFFER FULL OR HAS ALL OF THE 41 PROGRAM BEEN READ ?
NO
YES
FINISH COMPUTING THE VALUES FOR THE HEADER (BYTES 1, 2 & 3)
PRINT THE BAR CODE ROW. INITIALIZE THE BAR CODE ROW BUFFER
HAS ALL OF THE 41 PROGRAM BEEN READ ?
NO
YES
A

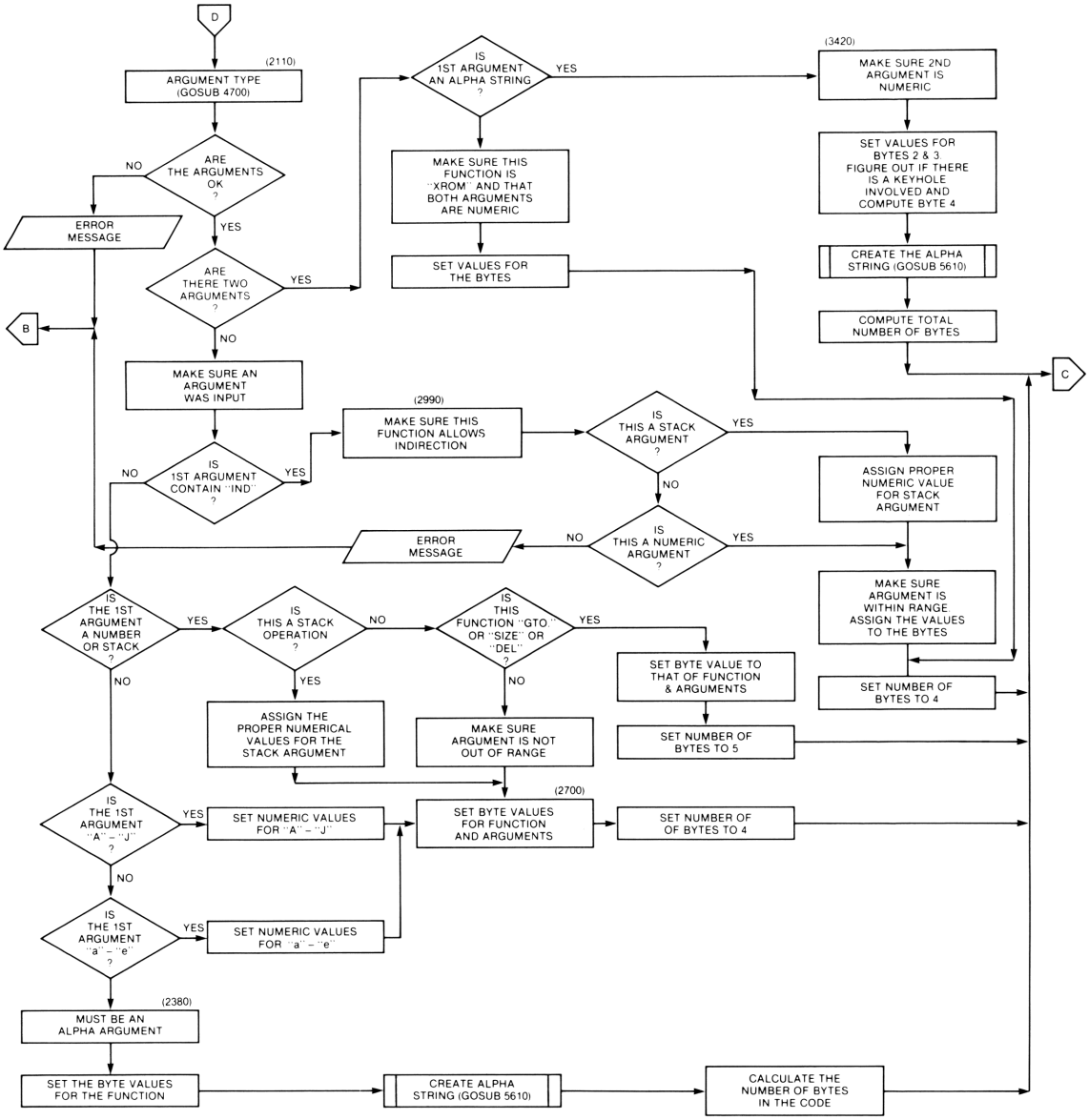

A
F
"COMPILE"
(1855)
INITIALIZE
ARRAY INDEX
FOR J = 1 TO 2233
(18B0)
AA
FIND VALID POINTER
AND GET INSTRUC
ARE
THERE
ADJACENT DIGIT
ENTRY
STRINGS
?
YES
NO
INSERT A NULL
INITIALIZE FLAGS
(3400)
ERROR CORRECTION
SUBROUTINE
BB
OUTPUT MESSAGE
THAT COMPILATION
IS COMPLETE
SET FINAL END WITH
2ND BYTE UNCOMPILED
(3355)
IS
INSTRUCTION
AN "END"
?
IS
IT A
TEXT ENTRY
INSTRUCTION
?
IS
IT AN
ALPHA-APPEND
?
SET ALPHA-APPEND
FLAG
GET FIRST
OPERAND
IS
THERE AN
OPERAND
?
IS
THIS A
DIGIT ENTRY
STRING
?
FIND END OF
TEXT STRING
SET "ONE WORD"
INSTRUCTION FLAG
SET DIGIT ENTRY
FLAG
IS
TEXT STRING
TOO LONG
?
IS
IT AN
ALPHA OPERAND
?
IS
FINAL
QUOTE MARK
THERE
?
IS
THE LENGTH
OF THE OPERAND
GREATER THAN 9
?
IS
THERE
A SECOND
OPERAND
?
TRIM BLANKS FROM
OPERAND. SET ALPHA FLAG
SET TEXT ENTRY
FLAG
IS
IT INDIRECT
?
CC
SET INDICRECT FLAG
TRIM AWAY BLANKS
IS
THE OPERAND
WITHIN BOUNDS
?
PRINT
ERROR MESSAGE
CC

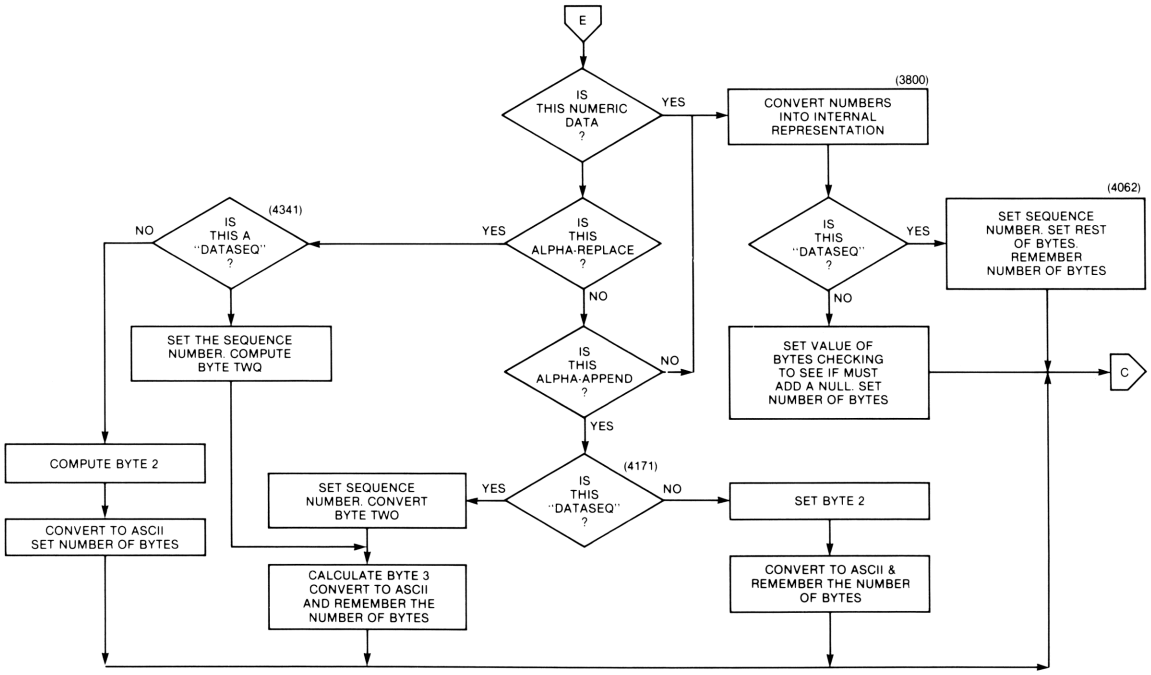

CC
IS THIS A TEXT ENTRY INSTRUCTION ?
NO
YES
ENTER THE LENGTH OF THE STRING
IS THIS AN APPEND INSTRUCTION ?
YES
NO
ADD THE EXTRA BYTE FOR THE APPEND CHARACTER AND ADD APPEND CHARACTER TO THE STRING
LOOK AT THE CHARACTER
IS THE CHARACTER VALID ?
NO
YES
ENTER THE CHARACTER INTO THE MACHINE CODE ARRAY
ARE THERE MORE CHARACTERS ?
YES
NO
PRINT ERROR MESSAGE
BB
EE
IS THIS A DIGIT ENTRY STRING ?
NO
YES
IS THE FIRST DIGIT A "+" OR A "–" ?
NO
YES
IS THE FIRST DIGIT A "–" ?
YES
NO
PUT CORRECT VALUE FOR "–" IN ARRAY
POINT TO THE FIRST NUMBER IN STRING
LOOK FOR AN EXPONENT. IF THERE IS ONE SET THE EXPONENT FLAG
ENTER MANTISSA INTO MACHINE CODE ARRAY
ANY ERRORS IN ENTRY ?
NO
YES
PRINT ERROR MESSAGE
ERROR CORRECTION SUBROUTINE
IS THERE AN EXPONENT ?
NO
YES
IS THE 1ST DIGIT AN "E" ?
NO
YES
CHECK FOR NEGATIVE EXPONENT
ENTER EXPONENT INTO MACHINE CODE ARRAY
ARE THERE MORE THAN 2 DIGITS IN EXPONENT ?
NO
YES
CALCULATE THE LENGTH OF THE FIRST ARGUMENT
IS THIS A ONE BYTE FUNCTION ?
YES
FF
NO
IS THE OPERAND ALPHA ?
YES
DD
NO
ARE THERE MORE THAN TWO DIGITS ?
YES
PRINT ERROR MESSAGE
AA
NO
ARE THERE LESS THAN TWO DIGITS ?
YES
NO
IF NECESSARY CALCULATE STACK OR LOCAL LABEL VALUE
CALCULATE VALUE OF THE TWO DIGIT OPERAND
IS THE VALUE OF THE OPERAND CORRECT ?
YES
NO
PRINT ERROR MESSAGE
AA
IS THE NUMBER BETWEEN 0 AND 99 ?
NO
YES
PRINT ERROR MESSAGE
AA
DD


DD
(2805)
ADD INDIRECT BIT IF NECESSARY
IS THIS FUNCTION "GTO" OR "XEQ" ?
YES
NO
IS THIS INDIRECT FLAG SET ?
NO
YES
SET HIGH ORDER BIT FOR "XEQ IND"
IS THE ALPHA OPERAND FLAG ≠ 1 ?
NO
YES
IS THIS SHORT FORM NUMERIC ?
NO
YES
SET LONG FORM NUMERIC OPERAND VALUES
SET OPERAND VALUES
LOAD FUNCTION CODE VALUE INTO "M" ARRAY. LOAD CHARACTER STRING PLUS NUMBER OF CHARS
ARE CHARACTERS VALID ?
YES
NO
PRINT ERROR MESSAGE
(3020)
IS THIS FUNCTION A "STO" ?
NO
YES
IS THIS FUNCTION A "RCL" ?
NO
IS IT SHORT FORM NUMERIC ?
YES
NO
SET VALUES INTO MACHINE CODE ARRAY
SET LONG FORM NUMERIC VALUES INTO "MACHINE CODE" ARRAY
IS THIS SHORT FORM NUMERIC ?
YES
NO
PUT SHORT FORM VALUES IN PLACE
PUT LONG FORM VALUES IN PLACE
AA
IS THIS AN "LBL" FUNCTION ?
NO
YES
(3265)
LOOK UP VALUES OF FUNCTIONS IN THE TABLE
FF
IS THIS AN ALPHA LABEL ?
NO
YES
IS THIS A SHORT FORM NUMERIC LABEL ?
YES
NO
PUT SHORT FORM VALUES IN PLACE
PUT LONG FORM VALUES IN PLACE
SET THE ALPHA VALUES
ARE ALL ASCII CHARACTERS CORRECT ?
YES
NO
ERROR MESSAGE
AA
IS THE FUNCTION IN THE TABLE ?
NO
YES
ERROR MESSAGE
SET THE FUNCTION CODE VALUES FOR THIS FUNCTION
ARE THE OPERANDS IN ORDER AND CORRECT ?
NO
YES
ERROR MESSAGE
SET CORRECT OPERAND VALUES
EE
NEXT J
END OF INSTRUCTION DECODE LOOP
ADD FINAL END INSTRUCTION SET COMPILATION FLAG
A


"GETPROG"
G
INPUT THE NAME OF THE FILE TO BE READ
SET THE COMPILED FLAG
IS THE FILE THERE ?
NO
YES
PRINT ERROR MESSAGE
CAN THE FILE BE READ ?
NO
YES
PRINT ERROR MESSAGE
READ (INPUT) THE NUMBER OF BYTES IN THE 41 PROGRAM
READ (INPUT) THE 41 MACHINE CODE ARRAY
CLOSE THE FILE
"SAVEPROG"
H
INPUT THE NAME OF THE FILE TO BE SAVED
CREATE THE FILE AND OPEN IT
WRITE THE NUMBER OF BYTES IN THE 41 PROGRAM
WRITE THE 41 MACHINE CODE ARRAY
CLOSE THE FILE
"GETTEXT"
I
INPUT THE NAME OF THE FILE TO BE READ
IS THE FILE THERE ?
YES
PRINT ERROR MESSAGE
CAN THE FILE BE READ ?
YES
PRINT ERROR MESSAGE
READ (INPUT) THE POINTER ARRAY AND INPUT TEXT STRING ARRAY
CLOSE THE FILE
RESTORE POINTER TO THE END OF THE TEXT
"SAVETEXT"
J
INPUT THE NAME OF THE FILE TO BE SAVED
CREATE AND OPEN THE FILE
WRITE THE POINTER ARRAY AND THE INPUT TEXT STRING ARRAY
CLOSE THE FILE
A


FULFCN
START
READ DATA INTO VARIABLES
WOULD USER LIKE INSTRUCTIONS ?
YES
NO
A
DISPLAY INSTRUCTION SET FOR 'FULFCN'
B
PROMPT USER FOR INPUT
WAS THERE INPUT ?
NO
YES
INITIALIZE VARIABLES DELETE LEADING & TRAILING BLANKS FROM INPUT STRING. REMEMBER LENGTH OF INPUT STRING
IS INPUT STRING = "STOP" ?
YES
STOP
NO
IS LENGTH OF INPUT STRING = 0 ?
YES
NO
USING SPACES AS DELIMITERS, PARSE OUT THE ARGUMENTS (IF ANY)
ARE THERE ARGUMENTS ?
YES
TRIM LEADING & TRAILING BLANKS FROM ARGUMENTS (IF ANY) AND REMEMBER THEIR LENGTHS
NO
IS THE ARGUMENT AN ALPHA STRING ?
NO
YES
LOOK FOR THE END OF THE STRING
IS THERE ONLY ONE ARGUMENT ?
NO
YES
ISOLATE THE TEXT ARGUMENT
IS THERE MORE THAN ONE ARGUMENT ?
NO
YES
PARSE OUT 2ND ARGUMENT
IS THIS A VALID INSTRUCTION ?
NO
ERROR MESSAGE
B
YES
REMEMBER THE LENGTHS OF:
1. FUNCTION STRING
2. 1ST ARGUMENT
3. 2ND ARGUMENT
ERROR MESSAGE
IS THE INSTRUCTION AN "??" ?
YES
A
NO
IS THIS INSTRUCTION A 'DATA' OR 'DATASEQ' ?
NO
SEARCH TABLE FOR FUNCTION
(3680)
ARGUMENT TYPE SUBROUTINE
MAKE SURE THIS IS A "DATA" ARGUMENT
E
WAS THE FUNCTION FOUND ?
NO
YES
IS THE FUNCTION ARGUMENTS ALLOWED ?
NO
YES
HAVE ARGUMENTS BEEN INPUT ?
YES
NO
ERROR MESSAGE
SET BYTE VALUES AND NUMBER OF BYTES
CALCULATE CHECKSUM (GOSUB 4580)
OUTPUT SUBROUTINE (GOSUB 5760)
C
D


D
(2110)
ARGUMENT TYPE (GOSUB 4700)
ARE THE ARGUMENTS OK ?
NO
YES
ERROR MESSAGE
ARE THERE TWO ARGUMENTS ?
YES
NO
B
MAKE SURE AN ARGUMENT WAS INPUT
IS 1ST ARGUMENT CONTAIN "IND" ?
NO
YES
(2990)
MAKE SURE THIS FUNCTION ALLOWS INDIRECTION
IS 1ST ARGUMENT AN ALPHA STRING ?
YES
MAKE SURE THIS FUNCTION IS "XROM" AND THAT BOTH ARGUMENTS ARE NUMERIC
SET VALUES FOR THE BYTES
(3420)
MAKE SURE 2ND ARGUMENT IS NUMERIC
SET VALUES FOR BYTES 2 & 3 FIGURE OUT IF THERE IS A KEYHOLE INVOLVED AND COMPUTE BYTE 4
CREATE THE ALPHA STRING (GOSUB 5610)
COMPUTE TOTAL NUMBER OF BYTES
C
IS THIS A STACK ARGUMENT ?
YES
NO
ASSIGN PROPER NUMERIC VALUE FOR STACK ARGUMENT
IS THIS A NUMERIC ARGUMENT ?
NO
YES
ERROR MESSAGE
MAKE SURE ARGUMENT IS WITHIN RANGE ASSIGN THE VALUES TO THE BYTES
IS THE 1ST ARGUMENT A NUMBER OR STACK ?
YES
NO
IS THIS A STACK OPERATION ?
NO
YES
IS THIS FUNCTION "GTO" OR "SIZE" OR "DEL" ?
YES
NO
SET BYTE VALUE TO THAT OF FUNCTION & ARGUMENTS
SET NUMBER OF BYTES TO 5
SET NUMBER OF BYTES TO 4
ASSIGN THE PROPER NUMERICAL VALUES FOR THE STACK ARGUMENT
MAKE SURE ARGUMENT IS NOT OUT OF RANGE
IS THE 1ST ARGUMENT "A" – "J" ?
YES
SET NUMERIC VALUES FOR "A" – "J"
(2700)
SET BYTE VALUES FOR FUNCTION AND ARGUMENTS
SET NUMBER OF OF BYTES TO 4
NO
IS THE 1ST ARGUMENT "a" – "e" ?
YES
SET NUMERIC VALUES FOR "a" – "e"
(2380)
MUST BE AN ALPHA ARGUMENT
SET THE BYTE VALUES FOR THE FUNCTION
CREATE ALPHA STRING (GOSUB 5610)
CALCULATE THE NUMBER OF BYTES IN THE CODE


E
IS THIS NUMERIC DATA ?
YES
(3800)
CONVERT NUMBERS INTO INTERNAL REPRESENTATION
(4062)
SET SEQUENCE NUMBER. SET REST OF BYTES. REMEMBER NUMBER OF BYTES
IS THIS "DATASEQ" ?
YES
NO
IS THIS ALPHA-REPLACE ?
YES
(4341)
IS THIS A "DATASEQ" ?
NO
SET THE SEQUENCE NUMBER. COMPUTE BYTE TWQ
NO
IS THIS ALPHA-APPEND ?
NO
SET VALUE OF BYTES CHECKING TO SEE IF MUST ADD A NULL. SET NUMBER OF BYTES
C
YES
(4171)
IS THIS "DATASEQ" ?
YES
NO
SET SEQUENCE NUMBER. CONVERT BYTE TWO
SET BYTE 2
COMPUTE BYTE 2
CONVERT TO ASCII SET NUMBER OF BYTES
CALCULATE BYTE 3 CONVERT TO ASCII AND REMEMBER THE NUMBER OF BYTES
CONVERT TO ASCII & REMEMBER THE NUMBER OF BYTES

Appendix B
HP-85A Adaptations of PRGMBR and FULFCN

The adaptation of PRGMBR required a modified version of the HP 9845A program in order to run on the minimum HP 85A configuration: HP 85A Personal Computer; 16K RAM Plugin; I/O ROM or Printer/Plotter ROM; and, RS232C or HP-IB Interface.

PRGMBR was segmented into three separate programs according to logical operations. The first segment, BEGIN, initializes all the variables and loads the necessary table data. The second segment, EDITR, is the interactive section where the HP-41 program text is input. This section also allows the user to save, retrieve, and edit the text. Giving the command COMPILE during execution of this segment loads the third segment CMPILE into the HP 85A.

CMPILE not only compiles the HP-41 program but also prints the bar code. As soon as the HP-41 program has been completely printed out in bar code this segment loads the EDITR segment back into the HP 85A ready for the next program.

This particular adaptation does not include the ability to save and retrieve the compiled HP-41 program code.

The source files for these HP 85A programs are available on cassette tape or floppy disc from:

Series 80 Users' Library
1000 N.E. Circle Blvd.
Corvallis, Oregon 97330

85A Adaptation of PRGMBR

```
 20 ! HP41C COMPILER
 30 ! & BAR CODE GEN PRGM
 40 ! FIRST OF THREE SEGMENTS
 50 ! THAT ARE CHAINED TOGETHER
 60 ! THIS IS AN ADAPTATION OF
 70 ! THE 9845A COMPILER PROGRAM
 80 ! THIS SEGMENT IS STORED AS
 90 ! FILE "BEGIN"
100 !
110 OPTION BASE 1
120 INTEGER I,J
130 COM INTEGER C2(60),P(2233),F
    6,P5,I1(103)
140 COM T$[60],T1$[30],T2$[30],A
    $[1500],S1$[27],I$[624],C$[6
    0]
150 T$,T1$,T2$=""
160 FOR I=1 TO 26 ! READ LOCAL L
    ABELS & STACK REGISTER MNEMO
    NICS INTO S1$
170 READ S1$[I,I]
180 NEXT I
190 FOR I=1 TO 103 ! READ SORTED
     INST MNEMONICS INTO I$, INS
    TRUC.
200 J=1+6*(I-1)
210 READ I$[J],I1(I) ! VALUES IN
    TO I1 FOR TABLE DRIVER
220 NEXT I
230 FOR I=1 TO 60 ! READ IN VALI
    D CHAR TABLES FOR CHAR CHECK
240 READ C$[I],C2(I) ! CHARS IN
    C$; CHAR CODE IN C2
250 NEXT I
260 DISP "1ST SEG DONE"
270 ! PUT CHAINING STATEMENTS
280 ! HERE TO PROGRAM "EDITR"
290 CHAIN "EDITR"
300 !
310 END
320 ! *DATA*
330 ! *LOCAL LABEL & STACK REG C
    HARS**
340 DATA A,B,C,D,E,F,G,H,I,J,T,Z
    ,Y,X,L
350 DATA " "," "," "," "," "," "
    ,a,b,c,d,e
360 ! *INSTN MNEMONICS & NUMERIC
     VALUES**
370 DATA %,76,%CH,77,&+,71,&-,72
    ,&REG,153
380 DATA *,66,"+",64,"-",65,/,67
    ,"1/X",96,"10^X",87,ABS,97
390 DATA ACOS,93,ADV,143,AOFF,13
    9,AON,140,ARCL,155,ASHF,136,
    ASIN,92
400 DATA ASTO,154,ATAN,94,AVIEW,
    126,BEEP,134,CF,169,CHS,84,C
    L&,112
410 DATA CLA,135,CLD,127,CLRG,13
    8,CLST,115,CLX,119,COS,90
420 DATA D-R,106,DEC,95,DEG,128,
    DSE,151,ENG,158,ENTER^,131,E
    ^X,85
430 DATA E^X-1,88,FACT,98,FC?,17
    3,FC?C,171,FIX,156,FRC,105
440 DATA FS?,172,FS?C,170,GRAD,1
    30,HMS,108,HMS+,73,HMS-,74,H
    R,109
450 DATA INT,104,ISG,150,LASTX,1
    18,LN,80,LN1+X,101,LOG,86
460 DATA MEAN,124,MOD,75,OCT,111
    ,OFF,141,P-R,78,PI,114
470 DATA PROMPT,142,PSE,137,R-D,
    107,R-P,79,RAD,129,RCL,144,R
    DN,117
480 DATA RND,110,RTN,133,R^,116,
    SCI,157,SDEV,125,SF,168
490 DATA SIGN,122,SIN,89,SQRT,82
    ,ST*,148
500 DATA ST+,146,ST-,147,ST/,149
    ,STO,145,STOP,132,TAN,91,TON
    E,159
510 DATA VIEW,152,X#0?,99,X#Y?,1
    21,X<0?,102,X<=0?,123,X<=Y?,
    70
520 DATA X<>,206,X<>Y,113,X<Y?,6
    8,X=0?,103,X=Y?,120,X>0?,100
530 DATA X>Y?,69,X^2,81,Y^X,83
540 ! *VALID 41C CHARS & CHAR CO
    DE**
550 DATA " ",32,#,29,$,36,%,37,&
    ,126,*,42,"+",43,",",44,"-",
    45,.,46,/,47
560 DATA "0",48,"1",49,"2",50,"3
    ",51,"4",52,"5",53,"6",54,"7
    ",55,"8",56
570 DATA "9",57,:,58,";",59,<,60
    ,=,61,>,62,?,63,@,13,A,65,B,
    66,C,67,D,68,E,69
580 DATA F,70,G,71,H,72,I,73,J,7
    4,K,75,L,76,M,77,N,78,O,79,P
    ,80,Q,81,R,82,S,83
590 DATA T,84,U,85,V,86,W,87,X,8
    8,Y,89,Z,90,^,94,a,97,b,98,c
    ,99,d,100,e,101
600 END
```

```
 20 ! HP41C COMPILER-EDITR
 30 ! & BAR CODE GEN PRGM
 40 ! SECOND OF THREE CHAINED
 50 ! PROGRAMS.  THIS ONE IS
 60 ! STORED UNDER FILE "EDITR"
 70 OPTION BASE 1
 80 INTEGER I,J,K,V,K1(2233),E1,
    E2,D,X1,V1,V2,V3
 90 COM INTEGER C2(60),P(2233),F
    6,P5,I1(103)
100 COM T$[60],T1$[30],T2$[30],A
    $[1500],S1$[27],I$[624],C$[6
    0]
110 ! MAIN PROGRAM: WRITES PRPT
    FOR TEXT OR COMMAND ENTRY AN
    D
120 ON ERROR GOTO 1760
130 FOR I=1 TO 2233
140 P(I)=-1
150 NEXT I
160 T$,T1$,T2$=""
170 A$=""
180 F6=0
190 P5=1
200 DISP "DO YOU WANT A LIST OF
    THE AVAILABLE COMMANDS?"
210 INPUT T1$
220 IF T1$="N" OR T1$="NO" THEN
    260
230 IF T$="SCRATCH" THEN 130
240 GOTO 1010
250 ! BEGIN PROMPTER SECTION
260 DISP ">"
270 INPUT T$
280 I=POS(T$," ")
290 V,K,D=0
300 IF I=0 THEN 550
310 T1$=T$[1,I-1]
320 IF T1$<>"DELETE" THEN 360
330 T1$=FNT$(T$[I+1])
340 I=LEN(T1$)+1
350 D=1
360 IF I-1>4 THEN 520
370 FOR J=I-1 TO 1 STEP -1
380 IF T1$[J,J]<"0" OR T1$[J,J]>
    "9" THEN 500
390 V=V+(NUM(T1$[J,J])-48)*10^K
400 K=K+1
410 NEXT J
420 IF V>2233 THEN 520
430 IF D<>1 THEN 460
440 P(V)=-1
450 GOTO 260
460 T$=FNT$(T$[I+1])
470 P(V)=LEN(A$)+1
480 A$=A$&T$&"!"
490 GOTO 260
500 PRINT "?? - GIVE NUMBERED ST
    ATEMENT OR A COMMAND"
510 GOTO 260
520 PRINT "STATEMENT NUMBER VALU
    E TOO LARGE"
530 GOTO 260
540 ! **COMMAND JUMP TABLE**
550 IF T$="NUMBER" THEN 700
560 IF T$="LIST" THEN 840
570 IF T$="COMPILE" THEN 930
580 IF T$="??" THEN 1010
590 IF T$="RENUMBER" THEN 1300
600 IF T$="SAVETEXT" THEN 1540
610 IF T$="GETTEXT" THEN 1650
620 IF T$="SCRATCH" THEN 130
630 IF T$="EXIT" THEN STOP
640 IF T$<>"C0MPILEPVT" THEN 670
650 P5=2
660 GOTO 930
670 PRINT "?? - UNRECOGNIZABLE C
    OMMAND"
680 GOTO 260
690 ! **'AUTO' ROUTINE**
700 DISP "GIVE STARTING VALUE AN
    D SIZE OF INCREMENT"
710 INPUT V,X1
715 PRINTER IS 1
720 IF V>2233 THEN 800
730 PRINT ">";V
740 INPUT T$
750 IF T$="EXIT" THEN 260
760 P(V)=LEN(A$)+1
770 A$=A$&T$&"!"
780 V=V+X1
790 GOTO 720
800 PRINTER IS 16
810 PRINT "STATEMENT NUMBER VALU
    E TOO LARGE"
820 GOTO 260
830 ! **"LIST' ROUTINE**
840 FOR I=1 TO 1500 ! MAX=2233
850 IF P(I)<0 THEN 910
860 FOR J=1 TO 50
870 K=P(I)+J
880 IF A$[K,K]="!" THEN 900
890 NEXT J
900 PRINT I;A$[P(I),K-1]
910 NEXT I
920 GOTO 260
930 ! AT THIS POINT CHAIN TO
940 ! COMPILE PROGRAM OF THE
950 ! THREE SECTIONS.
960 DISP "TEST OF 'EDITR' SEGMEN
    T DONE"
980 DISP "MOVING TO CMPILE"
990 CHAIN "CMPILE"
1000 ! *PRGM COMMANDS LIST SUB*
1010 PRINT " "
1020 PRINT "COMMANDS AVAILABLE I
     N THIS PROGRAM:"
1030 PRINT " "
1040 PRINT "COMPILE  - COMPILES
     41C PROGRAM CURRENTLY ENTER
     ED"
1050 PRINT "DELETE nn - DELETES
     NUMBERED INSTRC FROM PROGRA
     M"
1060 PRINT "EXIT     - HALTS PRO
     GRAM OR STOPS NUMBER GENERA
     TOR"
1070 PRINT "GETTEXT   - GETS PRO
     GRAM TEXT FROM TAPE"
1080 PRINT "LIST      - LISTS EN
     TIRE 41C PROGRAM IN MEMORY"
1090 PRINT "NUMBER    - GENERATE
     S 41C INSTRC NUMBERS - STOP
     PED BY "
1100 PRINT " TYPING 'EXIT'"
1110 PRINT "RENUMBER  - RENUMBER
     S 41C PROGRAM LINE NUMBERS"
1120 PRINT "COMPILEPVT- GENERATE
     S BAR CODE FOR PRIVATE PROG
     RAM"
```

```
1130 PRINT "SAVETEXT  - STORES P
     ROGRAM TEXT ON CASSETTE TAP
     E"
1140 PRINT "SCRATCH   - ERASES E
     NTIRE 41C PROGRAM"
1150 PRINT "??        - LISTS CO
     MMANDS AVAILABLE "
1160 PRINT " "
1170 PRINT "SYNTAX FOR INSTRUCTI
     ON ENTRY:"
1180 PRINT " A)INSTRC FORMAT: "
1190 PRINT "n <41C INSTRC>"
1200 PRINT "(BLANKS ARE USED AS
     DELIMITERS)"
1210 PRINT " B)SPECIAL SYMBOLS:"
1220 PRINT "1) USE '&' FOR SIGMA
      SIGN"
1230 PRINT "2) USE '@' FOR ANGLE
      SIGN"
1240 PRINT "3) USE '#' FOR 'NOT
     EQUAL' SIGN"
1250 PRINT "4) USE SINGLE QUOTES
      (') FOR DOUBLE QUOTES"
1260 PRINT " C)TEXT FORMAT: '<TE
     XT ENTRY>'(NOTE SINGLE QUOT
     ES)"
1270 PRINT "OR   A'<TEXT ENTRY>'
     (FOR APPENDING TEXT)"
1280 GOTO 260
1290 ! *'RENUMBER' SUB*
1300 DISP "ENTER OLD STARTING #,
     NEW STARTING #,AND INCREMEN
     T:"
1310 INPUT V1,V2,V3
1320 FOR I=V2+1 TO V1-1
1330 IF P(I)=-1 THEN 1360
1340 PRINT "ERROR - ATTEMPT MADE
      TO OVERWRITE EXISTING INST
     RUCTIONS"
1350 GOTO 260
1360 NEXT I
1370 FOR I=1 TO 2233
1380 K1(I)=P(I)
1390 IF I>=V1 THEN P(I)=-1
1400 NEXT I
1410 K=V1-1
1420 FOR I=V2 TO 2233 STEP V3
1430 K=K+1
1440 IF K>2233 THEN 260
1450 IF K1(K)<0 THEN 1430
1460 P(I)=K1(K)
1470 NEXT I
1480 PRINT "ERROR: INSTRUCTION N
     UMBER OUT OF BOUNDS"
1490 FOR I=1 TO 2233
1500 P(I)=K1(I)
1510 NEXT I
1520 GOTO 260
1530 ! *'SAVETEXT' ROUTINE*
1540 DISP "GIVE NAME OF FILE TO
     BE SAVED: "
1550 INPUT T1$
1560 CREATE T1$,LEN(A$) DIV 64+5
     0
1570 ASSIGN# 1 TO T1$
1580 PRINT# 1 ; A$
1590 FOR I=1 TO LEN(A$)
1600 PRINT# 1 ; P(I)
1610 NEXT I
1620 ASSIGN# 1 TO *
1630 GOTO 260
1640 ! *'GETTEXT' ROUTINE*
1650 DISP "GIVE THE NAME OF THE
     FILE TO BE READ"
1660 INPUT T1$
1670 A$=""
1680 ASSIGN# 1 TO T1$
1690 READ# 1 ; A$
1700 FOR I=1 TO LEN(A$)
1710 READ# 1 ; P(I)
1720 NEXT I
1730 ASSIGN# 1 TO *
1740 GOTO 260
1750 ! *ERROR CONDITION HANDLING
      ROUTINE*
1760 E1=ERRN
1770 E2=ERRL
1780 IF E1<>60 THEN 1830
1790 PRINT "TAPE IS WRITE PROTEC
     TED. PLEASE FIX!"
1800 IF E2=3850 THEN 1540
1810 IF E2=3940 THEN 1650
1820 GOTO 260
1830 IF E1<>61 THEN 1860
1840 PRINT "TAPE OR FILE IS FULL
     . USE ANOTHER TAPE."
1850 GOTO 1800
1860 IF E1<>62 THEN 1890
1870 PRINT "TAPE NOT IN TAPE DRI
     VE."
1880 GOTO 1800
1890 IF E1<>63 THEN 1920
1900 PRINT "DUPLICATE FILE NAME.
     "
1910 GOTO 1800
1920 IF E1<>64 THEN 1950
1930 PRINT "FILE NOT THERE. TRY
     AGAIN."
1940 GOTO 1800
1950 IF E1=65 THEN 1840
1960 IF E1<>66 THEN 1990
1970 PRINT "FILE CLOSED"
1980 GOTO 1800
1990 IF E1<>67 THEN 2020
2000 PRINT "FILE NAME ERROR."
2010 GOTO 1800
2020 PRINT "ERROR # ";E1;"SEEN F
     T LINE # ";E2
2030 GOTO 260
2040 END
2050 ! *FUNCTION DEFINITIONS*
2060 ! TRIM FUNCTION
2070 DEF FNT$(D$)
2080 INTEGER L,M
2085 DIM D$[60]
2090 FOR L=1 TO LEN(D$)
2100 IF D$[L,L]<>"" THEN 2150
2110 NEXT L
2130 DISP "TRYING TO TRIM BLANK
     STRING!"
2140 GOTO 260
2150 FOR M=LEN(D$) TO L STEP -1
2160 IF D$[M,M]<>"" THEN 2190
2170 NEXT M
2180 GOTO 2130
2190 FNT$=D$[L,M]
2200 FN END
2220 ! ** END OF PRGM **
```

```
 10 ! HP41C COMPILER-CMPILE
 20 ! THIRD CHAINED PROGRAM IN
 30 ! GROUP OF THREE.IT COMPILES
 40 ! THE 41C PROGRAM AND GENERA
    TES THE
 50 ! BAR CODE.  IT IS STORED
 60 ! IN FILE "CMPILE"
 70 OPTION BASE 1
160 INTEGER I,J,K,L,V,M(2233),M1
    ,P1,P2,C,D,H1,H2,S1(16),S2
165 INTEGER F1,F2,F3,F4,F5,F7,F9
    ,E4,X,Y,Z,B(132),C1,D1
170 DIM H1$[3],H2$[3],H4$[9]
173 COM INTEGER C2(60),P(2233),F
    6,P5,I1(103)
175 COM T$[60],T1$[30],T2$[30],A
    $[1500],S1$[27],I$[624],C$[6
    0]
220 ON ERROR GOTO 4250
1805 ! * 'COMPILE' ROUTINE *
1810 PRINTER IS 2
1855 M1=1
1875 FOR J=1 TO 2233
1880 IF P(J)<0 THEN 3335
1881 FOR I=1 TO 50
1882 K=P(J)+I
1883 IF A$[K,K]="!" THEN 1885
1884 NEXT I
1885 T$=A$[P(J),K-1]
1887 E4=M1
1895 ! SCANNER SECTION
1930 T1$,T2$=" "
1935 P1,P2=0
1937 V=-1
1940 IF F6<>1 THEN 1955
1945 M(M1)=0
1950 M1=M1+1
1955 F1,F2,F3,F4,F5,F6=0
1965 IF T$="END" OR T$=".END." T
     HEN 3355
1970 IF LEN(T$)<2 THEN 2055
1975 IF T$[1,1]<>"'" AND T$[1,2]
     <>"A'" THEN 2055
1980 IF T$[1,2]<>"A'" THEN 2000
1985 T$=T$[2]
1990 F1=1
2000 L=LEN(T$)
2005 IF L<18 THEN 2020
2010 PRINT "ALPHA STRING TOO LON
     G IN LINE # ";J
2015 GOTO 2230
2020 IF T$[L,L]="'" THEN 2040
2025 PRINT "ERROR IN ALPHA REGIS
     TER ENTRY INSTRUCTION AT LI
     NE # ";J
2035 GOTO 2230
2040 F2=1 ! SET TEXT FLAG
2045 GOTO 2275
2055 FOR I=1 TO LEN(T$)
2060 T1$=T$[I,I]
2065 IF T1$>="0" AND T1$<="9" TH
     EN 2085
2070 IF (T1$="+" OR T1$="-") AND
      LEN(T$)>1 THEN 2085
2075 IF T1$=" " OR T1$="E" OR T1
     $="." THEN 2085
2080 GOTO 2105
2085 NEXT I
2090 F6=1
2095 GOTO 2275
2105 P1=POS(T$," ")
2110 IF P1<>0 THEN 2130
2115 F3=1
2120 GOTO 2275
2130 T1$=FNT$(T$[P1+1])
2135 T$=T$[1,P1-1]
2140 L=LEN(T1$)
2145 IF T1$[1,1]<>"'" OR T1$[L,L
     ]<>"'" THEN 2175
2150 IF L>9 THEN 2010
2155 T1$=T1$[2,L-1]
2160 F4=1
2165 GOTO 2275
2175 P2=POS(T1$," ")
2180 IF P2=0 THEN 2220
2185 T2$=T1$[1,P2-1]
2190 IF T2$="IND" THEN 2210
2195 PRINT "OPERAND ERROR IN LIN
     E # ";J
2205 GOTO 2230
2210 F5=1
2215 T1$=FNT$(T1$[P2+1])
2220 IF LEN(T1$)<=2 THEN 2275
2225 PRINT "ERROR IN NUMERIC OPE
     RAND IN LINE # ";J
2230 GOSUB 3400
2235 GOTO 1930
2250 ! INTERP SECTION
2275 IF F2<>1 THEN 2395
2285 M(M1)=240+L-2
2315 IF F1<>1 THEN 2333
2316 M(M1)=M(M1)+1
2317 M1=M1+1
2320 M(M1)=127
2333 M1=M1+1
2335 FOR I=2 TO L-1
2339 D$=T$[I,I]
2340 GOSUB 4300
2345 IF Z<>0 THEN 2370
2350 PRINT "INVALID CHARACTER IN
      LABEL OR TEXT"
2365 GOTO 2333
2370 M(M1)=C2(Z)
2372 M1=M1+1
2375 NEXT I
2380 GOTO 3335
2395 IF F6<>1 THEN 2660
2400 F9=0
2410 IF T$[1,1]<>"+" AND T$[1,1]
     <>"-" THEN 2435
2415 IF T$[1,1]<>"-" THEN 2430
2420 M(M1)=28
2425 M1=M1+1
2430 T$=T$[2]
2435 L1=POS(T$," ")
2440 L2=POS(T$,"E")
2445 IF L1=0 THEN L1=LEN(T$)
2450 IF L2<>0 THEN L1=L2-1
2460 FOR I=1 TO L1
2465 IF T$[I,I]<>"." THEN 2495
2470 IF F9=1 THEN 2530
2475 F9=1
2480 M(M1)=26
2485 M1=M1+1
2490 GOTO 2510
2495 IF T$[I,I]<"0" OR T$[I,I]>"
     9" THEN 2490
2500 M(M1)=NUM(T$[I,I])-32
2505 M1=M1+1
2510 NEXT I
2520 IF L1=LEN(T$) AND L2=0 THEN
      3335
2525 IF I=L1 OR I=L2 THEN 2555
```

```
2530 PRINT "DIGIT ENTRY INSTRUCT
     ION ERROR IN LINE # ";J
2535 GOSUB 3400
2545 GOTO 2400
2555 T$=FNT$(T$[I])
2560 IF T$[1,1]<>"E" THEN 2530
2565 M(M1)=27
2570 M1=M1+1
2575 T$=T$[2]
2580 IF T$[1,1]<>"-" THEN 2600
2585 M(M1)=28
2590 M1=M1+1
2600 IF T$[1,1]="-" OR T$[1,1]="
     +" THEN T$=T$[2]
2605 T$=FNT$(T$)
2610 L=LEN(T$)
2615 IF L>2 THEN 2530
2620 FOR I=1 TO L
2625 IF T$[I,I]<"0" OR T$[I,I]>"
     9" THEN 2530
2630 M(M1)=NUM(T$[I,I])-32
2635 M1=M1+1
2640 NEXT I
2645 GOTO 3335
2660 L=LEN(T1$)
2665 IF F3=1 THEN 3265
2670 IF F4=1 THEN 2805
2680 IF L<=2 THEN 2705
2685 PRINT "NUMERIC OPERAND TOO
     LONG IN LINE # ";J
2695 GOTO 2333
2705 IF L<2 THEN 2740
2710 V=(NUM(T1$[1,1])-48)*10+NUM
     (T1$[2,2])-48
2715 IF V>=0 AND V<=99 THEN 2805
2720 PRINT "INCORRECT NUMERIC OP
     ERAND IN LINE # ";J
2730 GOTO 2333
2740 V=0
2745 FOR I=1 TO 26
2750 IF T1$=S1$[I,I] THEN V=I+10
     1
2755 NEXT I
2760 IF V<>0 THEN 2805
2765 V=NUM(T1$)-48
2770 IF V>=0 AND V<=9 THEN 2805
2775 PRINT "INCORRECT STACK OR S
     INGLE DIGIT OPERAND IN LINE
      # ";J
2785 GOTO 2333
2805 IF F5=1 THEN V=V+128
2815 IF T$<>"GTO" AND T$<>"XEQ"
     THEN 3020
2825 IF F5<>1 THEN 2860
2830 M(M1)=174
2835 IF T$="GTO" THEN V=V-128 !
     SET HIGH BIT FOR 'XEQ IND'
2840 M(M1+1)=V
2845 M1=M1+2
2850 GOTO 3335
2860 IF F4<>1 THEN 2955 !
2880 M(M1)=29
2885 IF T$="XEQ" THEN M(M1)=30
2890 L=LEN(T1$)
2895 M(M1+1)=240+L
2900 FOR I=1 TO L
2904 D$=T1$[I,I]
2905 GOSUB 4300
2910 IF Z<>0 THEN 2935
2915 PRINT "INVALID CHARACTER IN
      ALPHA LABEL"
2930 GOTO 2333
2935 M(M1+I+1)=C2(Z)
2936 NEXT I
2940 M1=M1+L+2
2945 GOTO 3335
2955 IF V>14 OR T$="XEQ" THEN 29
     85
2960 M(M1)=177+V
2965 M(M1+1)=0
2970 M1=M1+2
2975 GOTO 3335
2985 M(M1)=208
2990 IF T$="XEQ" THEN M(M1)=224
2995 M(M1+1)=0
3000 M(M1+2)=V
3005 M1=M1+3
3010 GOTO 3335
3020 IF T$<>"STO" THEN 3080
3030 IF V>15 THEN 3055
3035 M(M1)=48+V
3040 M1=M1+1
3045 GOTO 3335
3055 M(M1)=145
3060 M(M1+1)=V
3065 M1=M1+2
3070 GOTO 3335
3080 IF T$<>"RCL" THEN 3140
3090 IF V>15 THEN 3115
3095 M(M1)=32+V
3100 M1=M1+1
3105 GOTO 3335
3115 M(M1)=144
3120 M(M1+1)=V
3125 M1=M1+2
3130 GOTO 3335
3140 IF T$<>"LBL" THEN 3265
3150 IF F4<>1 THEN 3210
3170 M(M1)=192
3175 M(M1+1)=0
3180 L=LEN(T1$)
3185 M(M1+2)=241+L
3190 M(M1+3)=0
3195 M1=M1+2
3200 GOTO 2900
3210 IF V>14 THEN 3235
3215 M(M1)=1+V
3220 M1=M1+1
3225 GOTO 3335
3235 M(M1)=207
3240 M(M1+1)=V
3245 M1=M1+2
3250 GOTO 3335
3265 X=1
3270 Y=103
3271 C=1
3272 D=LEN(T$)
3276 GOSUB 4370
3280 IF Z<>0 THEN 3310
3290 PRINT "UNRECOGNIZABLE INSTR
     UCTION GIVEN IN LINE # ";J
3300 GOTO 2333
3310 M(M1)=I1(Z)
3312 M1=M1+1
3314 IF I1(Z)<64 OR I1(Z)>143 OR
      F3 THEN 3320
3316 PRINT "ERROR: EXTRANEOUS OP
     ERAND IN INSTRUCTION"
3318 GOTO 2333
3320 IF I1(Z)<144 THEN 3335
3322 IF I1(Z)<144 OR I1(Z)>191 O
     R V>=0 THEN 3327
3323 PRINT "ERROR: MISSING OPERA
     ND"
3325 GOTO 2333
3327 M(M1)=V
```

```
3330 M1=M1+1
3335 NEXT J ! *END OCODE LOOP*
3355 M(M1)=192
3360 M(M1+1)=0 ! IN 2ND BYTE
3365 M(M1+2)=47
3370 S2=M1+2
3375 PRINT "COMPILATION COMPLETE
     D"
3380 GOTO 3500
3395 ! *ERR CORRECTION SUB*
3400 PRINTER IS 2
3402 M1=E4
3405 PRINT "THE INSTRUCTION GIVE
     N WAS: ";A$[P(J)+1,K-1]
3407 PRINT "GIVE CORRECTED INSTR
     UCTION (WITHOUT LINE #)  "
3410 DISP " (TO ABORT COMPILE, T
     YPE 'ABORT') "
3412 INPUT T$
3415 IF T$="ABORT" THEN 4115
3417 P(J)=LEN(A$)+1
3420 A$=A$&T$&"!"
3435 RETURN
3450 ! **'RUN' ROUTINE**
3500 DISP "ENTER THE TITLE OF TH
     E PROGRAM (60 CHARS OR LESS
     )"
3505 INPUT T$
3510 ! *WRITE TO A PRINTING DEVI
     CE
3511 H1$=CHR$(27)&CHR$(31)&CHR$(
     2) ! SET UP DIABLO CNTRL CO
     DES
3512 H2$=CHR$(27)&CHR$(83)
3513 H4$="                "
3515 PRINTER IS 2 ! 9 FOR DIABLO
     USING RS232c INTERFACE
3520 PRINT CHR$(12)
3525 PRINT USING "10X,50A" ; T$
3530 PRINT " "
3535 X=S2 DIV 7
3540 IF S2 MOD 7>0 THEN X=X+1
3545 PRINT "          PROGRAM RE
     GISTERS NEEDED: ";X
3550 PRINT " "
3555 ! SET PRINTER BACK TO 85a
3560 ! *END DIABLO OUTPUT*
3565 F1,F3,F4,F5,F7,H1,H2,F9=0
3570 F2=3
3575 F6=1
3580 FOR I=1 TO 132
3585 B(I)=0
3590 NEXT I
3600 ! INSTN TRANSLATION SUB
3605 S1(F2+1)=M(F1+1)
3610 F1=F1+1
3615 F2=F2+1
3620 F3=F3-1
3625 IF F3<>0 THEN 3640
3630 F5=0
3635 GOTO 3865
3640 IF F3<0 THEN 3655
3645 F5=F5+1
3650 GOTO 3865
3655 IF S1(F2)<>0 THEN F7=F7+1
3660 IF M(F1)>143 THEN 3770
3665 ! *1 BYTE INSTNS*
3675 IF M(F1)<>29 AND M(F1)<>30
     THEN 3700
3680 F3=M(F1+1) MOD 16+1
3685 F5=F5+1
3690 GOTO 3865
3700 IF M(F1)<16 OR M(F1)>28 THE
     N 3750
3705 I=F1+1
3710 IF M(I)<16 OR M(I)>28 THEN
     3725
3715 I=I+1
3720 GOTO 3710
3725 F3=I-F1-1
3730 F5=F5+1
3735 IF F3<>0 THEN 3865
3740 F5=0
3745 GOTO 3865
3750 F3=0
3755 F5=0
3760 GOTO 3865
3765 ! *2 BYTE INSTNS*
3770 IF M(F1)>207 THEN 3830
3775 IF M(F1)<192 OR M(F1)>205 T
     HEN 3810
3780 F5=F5+1
3785 IF F1+2<S2 THEN 3800
3790 F3=2
3795 GOTO 3865
3800 F3=M(F1+2) MOD 16+2
3805 GOTO 3865
3810 F3=1
3815 F5=F5+1
3820 GOTO 3865
3825 ! *3 BYTE INST**
3830 IF M(F1)>240 THEN 3850
3835 F3=2
3840 F5=F5+1
3845 GOTO 3865
3850 F3=M(F1) MOD 16
3855 F5=F5+1
3860 ! BAR CODE ROW SETUP
3865 IF F2<16 AND F1<S2 THEN 360
     5
3870 H1=H2
3875 H2=F3
3880 S1(3)=H1 MOD 16*16+F5 MOD 1
     6
3885 IF H1<=F2-3 THEN 3905
3890 H1=F2-3
3895 S1(3)=H1 MOD 16*16+H1 MOD 1
     6
3905 S1(2)=P5*16+F4 MOD 16
3910 FOR I=2 TO F2
3915 F9=F9+S1(I) MOD 256
3920 IF F9>=256 THEN F9=F9 MOD 2
     56+1
3925 NEXT I
3930 S1(1)=F9
3935 ! CONVERSION SECTION:
3940 F4=F4+1
3945 FOR I=1 TO F2
3950 X=S1(I)
3955 FOR Y=2+I*8 TO 3+(I-1)*8 ST
     EP -1
3960 B(Y)=X MOD 2
3965 X=X DIV 2
3970 NEXT Y
3975 NEXT I
3980 B(1)=0
3985 B(2)=0
3990 B(F2*8+3)=1
3995 B(F2*8+4)=0
4000 ! *BAR CODE OUTPUT*
4005 I=F4+1-1
4006 T1$="ROW "&FNF$(I)&" ("&FNF
     $(F6)
4007 IF F6=F7 THEN 4009
```

```
4008 T1$=T1$&" - "&FNF$(F7)
4009 T1$=T1$&")"
4010 PRINTER IS 2 ! LU FOR OUTPU
     T DEVICE (PRINTER/PLOTTER)
4015 PRINT USING "3X,20A" ; T1$
4020 A$=" "
4023 FOR I=1 TO F2*8+4
4025 IF B(I)=0 THEN A$=A$&">>
     "
4027 IF B(I)=1 THEN A$=A$&">>>>
       "
4029 NEXT I
4030 PRINT USING "3X,3A,10A,915A
     ,2A" ; H1$,H4$,A$,H2$
4035 PRINT " "
4040 IF F4 MOD 18=0 THEN PRINT C
     HR$(12)
4045 ! RESET PRINTER TO 85A
4050 ! CLEANUP SECTION:
4055 FOR I=1 TO 16 ! ZERO OUT 16
      BYTE BAR CODE ROW BUFFER
4060 S1(I)=0
4065 B(I)=0
4070 NEXT I
4075 FOR I=17 TO 132
4080 B(I)=0
4085 NEXT I
4090 F2=3
4095 F6=F7
4100 IF F3=0 THEN F6=F6+1
4105 IF F1<S2 THEN 3605
4110 PRINT "BAR CODE GENERATION
     COMPLETED"
4115 CHAIN "EDITR"
4130 !
4250 PRINT "ERROR # ";ERRN;" SEE
     N AT LINE # ";ERRL
4255 PRINT " COMPILE/RUN ABORTED
     "
4260 STOP
4300 X,C=1
4301 Y=59
4305 Z=(X+Y) DIV 2
4310 IF D$[C,C]=C$[Z,Z] THEN RET
     URN
4320 IF D$[C,C]>C$[Z,Z] THEN X=Z
     +1
4330 IF D$[C,C]<C$[Z,Z] THEN Y=Z
     -1
4340 IF X<=Y THEN 4305
4350 Z=0
4360 RETURN
4370 Z=(X+Y) DIV 2
4380 C1=Z*(6-C+1)+1
4390 D1=C1+(D-C)
4400 IF T$[C,D]=I$[C1,D1] THEN 4
     450
4410 IF T$[C,D]>I$[C1,D1] THEN X
     =Z+1
4420 IF T$[C,D]<I$[C1,D1] THEN Y
     =Z-1
4430 IF X<=Y THEN 4370
4440 GOTO 4350
4450 Z=Z+1
4460 RETURN
4670 END
4685 ! *FUNCTION DEFINITIONS*
4785 ! NUMBER FORMAT FCN
4805 DEF FNF$(N)
4815 DIM N$[5]
4820 IF N>=10 THEN 4840
4825 N$=CHR$(N+48)
4830 GOTO 4910
4840 IF N>=100 THEN 4860
4845 N$=CHR$(N DIV 10+48)&CHR$(N
      MOD 10+48)
4850 GOTO 4910
4860 IF N>1000 THEN 4890
4865 I=N DIV 100
4870 J=N MOD 100 DIV 10
4875 N$=CHR$(I+48)&CHR$(J+48)&CH
     R$(N MOD 10+48)
4880 GOTO 4910
4890 I=N DIV 1000
4895 J=N MOD 1000 DIV 10
4900 K=N MOD 100 DIV 10
4905 N$=CHR$(I+48)&CHR$(J+48)&CH
     R$(K+48)&CHR$(N MOD 10+48)
4910 FNF$=N$
4915 FN END
5140 ! TRIM FUNCTION**
5150 DEF FNT$(D$)
5155 INTEGER E,F
5160 FOR E=1 TO LEN(D$)
5170 IF D$[E,E]<>"" THEN 5220
5180 NEXT E
5200 DISP "TRYING TO TRIM BLANK
     STRING!"
5210 GOTO 4115
5220 FOR F=LEN(D$) TO E STEP -1
5230 IF D$[F,F]<>"" THEN 5260
5240 NEXT F
5250 GOTO 5200
5260 FNT$=D$[E,F]
5270 FN END
5280 END
```

85A Adaptation of FULFCN with Data Table

```
  3 ! ASKS FOR HP-41C COMMANDS T
    O MAKE 41C BAR CODE.
  5 OPTION BASE 1
  6 DIM I$[90],A1$[50],A2$[50],F
    $[10],B$[915],X$[10],T$[732]
  7 DIM A$[50],T9$[30],I9$[512]
  8 DIM B1(16),D(30),D1(122),D2(
    122),A1(122),A2(122),D3(16)
  9 DIM B2(150),W$[1]
100 ! READ IN 41-C FUNC TABLE
105 ASSIGN# 1 TO "41FCNS"
107 ON ERROR GOTO 8000
110 FOR I=1 TO 122
120 J=1+6*(I-1)
130 READ# 1 ; T$[J],D1(I),D2(I),
    A1(I),A2(I)
150 NEXT I
160 ASSIGN# 1 TO *
180 T2=I-1
185 ON ERROR GOTO 4130
190 DISP "WOULD YOU LIKE HELP?"
200 INPUT X$
210 X$=X$[1,1]
220 IF X$="N" THEN 420
230 PRINTER IS 2
240 PRINT "THIS PGM ASKS YOU FOR
     HP-41C BAR CODE TYPE COMMAN
    DS."
250 PRINT "TYPE THE COMMAND AS Y
    OU WOULD ON THE 41C WITH FOL
    LOWING EXCEPTIONS:"
260 PRINT
270 PRINT "FOR NUMERIC DATA: DAT
    A **...** "
280 PRINT "FOR ALPHA REPLACE DAT
    A:              DATA '**...**
    '"
290 PRINT "FOR ALPHA APPEND DATA
    :              DATA A'**...**
    '"
300 PRINT "FOR SEQUENCED DATA US
    E FOLLOWING FORMAT:"
310 PRINT " DATASEQ SEQ#: **..**
    "
320 PRINT " FOR INSTANCE, DATASE
    Q 45: 1.234 OR DATASEQ  13:
    'TEST'"
330 PRINT " SIMILARLY, ALPHA TEX
    T STRING    ARGUMENTS, LIKE
    XEQ 'SQR'"
340 PRINT " ARE DELIMITED WITH '
    s."
350 PRINT
355 PRINT "FOR FUNCTION NAMES US
    ING 'Σ'"
357 PRINT "USE '&', i.e. '&+',OR
     '&REG 10' OR 'CL&'."
360 PRINT "IN FUNCTIONS USING TH
    E NOT EQUALSIGN USE '#' INST
    EAD.i.e.'X#Y?'."
370 PRINT "IN TEXT STRINGS USE '
    #' FOR NOT EQUAL, '&' FOR Σ,
     AND '@' FOR   ANGLE."
380 PRINT
390 PRINT "ENTER ?? FOR HELP AT
    ANY TIME."
400 PRINT "TYPE '*STOP' TO STOP
    THE PROGRAM"
410 PRINT "END ALL ENTRIES WITH
    THE 'ENDLINE' KEY"
420 DISP "TYPE FUNCTION TO BE CO
    DED"
430 INPUT I$
440 L=LEN(I$)
450 ! INITIALIZE STRINGS
460 F$=""
470 A1$=""
480 A2$=""
490 A$=""
500 ! *PARSE OUT THE FUNCTION AN
    D ANY ARGUMENTS*
510 T9$=I$
520 GOSUB 5000
530 IF T9 THEN 910
540 I$=T9$
550 IF I$="*STOP" THEN STOP
570 IF L<=0 THEN 420
580 FOR I=1 TO L ! PARSE LOOP
590 IF I$[I,I]=" " THEN 630
600 F$=F$&I$[I,I]
610 NEXT I
620 GOTO 900 ! NO ARGS SO JUMP
630 T9$=I$[I,L]
640 GOSUB 5000
650 IF T9 THEN 910
660 A$=T9$
665 IF LEN(A$)<2 THEN 680
670 IF A$[1,1]="'" OR A$[1,1]="A
    " AND A$[2,2]="'" THEN 780
680 FOR I=1 TO L
690 IF A$[I,I]=" " THEN 730
700 A1$=A1$&A$[I,I]
710 NEXT I
720 GOTO 900 ! ONLY ONE ARG
730 T9$=A$[I,L]
740 GOSUB 5000
750 IF T9 THEN 910
760 A2$=T9$
770 GOTO 900
780 I=3
790 IF A$[1,1]="'" THEN I=2
800 FOR I=I TO L-1
810 IF A$[I,I]="'" THEN 850
820 NEXT I
830 A1$=A$
840 GOTO 900
850 A1$=A$[1,I]
860 I=I+1
870 GOTO 730
880 ! *END OF PARSE ROUTINE*
890 ! WAS THERE AN INST.?
900 IF LEN(F$)>0 THEN 930
910 DISP "ERROR IN ENTRY",I$
920 GOTO 420
930 L1=LEN(F$)
940 L4=LEN(A1$)
950 L5=LEN(A2$)
1000 IF F$="??" THEN 240
1010 IF F$="DATA" THEN 2300
1020 IF F$="DATASEQ" THEN 2285
1030 ! #HERE FOR FUNCTIONS#
1040 X=1
1045 Y=121
1046 I=LEN(F$)
1050 Z=(X+Y) DIV 2
1055 Z1=(Z-1)*6+1
1060 B1(2)=64
1065 B1(3)=D1(Z)*16+D2(Z)
1090 IF F$[1,I]=T$[Z1,Z1+I-1] TH
     EN 1130
```

```
1095 IF F$[1,I]<T$[Z1,Z1+I-1] TH
     EN Y=Z-1
1100 IF F$[1,I]>T$[Z1,Z1+I-1] TH
     EN X=Z+1
1105 IF X<=Y THEN 1050
1110 DISP "FUNCTION NOT FOUND IN
      TABLE",F$
1120 GOTO 420 ! ASK FOR INSTRUC
1130 IF A1(Z)>0 THEN 1230
1140 ! #FUNCS WITH NO ARGS#
1150 IF L4=0 AND L5=0 THEN 1210
1160 DISP "THERE SHOULD BE NO AR
     GUMENTS FOR THIS FUNCTION "
     ,I$
1170 DISP "PLEASE ENTER THE FUNC
     TION AGAIN"
1180 GOTO 430
1210 N=3
1220 GOTO 1460
1230 ! FUNCS WITH ARGUMENTS
1240 GOSUB 3300 ! TYPE OF ARG?
1250 IF A4>4 OR A5>4 THEN 420
1260 IF A1(Z)=2 THEN 2050
1270 ! #SINGLE ARG FUNCTIONS#
1280 IF L4>0 THEN 1310
1290 DISP "ARG REQUIRED";I$
1300 GOTO 1170
1310 IF A1$="IND" THEN 1840
1320 IF A4=0 THEN 1490
1330 IF A4=3 THEN 1640 ! (A-J)
1340 IF A4=3.5 THEN 1650 ! (a-e)
1350 IF A4=4 THEN 1490 ! STACK
1360 ! SINGLE ARG, ALPHA TYPE
1370 IF A2(Z)=1 THEN 1420
1380 DISP "ALPHA ARGUMENTS NOT A
     LLOWED FOR THIS FUNCTION"
1390 GOTO 1170
1420 S1=4
1430 K=1
1440 N=3+LEN(A1$)
1450 GOSUB 3890
1460 GOSUB 3080 ! CHECKSUM
1470 GOSUB 3990 ! OUTPUT
1480 GOTO 420
1490 ! #SINGLE ARG,NUMERIC TYPE#
1500 IF A4<>4 THEN 1570 ! STACK?
1510 IF A1$="X" THEN X=115
1520 IF A1$="Y" THEN X=114
1530 IF A1$="Z" THEN X=113
1540 IF A1$="T" THEN X=112
1550 IF A1$="L" THEN X=116
1560 GOTO 1680
1570 IF F$="GTO." THEN 1770
1580 IF F$="SIZE" THEN 1770
1590 IF F$="DEL" THEN 1770
1600 X=INT(VAL(A1$))
1610 IF X>=0 AND X<=99 THEN 1680
1620 DISP " ARGUMENT OUT OF RANG
     E "
1630 GOTO 1170
1640 IF A4=3 THEN X=NUM(A1$)+37
1650 IF A4=3.5 THEN X=NUM(A1$)+2
     6
1680 IF F$="XEQ" THEN B1(3)=224
1690 IF F$="GTO" THEN B1(3)=208
1700 IF F$="LBL" THEN B1(3)=207
1710 B1(4)=X
1720 N=4
1730 GOTO 1460
1740 ! %HERE FOR DEL/GTO./SIZE%
1770 IF F$="GTO." AND A1$="." TH
     EN B1(4)=15 !
1780 IF F$="GTO." AND A1$="." TH
     EN B1(5)=255
1790 IF F$="GTO." AND A1$="." TH
     EN 1820 !
1800 B1(4)=VAL(A1$) DIV 256
1810 B1(5)=VAL(A1$) MOD 256
1820 N=5
1830 GOTO 1460
1840 IF L5>0 THEN 1870
1850 DISP "MISSING ARGUMENT FOR
     'IND' ",I$
1860 GOTO 1170
1870 IF A5<>4 THEN 1940
1880 IF A2$="X" THEN X=115
1890 IF A2$="Y" THEN X=114
1900 IF A2$="Z" THEN X=113
1910 IF A2$="T" THEN X=112
1920 IF A2$="L" THEN X=116
1930 GOTO 2010
1940 IF A5=0 THEN 1970
1950 DISP "IMPROPER ARGUMENT FOR
      'IND' ",I$
1960 GOTO 1170
1970 X=INT(VAL(A2$))
1980 IF X<0 OR X>99 THEN 1950
2010 B1(4)=128+X
2020 N=4
2030 GOTO 1460
2040 ! #TWO ARG FUNCTIONS#
2050 IF L4>0 AND L5>0 THEN 2080
2060 DISP "TWO ARGUMENTS ARE REQ
     UIRED."
2070 GOTO 1170
2080 IF A4=1 THEN 2110
2090 DISP "IMPROPER 1ST ARG"
2100 GOTO 1170
2110 IF A5=0 THEN 2160
2120 DISP "IMPROPER 2ND ARG"
2130 GOTO 1170
2160 A6=VAL(A2$)
2170 IF ABS(A6)<85 THEN 2200
2180 DISP "KEYCODE IS OUT OF RAN
     GE ";A2$
2190 GOTO 1170
2200 U4=ABS(A6) DIV 10
2210 D4=ABS(A6) MOD 10
2220 IF A6<0 THEN U4=(ABS(A6) DI
     V 10+8) MOD 16
2230 B1(4)=U4*16+D4
2240 S1=5
2250 K=1
2260 N=4+LEN(A1$)
2270 GOSUB 3890
2280 GOTO 1460
2282 ! DATASEQ TYPE CODE
2285 A$=A1$[1,L4-1] ! STRIP ':'
2287 A1$=A2$
2290 A2$=A$
2293 L4=LEN(A1$)
2296 L5=LEN(A2$)
2299 ! DATA TYPE BAR CODE
2300 GOSUB 3300 ! FIND ARG TYPE
2310 IF L4=0 THEN 2740
2320 IF A4>4 OR A5>4 THEN 420
2330 IF A4=0 THEN 2380
2340 IF A4=1 THEN 2940
2350 IF A4=2 THEN 2770
2360 ! # NUMERIC DATA #
2370 !   SET THE DIGITS
2380 FOR I=1 TO L4
2390 X1=NUM(A1$[I,I])
2400 IF X1>=48 AND X1<=57 THEN D
     (I)=VAL(A1$[I,I])
```

```
2410 IF A1$[I,I]=" " THEN D(I)=1
     1
2420 IF A1$[I,I]="+" THEN D(I)=1
     2
2430 IF A1$[I,I]="-" THEN D(I)=1
     3
2440 IF A1$[I,I]="E" THEN D(I)=1
     4
2450 IF D(I)<0 OR D(I)>14 THEN 2
     740
2460 NEXT I
2470 IF F$="DATASEQ" THEN 2590
2471 IF LEN(A1$)>1 THEN 2476
2472 B1(2)=96+10
2473 B1(3)=160+D(1)
2474 N=3
2475 GOTO 1460
2476 I$=A1$
2480 IF L4 MOD 2=0 THEN 2520
2490 B1(2)=96+D(1)
2500 J=2
2510 GOTO 2530
2520 B1(2)=106
2525 J=1
2530 K=3
2535 FOR I=J TO L4-1 STEP 2
2540 B1(K)=16*D(I)+D(I+1)
2550 K=K+1
2560 NEXT I
2570 N=2+LEN(A1$) DIV 2
2580 GOTO 1460
2590 S2=VAL(A2$)
2600 B1(2)=9*16+(S2-1) DIV 256
2610 B1(3)=(S2-1) MOD 256
2620 K=4
2630 J=1
2640 IF LEN(A1$) MOD 2=0 THEN 26
     80
2650 K=5
2660 J=2
2670 B1(4)=16*10+D(1)
2680 FOR I=J TO LEN(A1$) STEP 2
2690 B1(K)=16*D(I)+D(I+1)
2700 K=K+1
2710 NEXT I
2720 N=3+LEN(A1$) MOD 2+LEN(A1$)
      DIV 2
2730 GOTO 1460
2740 DISP "ERROR IN NUMERIC DATA
     ",I$
2750 GOTO 1170
2760 ! ALPHA APPEND DATA
2770 IF F$="DATASEQ" THEN 2850
2775 I$="APPEND"&" '"&A1$&"'"
2780 B1(2)=128+L4
2790 ! SET BYTES TO ASCII VALUE
2800 S1=3
2810 N=LEN(A1$)+2
2820 GOSUB 3890
2840 GOTO 1460
2850 S2=VAL(A2$)
2855 I$=F$&" "&A2$&": APPEND"&"
     '"&A1$&"'"
2860 B1(2)=11*16+(S2-1) DIV 256
2890 N=LEN(A1$)+3
2920 GOTO 3025
2930 ! # ALPHA REPLACE #
2940 IF F$="DATASEQ" THEN 3010
2945 I$=A1$
2950 B1(2)=112+L4
2960 GOTO 2800
3010 S2=VAL(A2$)
3020 B1(2)=10*16+(S2-1) DIV 256
3022 N=LEN(A1$)+3
3025 B1(3)=(S2-1) MOD 256
3028 S1=4
3030 GOSUB 3890
3035 GOTO 1460
3040 ! *END OF MAIN ROUTINE*
3050 ! CALCULATES AN 8 BIT CKSUM
      FOR N BYTE BAR CODE.
3060 ! N MUST BE INITIALIZED TO
     NUMBER OF BYTES IN THE CODE
3070 ! BEFORE THIS ROUTINE IS CA
     LLED.
3080 C=0
3090 FOR I=2 TO N
3100 C=C+B1(I)
3120 NEXT I
3130 C=C DIV 256+C MOD 256
3140 C=C DIV 256+C MOD 256
3150 B1(1)=C
3170 RETURN
3180 !   ARGUMENT TYPE ROUTINE
3190 ! DECIDES TYPE OF 2 POSSIBL
     E ARGUMENTS AND
3200 ! RETURNS INFO TO MAIN PROG
     RAM IN A4 FOR THE 1ST ARG
3210 ! AND A5 FOR THE 2ND.THE AR
     G TYPES ARE DEFINED:
3220 !  TYPE           MEANING
3230 !   0          NUMERIC DATA
3240 !   1    ALPHA REPLACE DATA
3250 !   2     ALPHA APPEND DATA
3260 !   3        A THROUGH J
3270 !  3.5       a THROUGH e
3280 !   4       STACK ARGUMENTS
3290 !   5       ERRORS IN ARGS
3300 A4=0
3310 A5=0
3320 IF L4=0 THEN RETURN
3330 IF A1$="IND" THEN 3610
3340 IF L4=1 AND A1$="E" THEN 34
     10
3350 FOR I=1 TO L4
3360 X1=NUM(A1$[I,I])
3370 IF (X1<43 OR X1>57) AND X1<
     >69 THEN 3400
3380 NEXT I
3390 GOTO 3610
3400 IF L4>1 THEN 3490
3410 IF NUM(A1$)>64 AND NUM(A1$)
     <75 THEN A4=3
3420 IF NUM(A1$)>96 AND NUM(A1$)
     <102 THEN A4=3.5
3430 IF A1$="X" THEN A4=4
3440 IF A1$="Y" THEN A4=4
3450 IF A1$="Z" THEN A4=4
3460 IF A1$="T" THEN A4=4
3470 IF A1$="L" THEN A4=4
3480 GOTO 3590
3490 IF A1$[1,1]="A" AND A1$[2,2
     ]="'" AND A1$[L4,L4]="'" TH
     EN A4=2
3500 IF A1$[1,1]="'" AND A1$[L4,
     L4]="'" THEN A4=1
3510 IF A4>0 THEN 3550
3520 A4=5
3530 DISP "ERROR IN DATA.  CHECK
      FOR MISSING ' OR IMPROPER
     NUMBER"
3540 RETURN
3550 IF A4=1 THEN A1$=A1$[2,L4-1
     ]
3560 IF A4=2 THEN A1$=A1$[3,L4-1
     ]
```

```
3570 IF A4=3 THEN A1$=A1$
3580 IF A4=3.5 THEN A1$=A1$
3590 L4=LEN(A1$)
3600 ! SECOND ARGUMENT
3610 IF L5=0 THEN RETURN
3620 FOR I=1 TO L5
3630 X1=NUM(A2$[I,I])
3640 IF (X1<43 OR X1>57) AND X1<
     >69 THEN 3670
3650 NEXT I
3660 GOTO 3810
3670 IF L5>1 THEN 3760
3680 IF A2$="X" THEN A5=4
3690 IF A2$="Y" THEN A5=4
3700 IF A2$="Z" THEN A5=4
3710 IF A2$="T" THEN A5=4
3720 IF A2$="L" THEN A5=4
3730 IF NUM(A2$)>64 AND NUM(A2$)
     <75 THEN A5=3
3740 IF NUM(A2$)>96 AND NUM(A2$)
     <102 THEN A5=3.5
3750 GOTO 3810
3760 IF A2$[1,1]="A" AND A2$[2,2
     ]="'" AND A2$[L5,L5]="'" TH
     EN A5=2
3770 IF A2$[1,1]="'" AND A2$[L5,
     L5]="'" THEN A5=1
3780 IF A5>0 THEN 3810
3790 A5=5
3800 GOTO 3530
3810 IF A5=1 THEN A2$=A2$[2,L5-1
     ]
3820 IF A5=2 THEN A2$=A2$[3,L5-1
     ]
3830 IF A5=3 THEN A2$=A2$
3840 IF A5=3.5 THEN A2$=A2$
3850 IF A5=4 THEN A2$=A2$
3860 L5=LEN(A2$)
3870 RETURN
3880 ! ALPHA STRNG PLACEMENT SUB
3890 K=1
3900 FOR J=S1 TO N
3910 B1(J)=NUM(A1$[K,K])
3920 IF A1$[K,K]="&" THEN B1(J)=
     126
3930 IF A1$[K,K]="@" THEN B1(J)=
     13
3940 IF A1$[K,K]="#" THEN B1(J)=
     29
3950 K=K+1
3960 NEXT J
3970 RETURN
3980 ! OUTPUT STRING GENERATION
3990 FOR J=1 TO N
3995 D3(J)=B1(J)
4000 FOR I=J*8 TO (J-1)*8+1 STEP
      -1
4005 B2(I)=D3(J) MOD 2
4010 D3(J)=D3(J) DIV 2
4015 NEXT I
4020 NEXT J
4025 B$=">>  >>  "
4030 FOR I=1 TO 8*N
4035 IF B2(I)=0 THEN B$=B$&">>
     "
4040 IF B2(I)=1 THEN B$=B$&">>>>
      "
4045 NEXT I
4050 B$=B$&">>>>  >>  "
4055 PRINT I$&CHR$(27)&CHR$(31)&
     CHR$(2)
4060 IF F$="DATA" OR F$="DATASEQ
     " THEN 4085
4065 IF A1(Z)=0 THEN PRINT USING
      "200A" ; B$
4070 IF A1(Z)=1 THEN 4085
4075 IF A1(Z)=2 THEN 4085
4080 GOTO 4090
4085 PRINT USING "850A" ; B$
4090 PRINT CHR$(27)&CHR$(83)
4095 L=L+1
4100 IF L<15 THEN RETURN
4105 PRINT CHR$(12)
4110 L=0
4115 RETURN
4120 !  ERRORS
4130 IF ERRN<>56 THEN 8020
4140 DISP "INPUT STRING WAS TOO
     LONG, PLEASE CHECK IT.";ERR
     L
4200 GOTO 1170
5000 T9=0 ! FLAG FOR NULL STR
5010 FOR I=1 TO LEN(T9$)
5020 IF T9$[I,I]<>" " THEN 5060
5030 NEXT I
5040 T9=1 ! GET HERE IF NULL ST
5050 RETURN
5060 T9$=T9$[I]
5070 L=LEN(T9$)
5080 IF T9$[L,L]<>" " THEN 5050
5090 T9$=T9$[1,L-1]
5100 GOTO 5070
8000 E=ERRN
8010 E1=ERRL
8020 IF E<>62 THEN 8050
8030 DISP "CARTRIDGE NOT IN PLAC
     E."
8040 GOTO 8100
8050 IF E<>66 THEN 8080
8060 DISP "FILE NOT OPENED, WILL
      TRY AGAIN."
8070 GOTO 105
8080 IF E<>67 AND E<>73 THEN 816
     0
8090 DISP "CAN NOT FIND FILE '41
     FCNS'"
8100 DISP "GET TAPE WITH '41FCNS
     ' FILE ON IT."
8110 DISP "HAVE YOU INSERTED THE
      TAPE? 'Y'OR'N'"
8120 INPUT X$
8130 IF X$="Y" THEN 105
8140 IF X$<>"N" THEN 8110
8150 GOTO 9150
8160 DISP "TAPE READ ERROR."
8170 GOTO 9140
9120 E=ERRN
9130 E1=ERRL
9140 PRINT "ERROR";E;"AT LINE";E
     1
9150 DISP "ABORTED"
9170 END
```

```
1   .%       , 4 , 12 , 0 , 0
2   .%CH     , 4 , 13 , 0 , 0
3   .&+      , 4 , 7 , 0 , 0
4   .&-      , 4 , 8 , 0 , 0
5   .&REG    , 9 , 9 , 1 , 0
6   .*       , 4 , 2 , 0 , 0
7   .+       , 4 , 0 , 0 , 0
8   .-       , 4 , 1 , 0 , 0
9   ./       , 4 , 3 , 0 , 0
10  .1/X     , 6 , 0 , 0 , 0
11  .10^X    , 5 , 7 , 0 , 0
12  .ABS     , 6 , 1 , 0 , 0
13  .ACOS    , 5 , 13 , 0 , 0
14  .ADV     , 8 , 15 , 0 , 0
15  .AOFF    , 8 , 11 , 0 , 0
16  .AON     , 8 , 12 , 0 , 0
17  .ARCL    , 9 , 11 , 1 , 0
18  .ASHF    , 8 , 8 , 0 , 0
19  .ASIN    , 5 , 12 , 0 , 0
20  .ASN     , 0 , 15 , 2 , 1
21  .ASTO    , 9 , 10 , 1 , 0
22  .ATAN    , 5 , 14 , 0 , 0
23  .AVIEW   , 7 , 14 , 0 , 0
24  .BEEP    , 8 , 6 , 0 , 0
25  .BST     , 0 , 7 , 0 , 0
26  .CAT     , 0 , 0 , 0 , 0
27  .CF      , 10 , 9 , 1 , 0
28  .CHS     , 5 , 4 , 0 , 0
29  .CL&     , 7 , 0 , 0 , 0
30  .CLA     , 8 , 7 , 0 , 0
31  .CLD     , 7 , 15 , 0 , 0
32  .CLP     , 0 , 4 , 2 , 1
33  .CLRG    , 8 , 10 , 0 , 0
34  .CLST    , 7 , 3 , 0 , 0
35  .CLX     , 7 , 7 , 0 , 0
36  .COPY    , 0 , 3 , 1 , 1
37  .COS     , 5 , 10 , 0 , 0
38  .D-R     , 6 , 10 , 0 , 0
39  .DEC     , 5 , 15 , 0 , 0
40  .DEG     , 8 , 0 , 0 , 0
41  .DEL     , 0 , 2 , 1 , 0
42  .DELETE  , 0 , 11 , 0 , 0
43  .DSE     , 9 , 7 , 1 , 0
44  .END     , 12 , 0 , 0 , 0
45  .ENG     , 9 , 14 , 1 , 0
46  .ENTER^  , 8 , 3 , 0 , 0
47  .E^X     , 5 , 5 , 0 , 0
48  .E^X-1   , 5 , 8 , 0 , 0
49  .FACT    , 6 , 2 , 0 , 0
50  .FC?     , 10 , 13 , 1 , 0
51  .FC?C    , 10 , 11 , 1 , 0
52  .FIX     , 9 , 12 , 1 , 0
53  .FRC     , 6 , 9 , 0 , 0
54  .FS?     , 10 , 12 , 1 , 0
55  .FS?C    , 10 , 10 , 1 , 0
56  .GRAD    , 8 , 2 , 0 , 0
57  .GTO     , 1 , 13 , 1 , 1
58  .GTO.    , 0 , 1 , 1 , 0
59  .HMS     , 6 , 12 , 0 , 0
60  .HMS+    , 4 , 9 , 0 , 0
61  .HMS-    , 4 , 10 , 0 , 0
62  .HR      , 6 , 13 , 0 , 0
63  .INT     , 6 , 8 , 0 , 0
64  .ISG     , 9 , 6 , 1 , 0
65  .LASTX   , 7 , 6 , 0 , 0
66  .LBL     , 12 , 13 , 1 , 1
67  .LN      , 5 , 0 , 0 , 0
68  .LN1+X   , 6 , 5 , 0 , 0
69  .LOG     , 5 , 6 , 0 , 0
70  .MEAN    , 7 , 12 , 0 , 0
71  .MOD     , 4 , 11 , 0 , 0
72  .OCT     , 6 , 15 , 0 , 0
73  .OFF     , 8 , 13 , 0 , 0
74  .ON      , 0 , 9 , 0 , 0
75  .P-R     , 4 , 14 , 0 , 0
76  .PACK    , 0 , 10 , 0 , 0
77  .PI      , 7 , 2 , 0 , 0
78  .PROMPT  , 8 , 14 , 0 , 0
79  .PSE     , 8 , 9 , 0 , 0
80  .R-D     , 6 , 11 , 0 , 0
81  .R-P     , 4 , 15 , 0 , 0
82  .R/S     , 0 , 5 , 0 , 0
83  .RAD     , 8 , 1 , 0 , 0
84  .RCL     , 9 , 0 , 1 , 0
85  .RDN     , 7 , 5 , 0 , 0
86  .RND     , 6 , 14 , 0 , 0
87  .RTN     , 8 , 5 , 0 , 0
88  .R^      , 7 , 4 , 0 , 0
89  .SCI     , 9 , 13 , 1 , 0
90  .SDEV    , 7 , 12 , 0 , 0
91  .SF      , 10 , 8 , 1 , 0
92  .SIGN    , 7 , 10 , 0 , 0
93  .SIN     , 5 , 9 , 0 , 0
94  .SIZE    , 0 , 6 , 1 , 0
95  .SQRT    , 5 , 2 , 0 , 0
96  .SST     , 9 , 2 , 0 , 0
97  .ST*     , 9 , 4 , 1 , 0
98  .ST+     , 9 , 2 , 1 , 0
99  .ST-     , 9 , 3 , 1 , 0
100 .ST/     , 9 , 5 , 1 , 0
101 .STO     , 9 , 1 , 1 , 0
102 .STOP    , 8 , 4 , 0 , 0
103 .TAN     , 5 , 11 , 0 , 0
104 .TONE    , 9 , 15 , 1 , 0
105 .VIEW    , 9 , 8 , 1 , 0
106 .X#0?    , 6 , 3 , 0 , 0
107 .X#Y?    , 7 , 9 , 0 , 0
108 .X<0?    , 6 , 6 , 0 , 0
109 .X<=0?   , 7 , 11 , 0 , 0
110 .X<=Y?   , 4 , 6 , 0 , 0
111 .X<>     , 12 , 14 , 1 , 0
112 .X<>Y    , 7 , 1 , 0 , 0
113 .X<Y?    , 4 , 4 , 0 , 0
114 .X=0?    , 6 , 7 , 0 , 0
115 .X=Y?    , 7 , 8 , 0 , 0
116 .X>0?    , 6 , 4 , 0 , 0
117 .X>Y?    , 4 , 5 , 0 , 0
118 .XEQ     , 1 , 14 , 1 , 1
119 .XROM    , 10 , 0 , 2 , 0
120 .X^2     , 5 , 1 , 0 , 0
121 .Y^X     , 5 , 3 , 0 , 0
122 .*END*   , 0 , 0 , 0 , 0
```

1000 N.E. Circle Blvd., Corvallis, OR 97330

82153-90019

Printed in U.S.A.